SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE HERMOSILLO

INSTITUTO TECNOLÓGICO DE HERMOSILLO

DEPARTAMENTO DE SISTEMAS COMPUTACIONALES

REPORTE INTERMEDIO DE AÑO SABÁTICO

23 DE ENERO A 22 DE JULIO DEL 2017

MANUAL DE PRÁCTICAS

DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

MARTHA ALICIA ROMERO DUEÑAS

DICTAMEN No. A5-1-89/2017

HERMOSILLO, SONORA **22 DE JULIO 2017**

Í N D I C E

INTRODUCCIÓN

Para desarrollar una buena aplicación móvil, lo primero que debemos hacer es aprender de la habilidades que utilizan los mejores desarrolladores. El propósito es conocer sistemas que nos permiten visualizar aplicaciones de la misma forma que la veríamos en un smartphone o tablet.

Este manual tiene un carácter eminentemente práctico y fomenta el aprendizaje del estudiante de Ingeniería en Informática mediante la incorporación gradual de funcionalidades a una aplicación.

El objetivo de este trabajo es elaborar un Manual de prácticas que permita a los estudiantes diseñar y desarrollar de manera lógica y coherente con base en el dominio de competencias para abordar cada uno de sus elementos, donde se conocerán la evolución, entorno arquitecturas y metodología para desarrollar aplicaciones móviles

El beneficio para los estudiantes de contar con un Manual de Prácticas de Desarrollo de Aplicaciones Móviles, es que podrán tener una idea más precisa de lo que deben hacer para elaborar dichas aplicaciones. También será útil para los docentes que imparten esta materia, ya que, se podrán uniformar los criterios y las competencias para su elaboración, tomando en cuenta que es una materia que se imparte en todas las carreras de los institutos del Tecnológico Nacional de México.

El manual contiene una breve descripción de cada una de las cuatro unidades y de los temas que conforman el programa de la materia de Desarrollo de aplicaciones para dispositivos móviles, incluye ejercicios por tema y al final de cada unidad se incluyen las prácticas programadas para este manual, las cuales se resolverán en forma individual y en trabajo de equipo. En total son 12 prácticas

Cada práctica cumple con el formato establecido de 10 puntos: número de práctica, título, objetivo, introducción, correlación con los temas y subtemas del programa de estudio, material y equipo necesario, metodología, sugerencias didácticas, reporte del alumno (resultados) y bibliografía preliminar.

OBJETIVO GENERAL

El principal objetivo de esta asignatura es que al final del curso los estudiantes sean capaces de diseñar y construir aplicaciones para dispositivos móviles usando la plataforma Android. Para alcanzar este objetivo, los estudiantes tienen que desarrollar paso a paso una aplicación Android completa. Esta asignatura ayuda a adquirir las siguientes competencias de acuerdo con el currículo:

TI3. Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

TI6. Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

IS3. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

CC3. Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

Elaborar un Manual de prácticas que permita a los estudiantes de Ingeniería en Informática, con base en el dominio de competencias:

- Identificar las implicaciones actuales de la programación móvil.
- Identificar las características de los diferentes emuladores para dispositivos móviles.
- Identificar los problemas de comunicación entre sistemas.
- Utilizar técnicas de modelado para la solución de problemas.
- Aplicar la sintaxis de un lenguaje para aplicaciones móviles.
- Aplicar un lenguaje para la solución de problemas para dispositivos móviles.

Además:

1. Presentar el panorama actual de hardware y software en lo referente a dispositivos móviles.
2. Describir en sus aspectos básicos los entornos iOS y Android, así como sus aplicaciones nativas
3. Mostrar las posibilidades del uso de los dispositivos móviles en la actividad docente, tanto en lo referido a los aspectos tutoriales como académicos.
4. Dar a conocer buenas prácticas llevadas a cabo con dispositivos móviles,
5. Establecer la situación actual y las perspectivas de futuro en el uso de los dispositivos móviles y los entornos que generan en educación.

