2020

# Reporte de plataformas de dispositivos y lenguajes de programación Móvil


DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MOVILES

CARLOS HUMBERTO TEJEDA OSORIO

INSTITUTO TECNOLOGICO SUPERIOR DE CHICONTEPEC

# INSTITUTO TECNOLOGICO SUPERIOR DE CHICONTEPEC

## INGENIERIA EN SISTEMAS COMPUTACIONALES

NOMBRE DE LA MATERIA:

Desarrollo de aplicaciones para dispositivos móviles

SEMESTRE:

5º semestre

NOMBRE DEL ALUMNO:

Carlos Humberto Tejeda Osorio.

NOMBRE DEL DOCENTE:

Ing. Francisco Javier Hernández Hernández

TRABAJO:

Investigación Arquitectura, Entornos de desarrollo y emuladores para disp. Moviles.

Chicontepec, Ver 4 de noviembre de 2020

# Índice

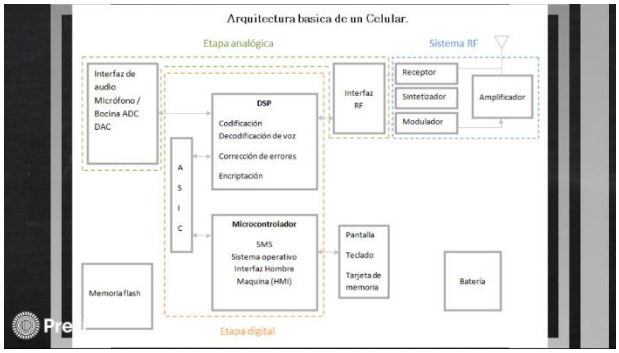
## Introducción

Los dispositivos móviles conocidos como Smartphone, laptops, tablas digitales, PDA, GPS entre otros están a la punta de la tecnología mundial, es muy interesante saber su arquitectura, como tal el cómo funciona estos dispositivos. Este trabajo estará enfocado especialmente a su arquitectura “hardware”, los componentes los cuales ayudan a su funcionamiento, pasaremos por una historia breve de estos dispositivos los cuales manejamos a diario esto con el fin de adquirir un conocimiento en características, tipos de procesadores, sus funciones, para llegar a una conclusión de tener similitudes y diferencias arquitectónicas y de desempeño frente a los dispositivos que no son móviles como los computadores los cuales hemos visto en clase adicionalmente sistemas operativos los cuales están enfocados en estos dispositivos y breve mente describiremos componentes de audio, de video, de voz, comunicaciones, aplicaciones y demás.

Para este trabajo tendremos en cuenta la descripción de los componentes internos y sus funciones, la explicación de sus diagramas en bloques, las características y funciones de su(s) procesador(es), las características, organización y uso de la(s) memoria(s), la segmentación de instrucciones (pipelining) si la tienen, las arquitecturas predominantes y sus fabricantes, las características generales que deben reunir los circuitos integrados destinados a estos dispositivos, y el comparativo con las arquitecturas no móviles.

# Marco Teórico

## ¿Qué es una Arquitectura?




Arquitectura Informática. Es el diseño conceptual y la estructura operacional fundamental de un sistema de Computadora. Es decir, es un modelo y una descripción funcional de los requerimientos y las i implementaciones de diseño para varias partes de una computadora, con especial interés en la forma en que la unidad central de proceso (UCP) trabaja internamente y accede a las direcciones de memoria. También suele definirse como la forma de seleccionar e interconectar componentes de hardware para crear computadoras según los requerimientos de funcionalidad, rendimiento y costo.

Los problemas de diseño no pueden ser resueltos sin la ayuda de un computador, siendo la maquina un complemento y no un substituto del talento creativo, la computadora mientras no pueda inventar, puede explorar relaciones muy rápida y sistemáticamente de acuerdo a reglas pre establecidas. El computador funciona como una extensión natural de la habilidad analítica del hombre.

Término general que se aplica a la estructura de un sistema informático o de una parte del mismo. El término se aplica también al diseño del software de sistema, por ejemplo, el sistema operativo, y a la combinación de Hardware y Software básico que comunica los aparatos de una Red informática. La arquitectura de ordenadores se refiere a toda una estructura y a los detalles necesarios para que sea funcional, es decir, cubre sistemas informáticos, microprocesadores, circuitos y programas del sistema. Por lo general, el término no suele referirse a los programas de aplicación, como hojas de cálculo o procesadores de textos, que son necesarios para realizar una tarea pero no para que el sistema funcione.

# Tipos de arquitecturas para dispositivos moviles

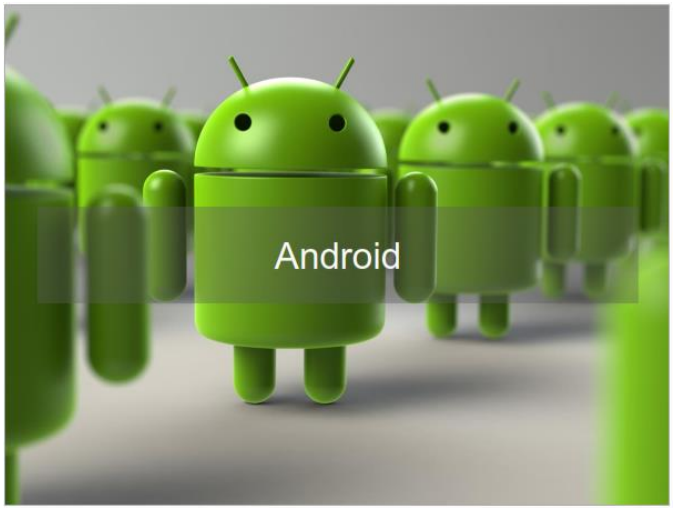
## Arquitectura de aplicaciones: antes

- Arquitectura de 3 capas: cliente móvil, servidor y BD
- Transaccionales
- Modo offline y sincronización con servidor
- Aplicación típica: automatización de fuerza de ventas

## Arquitectura de aplicaciones: ahora

- Arquitectura de múltiples capas
- Orientada a servicios: Twitter, Facebook, aws, maps.
- Grandes volúmenes de información (Big Data, Bases de datos No SQL)
- Procesamiento a gran escala
- Aplicaciones para múltiples plataformas.
- Aplicaciones que aprendan de: Información previa, el contexto, interacción del usuario, preferencias de usuario, información de otros usuarios.
- Aplicaciones que reconozcan: Patrones del entorno, imágenes, gestos, sonidos, movimientos.

## Plataforma Android




La arquitectura de Android está compuesta por cuatro capas, la primera de ellas es un Kernel basado en Linux, le siguen las bibliotecas entre las que se encuentran las básicas correspondientes a la máquina virtual, a continuación, está el marco de aplicaciones o framework y finalmente las aplicaciones. La descripción se da a continuación:

**Núcleo Linux:** Android se basa en Linux para los servicios base del sistema como gestión de memoria y de procesos, pila de red, modelo de controladores y seguridad. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software. Esto permite que se pueda acceder a los componentes sin necesidad de conocer el modelo o características precisas que están instalados en cada dispositivo.

**Bibliotecas:** esta capa incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas bibliotecas se ofrecen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), biblioteca de medios, biblioteca de gráficos, 3D y SQLite, entre otras.

Algunas bibliotecas que se incluyen habitualmente son: Gestor de superficies, SGL (Scalable Graphics Library), Open GL ES (Open GL for Embedded Systems), Bibliotecas multimedia, WebKit, SSL (Secure Sockects Layer), Free Type, SQLite y Biblioteca C de sistema (libc).

**Runtime de Android:** en el mismo nivel están las bibliotecas de entorno de ejecución (no se considera una capa en sí mismo, dado que también está formado por bibliotecas), Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas habituales del lenguaje Java.

**Marco de trabajo de aplicaciones**: La arquitectura está diseñada para simplificar la reutilización de componentes; una aplicación puede publicar sus capacidades y después otra aplicación puede hacer uso de las mismas siempre que atienda a las reglas de seguridad del framework. Los desarrolladores tienen acceso total a los mismos APIs (Interfaz de Programación de Aplicaciones) del framework usados por las aplicaciones base.

La mayoría de los componentes de esta capa son bibliotecas Java que acceden a los recursos a través de la máquina virtual Dalvik. Entre las más importantes se encuentran las siguientes: Administrador de actividades, Administrador de ventanas, Proveedor de contenidos, Vistas, Administrador de notificaciones, Administrador de paquetes, Administrador de telefonía, Administrador de recursos, Administrador de ubicaciones, Administrador de sensores, Cámara y Multimedia.

**Aplicaciones:** las aplicaciones base incluyen un navegador, cliente de correo electrónico, programa de mensajería, calendario, mapas, contactos, juegos y el inicio (launcher) que es donde se muestran las aplicaciones instaladas permitiendo lanzarlas, así como mostrar widgets (pequeñas aplicaciones). Las aplicaciones están escritas en lenguaje Java o C/C++.

## Especificaciones técnicas

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits, x86 32 y 64 bits, MIPS y NEON. |
| *Kernel* | Kernel de Linux 3.X dependiendo de la versión de Android y el dispositivo. |
| *Sistema de ficheros* | Soporta varios, pero principalmente utiliza EXT4 o JFFS2 para el sistema de ficheros internos. Acepta tarjetas de memoria formateadas en FAT32. Soporta cifrado de disco desde la versión 4.3. |
| *Ejecutables* | Bytecode. Ejecutable por la máquina Dalvik o por el más reciente Android Runtime (ART). |
| *Plataforma del sistema* | Basado en Linux. |
| *Licencia* | Licencia Apache 2.0. Los fabricantes modifican el código del sistema y lo adaptan a los dispositivos con libertad dentro de unos parámetros mínimos comunes. |

## Estructura de un APK

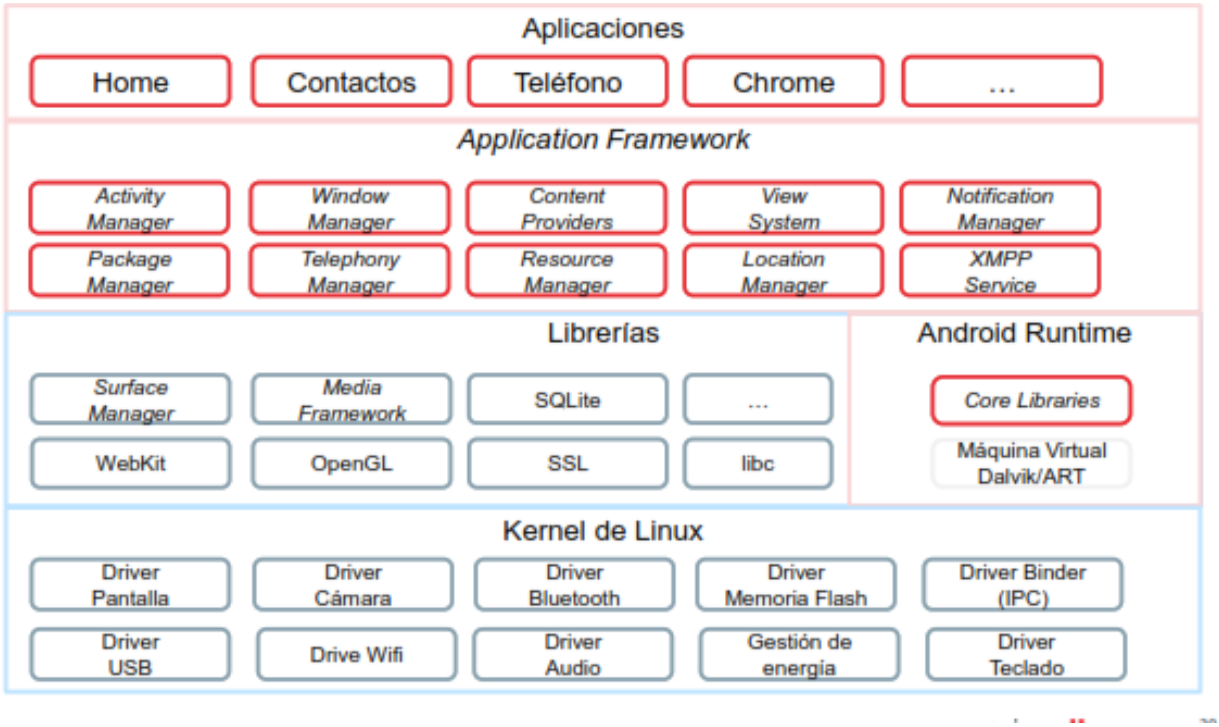
Las aplicaciones Android se empaquetan en ficheros APK

(zip), los cuales contienen los siguientes elementos:

- META-INF: directorio que contiene un listado de ficheros (MANIFEST.MF). El certificado de la app (CERT.RSA), una lista de recursos de la app y hashes SHA-1 de los ficheros indicados en el listado.
- lib: directorio con el código nativo utilizado por la app para plataformas específicas (ARM, x86, MIPS).
- assets: directorio con diferentes elementos utilizados por la app.

- res: directorio con recursos utilizados por la app (iconos, imágenes, constantes, etc.).
- resources. arsc: ficheros XML pre-compilados con definiciones del interfaz de usuario.
- classes.dex: código compilado de la aplicación para el Runtime de Android.
- AndroidManifest.xml: fichero de información de la aplicación.

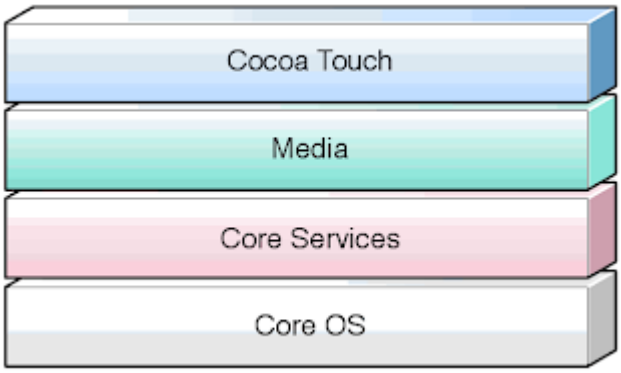
## Arquitectura del sistema operativo



## Plataforma iOS

La capa de aplicación dentro del marco de iOS se llama Cocoa Touch.

A continuación, se resumen algunos de los marcos que se encuentran en cada capa de la pila de IOS, a partir de la capa de base.




Core OS. Este nivel contiene el Kernel, el sistema de archivos, la infraestructura de red, seguridad, administración de energía, y una serie de controladores de dispositivos. También incluye las API (Interfaz de programación de aplicaciones) a nivel de sistema para muchos servicios.

Core Services. Los marcos de este nivel proveen servicios básicos, tales como la manipulación de cadenas, gestión de cobro, las redes, los servicios públicos de URL, gestión de contactos y preferencias. También ofrecen servicios basados en las características de hardware de un dispositivo, como el GPS, brújula, acelerómetro y giroscopio. Ejemplos de marcos en esta capa son Core Location, Core Motion y configuración del sistema.

Esta capa incluye tanto la Foundation y Foundation Core, los frameworks que proporcionan abstracciones para tipos de datos comunes, tales como cadenas y colecciones. La capa Core Frameworks también contiene datos básicos, un marco para la gestión del gráfico de objetos y la persistencia de objetos.

Media. Los marcos y servicios de esta capa dependen de la capa Core Services y proporcionan servicios gráficos y multimedia para la capa de Cocoa Touch. Incluyen Core Graphics, Core Text, OpenGL ES, Core Animation, AVFoundation, Core Audio, y la reproducción de vídeo.

Cocoa Touch. Los frameworks de esta capa apoyan directamente aplicaciones basadas en iOS. Incluyen marcos como Game Kit, Kit Mapa y iAd.

La capa de Cocoa Touch y la capa Core Services cada uno tiene un framework de Objective-C que es especialmente importante para el desarrollo de aplicaciones para iOS. Estos son los frameworks fundamentales de Cocoa Touch en iOS:

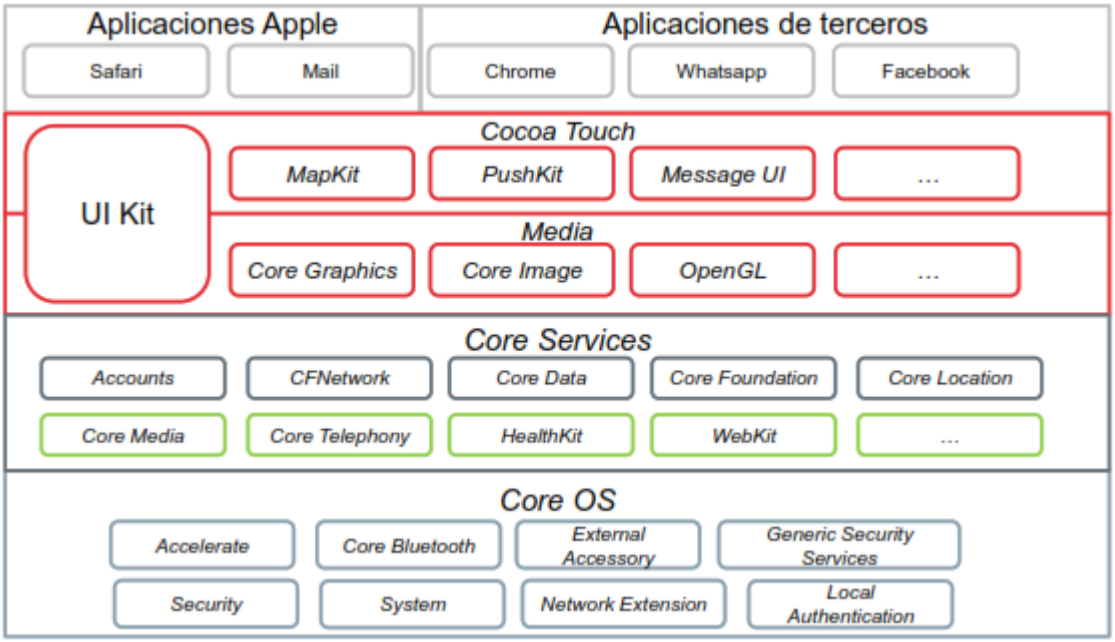
• UIKit: Este marco de trabajo proporciona los objetos que una aplicación muestra en su interfaz de usuario y define la estructura para el comportamiento de la aplicación, incluyendo la gestión de eventos y el dibujo.

• Foundation: Este marco define el comportamiento básico de los objetos, establece mecanismos para su gestión, y proporciona objetos para los tipos primitivos de datos, colecciones y servicios del sistema operativo. Foundation es esencialmente una versión orientada a objetos del marco de Foundation Core.

## Especificaciones técnicas

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits. |
| *Kernel* | XNU. Basado en el kernel Darwin de OS X. |
| *Sistema de ficheros* | HFSX. Basado en HFS+ de iOS. Sin almacenamiento extraíble y con cifrado de disco por defecto. |
| *Ejecutables* | Formato Match-O, el mismo que el utilizado para aplicaciones de OS X. |
| *Plataforma del sistema* | Basado en BSD, igual que OS X. |
| *Licencia* | Propietario. No se permite su instalación en equipos que no sean manufacturados por Apple. Su licencia prohíbe la ingeniería inversa del sistema. |

## Arquitectura del sistema operativo




## Componentes de una aplicación

Las aplicaciones en iOS están compuestas por una serie de elementos comunes:

- AppDelegate: es el encargado de mediar entre el sistema operativo y la aplicación. La mayoría de eventos importantes que pueden interferir en el uso de una app (ej.: llamada telefónica) son manejados por él.
- ViewControlers: son los componentes principales de la aplicación. Cada aplicación debe tener al menos uno, aunque suelen tener varios. Un ViewController maneja la porción del interfaz mostrado al usuario y gestiona la interacción entre el interfaz de usuario y los datos.
- Storyboards: definen las interfaces de usuario y transiciones entre las diferentes pantallas de la aplicación, aunque también se pueden definir de forma programática.

## Windows

Windows 10 trata de unificar la arquitectura de todas las plataformas de Microsoft en una sola plataforma integrada con:

- Un tipo único de kernel.
- Un único sistema de archivos.
- Un modelo de aplicación único para todas las plataformas.

Windows 10 integra las siguientes plataformas:

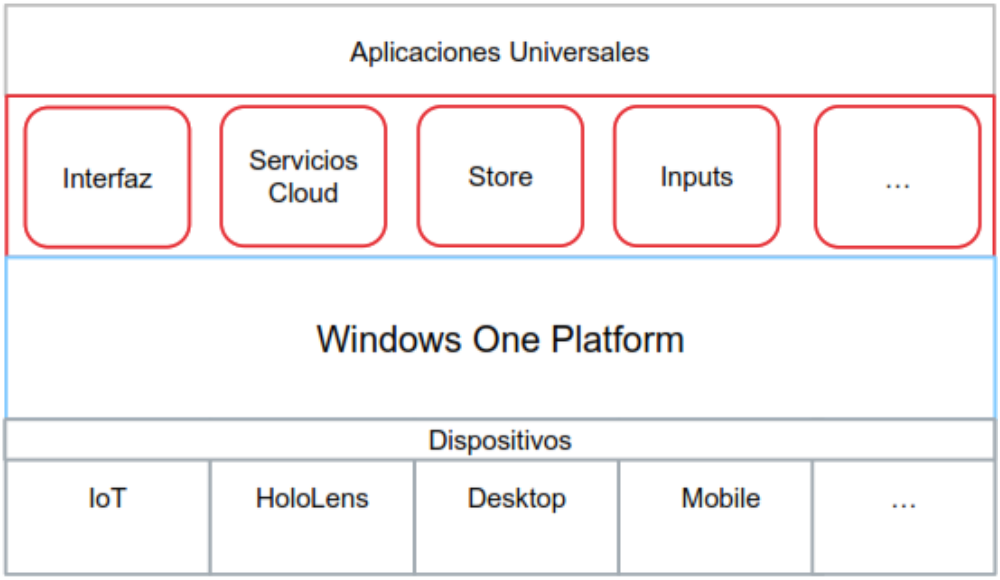
- Windows para escritorio.
- Windows Phone.
- Xbox One.
- Windows para IoT.

De esta manera, una aplicación desarrollada para Windows 10 podrá ser ejecutada en cualquier plataforma de Microsoft (Universal App Platform).

## Especificaciones técnicas

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits, X86 y X64 y otras arquitecturas IoT. |
| *Kernel* | Windows One Core para todas las plataformas. |
| *Sistema de ficheros* | NFTS por defecto. FAT en tarjetas SD. |
| *Ejecutables* | Universal Windows Apps (C#, C, C++, JavaScript). |
| *Plataforma del sistema* | Windows. Específica para cada dispositivo en el que se instala (Xbox, móvil, escritorio, etc.). |
| *Licencia* | Propietario. |

## Arquitectura del sistema operativo




## Aplicaciones

Las aplicaciones de Windows se desarrollan en base a:

- Un código común a todas las plataformas que incluye los interfaces de usuario que se van a mostrar. Debido al diseño de los interfaces de Windows, el contenido de los mismos se puede adaptar de forma sencilla a los diferentes dispositivos que ejecutan las aplicaciones.
- Un código específico en forma de extensión por cada plataforma en la que se quiere ejecutar la aplicación.

- El código común de una aplicación suele contener la lógica de negocio suficiente para ejecutar la aplicación en dispositivos móviles y escritorio.
- Las extensiones permiten a la aplicación acceder a funcionalidad específica de la plataforma en la que se está ejecutando.
- Las aplicaciones deben comprobar de forma activa que se están ejecutando en la plataforma antes de acceder a la funcionalidad de la extensión.