ESQUEMA DE PRÁCTICAS POR UNIDAD

UNIDAD, TEMAS Y SUBTEMAS	NÚMERO, NOMBRE Y OBJETIVO ESPECÍFICO DE CADA PRÁCTICA
UNIDAD 1. Introducción a las tecnologías de móviles 1.1 Evolución de los dispositivos móviles. 1.2 Introducción a las tecnologías y herramientas móviles. 1.3 Tecnologías emergentes. 1.4 Tecnología de clientes ligeros: tecnología inalámbrica, redes de datos de radio, tecnología de microondas, redes de radio móvil, asistentes personales digitales, tarjetas inteligentes	PRÁCTICA #1 DESCARGAR S.O. Objetivo: Identificar las características de los diferentes emuladores para dispositivos Móviles. Conozcan y apliquen los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles.
UNIDAD 2.Arquitecturas y entorno de desarrollo 2.1 Sistemas operativos para dispositivos ligeros 2.2 Arquitecturas 2.3 Entorno de desarrollo 2.4 Requerimientos de los dispositivos ligeros 2.5 Lenguajes de programación 2.6 Configuraciones 2.7 Perfiles	PRÁCTICA #2 INSTALAR S.O. Objetivo: El propósito es que los estudiantes conozcan y apliquen los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles Que muestre la estructura, configuración y operación de un entorno de desarrollo para aplicaciones móviles mediante PRÁCTICA #3 CONFIGURAR S.O. PRÁCTICA #4 ENTORNO DE DESARROLLO EMULADORES A UTILIZAR Objetivo: Conocer y aplicar los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles**.** El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio PRÁCTICA #4 ENTORNO DE DESARROLLO EMULADORES A UTILIZAR Objetivo: Desarrollar aplicaciones móviles nativas, web e híbridas para atender las necesidades del entorno. El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio
UNIDAD 3. Desarrollo de aplicaciones móviles 3.1 Metodología de desarrollo y ejecución. 3.2 Uso de formularios Web móvil. 3.3 Uso de controles. 3.4 Creación Interfaces de usuario. 3.5 Temas selectos de programación para móviles.	PRÁCTICA #5 ENTORNO DE DESARROLLO LENGUAJE A UTILIZAR Objetivo: Desarrolla aplicaciones móviles nativas, web e híbridas para atender las necesidades del entorno**.** El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio

	PRÁCTICA #6 INSTALACION DE UN SISTEMA GESTOR DE BASE DE DATOS PARA MOVILES Objetivo: Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD PRÁCTICA #7 DESARROLLO DE PLICACIONES MOVILES NATIVAS QUE RESUELVAN PROBLEMATICAS DIVERSAS Objetivo: Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es identifique la diferencia entre aplicaciones movibles nativas, web e hibridas
UNIDAD 4. Administración de datos en dispositivos móviles 4.1 Introducción. 4.2 Modelo de objetos de acceso a datos. 4.3 Manipulación de datos 4.4 XML. 4.5 JSON.	PRÁCTICA #8 DISEÑO DE UNA ESTRUCTURA DE APLICACIÓN EN UN AMBIENTE CLIENTE-SERVIDOR Objetivo: Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD PRÁCTICA #9 DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MOVILES CON ENFOQUE CLIENTE Y SERVIDOR Objetivo: Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD PRÁCTICA #10 DISEÑO DE UNA APLICACIÓN MOVEL QUE INTEGRE BASES DE DATOS Objetivo: Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD PRÁCTICA #11 DESARROLLO DE UNA APLICACICON MOVIL QUE INTEGRE BASES DE DATOS Objetivo:

	Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD PRÁCTICA #12 PROYECTO INTEGRADOR Objetivo: Aplicar tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles. El propósito es que los estudiantes haya dominada el diseño y desarrollo de las aplicaciones móviles.

PRÁCTICA #1

DESCARGAR S.O.

Competencia(s) a desarrollar.

Identificar las características de los diferentes emuladores para dispositivos Móviles.

Introducción

Si pensamos en dispositivos móviles, lo primero que nos viene a la cabeza es un teléfono móvil. Pero en la actualidad son varios los dispositivos móviles disponibles en el mercado: PC portátiles, PocketPC, tabletas, etc.

Esta diversidad comporta una importante problemática para quien debe programarlos, ya que cada uno tiene unas características particulares: dispone de una memoria determinada o ha de soportar un lenguaje y un entorno específicos.

Por todo ello veremos las características generales de los dispositivos móviles para después clasificarlos. A nivel más técnico, veremos los componentes específicos que pueden tener y repasaremos las redes a las que pueden acceder.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con los temas de la unidad 1.

El propósito es que los estudiantes Conozcan y apliquen los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles.

Material y equipo necesario

Sistema operativo.

Una computadora que corra con Windows, Linux o Mac OS.

En Windows y en Linux no importa la arquitectura (32 bits o 64 bits).