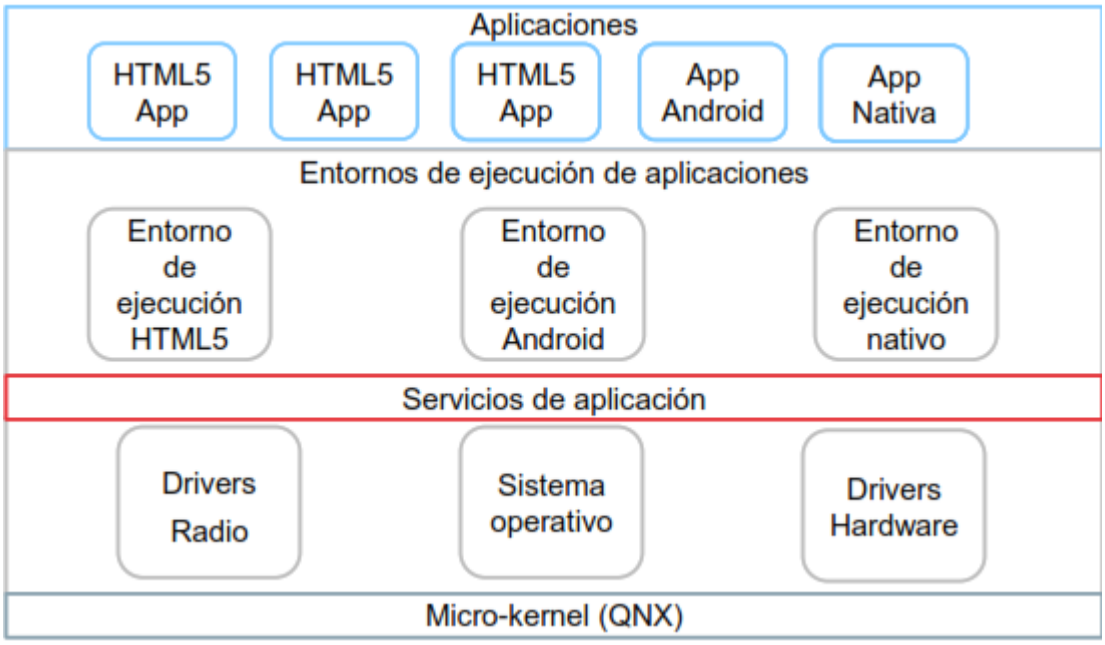
## BlackBerry

- BlackBerry es un sistema operativo de micro-kernel (QNX Neutrino RTOS).
- La mayoría de funciones del sistema se implementan fuera del kernel.
- El sistema operativo y los drivers del sistema utilizan el micro-kernel para acceder al hardware del dispositivo a bajo nivel.
- Por encima del sistema operativo se sitúan los servicios y plataformas que ofrece el sistema operativo a cada plataforma de ejecución: HTML5, Android y Cascadas (Nativas).
- Cada aplicación es ejecutada en su plataforma específica de ejecución.
- El sistema de ficheros de BlackBerry se divide en cuatro particiones:
  - Sistema Operativo: partición de solo lectura con el sistema operativo.
  - Base File System: contiene ficheros del sistema y también es de solo lectura.
  - Work File System: contiene las aplicaciones y datos del entorno de trabajo. Se cifra por defecto.
  - Personal File System: contiene las aplicaciones personales del usuario. No se cifra por defecto.
  -

## Especificaciones técnicas

| | |
|---|---|
| *Arquitecturas válidas* | ARM. |
| *Kernel* | RTOS Microkernel. |
| *Sistema de ficheros* | QMX para la memoria interna. FAT en tarjetas SD. |
| *Ejecutables* | C++, HTML5, Java para aplicaciones portadas de Android. |
| *Plataforma del sistema* | QMX OS (tipo UNIX). |
| *Licencia* | Propietario. |

## Arquitectura del sistema operativo



## Resumen Comparativo de Especificaciones Técnicas

**Android**

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits, x86 32 y 64 bits, MIPS y NEON. |
| *Kernel* | Kernel de Linux 3.X dependiendo de la versión de Android y el dispositivo. |
| *Sistema de ficheros* | Soporta varios, pero principalmente utiliza EXT4 o JFFS2 para el sistema de ficheros internos. Acepta tarjetas de memoria formateadas en FAT32. Soporta cifrado de disco desde la versión 4.3. |
| *Ejecutables* | Bytecode. Ejecutable por la máquina Dalvik o por el más reciente Android Runtime (ART). |
| *Plataforma del sistema* | Basado en Linux. |
| *Licencia* | Licencia Apache 2.0. Los fabricantes modifican el código del sistema y lo adaptan a los dispositivos con libertad dentro de unos parámetros mínimos comunes. |

**iOS**

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits. |
| *Kernel* | XNU. Basado en el kernel Darwin de OS X. |
| *Sistema de ficheros* | HFSX. Basado en HFS+ de iOS. Sin almacenamiento extraíble y con cifrado de disco por defecto. |
| *Ejecutables* | Formato Match-O, el mismo que el utilizado para aplicaciones de OS X. |
| *Plataforma del sistema* | Basado en BSD, igual que OS X. |
| *Licencia* | Propietario. No se permite su instalación en equipos que no sean manufacturados por Apple. Su licencia prohíbe la ingeniería inversa del sistema. |

**BlackBerry**

| | |
|---|---|
| *Arquitecturas válidas* | ARM. |
| *Kernel* | RTOS Microkernel. |
| *Sistema de ficheros* | QMX para la memoria interna. FAT en tarjetas SD. |
| *Ejecutables* | C++, HTML5, Java para aplicaciones portadas de Android. |
| *Plataforma del sistema* | QMX OS (tipo UNIX). |
| *Licencia* | Propietario. |

**Windows**

| | |
|---|---|
| *Arquitecturas válidas* | ARM 32 y 64 bits, X86 y X64 y otras arquitecturas IoT. |
| *Kernel* | Windows One Core para todas las plataformas. |
| *Sistema de ficheros* | NFTS por defecto. FAT en tarjetas SD. |
| *Ejecutables* | Universal Windows Apps (C#, C, C++, JavaScript). |
| *Plataforma del sistema* | Windows. Específica para cada dispositivo en el que se instala (Xbox, móvil, escritorio, etc.). |
| *Licencia* | Propietario. |

## Lenguajes de Programación.

## Que es un lenguaje de programación.


Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para, de esa forma, crear programas que controlen el comportamiento físico y lógico de una máquina.

Mediante este lenguaje se comunican el programador y la máquina, permitiendo especificar, de forma precisa, aspectos como: cuáles datos debe operar un software específico; cómo deben ser almacenados o transmitidos esos datos; las acciones que debe tomar el software dependiendo de las circunstancias variables.

Para explicarlo mejor (en otras y con menos palabras), el lenguaje de programación es un sistema estructurado de comunicación, el cual está conformado por conjuntos de símbolos, palabras claves, reglas semánticas y sintácticas que permiten el entendimiento entre un programador y una máquina.

Es importante recalcar que existe el error común de usar como sinónimos el lenguaje de programación y el lenguaje informático, pero ¿por qué no debemos confundirlos?

Pues, es debido a que el lenguaje de programación obedece a un conjunto de reglas que permiten expresar las instrucciones que serán interpretadas por el programador. Y el lenguaje informático comprende otros lenguajes que dan formato a un texto pero no son programación en sí mismos.

Entonces, no todos los lenguajes informáticos son de programación, pero todos los lenguajes de programación son a la vez informáticos.

## ¿Qué tipos de lenguaje de programación existen?

El lenguaje de programación es la base para construir todas las aplicaciones digitales que se utilizan en el día a día y se clasifican en dos tipos principales: lenguaje de bajo nivel y de alto nivel.

### Lenguaje de programación de bajo nivel

Son lenguajes totalmente orientados a la máquina.

Este lenguaje sirve de interfaz y crea un vínculo inseparable entre el hardware y el software.

Además, ejerce un control directo sobre el equipo y su estructura física. Para aplicarlo adecuadamente es necesario que el programador conozca sólidamente el hardware. Éste se subdivide en dos tipos:

**Lenguaje máquina**

Es el más primitivo de los lenguajes y es una colección de dígitos binarios o bits (0 y 1) que la computadora lee e interpreta y son los únicos idiomas que las computadoras entienden.

Ejemplo: 10110000 01100001

No entendemos muy bien lo que dice ¿verdad? Por eso, el lenguaje ensamblador nos permite entender mejor a qué se refiere este código.

**Lenguaje ensamblador**

El lenguaje ensamblador es el primer intento de sustitución del lenguaje de máquina por uno más cercano al utilizado por los humanos.

Un programa escrito en este lenguaje es almacenado como texto (tal como programas de alto nivel) y consiste en una serie de instrucciones que corresponden al flujo de órdenes ejecutables por un microprocesador.

Sin embargo, dichas máquinas no comprenden el lenguaje ensamblador, por lo que se debe convertir a lenguaje máquina mediante un programa llamado Ensamblador.

Este genera códigos compactos, rápidos y eficientes creados por el programador que tiene el control total de la máquina.

## Lenguaje de programación de alto nivel

Tienen como objetivo facilitar el trabajo del programador, ya que utilizan unas instrucciones más fáciles de entender.

Además, el lenguaje de alto nivel permite escribir códigos mediante idiomas que conocemos (español, inglés, etc.) y luego, para ser ejecutados, se traduce al lenguaje de máquina mediante traductores o compiladores.

### Traductor

Traducen programas escritos en un lenguaje de programación al lenguaje máquina de la computadora y a medida que va siendo traducida, se ejecuta.

### Compilador

Permite traducir todo un programa de una sola vez, haciendo una ejecución más rápida y puede almacenarse para usarse luego sin volver a hacer la traducción.

## ¿Qué softwares de programación existen?

Por software de programación entendemos el conjunto de todas las herramientas que le permiten al programador, crear, escribir códigos, depurar, mantener y empaquetar los proyectos.

Algunos de los distintos programas por los que pasará el proyecto para gestionarlo son:

**Editores de código o texto**

Al escribir los códigos se auto-completan marcando los errores sintácticos y la refactorización.

**Compiladores**

Como mencionados anteriormente, éstos traducen el código ingresado a lenguaje de máquina generando un código binario ejecutable.

**Depuradores**

Sirven para optimizar el tiempo de desarrollo mediante el monitoreo de la ejecución de un programa, el seguimiento a los valores de ciertas variables, las referencias a objetos en memoria y por ende, nos ayuda a corregir errores.

**Enlazadores**

Este programa toma objetos generados en los primeros pasos del proceso de compilación y los recursos necesarios de la biblioteca, quita aquellos procesos y datos que no necesita, y enlaza el código con dicha biblioteca para así aumentar su tamaño y extensión.

**Interpretadores o traductores**

Como leíste en éste artículo, el traductor (o intérprete) carga el código ingresado y traduce las instrucciones para que el programa pueda ser ejecutado.

**IDE**

El IDE (Integrated Development Environment) o Entorno de Desarrollo Integrado, es una aplicación informática que proporciona una serie de servicios que facilitan la programación de software, tales como:

- funciones de autocompletado;
- un editor de código fuente;
- gestión de conexiones a bases de datos;
- integración con sistemas de control de versiones;
- simuladores de dispositivos;
- un depurador para agilizar el proceso de desarrollo de software, entre otros.

## Lenguajes de desarrollo para Android

### Java


Se mantiene c como el Lenguaje más popular para programar Apps y de programación en General a nivel mundial. Cuenta con una comunidad enorme de desarrolladores por lo que siempre contarás con soporte y ayuda mientras desarrollas con Java.

Actualmente puedes usar Java con los programas Android Studio, NetBeans, Eclipse entre otros.

Java es un Lenguaje Multiplataforma que Soporta el desarrollo para Apps Móviles y Desktop, pero fue Google quien le dio bastante popularidad a Java para el desarrollo de Aplicaciones Móviles mediante la creación del sistema operativo Android, entre las características de este Lenguaje están:

Programación Orientada a Objetos

El desarrollo con Java puede ser menos complicado si usas Android Studio el cual cuenta con muchas herramientas para crear impresionantes aplicaciones para Android.

Es un Lenguaje muy Robusto.

Cuenta con una Arquitectura Neutral.

Tiene muchos paquetes y librerías en GitHub realizados con Java, listos para implementarse en tu proyecto.

## Kotlin


Este lenguaje de programación casi Joven aún fue desarrollado por la empresa JetBrains, esta empresa es muy conocida entre los profesionales del medio, por haber creado el popular IDE, IntelliJ IDEA y que a su vez este IDE es la base y tiene elementos añadidos en Android Studio.

Kotlin fue creado para resolver problemas que existen en Java, la sintaxis de Kotlin es mucho más limpia y puedes resolver un problema escribiendo menos código. Puedes usar Java y Kotlin al mismo tiempo, esto lo hace muy potente, entre las características de Kotlin son:

Ayuda a escribir menos código repetitivo, es muy conciso.

Puedes evitar errores en el interior de las clases de tu proyecto.

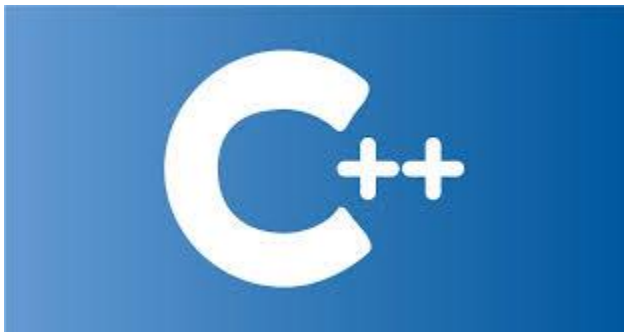
Puedes trabajar con JVM (Java Virtual Machine), Android y el Navegador.

Pude utilizarse en cualquier IDE de Java, de preferencia Android Studio tiene mejor compatibilidad con este.

Soporta Programación Orientada a Objetos.

Entre las aplicaciones más populares que han introducido Kotlin tenemos a Pinterest, Evernote, Uber, Coursera, Corda, Gradle, Esencial, Atlassian entre otras.

## C++




C++ no es fácil de aprender y definitivamente no es un punto de partida recomendado. Es un lenguaje orientado a objetos más antiguo con muchas re glas y limitaciones. Está estandarizado por la Organización Internacional de Normalización. Que ayuda al idioma y ralentiza su desarrollo.

Existe una amplia gama de soporte cuando se encuentra con problemas, pero las funcionalidades más recientes tardan en agregarse debido al proceso de revisión. Si eres un alma valiente, ¡súbete! C ++ ofrece el más alto nivel de control con acceso a librerías C ++ y desarrollo nativo. Esto significa que su código se comunicará directamente con el dispositivo.

Sin embargo, si esto no es importante para usted. Debe comenzar en otro lugar. C ++ es un lenguaje complejo de aprender, y puede encontrarse con serios problemas como principiante. Por un lado, no hay recolección de basura automática. Esto significa que los objetos no se eliminan de la memoria del dispositivo. A menos que codifique esto manualmente, la memoria continuará llenándose y eventualmente la aplicación se bloqueará.

C ++ también es muy implacable cuando comete errores en el código. No se darán mensajes de error o excepciones. Su código se bloqueará sin explicación.

# Lenguajes de desarrollo para iOS

## Swift

Swift como el resto de lenguajes de programación existentes cuenta con características que lo distinguen. El objetivo detrás del lenguaje de programación de Apple es crear el mejor lenguaje disponible para usos que van desde la programación de sistemas, aplicaciones para móviles y de escritorio, llegando a servicios en la nube.

Más importante aún, Swift está diseñado para hacer que la tarea de escribir y mantener programas sea más fácil para el desarrollador. Para lograr este objetivo, los desarrolladores de Apple y de la comunidad, han abogado porque el lenguaje Swift sea:

- Seguro: La manera más obvia para escribir código también debe ser de una manera segura. Un comportamiento indefinido es el enemigo de la seguridad, y los errores del desarrollador deben ser detectados antes de que el software esté en producción. Optar por la seguridad significa que Swift en algunas ocasiones se sentirá estricto, pero la claridad ahorra tiempo a largo plazo.
- Rápido: Swift está pensado como un reemplazo para los lenguajes basados en C (C, C ++ y Objective-C). Como tal, Swift debe ser comparado a estos lenguajes en el rendimiento que logra para la mayoría de tareas, un rendimiento similar y en algunas pruebas incluso hasta mejor. El rendimiento también debe ser predecible y consistente, hay un montón de lenguajes con características novedosas como Swift, pero donde la rapidez muchas veces no se logra, de hecho, rara vez podemos contar con lenguajes comparables en este aspecto a los basados en C.

- Expresivo: El lenguaje Swift se beneficia de décadas de avance en la ciencia de la computación y ofrece una sintaxis que es sin dudas todo un placer, con las características modernas que los desarrolladores de hoy en día esperan encontrar. Pero Swift se encuentra en constante desarrollo, bajo un estricto seguimiento de los avances que va teniendo el lenguaje, en una continua evolución para hacer Swift aún mejor.

Como es lógico las herramientas son una parte fundamental del ecosistema Swift. Por esto hay un esfuerzo más que vidente en la plena integración con las herramientas de desarrollo, permitiendo programar rápidamente, presentar excelentes diagnósticos y brindar una experiencia de desarrollo cada vez más interactiva.

Swift incluye características que hacen mucho más fluida y fácil la lectura y escritura de código, mientras que el desarrollador mantiene el control necesario sobre un lenguaje de programación de sistemas verdadero.

Swift admite tipos inferidos de datos para hacer el código más limpio y menos propenso a errores, y los módulos han eliminado los encabezados (headers) y proporcionan espacios de nombres (namespaces).

El manejo de memoria se realiza de forma automática, y no es necesario escribir un punto y coma al final de cada línea.

Las características de Swift están diseñadas para trabajar juntas y crear así un lenguaje potente, pero divertido de usar. Algunas características adicionales de Swift serían:

- Unificación de Closures con funciones punteros.
- Tuplas y valores de retorno múltiples.
- Genéricos.
- Iteracción rápida y concisa sobre un rango o colección.
- Estructuras con soporte para métodos, extensiones y protocolos.
- Patrones de programación funcional, por ejemplo: map y filter.
- Potente gestión de errores.

## Objetive-C


A la hora de programar aplicaciones para el sistema operativo de Apple, iOS, y por lo tanto para crear apps para iPhone y iPad, debes utilizar el lenguaje Objective-C.

Este lenguaje extiende al clásico lenguaje de programación C, añadiéndole capacidades de programación orientada a objetos y sobre todo intentando atajar los problemas de reusabilidad que tenía éste. Su desarrollo se inició en 1981 (¡hace más de 30 años!) por parte de dos programadores entusiastas de la empresa ITT, que luego fundaron su propia empresa para comercializarlo. Se popularizó a finales de la década de los '80 cuando lo licenció una pequeña empresa llamada NEXT, fundada por Steve Jobs tras haber sido expulsado de Apple. Cuando Apple compró NEXT unos años después (en 1996) y Jobs volvió triunfante a su casa, sus sistemas formaron la base de la nueva Apple, y con ellos el lenguaje Objective-C, que nos persigue hasta hoy.

Objective-C es un lenguaje bastante árido y con muchas diferencias frente a lenguajes de propósito más general como C# o Java. Por eso muchos programadores que se meten en la programación para Mac o para iPhone/iPad encuentran su principal barrera en comprender bien y utilizar Objective-C.

Una de las primeras cosas que llaman la atención es que Objective-C es un lenguaje compilado. pero también es al mismo tiempo un lenguaje enlazado. Esto quiere decir básicamente que el resultado del compilador no es el programa final sino que existe una segunda fase que lleva a cabo el enlazador (linker en inglés).

**Relación entre compilador y enlazador**

El enlazador es el responsable de recoger el código compilado y de "juntarlo" con otro código compilado: el de las librerías que tu programa utilice (sean de sistema o de terceros).

Este concepto te sorprenderá si vienes de C# o Java. En estos lenguajes el concepto de enlazado no existe: si quieres utilizar una librería (un assembly o bien un fichero .jar) te limitas a "referenciarlo" desde tu proyecto. Automáticamente, sin que tengas que hacer nada más, todas las clases definidas en esta librería las tienes disponibles. En tiempo de ejecución basta con que el ordenador de destino tenga las librerías instaladas para que tu programa funcione. En estos lenguajes esto es posible porque el código compilado (ya sean assemblies de .NET o archivos .class de Java) expone metadatos que informan de qué tipos existen, qué métodos tienen y en general todo lo que el compilador necesita para poder compilar el código.

En Objective-C esto no funciona así. Cuando usas una librería (ya sea del sistema o de terceros), el compilador no la necesita para nada. Tampoco puede hacer nada con ella: el código compilado no expone metadatos que permitan leerlo para saber qué tipos hay definidos y qué métodos tienen. Así que para poder utilizar la librería se debe suministrar al compilador uno, o varios, archivos de código fuente que definan la clase y todos sus métodos (tan solo la firma de los métodos, no la implementación). Dichos archivos reciben el nombre de archivos de cabecera o headers (del inglés), tienen (o suelen tener) la extensión .h y por supuesto, no los creas tú sino que los crea quien haya hecho la librería.

Así que cuando uses una librería en Objective-C, tendrás el código compilado y uno o varios archivos de cabecera que contienen la definición de las clases y métodos de esa librería. El compilador tan solo utiliza la información de los ficheros de cabecera y es el enlazador quien recoge el código compilado generado por el compilador, los binarios de las librerías y lo enlaza todo en el programa final.

## Entornos de Desarrollo:

## Definición

Un entorno de desarrollo es un conjunto de procedimientos y herramientas que se utilizan para desarrollar un código fuente o programa. Este término se utiliza a veces como sinónimo de entorno de desarrollo integrado (IDE), que es la herramienta de desarrollo de software utilizado para escribir, generar, probar y depurar un programa. También proporcionan a los desarrolladores una interfaz de usuario común (UI) para desarrollar y depurar en diferentes modos.

A la hora de elegir en entorno de desarrollo o IDE (Integrated Development Environment) es fundamental tener definido qué lenguaje de programación se va a utilizar tanto en el Frontend (la parte visible del programa/web) como en el Backend.

Las actividades mejor soportadas por herramientas de desarrollo son normalmente las centrales: codificación y pruebas de unidades. El conjunto de herramientas que soportan estas actividades constituye lo que se llama un entorno de programación. A veces se utilizan las siglas IDE (Integrated Development Environment) para designar estos entornos, aunque no son un entorno de desarrollo completo, sino sólo una parte de él.

- Siguiendo la terminología anterior, de niveles funcionales, es el banco de trabajo del programador.
- Da soporte a las actividades de la fase de codificación (preparación del código y prueba de unidades).
- Los mismos productos sirven también para el diseño detallado y para las pruebas de integración.
- Se sitúa, por tanto, en la parte central del ciclo de desarrollo.

## Función del IDE.

Como sabemos, los entornos de desarrollo están compuestos por una serie de herramientas software de programación, necesarias para la consecución de sus objetivos. Estas herramientas son:


- Un editor de código fuente.
- Un compilador y/o interprete
- Automatización de generación de herramientas
- Un depurador

**Las funciones de los IDE son:**

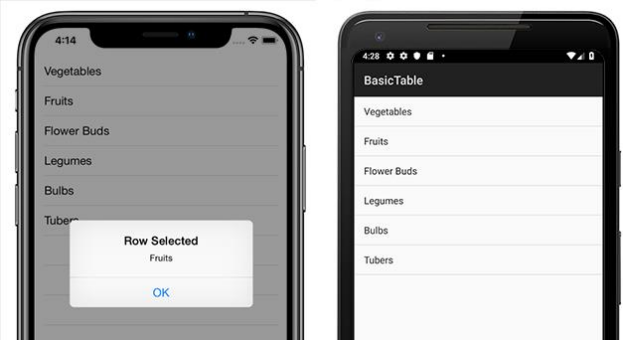
- Un editor de código fuente.
- Auto-completado de código, atributos y métodos de clases.
- Identificación automática de código.
- Herramientas de concepción visual para crear y manipular componentes visuales.
- Asistentes y utilidades de gestión y generación de código.
- Archivos fuente en unas carpetas y compilados a otras.
- Compilación de proyectos complejos en un solo paso.
- Control de versiones: tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de auto-recuperación a un estado anterior estable.
- Soporta cambios de varios usuarios de manera simultánea.
- Generador de documentación integrado.
- Detección de errores de sintaxis en tiempo real.