Acceso a Internet

Metodología

Mediante esta práctica se pretende que obtengas los siguientes objetivos:

1. Entender que son los dispositivos móviles y cuáles son sus características.
2. Conozcas los tipos de dispositivos móviles existentes.
3. Tengas una visión general de las diferencias que puede haber entre dispositivos móviles en función de sus características.
4. Tengas una visión histórica de la evolución de los dispositivos móviles.

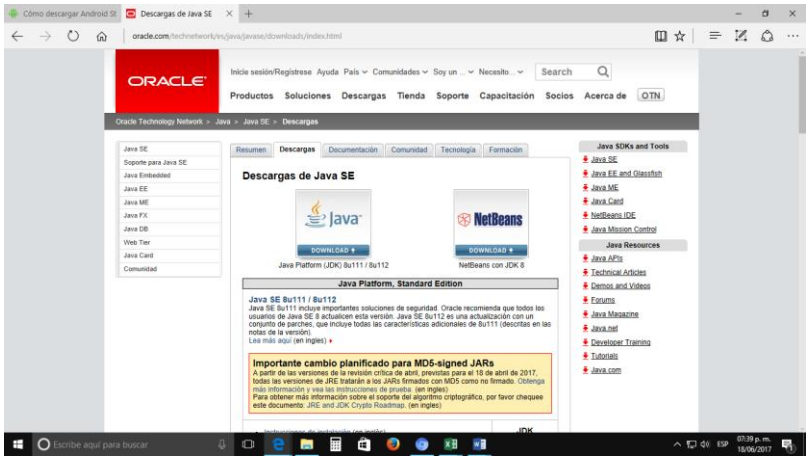
Descargar el Sistema Operativo para aplicaciones móviles

Herramientas de software:

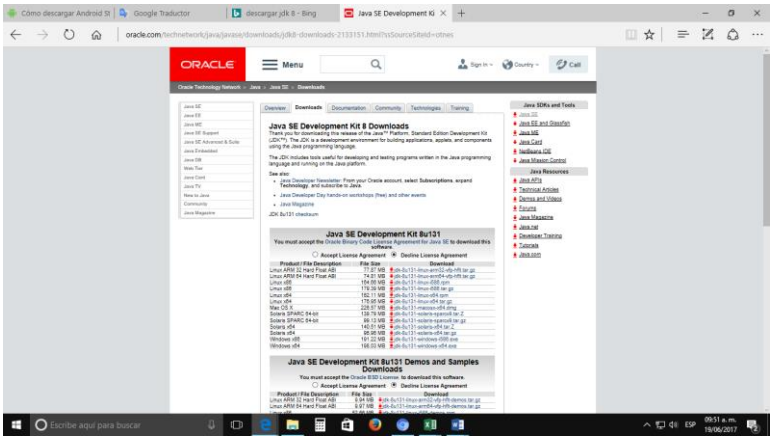
Sistema operativo. Una computadora que corra con Windows, Linux o Mac OS. En Windows y en Linux no importa la arquitectura (32 bits o 64 bits).

Java. Se recomienda la última versión de Java disponible, puede descargarse desde la siguiente página.

http://www.oracle.com/technetwork/es/java/javase/downloads/index.html



Seleccionas la Java Platform (JDK) y te aparecerá la siguiente pagina



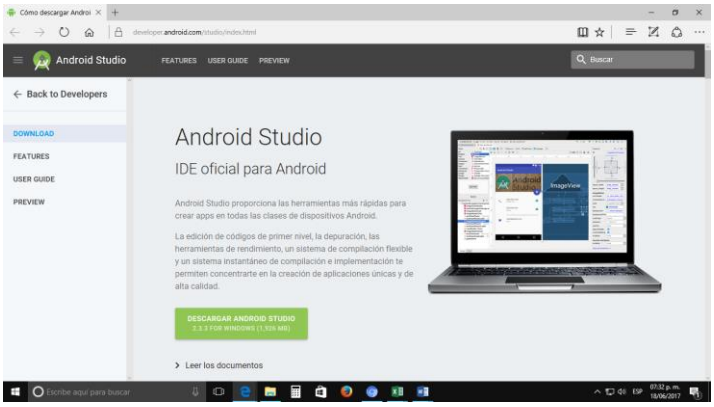
Selecciona la casilla **Accept License Agreement para poder descargar la versión de Java SE Development Kit que corresponda a tu equipo**