Otras funciones importantes son:

- Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo, cambiar el nombre a una variable).
- Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
- Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
- Aumento de funcionalidades a través de la gestión de sus módulos y plugins.
- Administración de las interfaces de usuario (menús y barras de herramientas).
- Administración de las configuraciones del usuario.

# Entornos de Desarrollo

## Xamarin




Xamarin es una plataforma de código abierto para crear aplicaciones modernas y de alto rendimiento para iOS, Android y Windows con .NET. Xamarin es una capa de abstracción que administra la comunicación del código compartido con el código de la plataforma subyacente. Xamarin se ejecuta en un entorno administrado que proporciona comodidades como la asignación de memoria y la recolección de elementos no utilizados.

Xamarin permite a los desarrolladores compartir un promedio del 90% de su aplicación entre plataformas. Este patrón permite a los desarrolladores escribir toda su lógica empresarial en un solo idioma (o reutilizar el código de la aplicación existente) pero lograr un rendimiento, una apariencia y un funcionamiento nativos en cada plataforma.

Las aplicaciones de Xamarin se pueden escribir en PC o Mac y compilarse en paquetes de aplicaciones nativas, como un archivo .apk en Android o un archivo. ipa en iOS.

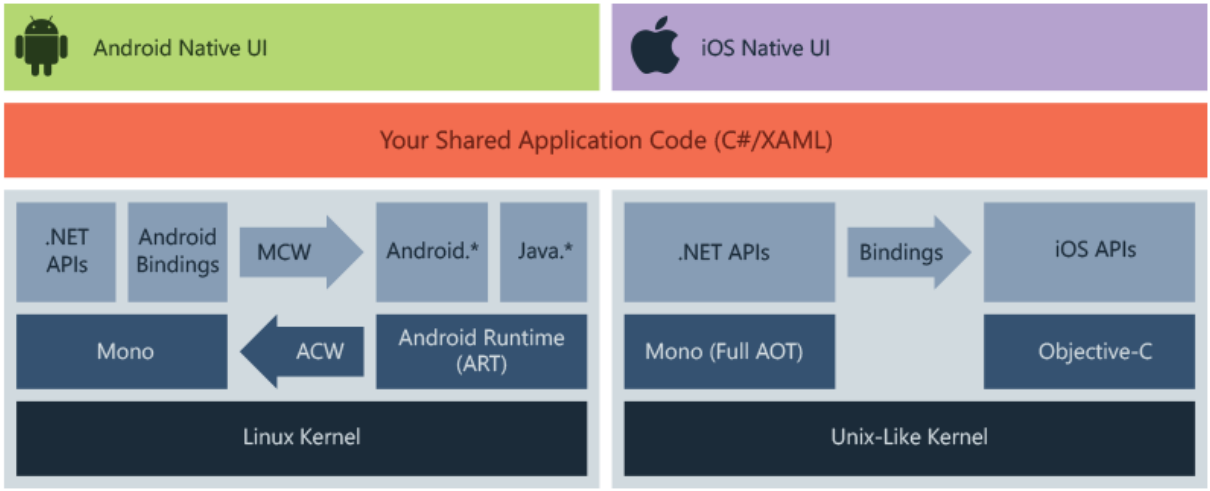
# Para quién es Xamarin

Xamarin es para desarrolladores con los siguientes objetivos:

Comparta código, prueba y lógica empresarial entre plataformas.

Escriba aplicaciones multiplataforma en C # con Visual Studio.

## Cómo funciona Xamarin




El diagrama muestra la arquitectura general de una aplicación de Xamarin multiplataforma. Xamarin le permite crear una interfaz de usuario nativa en cada plataforma y escribir lógica empresarial en C # que se comparte entre plataformas. En la mayoría de los casos, el 80% del código de la aplicación se puede compartir con Xamarin.

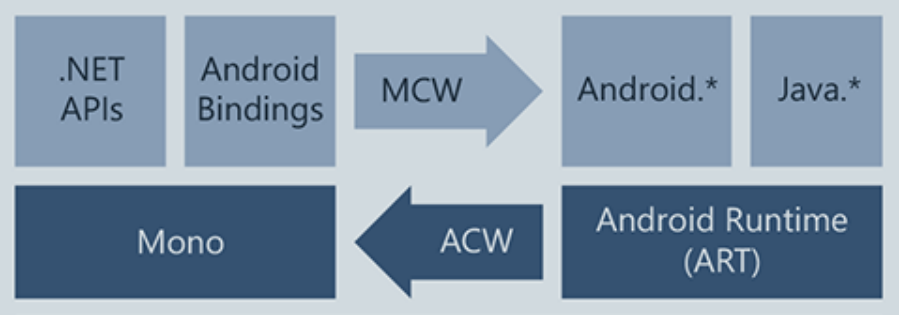
Xamarin se basa en .NET, que maneja automáticamente tareas como la asignación de memoria, la recolección de basura y la interoperabilidad con plataformas subyacentes.

Xamarin combina las capacidades de las plataformas nativas y agrega una serie de características, que incluyen:

- **Enlace completo para los SDK subyacentes:** Xamarin contiene enlaces para casi todos los SDK de la plataforma subyacente tanto en iOS como en Android. Además, estos enlaces están fuertemente tipados, lo que significa que son fáciles de navegar y usar, y proporcionan una sólida verificación de tipos en tiempo de compilación y durante el desarrollo. Los enlaces fuertemente tipados conducen a menos errores en tiempo de ejecución y aplicaciones de mayor calidad.

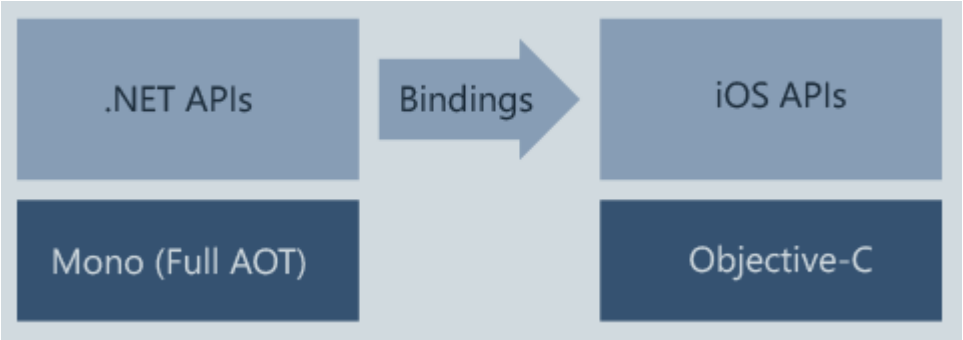
- **Interoperabilidad de Objective-C, Java, C y C ++:** Xamarin proporciona funciones para invocar directamente bibliotecas de Objective-C, Java, C y C ++, lo que le permite utilizar una amplia variedad de código de terceros. Esta funcionalidad le permite usar bibliotecas de iOS y Android existentes escritas en Objective-C, Java o C / C ++. Además, Xamarin ofrece proyectos vinculantes que le permiten vincular bibliotecas nativas de Objective-C y Java mediante una sintaxis declarativa.
- **Construcciones de lenguaje moderno:** las aplicaciones de Xamarin están escritas en C #, un lenguaje moderno que incluye mejoras significativas sobre Objective-C y Java, como características de lenguaje dinámico, construcciones funcionales como lambdas, LINQ, programación paralela, genéricos y más.
- **Biblioteca de clases base robusta (BCL):** las aplicaciones de Xamarin usan .NET BCL, una gran colección de clases que tienen características integrales y optimizadas, como compatibilidad con XML, base de datos, serialización, E / S, cadenas y redes, y más. El código C # existente se puede compilar para su uso en una aplicación, lo que proporciona acceso a miles de bibliotecas que agregan funcionalidad más allá de BCL.
- **Entorno de desarrollo integrado moderno (IDE):** Xamarin usa Visual Studio, un IDE moderno que incluye características como la finalización automática de código, un sofisticado sistema de gestión de soluciones y proyectos, una biblioteca de plantillas de proyectos integral, control de fuente integrado y más.
- **Soporte multiplataforma móvil:** Xamarin ofrece soporte multiplataforma sofisticado para las tres plataformas principales de iOS, Android y Windows. Las aplicaciones se pueden escribir para compartir hasta el 90% de su código, y Xamarin.Essentials ofrece una API unificada para acceder a recursos comunes en las tres plataformas. El código compartido puede reducir significativamente los costos de desarrollo y el tiempo de comercialización para los desarrolladores de dispositivos móviles.

## Xamarin.Android




Las aplicaciones de Xamarin.Android se compilan desde C # en lenguaje intermedio (IL) que luego se compila Just-in-Time (JIT) en un ensamblado nativo cuando se inicia la aplicación. Las aplicaciones de Xamarin.Android se ejecutan dentro del entorno de ejecución Mono, junto con la máquina virtual de Android Runtime (ART). Xamarin proporciona enlaces .NET a los espacios de nombres de Android. * Y Java. *. El entorno de ejecución Mono llama a estos espacios de nombres a través de Managed Callable Wrappers (MCW) y proporciona Android Callable Wrappers (ACW) al ART, lo que permite que ambos entornos invoquen código entre sí.

## Xamarin.iOS




Las aplicaciones Xamarin.iOS están completamente compiladas con anticipación (AOT) desde C # en código de ensamblaje ARM nativo. Xamarin usa selectores para exponer Objective-C a C # administrado y registradores para exponer código C #

administrado a Objective-C. Los selectores y registradores se denominan colectivamente "enlaces" y permiten que Objective-C y C # se comuniquen.

## Flutter


Flutter es un framework de código abierto desarrollado por Google para crear aplicaciones nativas de forma fácil, rápida y sencilla. Su principal ventaja radica en que genera código 100% nativo para cada plataforma, con lo que el rendimiento y la UX es totalmente idéntico a las aplicaciones nativas tradicionales.

En el mercado de desarrollo de aplicaciones móviles para iOS y Android hay una gran cantidad de frameworks o herramientas que permiten utilizar un mismo código fuente para ambas plataformas. Pero ninguna genera código 100% nativo.

La principal y más importante ventaja de Flutter es que desarrollas un solo proyecto para todos los sistemas operativos, lo que significa una reducción de costes y tiempo de producción.

### Funcionalidades de Flutter

Calidad nativa: Las aplicaciones nativas se desarrollan específicamente para un sistema operativo, Flutter utiliza todas las ventajas de las aplicaciones nativas para conseguir calidad en el resultado final.

Experiencia de usuario: Flutter incluye Material Design de Google y Cupertino de Apple, con lo que la experiencia de usuario es óptima y los interfaces de usuario idénticos a los de las aplicaciones desarrolladas por las propias compañías.

Tiempo de carga: Una de las principales causas de abandono de una aplicación es el tiempo que tarda en cargar, con Flutter se experimentan tiempos de carga por debajo de un segundo en cualquiera de los soportes iOS o Android.

Desarrollo ágil y rápido: Gracias a la característica hot-reload, puedes programar y ver los cambios en tiempo real en tu dispositivo o en los simuladores.

Otra de las ventajas de utilizar este framework es que da igual el sistema operativo que utilices, ya que puedes descargarlo para Windows, Mac o Linux desde este enlace.

Seguimos contando beneficios al utilizar Flutter, el framework tiene una comunidad en la que podrás compartir tus conocimientos, experiencias y/o buscar ayuda para tus dudas o problemas puntuales.

## NativeScript


NativeScript es un framework para construir aplicaciones móviles nativas multiplataforma. Esto les permite a los desarrolladores usar XML, CSS y JavaScript para construir aplicaciones para Android, iOS, e incluso Windows Universal Platform. A diferencia de Cordova, que usa un WebView para renderizar la UI de la aplicación, NativeScript usa el motor de renderizado de la plataforma, lo que significa que proporciona una experiencia de usuario verdaderamente nativa.

NativeScript difiere de otros entornos de desarrollo de muchas maneras. La más importante es que se trata de un entorno de trabajo que verdaderamente permite escribir una sola vez y entregar a todas las plataformas. Con Nativescript podemos:

Aprovechar el conocimiento existente (no tienes que saber Objective C, Swift o Java). NativeScript ha sido diseñado para ser aprovechado por desarrolladores con diferentes formaciones.

Implementación de hojas de estilo CSS. Podemos cambiar la apariencia y estilo de vistas y elementos en una aplicación NativeScript de manera similar a como lo hacemos en una aplicación web, usando hojas de estilo o cambiando el estilo del objeto de los elementos en JavaScript. Sin embargo, sólo un subconjunto del lenguaje CSS es soportado.

Acceso a las APIs nativas de la plataforma. Si el framework de Nativescript no expone una API nativa que necesitemos, podemos implementar plugins con NativeScript.

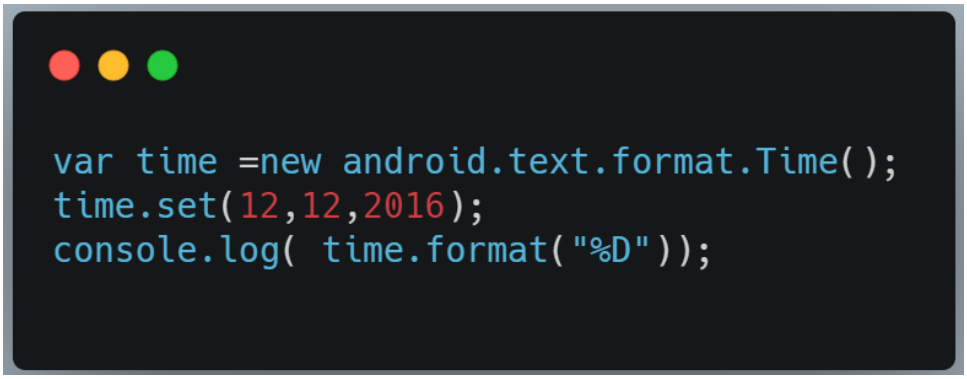
Tu código se escribe una vez. En experiencias pasadas, trabajando con otros frameworks cross-platform, he visto que se requiere usar muchos shims (código de cuña) para que la aplicación funcione bien en las distintas plataformas; por ejemplo,

puede que necesitemos agregar un poco de código para desplegar un botón sólo en la versión de Android o, tal vez, necesitamos escribir código adicional para hacer que un menú de lista se vea bien en iOS. Pero en NativeScript esto rara vez es necesario.

**¿Cómo funciona NativeScript?**

Tal vez la característica más importante de NativeScript es su mecanismo para brindar acceso directo a las APIs nativas de cada plataforma. Pero, ¿cómo funciona?

Examinemos el siguiente código de una aplicación NativeScript para Android.



```
var time =new android.text.format.Time();
time.set(12,12,2016);
console.log( time.format("%D"));
```

Te preguntarás: ¿es código JavaScript creando una instancia de un objeto Java? Sí, JavaScript crea una instancia del objeto, android.text.format.Time(), invoca su método set() y luego manda al log de la consola el valor de retorno de su método format, el cual es la cadena “12/12/16”.

Veamos un ejemplo de una aplicación NativeScript para iOS.

```
var alert =new UIAlertView();
alert.message ="Hello world!";
alert.addButtonWithTitle("OK");
alert.show();
```

Este código JavaScript crea una instancia de una clase de Objective-C llamada UIAlertView, establece su propiedad message y luego llama a sus métodos addButtonWithTitle() y show().

Vale la pena aclarar que estos casos en los que accedemos a las APIs nativas invocando objetos específicos de Android o iOS, son excepciones. NativeScript incluye muchos módulos para tareas comunes, como hacer una petición HTTP, construir componentes UI, etc.  Sin embargo, a veces las aplicaciones necesitan acceso a las APIs, y el runtime de NativeScript hace este acceso muy sencillo cuando se requiere.

## Ionic


Ionic es una estructura tecnológica (Framework) de código abierto que se utiliza en el desarrollo de aplicaciones móviles híbridas, es decir, se combinan el HTML5, CSS y JavaScript dando como resultado aplicaciones con una interfaz amigable e intuitiva para el usuario que luego se comercializan o descargan en plataformas como Android o iOS.

La base de Ionic está desarrollada sobre AngularJs y Cordova, vio la luz en 2013 con la única intención de que desarrolladores pudieran crear aplicaciones móviles híbridas con la particularidad y beneficios de los dos framework sobre los que fue construida.

Una de las principales ventajas de trabajar con Ionic es que aprovecha todos los plugins (Hardware, software, imágenes, texto, códigos QR, etc) del marco de desarrollo móvil Cordova.

En 2016 se actualiza a la versión Ionic 2, en donde llega la modularización, una de las actualizaciones más de este framework, que permite separarlo por partes: core, angular, native, etc.

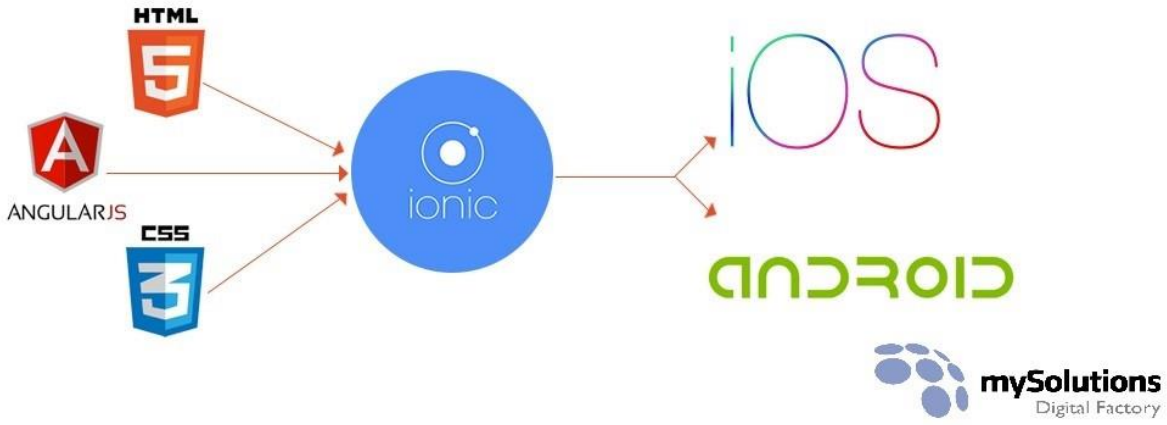
Ionic es la plataforma que facilita la creación de aplicaciones móviles con tecnologías web para los programadores web. La plantilla de Ionic permite a los desarrolladores crear aplicaciones móviles en diferentes plataformas que pueden instalarse en teléfonos con Android e iOS.

Desarrollado sobre AngularJS y lanzada su versión alfa el noviembre de 2013. Se lanzó una versión beta 1.0 en marzo de 2014, una versión final 1.0 en mayo de 2015 y varias versiones 2.0 en 2016. Ionic utiliza su base para proporcionarnos la estructura de aplicación mínima sobre la que poder comenzar a trabajar, mientras que Ionic en sí nos ofrecerá facilidades en el desarrollo de la interfaz de usuario. Con esta dupla,

AngularJS con su versatilidad y potencia para la creación de aplicaciones e Ionic Framework para el desarrollo de la interfaz, obtenemos una herramienta de creación de aplicaciones completísima, con la que ahorraremos tiempo y trabajo en el desarrollo de cada proyecto.

Ionic facilita la creación de aplicaciones híbridas utilizando HTML5 debido a su Framework de código abierto y también es completamente gratuito. El uso de Angular JS lo hace perfecto para el desarrollo de aplicaciones altamente interactivas y tiene una gran gama de herramientas y servicios que hacen que la ejecución de Ionic sea bastante simple.

Ionic viene con CLI (Command-Line Interface), que ayuda a los desarrolladores móviles a construir y probar aplicaciones iónicas en cualquier plataforma. Permite a los usuarios desplazarse por miles de listas sin que su rendimiento se vea afectado. Los usuarios pueden crear sus propias aplicaciones, personalizarlas para Android, iOS y desplegarlas.




**Las principales ventajas que ofrece son:**

- Desde una única fuente podremos llegar a las plataformas que soporta este Framework (Android e iOS).

- El desarrollo principal se realiza en HTML junto con CSS y JS, lenguajes muy extendidos por la comunidad de desarrolladores, con lo que la implantación de esta herramienta en la empresa, facilitará el desarrollo de proyectos de la forma más efectiva aun cuando la plantilla de desarrolladores contenga nuevas incorporaciones.
- Una herramienta tan "reciente" como Ionic sea capaz de dar soporte a un gigantesco Framework como AngularJS, nos hace plantearnos hasta dónde serán capaces de llegar, teniendo ya en vistas incluir otros Frameworks como EmberJS o KnockOut, por ejemplo. Si ya contamos con una aplicación web que queremos convertir en aplicación móvil, en la mayoría de los casos habremos hecho uso de JavaScript, por lo que el código es reutilizable.
- Para el caso de aplicaciones híbridas, tendremos con un único proceso de desarrollo e implementación, una aplicación para Android, iOS y Web.
- Ionic se centra en construir para los estándares web modernos y para dispositivos móviles modernos. Para Android, Ionic es compatible con Android 4.1 y versiones superiores. Para iOS, Ionic admite iOS 7 en adelante. Ionic 2 es compatible con la plataforma universal de Windows para crear aplicaciones de Windows 10. Ionic Framework, impulsado por AngularJS, es compatible con las aplicaciones BlackBerry 10.
- El uso de Ionic te permitirá crear, construir, y compilar aplicaciones en cualquier plataforma, todo con un solo comando. Por eso se considera un potente CLI.
- Si te desesperas con poco, te gustará Ionic. Está hecho para ser rápido.
- El Ionic Creator. Es el que te permitirá crear las Interfaces sin tener que meterte en código. Podrás crear la parte gráfica fácil sin tocar el código para nada (Es una herramienta de creación de interfaz de arrastrar y soltar).

**Las principales desventajas que contiene:**

- El rendimiento puede ser ligeramente menor que en aplicaciones desarrolladas de forma nativa, cosa que no debería ser un problema al menos que el proyecto sea para la creación de juegos con detallados gráficos u otras aplicaciones que hagan uso de grandes cantidades de recursos.
- Es una herramienta “joven” y puede ser difícil encontrar módulos compartidos por los usuarios, pero como dijimos, la comunidad está creciendo a pasos agigantados y en breve este inconveniente podría dejar de existir.
- El navegador como muestra de ejemplo para aplicaciones, no siempre da la información correcta sobre como se mostrará en el teléfono y puede haber fallos en las pruebas, debido a que el navegador solo mantiene las características más comunes de los teléfonos.
- Puede ser difícil de integrar con diferentes funcionalidades nativas.
- Las aplicaciones hibridas son más lentas que las aplicaciones nativas, pero conforme mejoran los dispositivos, esta diferencia se va reduciendo.

## React Native


React Native es un framework JavaScript para crear aplicaciones reales nativas para iOS y Android, basado en la librearía de JavaScript React para la creación de componentes visuales, cambiando el propósito de los mismos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Android. Es decir, en lugar de desarrollar una aplicación web híbrida o en HTML5, lo que obtienes al final como resultado es una aplicación real nativa, indistinguible de la que podrías desarrollar con tu código en Objective-C o Java.

React Native usa el mismo paradigma fundamental de construcción de bloques de UI (componentes visuales con los que interacciona el usuario) que las aplicaciones nativas reales de Android e iOS, pero gestiona la interacción entre los mismos utilizando las capacidades de JavaScript y React.

### Características

Con esta idea de construcción de aplicaciones React Native nos proporciona las siguientes funcionalidades:

- Compatibilidad Cross-Platform: ya que la mayoría de las APIs de React Native lo son de por sí, lo cual ayuda a los propios desarrolladores a crear aplicaciones que puede ser ejecutados tanto en iOS como Android simultáneamente con el mismo código base.
- Funcionalidad nativa: las aplicaciones creadas mediante React Native funcionan de la misma manera que una aplicación nativa real creada para cada uno de los sistemas usando su lenguaje nativo propio. La unión de React Native junto con JavaScript permite la ejecución de aplicaciones más complejas de

manera suave, mejorando incluso el rendimiento de las apps nativas y sin el uso de un WebView.

- Actualizaciones instantáneas (para desarrollo y/o test): con la extensión de JavaScript, los desarrolladores tienen la flexibilidad de subir los cambios contenidos en la actualización directamente al dispositivo del usuario sin tener que pasar por las tiendas de aplicaciones propias de cada sistema y sus tediosos ciclos de procesos obligatorios previos. Hay que aclarar que este uno es exclusivo de versiones de desarrollo o para test, es ilegal, y puede llegar a conllevar castigos que llegan hasta la retirada definitiva de la aplicación si se realizan cambios directos sobre código con aplicaciones ya publicados y en producción. La tienda de Apple lleva un control muy exhaustivo sobre este tipo de prácticas.
- Sencilla curva de aprendizaje: React Native es extremadamente fácil de leer y sencillo de aprender ya que se basa en los conceptos fundamentales del lenguaje JavaScript, siendo especialmente intuitivo tanto para los ya expertos en dicho lenguaje o incluso para las personas sin experiencia en él, ya que nos provee de un rango muy amplio de componentes, incluyendo ejemplo como los maps y filters típicos que se han usado siempre.
- Experiencia positiva para el desarrollador: si bien la curva de aprendizaje hemos dicho que es sencilla, también el propio lenguaje nos motiva y ayuda a la hora de la evolución según aumentamos nuestro conocimiento y dominio del mismo. Nos ofrece varias características importantes como, por ejemplo, el Hot reloading que nos refresca la app en el momento en que guardamos cambios, y nos ofrece una gran ventaja para el desarrollo y testing de nuevas versiones, como hemos comentado arriba. O el uso del flexbox layout engine gracias al cual nos permite abstraernos de muchos de los tediosos detalles de la generación de cada uno de los layouts correspondientes a iOS y Android. Así como el uso del debugger de las herramientas de desarrollados del navegador Google Chrome, facilitando de sobre manera la tarea de depuración de código.

Estas son cinco de las características principales que nos brinda React Native por defecto si lo elegimos como framework para nuestro desarrollo.

**El API de React Native**

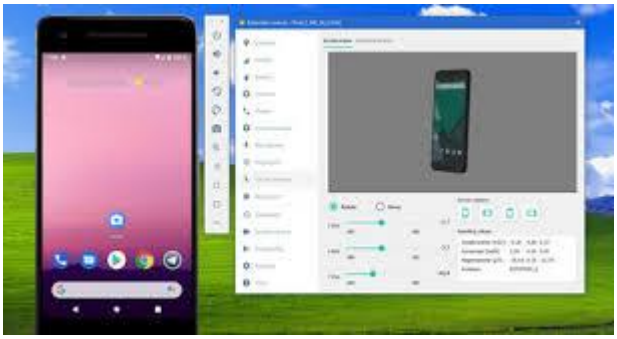
El API de React Native nos ofrece acceso a un gran número de funcionalidades nativas, y en Facebook están trabajando muy duro para ampliar cada vez más este abanico de posibilidades, pero puede darse el caso de que necesites añadir código nativo a tu desarrollo que todavía no esté accesible (o incluir alguna librería de terceros, por ejemplo). Puedes comunicar tu código JavaScript con código nativo mediante 'envoltorios', teniendo acceso total al hardware y a los APIs nativos.

Como hemos comentado, la principal ventaja que tiene React Native frente a sus competidores Web App es la experiencia de usuario nativa. Esta experiencia nativa se consigue principalmente utilizando componentes visuales nativos y mediante las animaciones.

«React Native tiene un futuro muy prometedor, una excelente comunidad y sobre todo el respaldo de una gran compañía como Facebook.»

La maquetación de nuestras vistas se añade mediante código JSX. Puede parecer que estamos mezclando HTML y JavaScript (una muy mala práctica), pero realmente el código JSX se transforma en código Javascript. En tiempo de ejecución este código es a su vez transformado a vistas nativas que son indistinguibles de las que se crean en un desarrollo nativo (a diferencia de las Web App que son una web dentro de un componente WebView). Para añadir estilos a nuestros componentes se utiliza una versión simplificada de css y Flexbox.

## Que es un emulador.



Este tipo de programa se diferencia de los simuladores, ya que lo que trata de modelar de manera precisa es el dispositivo original. Un emulador es un software originalmente pensado para ejecutar programas de diversas índoles, en una plataforma o sistema operativo diferente al programa que deseamos abrir o ejecutar.

Este tipo de programa se diferencia del simulador, ya que éste trata de modelar de manera precisa el dispositivo original para que el programa a ejecutar funcione correctamente en una plataforma distinta. En cambio, el simulador sólo reproduce el comportamiento original de un sistema determinado.

Podemos decir entonces, que un emulador es un programa diseñado para crear una plataforma virtual que pueda ejecutar un programa determinado, que no haya sido diseñado para ser ejecutado en el PC.

## Funcionamiento de un emulador

El principal objetivo del emulador, es crear las condiciones necesarias para poder ejecutar un programa diseñado para otras plataformas distintas al PC, de modo que estos programas son utilizados ampliamente para las siguientes actividades:

- Probar programas de plataformas informáticas diferentes.
- Ejecutar juegos de plataforma en un ordenador.
- Ejecutar programas de ordenadores antiguos.

Como podemos ver, los emuladores son ampliamente utilizados para poder jugar juegos antiguos o de plataformas diversas desde la comodidad de nuestro ordenador, utilizando para ello un programa o archivo llamado ROM.

Así también, los emuladores son muy utilizados para probar software de otros sistemas operativos, como los sistemas Linux y programas Windows o programas Mac en Windows.

## Emuladores para Android.

## Android Studio


Android Studio es el nombre de una plataforma de desarrollo de Android cuyo objetivo fue llegar a ser la herramienta indispensable en la creación de las aplicaciones del sistema operativo.

Esta herramienta cumplió un rol clave en el desarrollo de las aplicaciones para Android.

Android Studio es de vital importancia para el sistema operativo de Google porque le permite al desarrollador crear las aplicaciones que luego serán subidas en la Play Store.

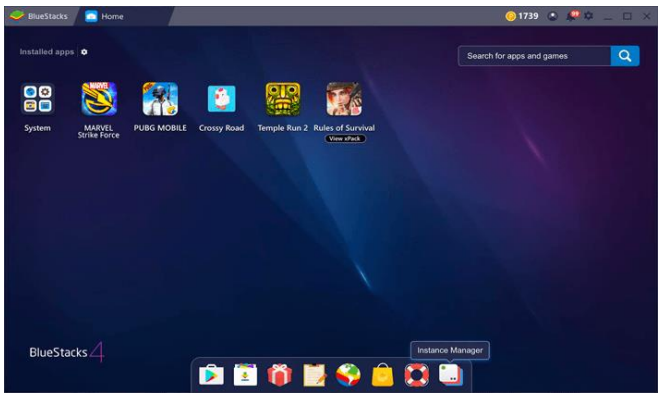
Con Android Studio se puede crear una aplicación, siendo esta el objetivo primordial de la herramienta en cuestión. El usuario podrá crear su propia aplicación de manera rápida y sencilla.

También se puede usar como emulador de Android. Esto permite probar las funciones del sistema operativo o probar las aplicaciones que haya creado el usuario en el teléfono, para ver si hay algo que modificar o si todo funciona según esperado.

Otra se las posibilidades que ofrece esta herramienta de Android es la de analizar o usar el trabajo de otros desarrolladores para aprender sobre cómo fueron diseñadas o para usar parte de su código en un proyecto propio. Y por último Android Studio posibilita utilizar las aplicaciones creadas en el dispositivo a través de la depuración USB. Con esto se logra verificar que la aplicación creada esté funcionando de manera correcta o que si presenta una falla el desarrollador pueda modificarla.

## BlueStacks

BlueStacks es uno de los emuladores de Android para PC más populares, sino el que más. Es algo así como tener un móvil o Tablet virtual en el PC, de modo que puedes jugar a juegos para móviles desde el PC, usando el teclado y el ratón.




BlueStacks se especializa en juegos, incluyendo varias ventajas añadidas como la posibilidad de abrir varios juegos a la vez, así como una completa configuración de los controles. Aquí veremos cómo puedes empezar a usar y sacarle todo el partido a este emulador de Android para Windows y Mac.

Por tanto, la función principal de BlueStacks es permitirnos ejecutar juegos y apps de Android en nuestros ordenadores. De esta forma, no es necesario que tengamos a mano todas las plataformas y apps que utilizamos a diario, nos basta con tener nuestra aplicación BlueStacks en el ordenador y ejecutar el juego o herramienta que necesitamos utilizar.

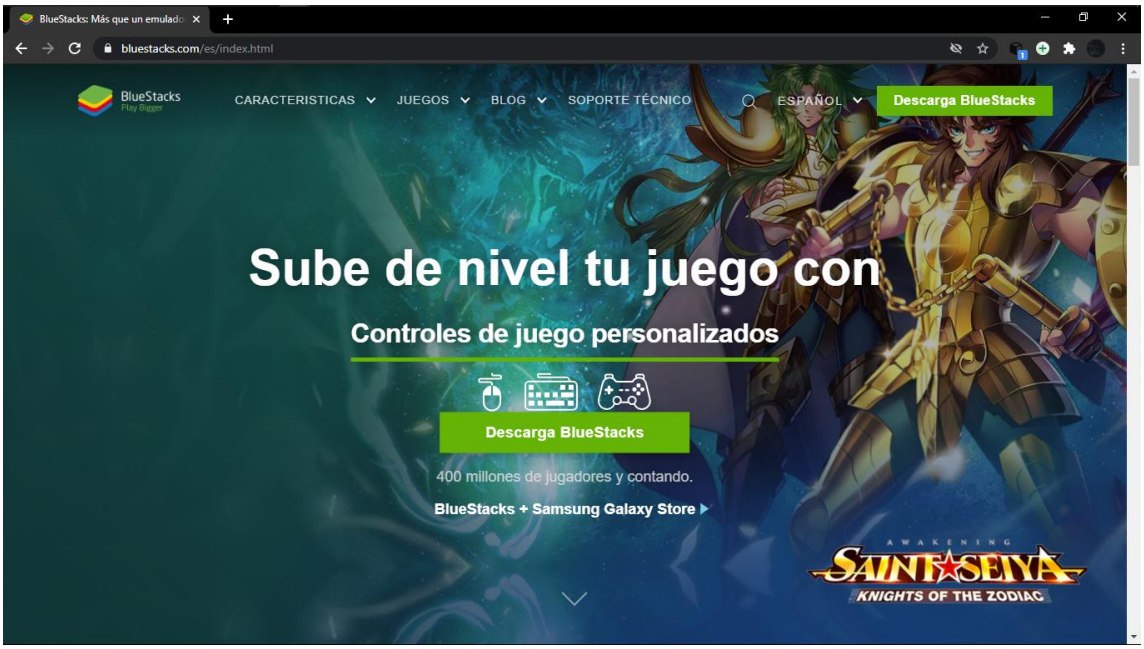
## Cómo instalar y configurar BlueStacks

La instalación del emulador de Android en PC o Mac es sencilla y no requiere demasiadas complicaciones. Lo primero que tienes que hacer es descargar el programa en la página oficial de BlueStacks. Te recomendamos que lo hagas en la web del producto porque es mucho más seguro que descargarlo de páginas desconocidas y así evitarás posibles virus o problemas con el ordenador.

Link de descarga:

https://www.bluestacks.com/es/index.html

Una vez que estés en la página web de BlueStacks y pinches en "Descargar BlueStacks", se abrirá una ventana en donde tendrás que seleccionar el lugar en el que quieres que se descargue el programa. Puedes dejarlo en el escritorio, la carpeta de descargas o el lugar en el que mejor lo vayas a encontrar después porque, por ahora, sólo se trata de un archivo de instalación. Cuando se descargue el archivo, tendrás que proceder a instalar el programa con lo que tendrás que hacer doble clic sobre el archivo de descarga de BlueStacks para que comience el proceso de extracción y se inicie la instalación.



## MEmuPlay

MEmu se presenta como un emulador de Android p ara PC muy completo, gracias a una serie de funciones que permiten obtener la mejor experiencia de uso posible. Nos va a dar la mejor experiencia de Android posible, gracias a que posee un diseño cómodo de usar, pero que no escatima en cuanto a funciones. Las características o funciones más destacadas que podemos encontrar en este emulador son:

- Simulador de GPS, para que mostremos en el mapa la ubicación que queramos.
- Personalización flexible (CPU, memoria, resolución, etc.).
- Compartir archivos entre Windows y Android.
- Crear copias con un clic.
- Personalizar/Mapear botones o joystick.

Gracias a este emulador de Android para PC podremos disfrutar de nuestros juegos favoritos para móvil desde el ordenador. Además, nos permite crear varias ventanas, de manera que podamos tener varios juegos o apps abiertos al mismo tiempo en el mismo. Ya que podremos tener abierta una app como WhatsApp en una ventana y un juego en otra, por ejemplo. También podremos ver contenidos o emisiones en directo desde el mismo.

## Requisitos de uso

Para poder disfrutar de MEmu en tu ordenador, hay que cumplir con una serie de requisitos, para asegurarte de que este emulador es compatible con el mismo. Los requisitos oficiales para poder proceder con su instalación son los siguientes:

- Sistema Operativo: Windows 7/Windows 8.1/Windows 10.

- Al menos 1 GB de espacio disponible en disco. 2 GB en caso de que se use la instalación offline.
- Tener una CPU Intel o AMD con Virtualization Technology activada.
- GPU con soporte para OpenGL 2.0 o versiones superiores.

Cuando abrimos MEmu en nuestro ordenador, podemos ver que el diseño que posee este emulador de Android para PC es sencillo. Nos podemos mover con facilidad entre sus funciones o aplicaciones. Si deseamos instalar algún juego o aplicación, solo tendremos que abrir la Play Store en el mismo. Tendremos que iniciar sesión en nuestra cuenta de Google para entrar en la tienda y poder descargar juegos. El funcionamiento es como en Android, solo tendremos que buscar el juego que queremos en la tienda y pulsar en el botón de instalar.

## Cómo se descarga

MEmu es un emulador que se mantiene muy actualizado, de hecho, su versión más reciente se lanzaba este mismo mes de enero. Lo podemos descargar en nuestro ordenador, ya sea una sobremesa, un portátil o un 2-en-1 desde su página web, en este enlace.

https://www.memuplay.com/

En su web tenemos varias opciones de descarga, como la descarga directa o la descarga offline, para que elijamos lo que deseamos.

La instalación de este emulador no presenta problemas, es como instalar cualquier otro programa o aplicación en el ordenador. Además, en su web tenéis todo tipo de información, disponible en multitud de idiomas, para saber más sobre el funcionamiento de este emulador de Android para PC.

## KoPlayer


KoPlayer es un emulador gratuito de Android para Windows que nos va a permitir utilizar el sistema operativo móvil de Google aprovechando el hardware de nuestro ordenador. Este emulador ocupa tan solo 300 MB y está formado, por un lado, por la herramienta de virtualización VirtualBox y, por otro lado, por una imagen de Android.

Las principales características de este emulador son:

- Permite iniciar y utilizar varias instancias a la vez.
- Cuenta con OpenGL y aceleración de hardware, por lo que es ideal para juegos.
- Permite grabar vídeo para compartirlo en las redes sociales.
- Cuenta con soporte para teclado, ratón y mandos de juego.
- Viene con la Google Play Store instalada de base, por lo que podemos descargar cualquier aplicación disponible en ella.
- Está basado en Android 4.4.

## Cómo funciona KoPlayer

Para poder utilizar este emulador, lo primero que debemos hacer es descargar de forma totalmente gratuita la versión más reciente del mismo desde su página web.

Link de descarga:

https://syzs.qq.com/sempage/jpppc2semsyzs/index.html?ADTAG=kaopuzhushou

La instalación del emulador es muy sencilla y se instala en pocos segundos, por defecto, en el disco duro con más espacio disponible (ya que con el uso la imagen ISO puede crecer bastante).

Una vez finalizada la instalación podemos ejecutar el emulador. El primer arranque puede tardar unos minutos hasta que finaliza la configuración inicial del programa.

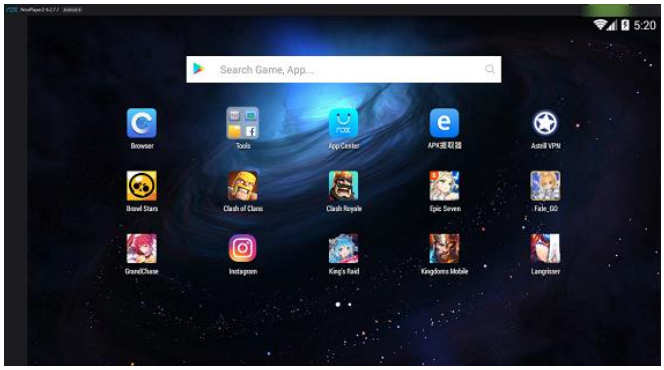
Una vez finalice ya podremos ver la interfaz principal de nuestro Android.


Lo primero que haremos será ir al menú ***Settings > Language*** y poner Android en nuestro idioma, para que sea más cómodo de utilizar.

Una vez en español, desde el propio menú de Settings, ahora llamado Ajustes, añadiremos una cuenta de Google para poder sincronizar nuestros datos y acceder sin problemas a la Play Store para descargar juegos y aplicaciones.

A partir de ahora podemos utilizar nuestro emulador de Android para Windows sin problemas, como si se tratara de un smartphone real. Si tenemos problemas o queremos cambiar algunos de los parámetros de KoPlayer, el menú de configuración del emulador nos permite cambiar la resolución del emulador y la memoria RAM de nuestro ordenador dedicada a Android.

## NoxPlayer




Los emuladores siempre requieren una cantidad de re cursos disponibles para mantenerse funcionando sin afectar al equipo huésped y en este punto Nox Player se destaca puesto que, no consume tantos como Bluestack, por ejemplo. En ese sentido, promete una excelente experiencia de uso, con la posibilidad de ejecutar juegos como Clash of Clans y PUBG.

Las características que hacen de NoxPlayer una excelente alternativa para disfrutar de Android en Windows empiezan por la posibilidad de mapear el teclado, de manera que, podemos desplazarnos en los juegos a través de las teclas, además de permitir la configuración de ciertas combinaciones personalizadas.

Asimismo, y si el juego en ejecución cuenta con soporte, es posible incorporar controles externos para jugar.

Nox Player también cuenta con una excelente integración con los dispositivos periféricos del sistema, permitiendo el uso de la cámara, teclado o micrófono en caso de ser requeridos por algún juego o aplicación.

El emulador presenta soporte para las versiones de Android 4.4.2, 5.1.1 y 7.1.2, asimismo, cabe destacar que también es compatible con la visualización de múltiples ventanas, permitiendo tener en primer plano varias aplicaciones a la vez.

Sin duda se trata de un emulador muy potente, con el valor adicional de no presentar un consumo excesivo de recursos.

Link de descarga:

https://es.bignox.com/

## PrimeOS


PrimeOS es una adaptación de Android 7 Nougat con una interfaz de escritorio que permite su uso sencillo con teclado y ratón. Ofrece un conjunto de herramientas aptas para sacarle el máximo partido sin necesidad de instalar nada. Viene bastante limpio de serie, dispone de la Google Play Store y, por supuesto, da acceso a la descarga de aplicaciones y de juegos Android.

Android x86 ya proporciona una versión del sistema operativo que puede escribir en un USB e instalar en cualquier PC x86 / x64 antiguo, siempre que sepas cómo hacerlo. Sin embargo, esta versión de PrimeOS se mantiene lo más cerca posible de AOSP, por lo que no hace muchos cambios en la experiencia del usuario para que sea más amigable con el escritorio.

Este emulador lleva las cosas un poco más lejos y se mantiene cerca de la experiencia pura de Android. Para ser específico, solo agrega características que considera necesarias para proporcionar una experiencia de escritorio verdadera. Cosas como un menú de inicio y una barra de tareas y ventanas múltiples con los botones para minimizar, maximizar y cerrar. Incluso es compatible con los métodos abreviados de teclado más conocidos, como Alt + Tab, Alt + F4, Win + D, etc. Y para las aplicaciones que no desees iniciar en el modo de ventana, también hay configuraciones para ellas.

PrimeOS también incluye algunas funciones para juegos de Android en modo de escritorio, lo que incluye el mapeo de teclas, incluso hay asignaciones predefinidas para juegos populares de Android como PUBG. Por supuesto, el rendimiento puede variar dependiendo del hardware de tu PC.

## Características destacadas

Experiencia de escritorio

- Múltiples características para dar experiencia de escritorio como menú de inicio, barra de tareas, etc.
- Soporte multi-ventana con maximizar, minimizar, cerrar, cambiar el tamaño, etc.
- Opción para desactivar la ventana múltiple para cualquier aplicación en caso de que lo necesite.
- Atajos de teclado generales como alt + tab, alt + f4, win + d etc.
- Cerca de la experiencia AOSP con solo las características de escritorio necesarias.
- Barra de tareas con capacidad para anclar aplicaciones, mostrar notificaciones e íconos del sistema.

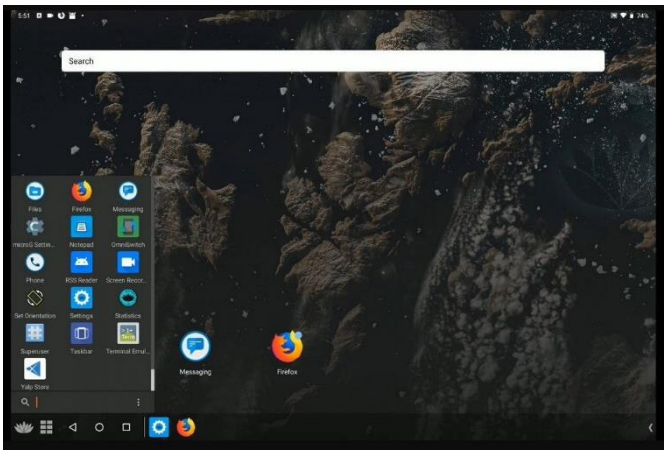
Juegos Android

- Herramienta de mapeo de teclas Decapro para jugar con el teclado y mouse (Presione F12).
- Pre mapea algunos juegos populares como PUBG, Subway surf etc.
- Las herramientas de GPU están disponibles para falsificar información de gpu para cualquier juego.

Link de descarga:

https://primeos.in/download/

## BlissOS




Bliss OS es un sistema de código abierto realizado por desarrolladores independientes, que destaca por ofrecer una solución de escritorio basado en Android.

Bliss OS es otro que tiene el mismo objetivo. O más ambicioso, ya que también tiene versión para móviles y pretende ser «un sistema operativo de calidad que pueda ejecutarse en cualquier dispositivo en el trabajo diario, sincronizando sus aplicaciones + configuraciones + personalizaciones en todas las plataformas en las que ejecute Bliss».

La versión para x86 que nos interesa está basada en Android 9 Pie y en los últimos días se ha renovado con una nueva versión que incluye las API Vulkan Graphics, lo que además de soportar las librerías en todas las GPUs integradas de Intel y también en algunas dedicadas de AMD y NVIDIA, añade compatibilidad parcial con los modos de suspensión.