Kit de desarrollo. La última versión del SDK de Android, puede descargarse de la siguiente página. https://developer.android.com/intl/es/sdk/index.html

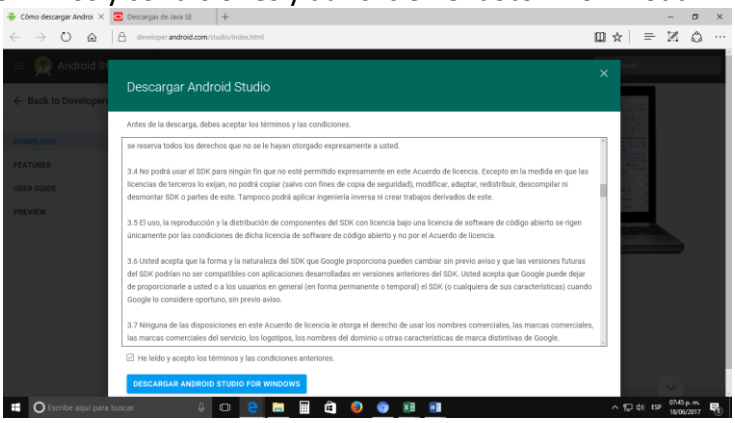
IDE. Android ha puesto a disposición de los desarrolladores su propio IDE, Android Studio. Este entorno de desarrollo ya incluye la última versión del SDK y puede ser descargado de la siguiente página.

https://developer.android.com/intl/es/sdk/index.html

1. Abrir un navegador y dirigirse a la siguiente dirección de Internet http://developer.android.com/sdk/index.html
2. Seleccionar la opción Download Android Studio.



3. Aceptar términos y condiciones y dar clic en el botón Download Android Studio.



4. Proceder a descargar.

Sugerencias didácticas.

Investigar y hacer un mapa mental de los Sistemas Operativos Ligeros y sus requerimientos

Reporte del alumno

Criterio para calificar	Sí	No
Entrega del Mapa Mental	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Julián david morillo pozo. (2014). *Introducción a los dispositivos móviles*. Retrieved 4 August, 2017, from https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_2).pdf

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #2

INSTALAR S.O.

Competencia(s) a desarrollar.

Conocer y aplicar los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles.

Introducción

Requisitos y proceso de instalación del entorno de desarrollo Android Studio, indispensable para realizar las prácticas a lo largo del curso y desarrollar aplicaciones en Android

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

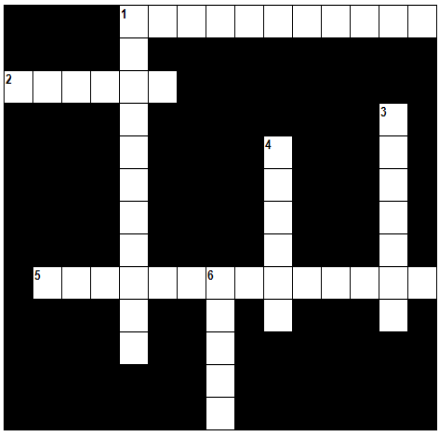
Esta práctica está directamente relacionados con los temas de la unidad 2.

El propósito es que los estudiantes conozcan y apliquen los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles.

Que muestre la estructura, configuración y operación de un entorno de desarrollo para aplicaciones móviles mediante una exposición.

Emplear tutoriales para comprender el proceso de instalación del ambiente de trabajo para el desarrollo de aplicaciones móviles.
Ejercicio. Realizar el siguiente crucigrama

CARACTERISTICAS ANDROID
CRUCIGRAMA



Check

Vertical:

1. Android es un sistema operativo que permite ejecutar varias aplicaciones al mismo tiempo
2. Android es propiedad de:
5. Esta basado para dispositivos que tienen:

Horizontal:

1. Consiste en una pantalla táctil o touchpad que reconoce simultáneamente múltiples puntos de contacto
3. es un sistema operativo diseñado especialmente para dispositivos moviles
4. Android nos brinda el soporte para base de datos a través de:
6. Android es un sistema operativo basado en:

Material y equipo necesario

Equipo Android. Uno de los puntos importantes que se recomienda, es tener es un teléfono que corra con Android. Esto es porqu**e** de esta manera se tendrá una auténtica forma de probar cómo es que corren las aplicaciones en un ambiente real, además de que existen algunos ejemplos prácticos que por sus características es altamente recomendable ejecutarlos en un dispositivo físico