## Cómo se instala Bliss OS

La instalación se realiza se manera similar a otros sistemas operativos. Descargas la imagen ISO (última versión «Bliss OS x86 v11.10«) y la quemas en un medio óptico o mejor en un pendrive o disco USB con herramientas como Rufus. Puedes instalarla en un PC o usarla en modo «Live» sin instalación.

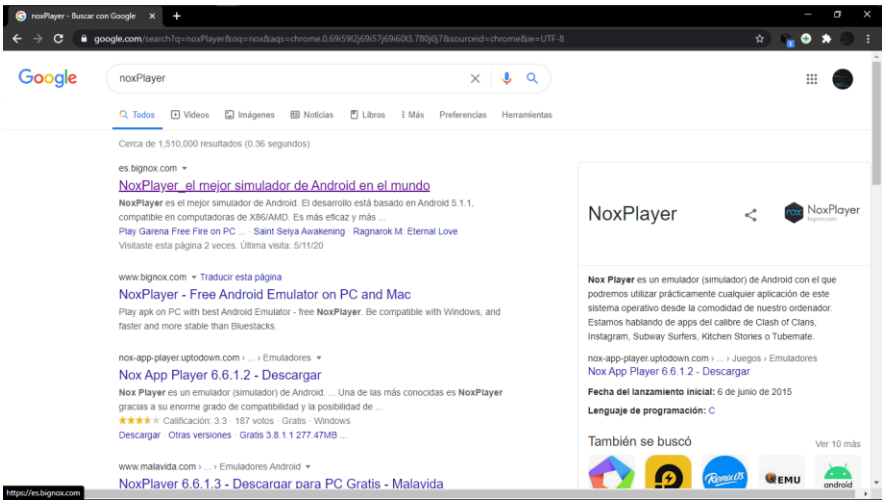
Y si no quieres tocar el sistema instalado en tu PC, puedes usar máquinas virtuales de manera muy sencilla y con bajo consumo de recursos de tu máquina por los bajos requisitos de Android.

Link de descarga: https://blissos.org/

# Metodología

## Descarga de NoxPlayer

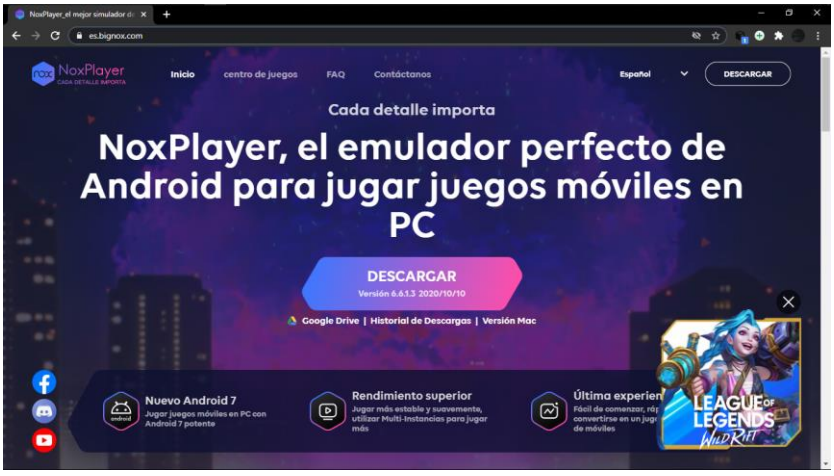
En este caso vamos a descargar el emulador Nox, para ello vamos a irnos al navegador y teclear el nombre del programa (Nox), y vamos a elegir la primera opción.




o bien hacemos clic en el siguiente link:

https://es.bignox.com/

Este link nos enviara a la página oficial de Nox, una vez ahí damos clic en el botón Download (Descargar).



## Instalar NoxPlayer

Una vez descargado nos vamos a dirigir a la ruta donde tengamos nuestro ejecutador, y vamos a seleccionarlo, posteriormente vamos a darle clic derecho para desplegar el menú de opciones, de las opciones que aparecen vamos a elegir la opción que dice ***Ejecutar como administrador,*** esto para poder acceder al menú de instalación e instalarlo en nuestro sistema operativo.

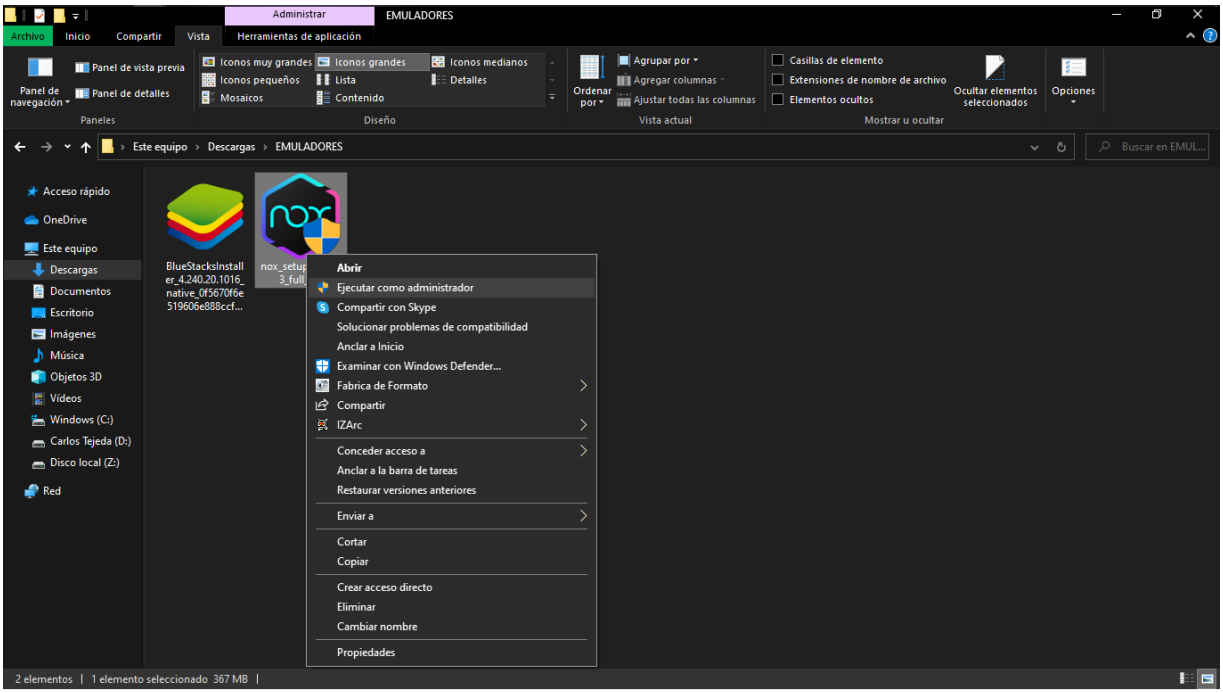
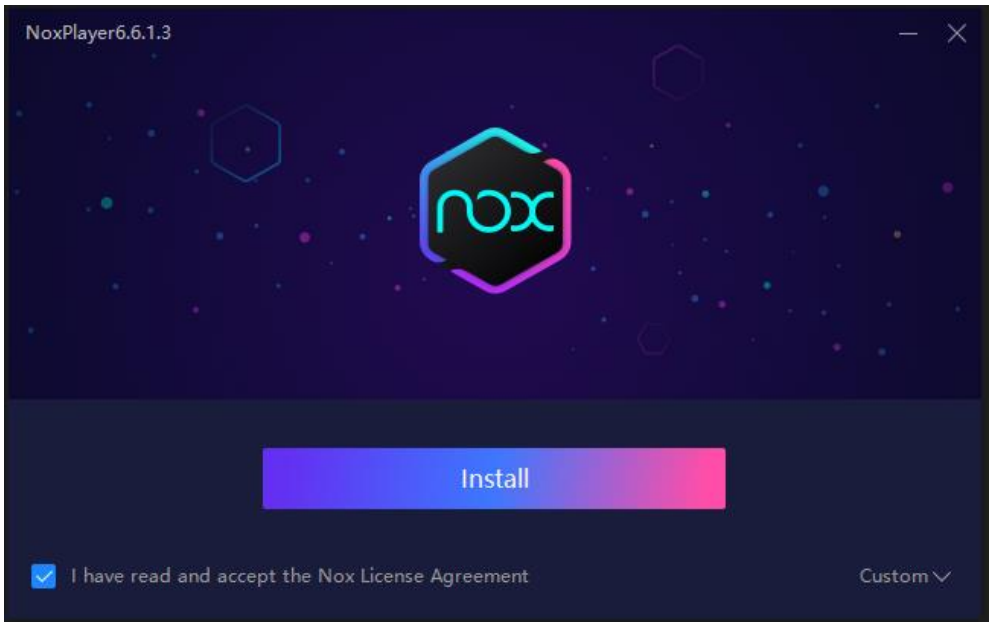


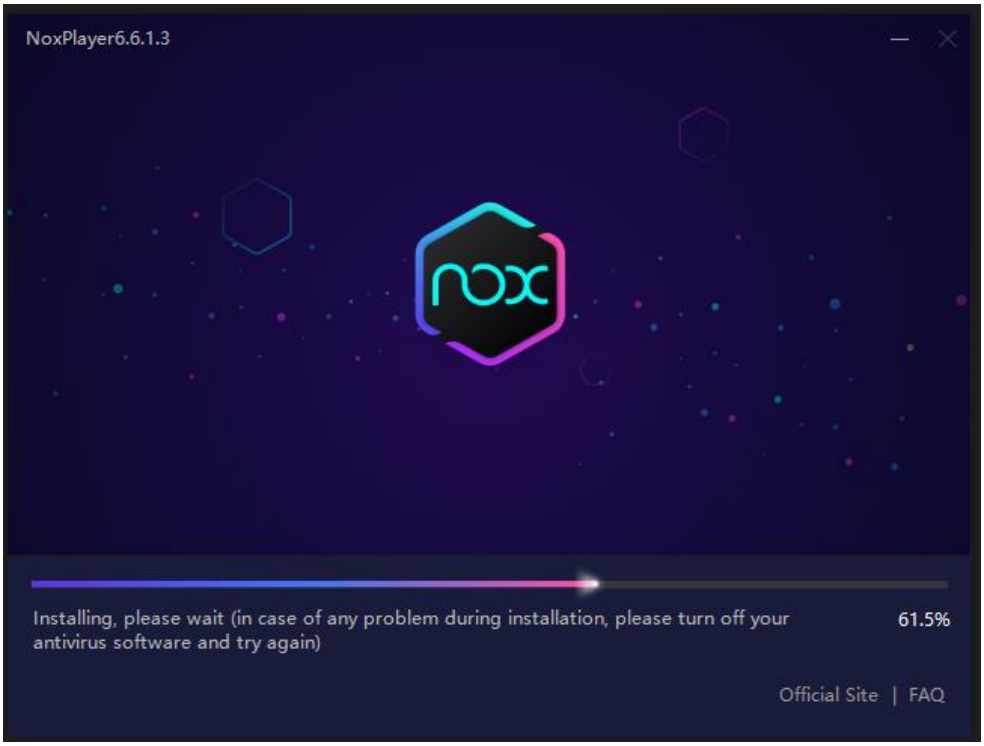

*Imagen 1.1 Ejecutar como administrador*

Una vez que le hayamos dado en la opción de ejecutar como administrador, nos mostrará una ventana como la siguiente en la cual nos dará un botón con la instrucción de instalar

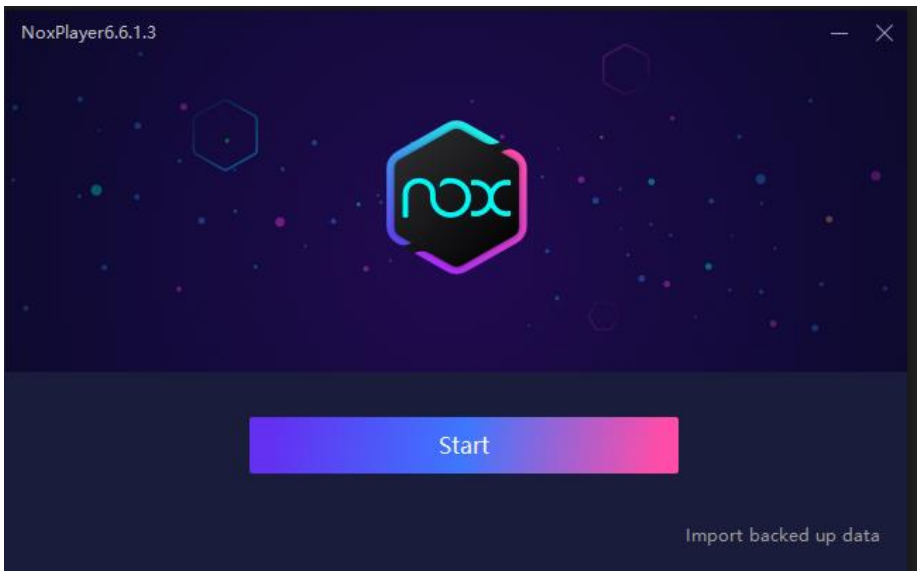



.

Una vez que le picamos en instalar nos mostrara el progreso de la instalación, esto puede tardar dependiendo de la velocidad del procesador de la máquina.



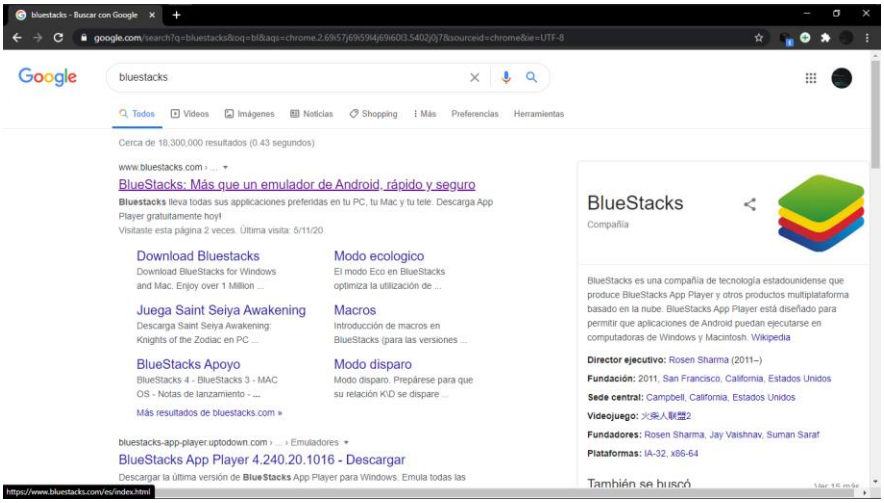

Una vez que se termine de instalar nos aparecerá una ventana con la opción de Start (Iniciar), para poder ver el entorno de este emulador vamos a darle clic en el botón.




Después de haberle dado clic al botón de iniciar nos mostrara la interfaz de nuestro emulador, la cual es la siguiente.

## Descarga de BlueStacks

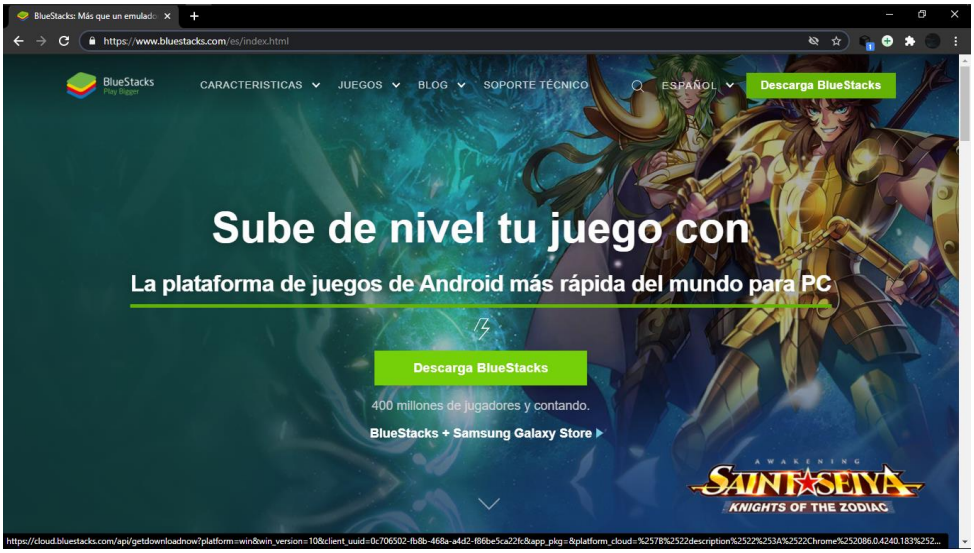
En este caso vamos a descargar el emulador BlueStack, para ello vamos a irnos al navegador y teclear el nombre del programa (BlueStacks), y vamos a elegir la primera opción




o bien hacemos clic en el siguiente link:

https://www.bluestacks.com/es/index.html

Este link nos enviara a la página oficial de BlueStacks, una vez ahí damos clic en el botón Download (Descargar).



## Instalar BlueStacks

Una vez descargado nos vamos a dirigir a la ruta donde tengamos nuestro ejecutador, y vamos a seleccionarlo, posteriormente vamos a darle clic derecho para desplegar el menú de opciones, de las opciones que aparecen vamos a elegir la opción que dice Ejecutar como administrador, esto para poder acceder al menú de instalación e instalarlo en nuestro sistema operativo.

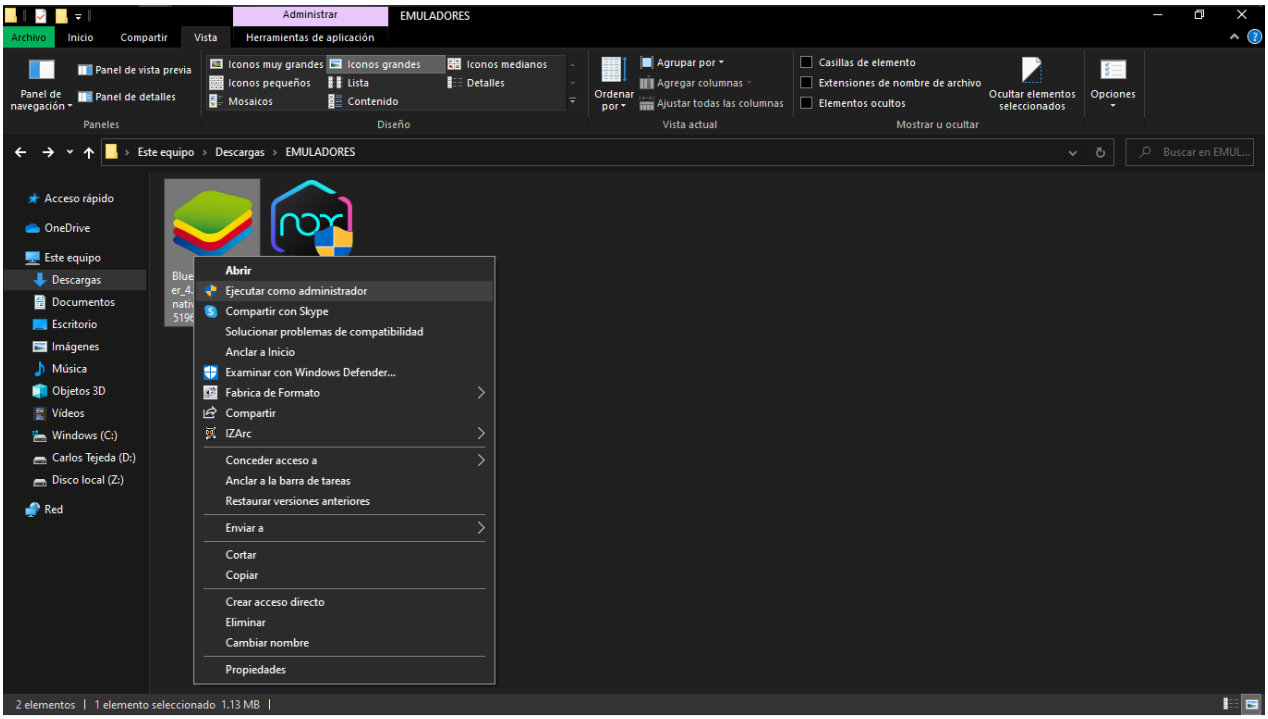



*Imagen 2.1 Ejecutar como administrador*

Una vez que le hayamos dado en la opción de ejecutar como administrador, nos mostrará una ventana como la siguiente en la cual nos dará un botón con la instrucción de instalar.

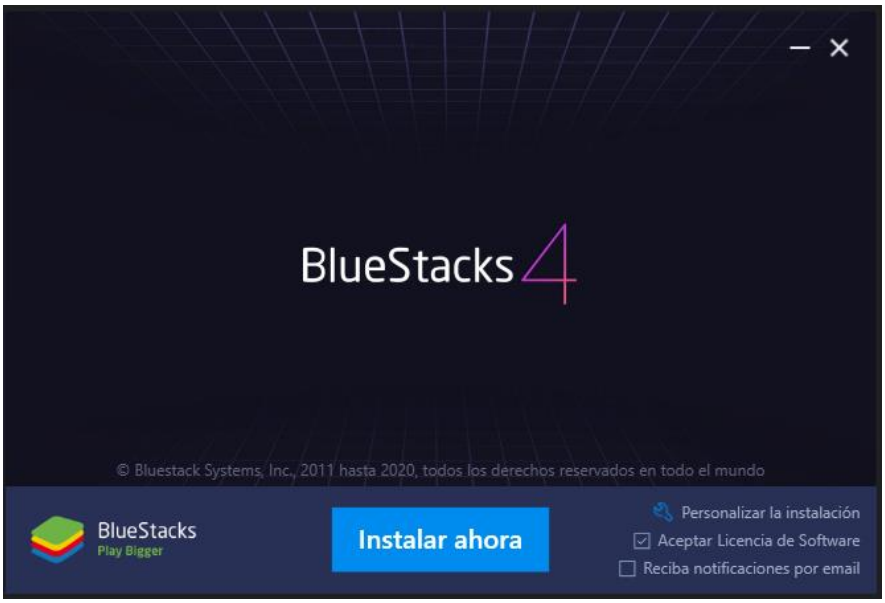



*Image 2.1.2 Inicio de Instalación*

Una vez que le picamos en instalar comenzara a descargar complementos de internet, esto puede tardar dependiendo de la velocidad del internet que tengamos.

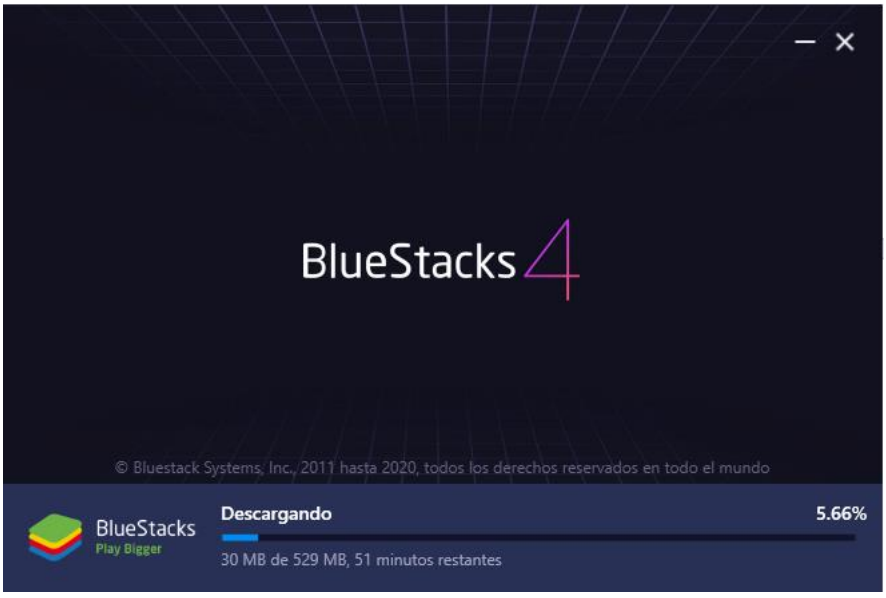



*Imagen 2.1.3 Proceso de Instalación*

Una vez que se termine de descargar se va a iniciar nuestro emulador BlueStacks, y podemos ver la interfaz gráfica como la siguiente.

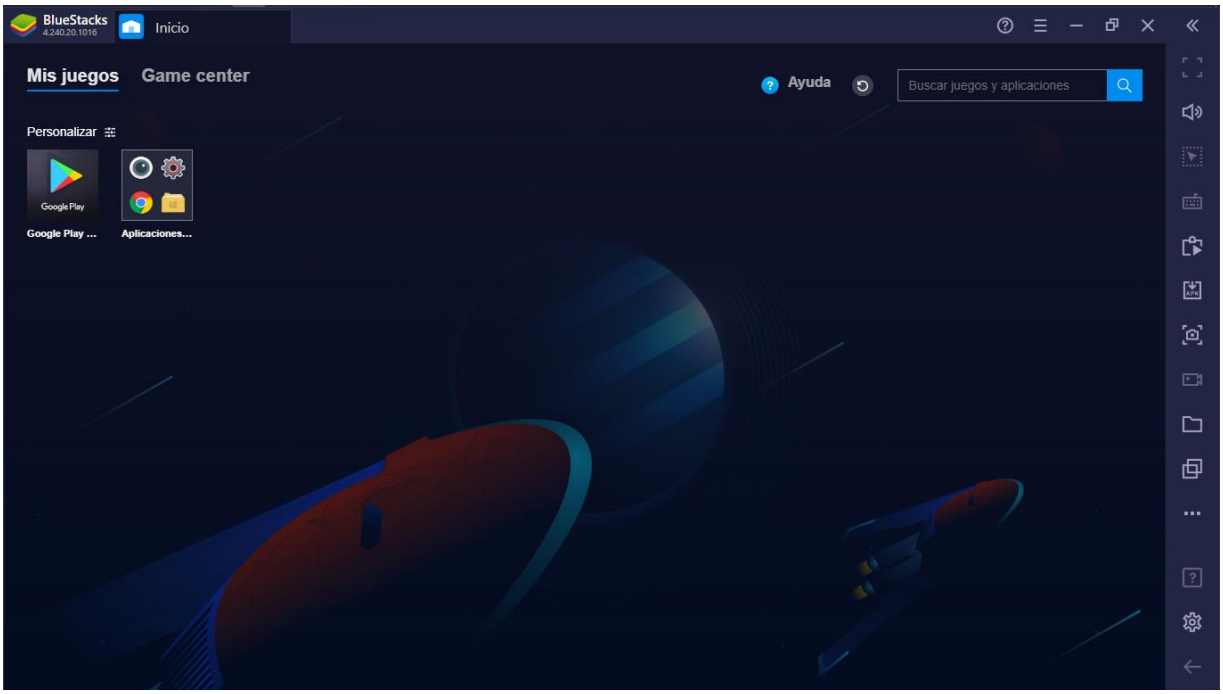



*Imagen 2.1.4 Interfaz del emulador BlueStack*

## Instalar Android Studio

Lo primero que debemos de hacer es descargar nuestro instalador, para ello vamos a irnos al navegador y escribir Android Studio, de las opciones que aparezcan elegimos la primera, y nos llevara a la pagina oficial de Android Studio, o bien podemos hacerlo desde este link.

Link de Descarga: https://developer.android.com/studio

Podemos ver la interfaz como la siguiente, para descargar solo damos clic al botón *DESCARGAR ANDROID STUDIO*

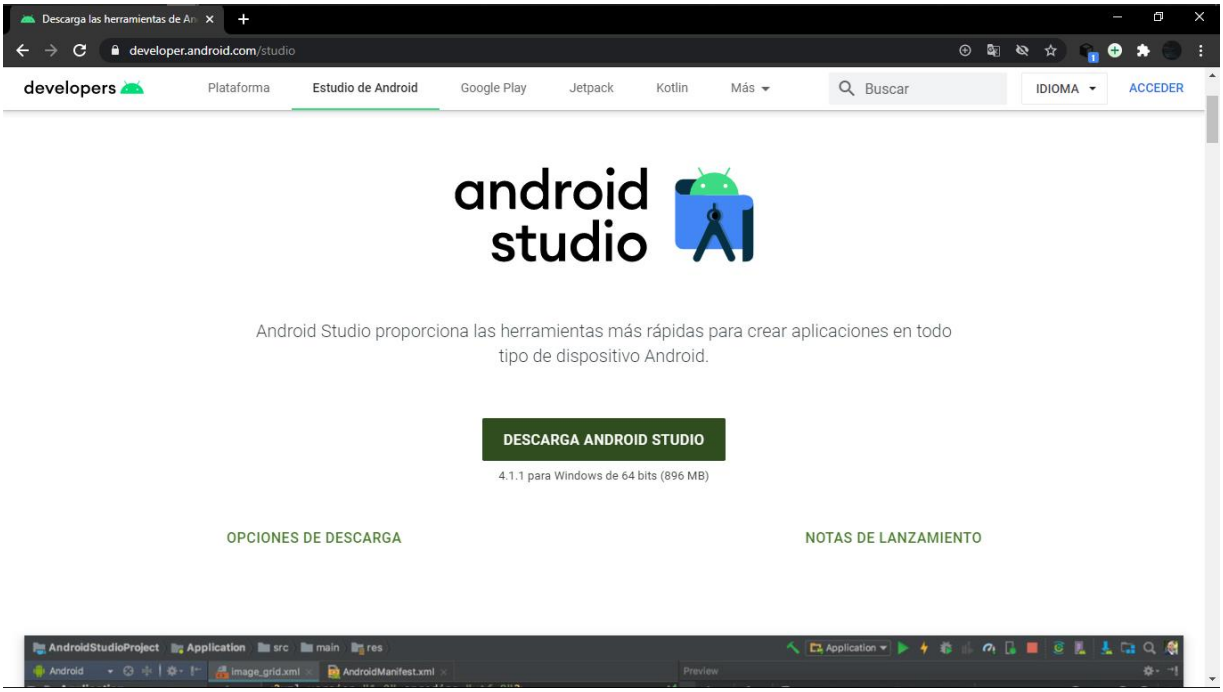



*Imagen 3.1.1 Descarga de Android Studio v4.1.1*

## Ejecutar el Archivo

Después de haber descargado nuestro archivo, nos vamos a dirigir al apartado donde se haya guardado, una vez ahí vamos a seleccionar el archivo y vamos a dar clic derecho sobre él, del menú desplegable vamos a dar un clic sobre la opción de ***Ejecutar como administrador.***

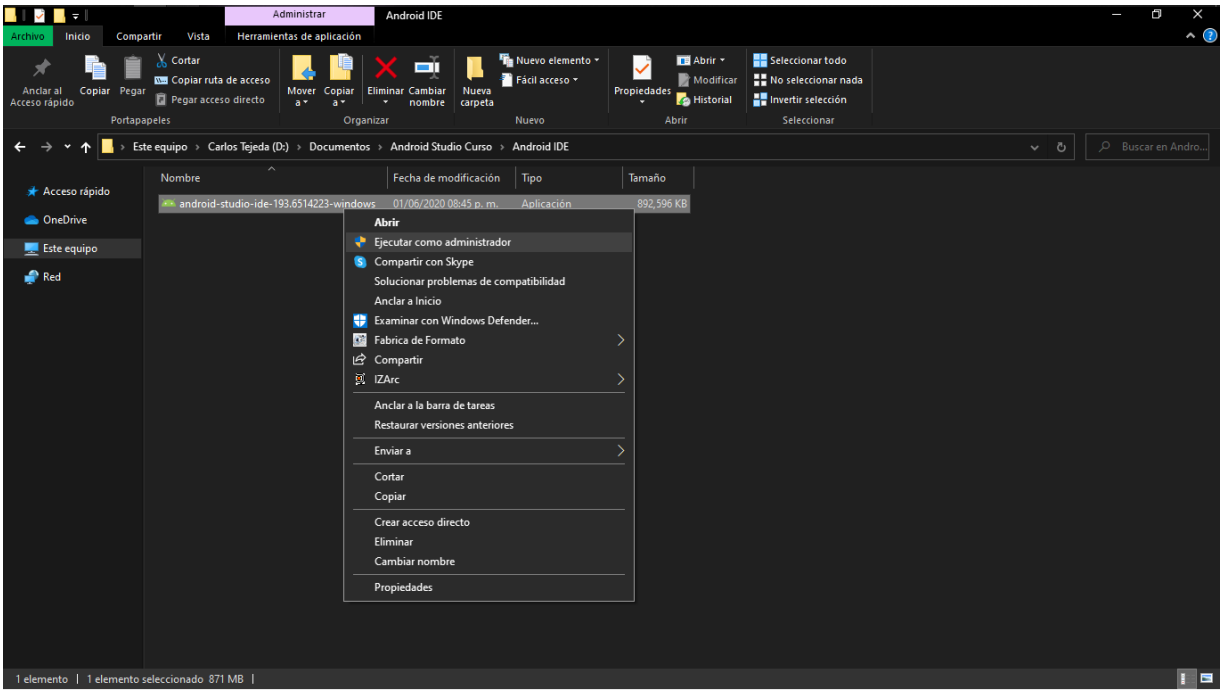



*Imagen 3.1.2 Ejecutar como Administrador*

Una vez que hayamos dado clic al botón de Ejecutar como administrador vamos a observar que se abre una ventana donde nos dará una bienvenida al asistente de instalación, para seguir con el proceso le daremos clic al botón ***Next>***, es importante mencionar que este IDE es muy completo, pero ocupa mucho recurso de memoria RAM.


*Imagen 3.1.3 Asistente de Instalación de Android Studio*

Una vez que le hayamos dado clic en el botón next en el paso anterior nos mandara a la ventana de selección de componentes, es importante que seleccionemos todos los componentes ya que si queremos utilizarlas mas adelante no se nos complique estar instalando complementos.

Una vez seleccionado los componentes, daremos clic al botón de ***Next>,*** para seguir con la instalación de nuestro IDE.


*Imagen 3.1.4 Selección de componentes de Android Studio*

Siguiendo con la instalación llegamos al apartado de los términos de licencia de Android Studio, debemos de leer los términos para saber a que se refiere, una vez que hayamos leído daremos en el botón de I Agree (Acepto) para poder seguir con la instalación de Android Studio.


*Imagen 3.1.5 Términos de licencia de Android Studio*

En el siguiente paso debemos elegir la ruta donde instalaremos nuestro Android Studio. Debemos elegir una ruta para el programa en sí y otra diferente para instalar el SDK, con bastante espacio disponible ya que las descargas y actualizaciones de los componentes de este suelen ocupar bastante espacio.

Una Vez que se hayamos configurado vamos a dar un clic al botón de ***Next>***

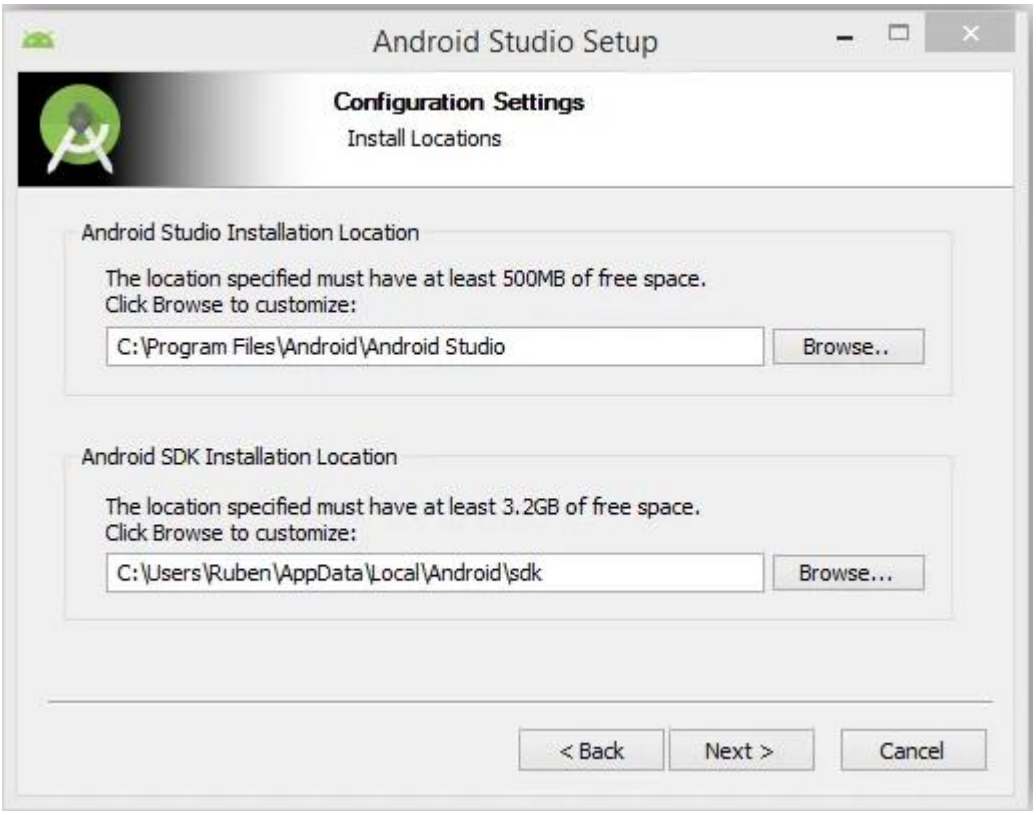


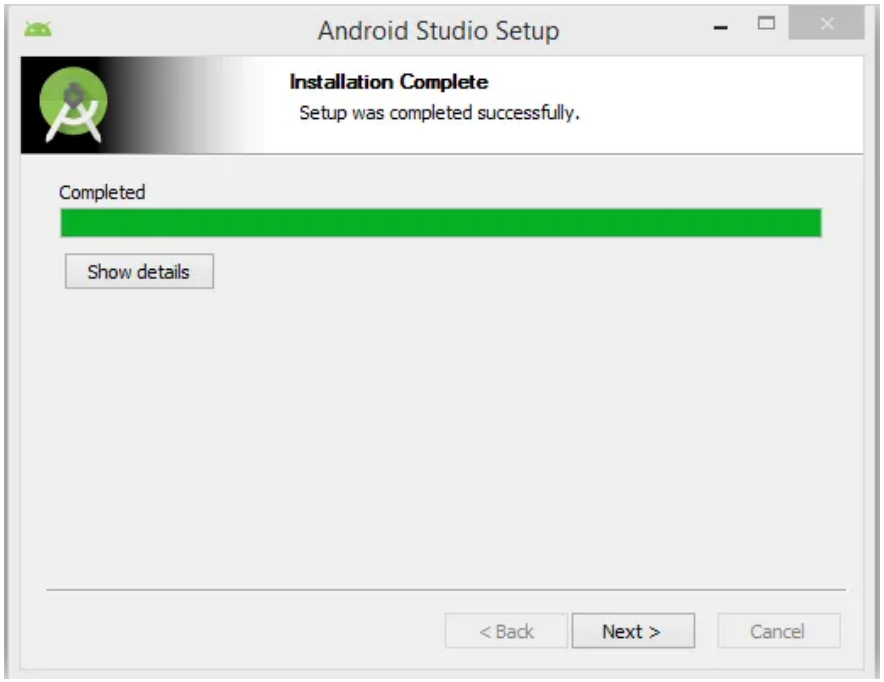

*Imagen 3.1.6 Configurar ruta de instalación*

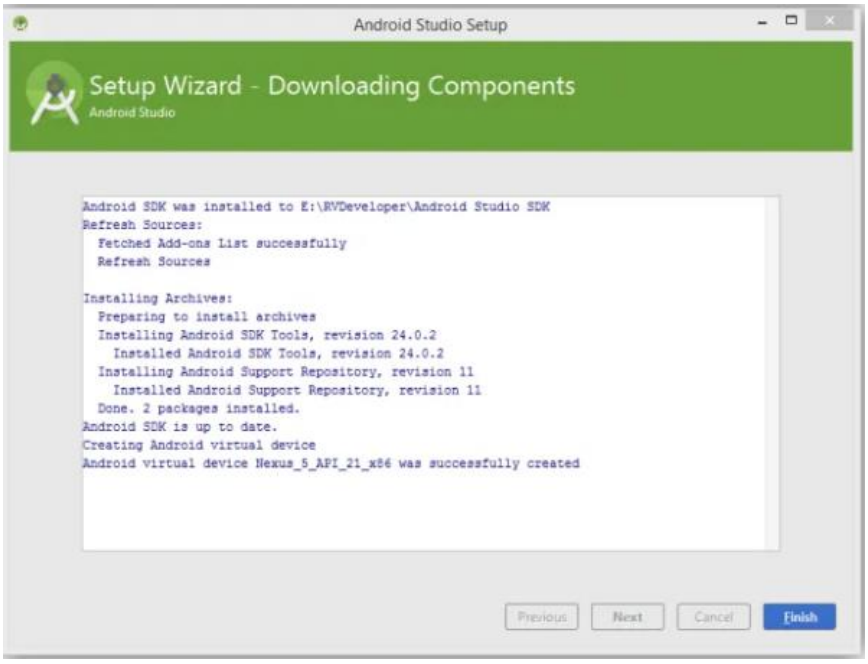
En el siguiente paso el asistente nos preguntará por la cantidad de memoria RAM que queremos asignar para el uso de máquinas virtuales y emuladores de Android. Cuanta mayor memoria mejor rendimiento tendrán estas, aunque debemos tener en cuenta que la mayoría del smartphone cuentan con un promedio de 2GB de memoria (de momento) y que el sistema operativo y las demás aplicaciones de nuestro ordenador también necesitan memoria, por lo que la cantidad que nos aparezca como recomendada será la que debemos dejar.

Una vez echo este paso daremos en el botón ***Next>.***


*Imagen 3.1.7 Configuración de memoria del emulador de Android*

Una vez hecho esto se comenzará a instalar nuestro IDE esto puede tardar varios minutos, depende de la velocidad del procesador y la del internet ya que descargará complementos del SDK.



## Bienvenida a Android Studio

Una vez configurado todo vamos a encontrar una ventana de bienvenida de Android Studio, en ella podemos ver varias opciones como Iniciar un nuevo proyecto, abrir un proyecto que ya hemos creado, importar un proyecto desde otra IDE como Eclipse u otros, como vamos iniciando vamos a dar un clic al botón de ***Star a new Android Studio Project***.

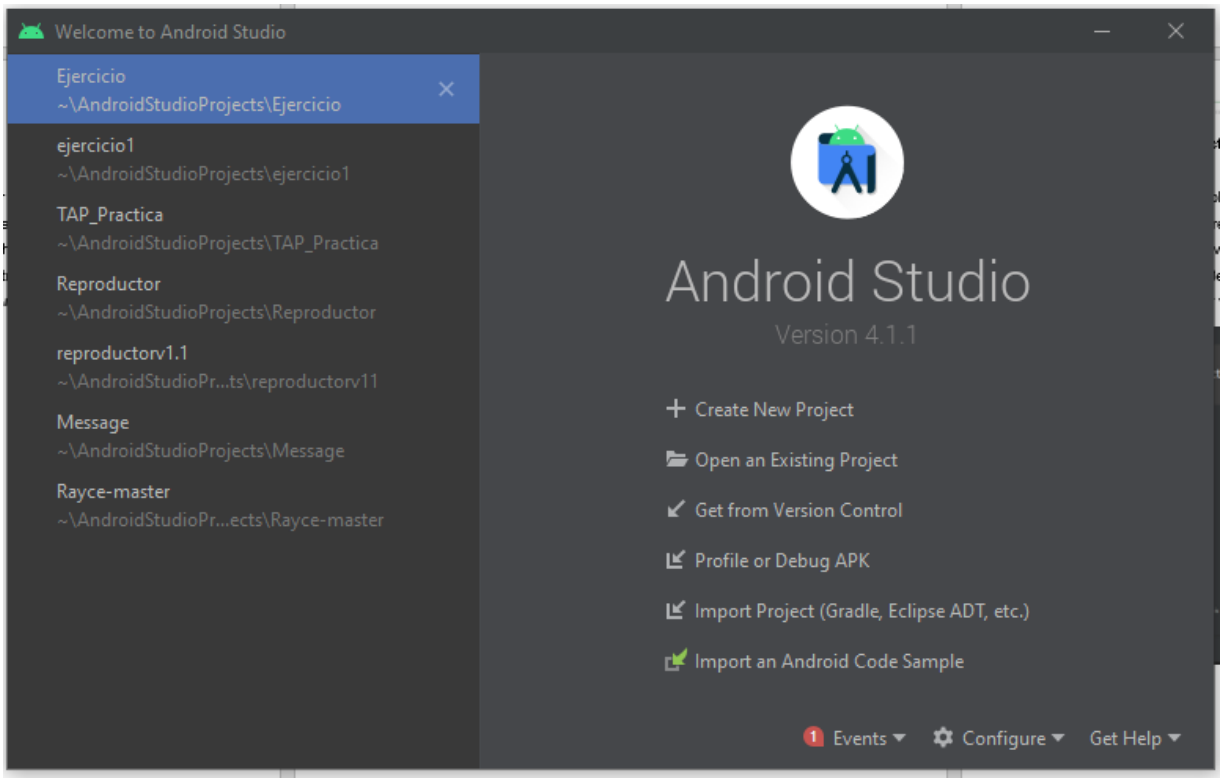



*Imagen 4.1 bienvenida a Android Studio*

## Selección de plantilla para el proyecto

Después de haber dado al botón de crear nuevo proyecto nos aparecerá, las opciones de Activity o plantillas que podemos agregar para apoyarnos mas en el diseño de la interfaz, para este proyecto vamos a agregar una plantilla vacía o Empty Activity, y daremos clic al botón de ***Next.***

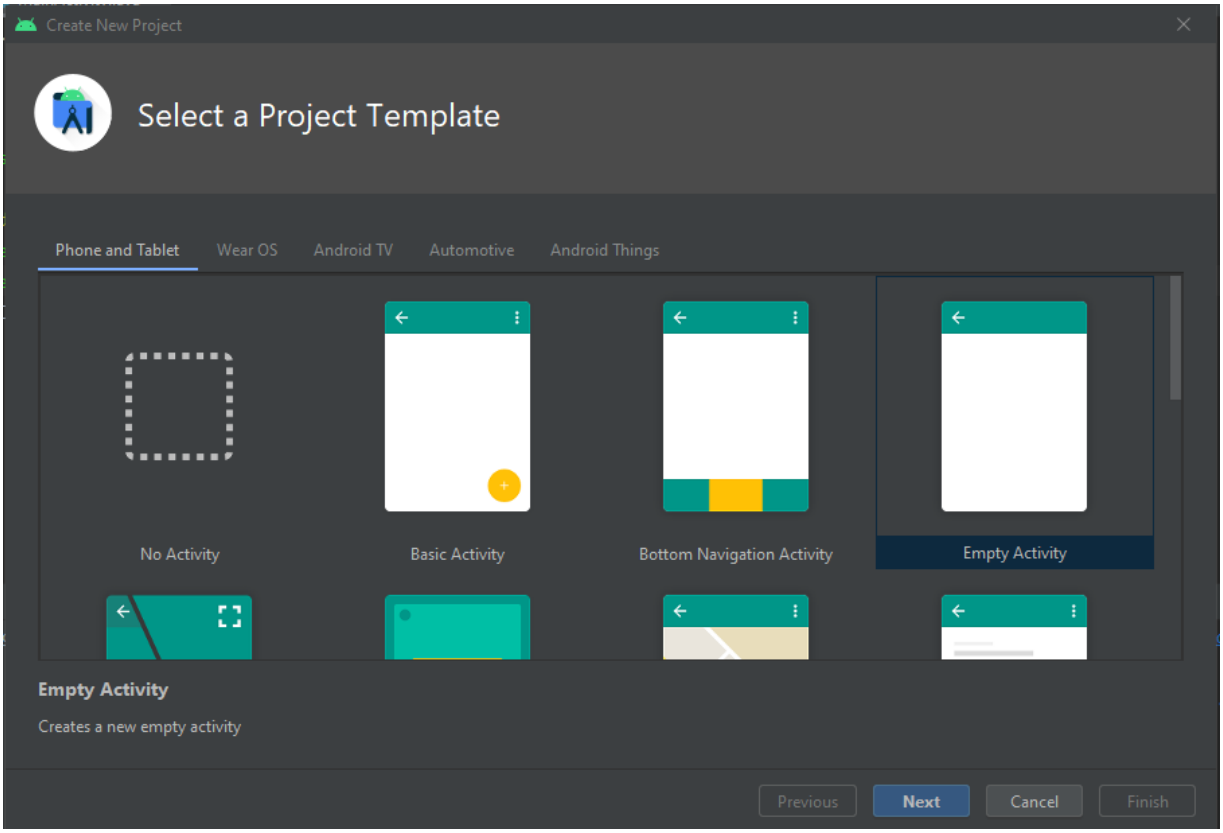



*Imagen 4.1.1 Selección de plantillas*

## Configuración del proyecto

Una vez seleccionado la plantilla vamos a nombrar a nuestro proyecto, vamos a colocar un nombre del paquete, seleccionaremos el lenguaje a programar en este caso Java, y el mínimo SDK en este apartado dejare el API 16: Android 4.1 (Jelly Bean) que tiene un rango del 99.8 % de posición en el mercado, para seguir vamos a dar clic al botón de ***Finish***.