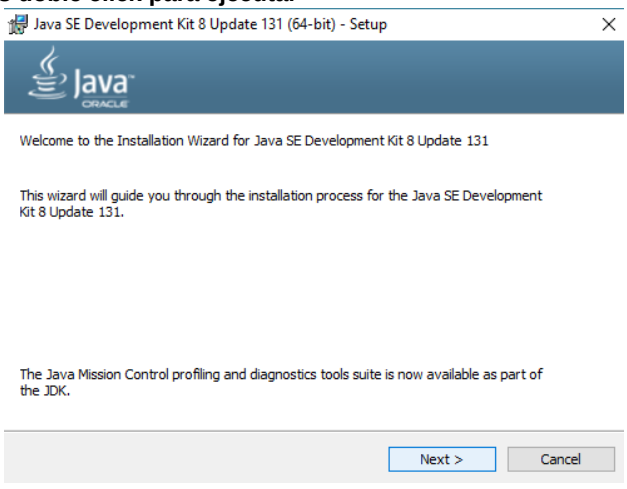
- Microsoft Windows 10/8/7/Vista/2003 (32 o 64 bits).
- 2 GB en RAM, 4 GB recomendado. 400 MB de espacio en disco duro para el IDE Android Studio
- Al menos 1 GB de espacio en disco duro para el SDK de Android.
- 1280 x 800 resolución de pantalla como mínimo.
- Oracle Java Development Kit (JDK) 7 o superior

Internet
PC o Laptop

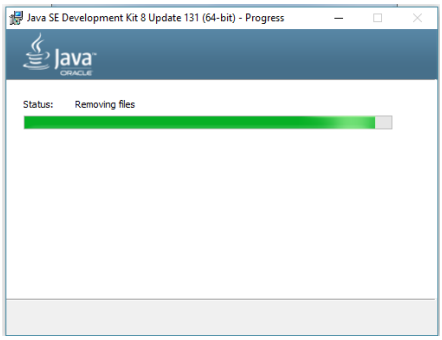
Metodología

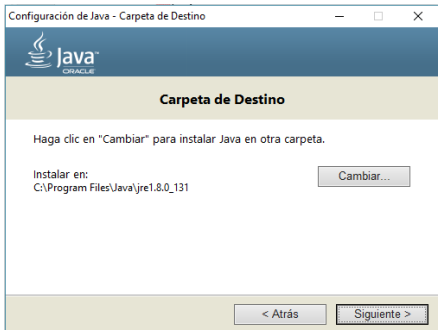
La siguiente guía lleva al usuario por el procedimiento para poder instalar el IDE Android Studio en alguna computadora que ejecute cualquiera de los sistemas operativos más usados; Windows, Mac o Linux.

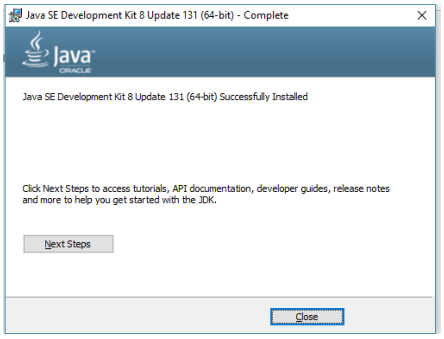
1. En la práctica no. 1 descargamos JKD, IDE
   a) Abrir la carpeta donde se descargó el arvhivo **Java SE Development Kit y le damos doble click para ejecutar**



Le damos Next



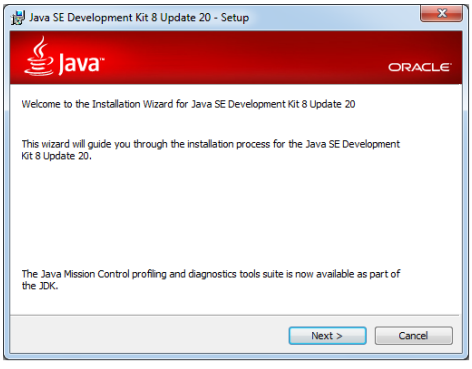




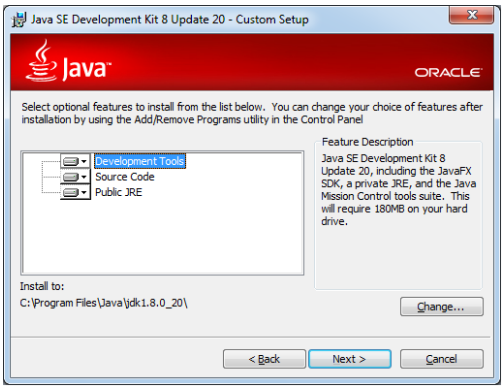
2) Instalación del jdk

Cuando haya descargado pulsamos doble clik para ejecutarlo, es probable que te pregunte si deseas permitir que el programa realice cambios en el equipo, escoge si.