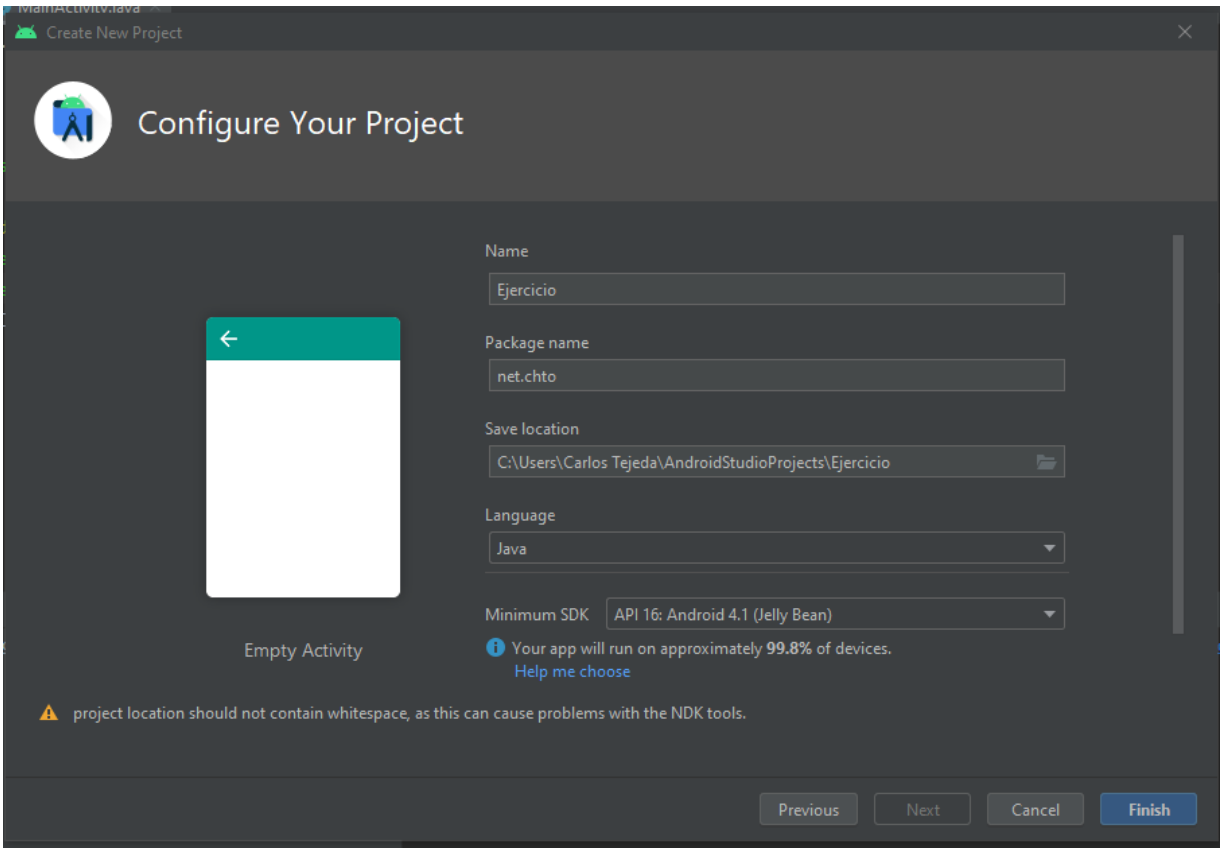



*Imagen 4.1.2 Configuración de plantilla*

## Revisando Área de Trabajo.

Una vez configurado lo anterior nos aparecerá la interfaz donde vamos a trabajar podemos ver como nuestra área de trabajo abre en el archivo principal con extensión .JAVA, el otro archivo con extensión .XML, es para visualizar de manera grafica como va a quedar nuestro programa.

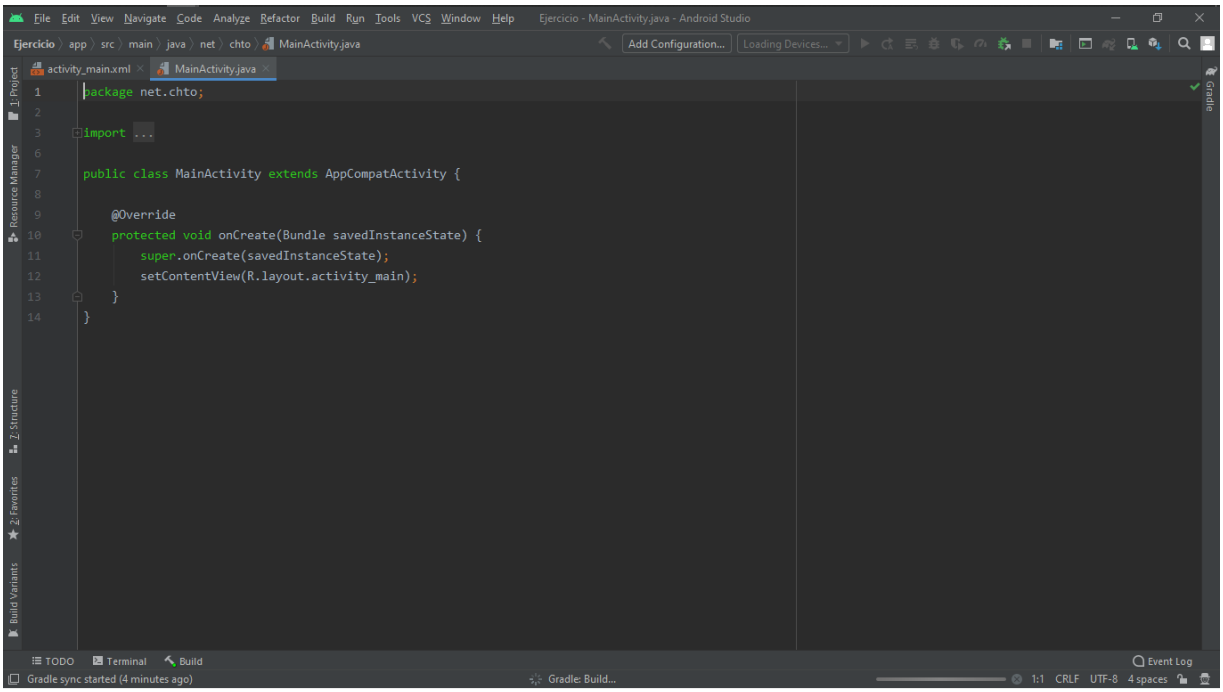



*Imagen 4.1.3 Revisando área de trabajo*

## Agregar Emuladores de Android Studio.

Para poder ir revisando nuestros avances Android studio nos proporciona varios dispositivos que podemos emular para ir probando las aplicaciones, para agregar uno de ellos vamos a ir a apartado de Tools y después en la opción de AVD Managers, daremos un clic en esta opción.

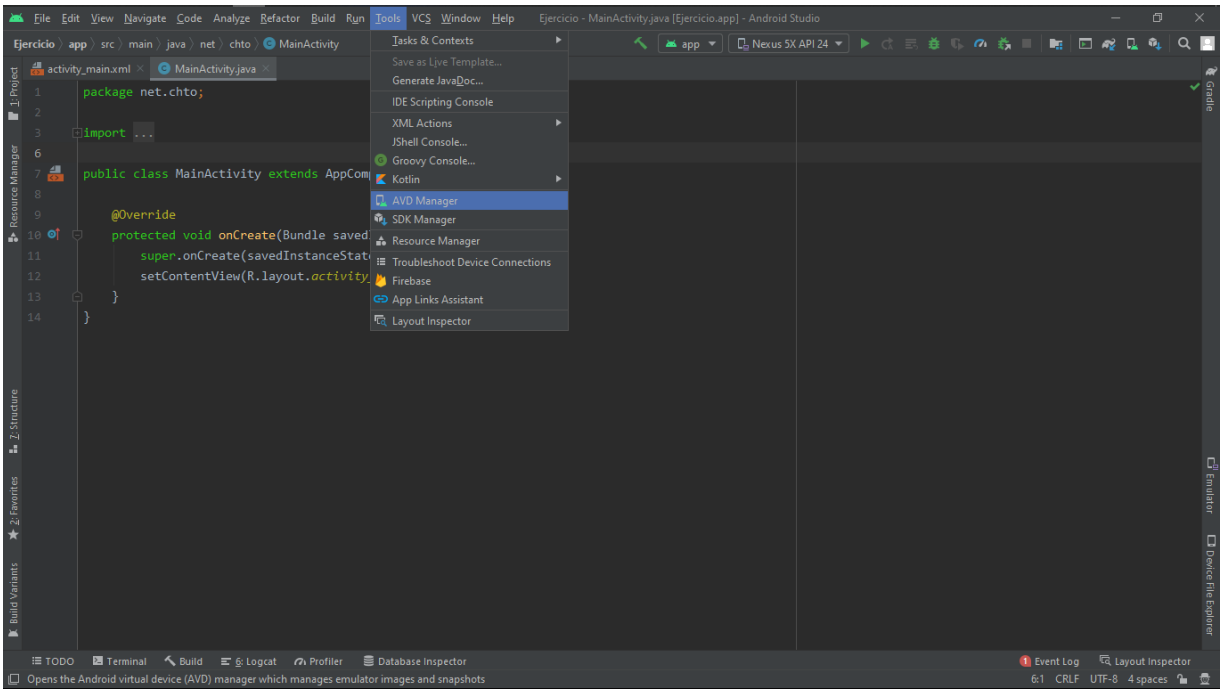



*Imagen 4.1.4 Agregar emuladores de Android*

## Agregar un nuevo equipo virtual

Una vez que le damos clic al botón de AVD Manager nos aparecerá esta ventana de los dispositivos que tenemos virtualizados, y de donde podemos agregar más, para agregar daremos clic al botón de Create Virtual Devices.

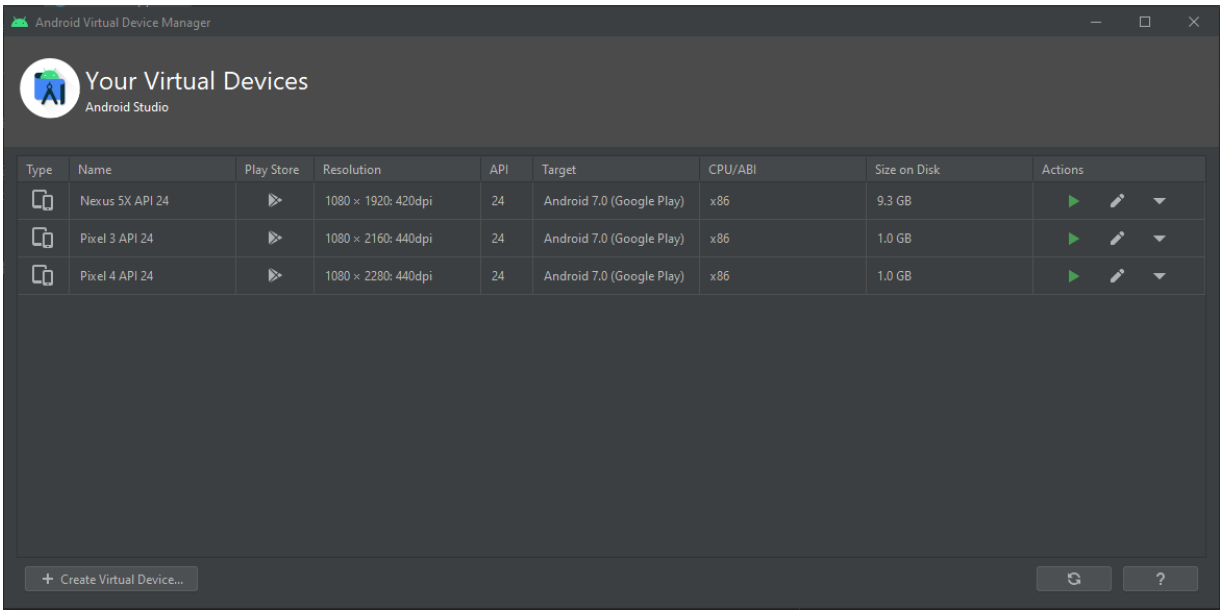



*Imagen 4.1.5 crear dispositivos virtuales*

## Selección del modelo del dispositivo a virtualizar

Una vez que hayamos dado clic al botón de crear dispositivo nos aparecerá esta ventana donde nos dice que seleccionemos el hardware que nosotros deseemos, encontramos varios dispositivos don diferente resolución y tamaño, vamos a elegir uno y damos al botón de ***Next.***

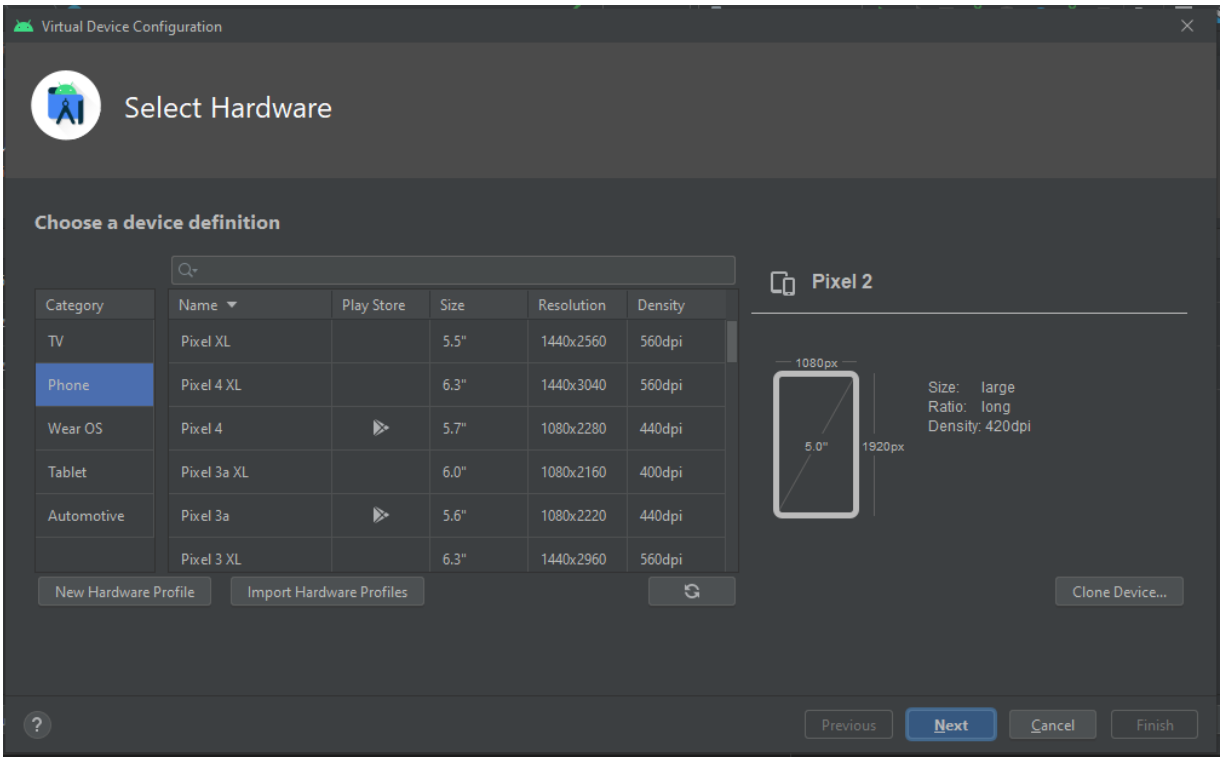



*Imagen 4.1.6 selección del hardware del emulador*

## Selección de la versión de Android

Una vez que seleccionemos los requisitos de hardware podemos seleccionar la opción de la imagen sistema a instalar en el emulador, en este caso vamos a seleccionar, la opción de R Android 11.0 la más actual.

Una vez echo esto vamos a dar clic al botón de ***Next,*** para continuar con la instalación.

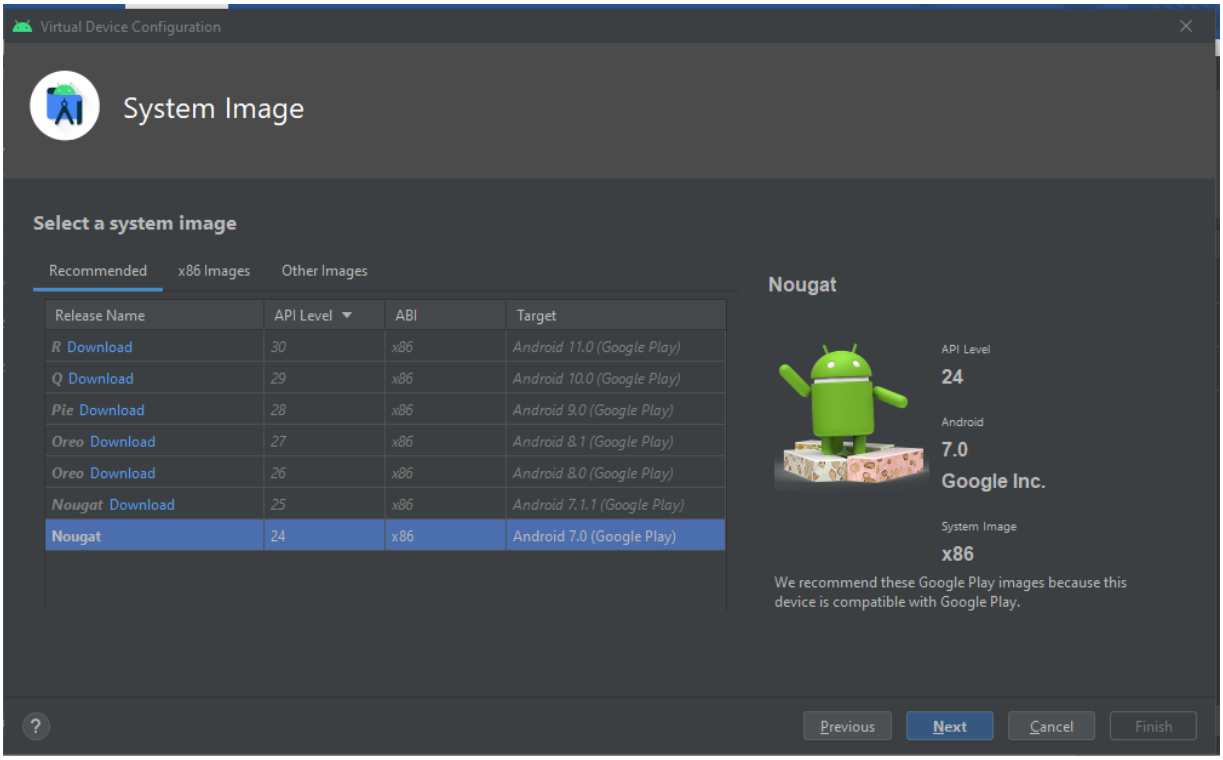



*Imagen 4.1.7 selección de la version de Android a instalar*

## Inicio de descarga de complementos

Una vez que hayamos terminado los procesos de configuración anteriores, se comenzara a descargar los complementos que se van a instalar en nuestro emulador virtual, esto dependerá en gran medida de la velocidad de nuestro internet, un promedio de descarga es de 2 a 4 horas.

Esto lo haremos nuevamente para crear otros dos emuladores más, de diferente resolución y tamaño.

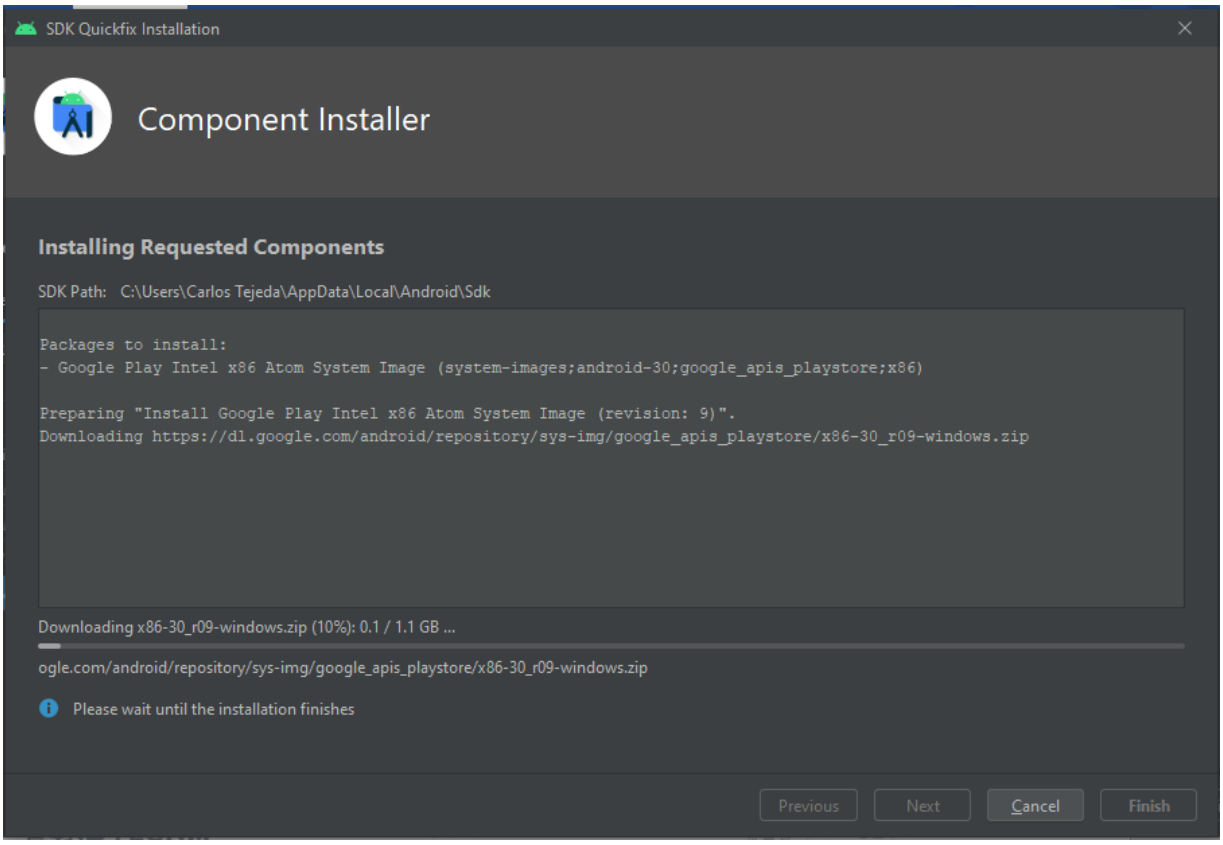



*Imagen 4.1.8 descarga de componentes del emulador creado*

## Verificación de instalación de los tres Dispositivos virtualizados

Si nosotros desplegamos el menú que esta al lado del icono de play, podremos ver que efectivamente hay tres emuladores de diferente nombre, esto quiere decir que tenemos ya instalados los tres emuladores en Android.

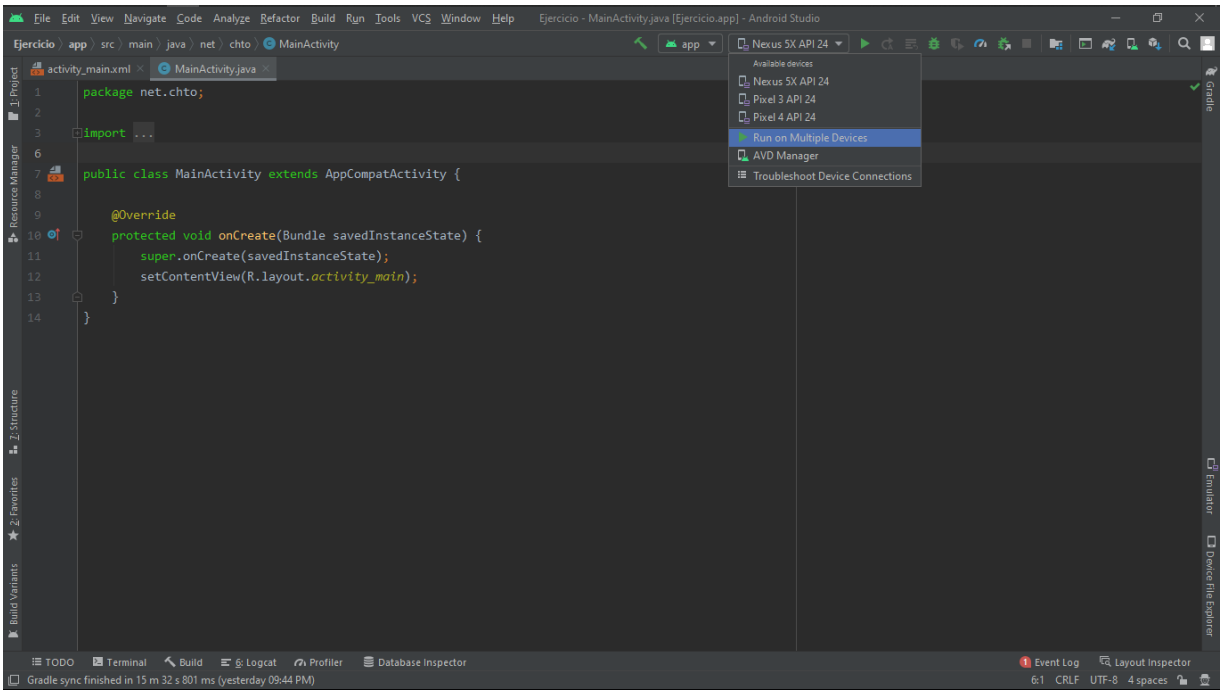



*Imagen 4.1.9 Verificación de instalar los emuladores en Android Studio*

## Resultados

## Interfaz del emulador BlueStacks y Nox Player.

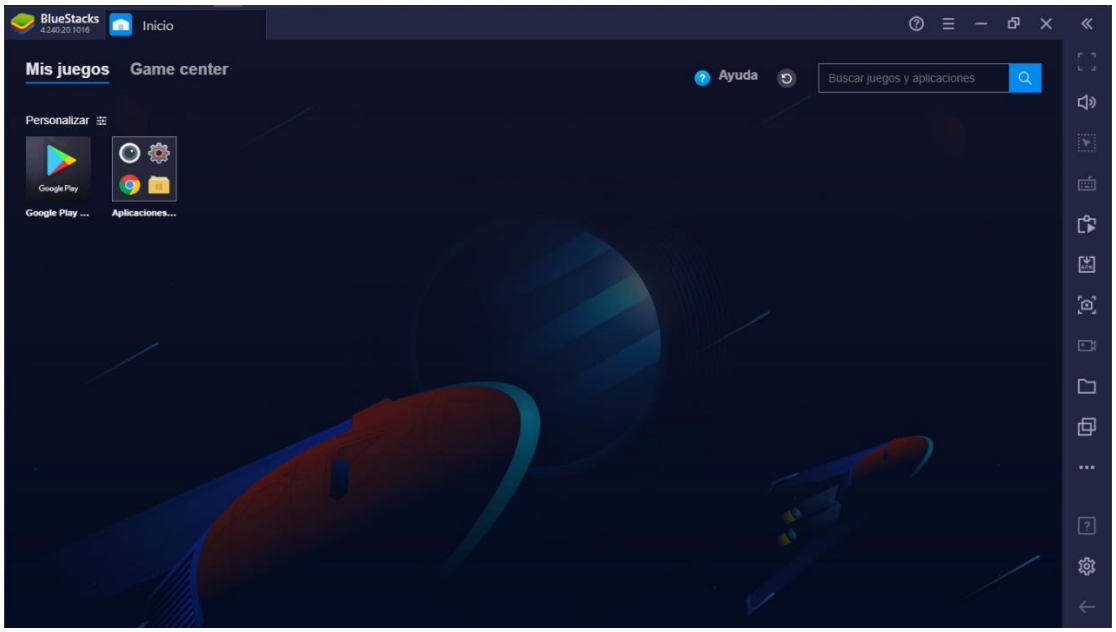



*Imagen 5.1 Interfaz del emulador Blue Stacks*

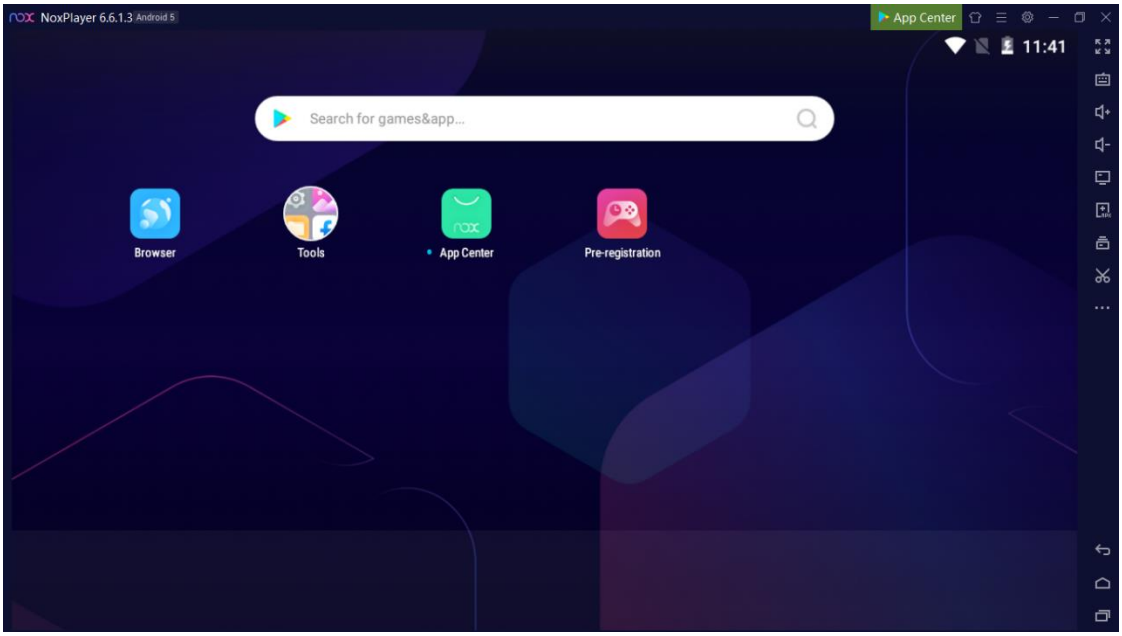



*Imagen 5.1.2 Interfaz del emulador Nox Player*

## Consumo de memoria de los emuladores

Ahora vamos a comparar que tanto de consumo de memoria tienen estos emuladores, el primer método es, irnos primero al símbolo del sistema (combinación de teclas ***Windows + R***, en la ventana ***Ejecutar*** escribir ***cmd*** y dar enter), una vez ahí escribir el comando ***tasklist |more,*** y dar enter. Podemos ver que ahí están los dos procesos de los emuladores, uno esta consumiendo 73,008 kb y el otro 141,428 kb.

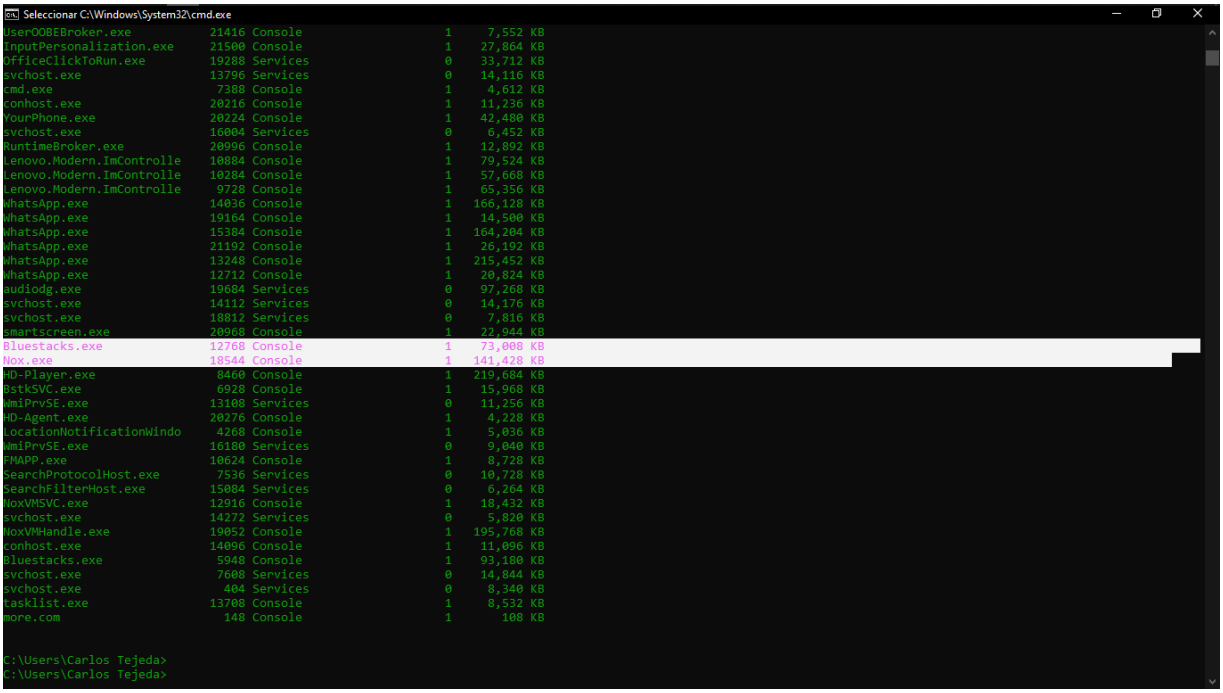



*Imagen 5.1.3 Consumo de memoria de los emuladores consultada desde cmd*

La segunda manera es irnos al administrador de tareas, daremos ***clic derecho en la barra de tareas***, elegimos la opción de ***administrador de tareas***, y se abrirá la ventana, en ella nos iremos al apartado de ***Detalles,*** y estando ahí vamos a buscar los procesos de los emuladores. Podemos ver que el BlueStacks esta ejecutando dos veces una con 22,932 k y la otra con 88,068, mientras que el NoxPlayer, está ejecutando tres veces una con 15,524, otra con 108,980, y la ultima de 3,928.

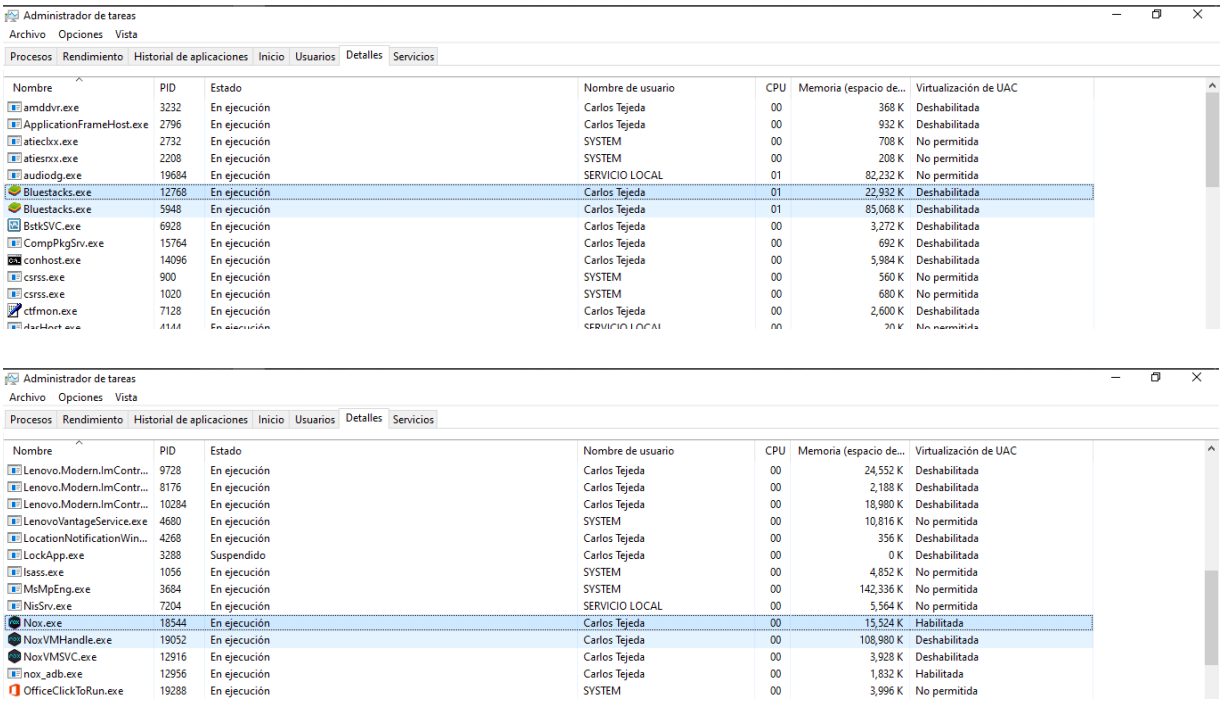



*Imagen 5.1.4 Verificación de consumo de memoria de manera grafica*

## Especificaciones del pc Utilizada

Estos emuladores fueron instalados en una laptop con las siguientes capacidades, de memoria y procesador.

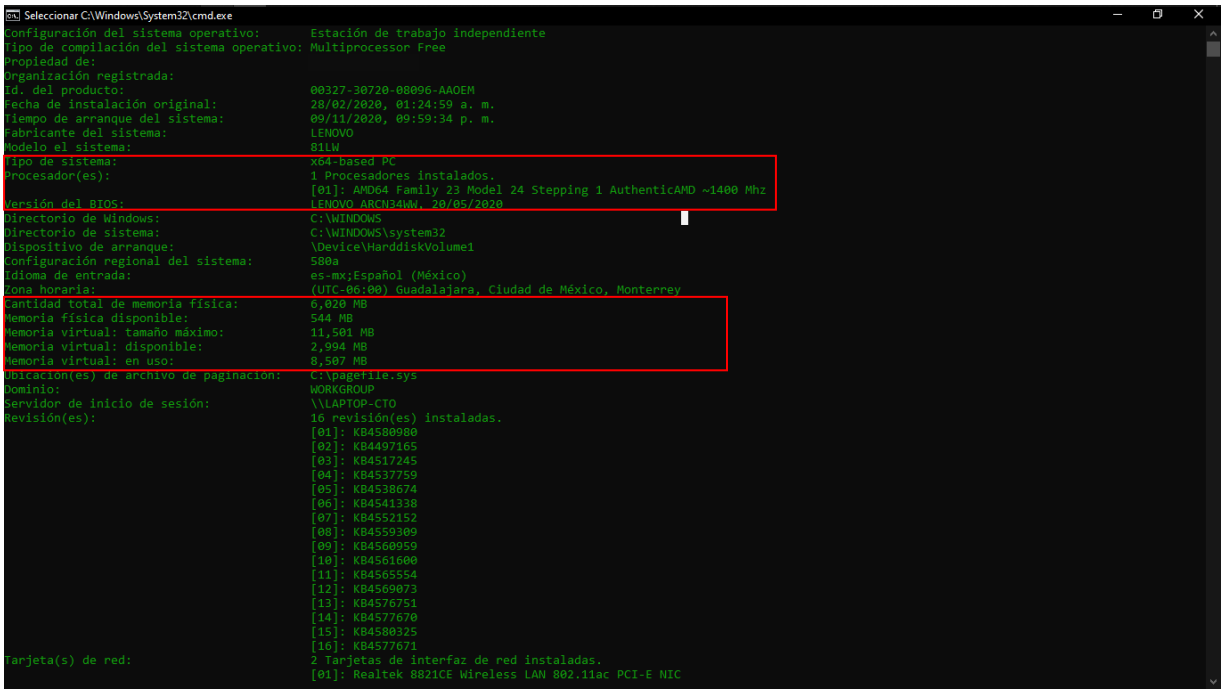



*Imagen 5.1.5 Revisión de capacidad de la computadora*

## Conclusión

En base a lo investigado y recopilado en este documento podemos entender y comprender que existen varias plataformas de desarrollo de software, entre los mas principales contamos con Android y iOS, existen otros como BlackBerry o Windows Mobile pero estos dos últimos ya están en desuso o descontinuados y los que lideran el mercado son Android y iOS, es por ello que nos dimos a la tarea de investigar las diferentes lenguajes de programación de estas dos plataformas de dispositivos móviles, entre ellas para el sistema Android esta el lenguaje Java, Kotlin y C++, y en el caso de iOS encontramos el Objetive-c y Swift, es importante conocer los lenguajes de programación y saber que tanto están influyendo en la actualidad, así como debemos de conocer de igual manera los entornos de desarrollo de cada lenguaje, y saber los requerimientos mínimos para poder instalarlos en nuestros equipos. Entre los emuladores mas conocidos podemos decir que para desarrollo de Android encontramos el famoso IDE Android Studio, que es un IDE muy completo, pero de igual manera es muy importante contar con los requerimientos de instalación ya que si no, el pc puede no soportarlo. En caso de iOS encontramos el Objetive-C que cuenta ya con su entorno de desarrollo.

# Referencias


(s.f.). Obtenido de https://tecnologiasmovilesenlaeducacionvirtual.wordpress.com/arquitecturas-moviles/

(s.f.). Obtenido de https://sites.google.com/site/tecnologiaiostm/desarrollo-de-aplicaciones/arquitectura-ios#:~:text=La%20arquitectura%20iOS%20est%C3%A1%20basada,Arquitectura%20de%20capas%20iOS.&text=Cocoa%20Touch%20es%20la%20capa,el%20desarrollo%20de%20aplicaciones%20iOS.

(s.f.). Obtenido de https://sites.google.com/site/lenguajedeprogramacionit3p1/unidad-i-introduccion-a-el-lenguaje-c/entornos-de-desarrollo

(s.f.). Obtenido de https://rockcontent.com/es/blog/que-es-un-lenguaje-de-programacion/

(s.f.). Obtenido de https://blog.nubecolectiva.com/lenguajes-de-programacion-para-desarrollar-aplicaciones-moviles/

(s.f.). Obtenido de https://swiftbycoding.dev/el-lenguaje-de-programacion-swift/

(06 de Noviembre de 2019). Obtenido de https://www.minutoneuquen.com/tecno/2020/1/18/que-es-para-que-sirve-android-studio-181273.html

A24.com. (s.f.). Obtenido de https://www.a24.com/actualidad/que-son-los-emuladores-para-pc-08012020_rkQiVwmgL

ANDROID, T. (28 de Marzo de 2019). Obtenido de https://jorgen.cubava.cu/2019/03/28/prime-os-una-forma-de-usar-android-desde-tu-ordenador/

ARIMETRICS. (s.f.). Obtenido de https://www.arimetrics.com/glosario-digital/entorno-de-desarrollo

BlueStacks. (s.f.). Obtenido de https://www.bluestacks.com/es/index.html

BlueStacks. (s.f.). Obtenido de https://www.bluestacks.com/es/index.html

Colmenarez, J. (7 de febrero de 2014). Obtenido de https://prezi.com/arz9udgmr3yo/arquitectura-de-dispositivos-moviles/

Culturacion. (s.f.). Obtenido de https://culturacion.com/que-es-un-emulador-y-para-que-se-utiliza/

Deloitte. (s.f.). Obtenido de https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html

EcuRed. (s.f.). Obtenido de https://www.ecured.cu/Arquitectura_Inform%C3%A1tica

Mercado, A. (s.f.). Obtenido de https://sg.com.mx/revista/53/desarrollo-aplicaciones-m-viles-nativescript

Microsoft. (s.f.). Obtenido de https://docs.microsoft.com/es-es/windows/msix/package/device-architecture

Microsoft. (28 de Mayo de 2020). Obtenido de https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

mySolutions. (s.f.). Obtenido de https://mysolutions.cl/que-es-ionic/

NOTICIAS. (3 de Octubre de 2019). Obtenido de https://www.muycomputer.com/2019/10/03/bliss-os-android-para-pcs/

NoxPlayer. (s.f.). Obtenido de https://es.bignox.com/

NoxPlayer. (s.f.). Obtenido de https://es.bignox.com/

OPENINNOVA. (s.f.). Obtenido de https://www.openinnova.es/cual-es-el-mejor-lenguaje-de-programacion-para-android-apps/

OS, B. (s.f.). Obtenido de https://blissos.org/

OS, p. (s.f.). Obtenido de https://primeos.in/download/

PLAY, M. (s.f.). Obtenido de https://www.memuplay.com/

Qualitydevs. (s.f.). Obtenido de https://www.qualitydevs.com/2019/07/05/que-es-flutter/

Qualitydevs. (31 de Mayo de 2019). Obtenido de https://www.qualitydevs.com/2019/05/31/que-es-ionic-desarrollador-web/

Tomas, E. (14 de Mayo de 2014). Obtenido de https://www.campusmvp.es/recursos/post/Objective-C-un-lenguaje-compilado-y-enlazado-para-programar-para-iPhone-y-iPad.aspx

VANDAL. (s.f.). Obtenido de https://vandal.elespanol.com/reportaje/que-es-y-como-configurar-bluestacks-

Xataka. (19 de Julio de 2019). Obtenido de https://www.xataka.com/basics/mejores-emuladores-android-para-pc

zone, S. s. (s.f.). Obtenido de https://www.softzone.es/2016/05/31/koplayer-emulador-gratuito-android-windows/