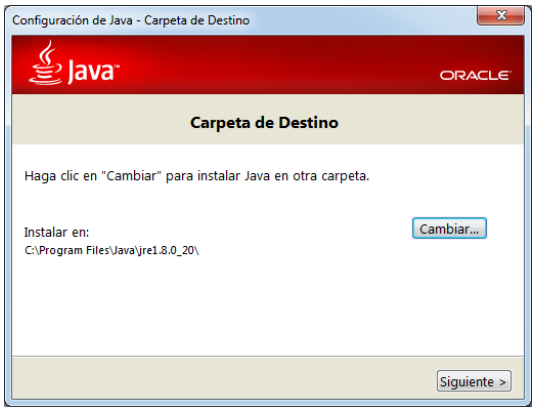
- Y damos clik en next



- Y damos clik en next , a tener en cuenta que si quieres cambiar la carpeta del jdk este es el paso indicado para hacer el cambio.



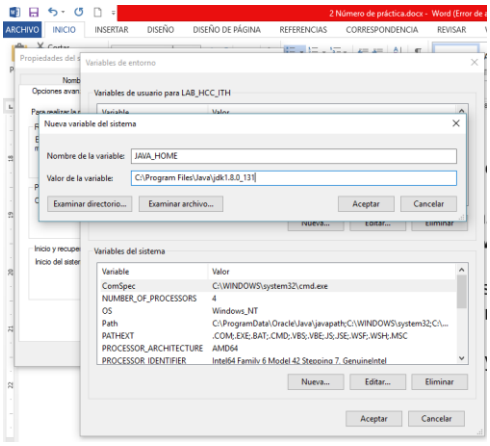
- Y damos clik en next, a tener en cuenta que si quieres cambiar la carpeta del jre este es el paso indicado para hacer el cambio.



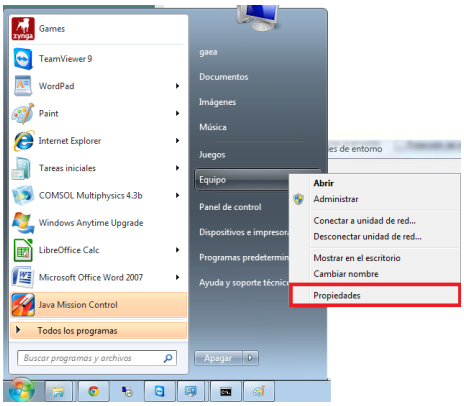
3. Configuración de variables de entorno.

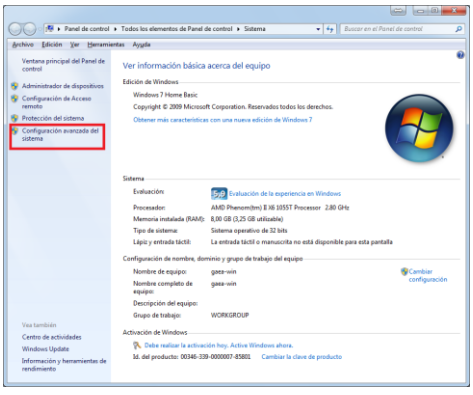
- Vamos a inicio->equipo y damos click derecho propiedades.
- Seleccionamos **Sistema**
- Vamos a la opción de **configuración avanzada del sistema**.

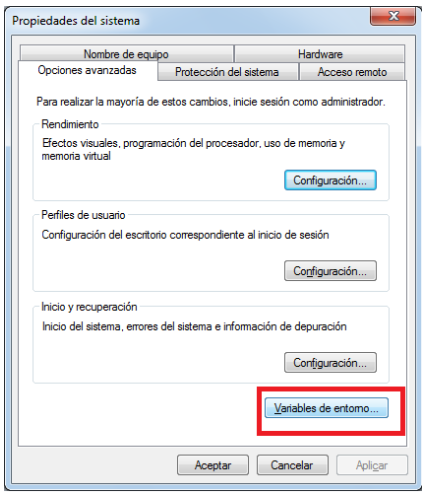
- En propiedades del sistema ->**opciones avanzadas** damos click en variables de entorno.
- Pulsamos en el botón nueva, y agregamos JAVA_HOME con la ruta de instalación que se haya configurado en el paso 2 en mi caso sería C:\Program Files\Java\jdk1.8.0_131

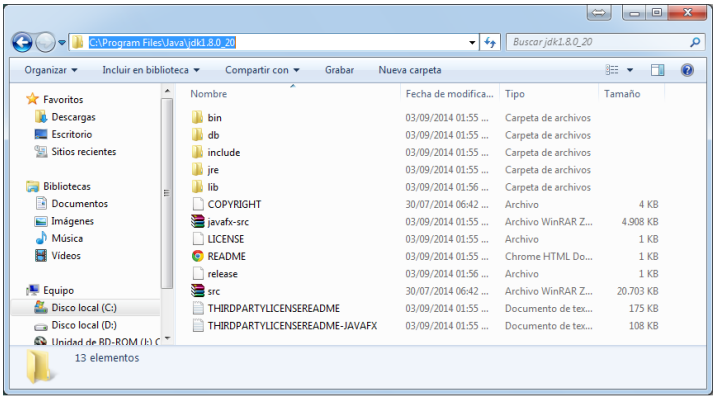


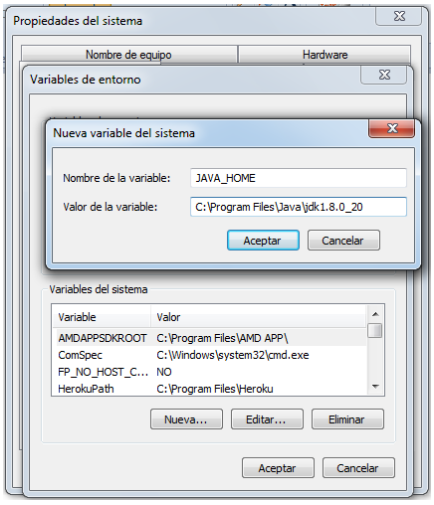
- En la misma ventana de variables de entorno buscamos la que diga Path y agregamos al final lo siguiente ;%JAVA_HOME%\bin
- Luego abrimos una terminal de comandos y revisamos que el sistema permita la ejecución del comando sin problemas. ejecutamos el comando "java-version"

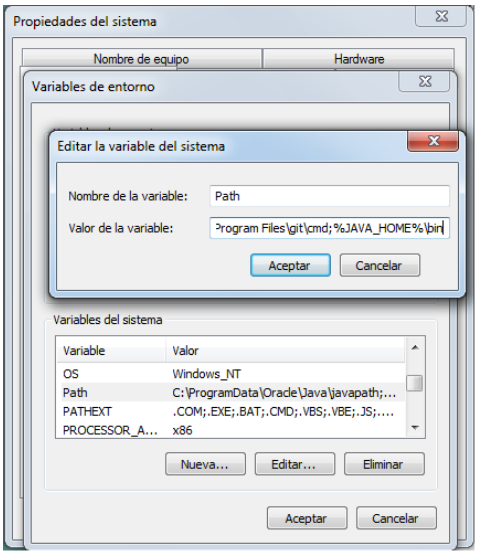


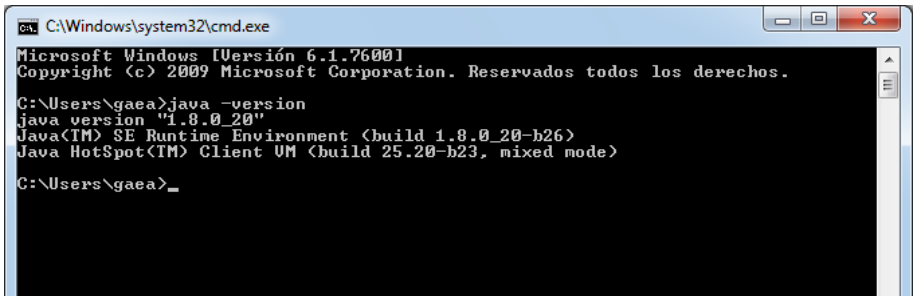



Propiedades del sistema
Nombre de equipo
Hardware
Opciones avanzadas
Protección del sistema
Acceso remoto
Para realizar la mayoría de estos cambios, inicie sesión como administrador.
Rendimiento
Efectos visuales, programación del procesador, uso de memoria y memoria virtual
Configuración...
Perfiles de usuario
Configuración del escritorio correspondiente al inicio de sesión
Configuración...
Inicio y recuperación
Inicio del sistema, errores del sistema e información de depuración
Configuración...
Variables de entorno...
Aceptar
Cancelar
Aplicar

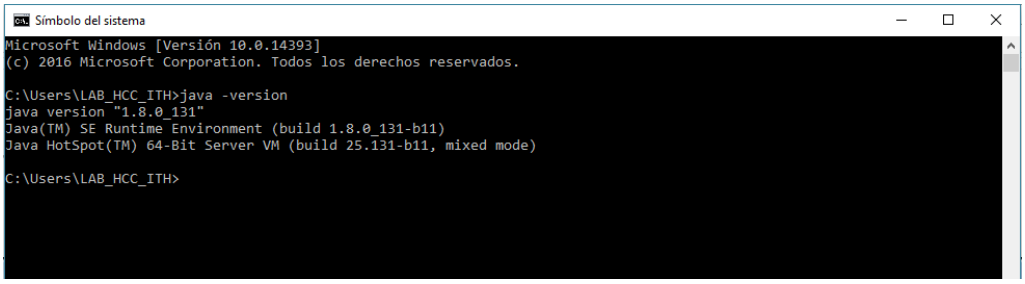

C:\Program Files\Java\jdk1.8.0_20
Buscar jdk1.8.0_20
Organizar
Incluir en biblioteca
Compartir con
Grabar
Nueva carpeta
Favoritos
Descargas
Escritorio
Sitios recientes
Bibliotecas
Documentos
Imágenes
Música
Vídeos
Equipo
Disco local (C:)
Disco local (D:)
Nombre
Fecha de modifica...
Tipo
Tamaño
bin
db
include
jre
lib
COPYRIGHT
javafx-src
LICENSE
README
release
src
THIRDPARTYLICENSEREADME
THIRDPARTYLICENSEREADME-JAVAFX
13 elementos


Propiedades del sistema
Nombre de equipo
Hardware
Variables de entorno
Nueva variable del sistema
Nombre de la variable:
JAVA_HOME
Valor de la variable:
C:\Program Files\Java\jdk1.8.0_20
Aceptar
Cancelar
Variables del sistema
Variable
Valor
AMDAPPSDKROOT
C:\Program Files\AMD APP\
ComSpec
C:\Windows\system32\cmd.exe
FP_NO_HOST_C...
NO
HerokuPath
C:\Program Files\Heroku
Nueva...
Editar...
Eliminar
Aceptar
Cancelar

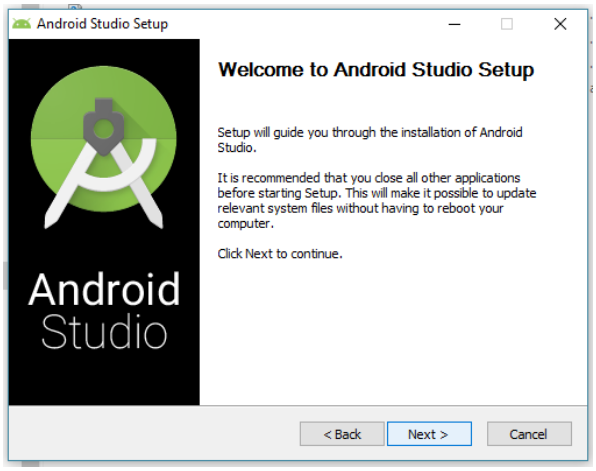




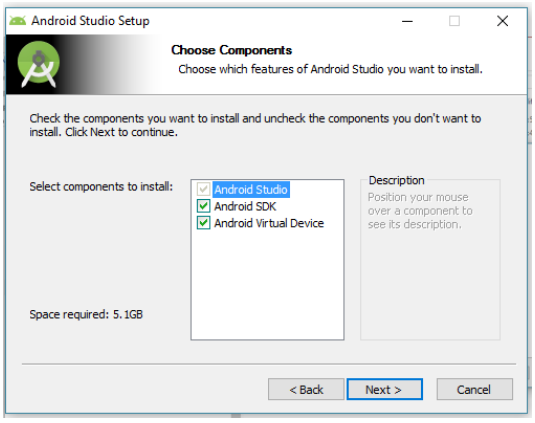
Y Listo terminamos.



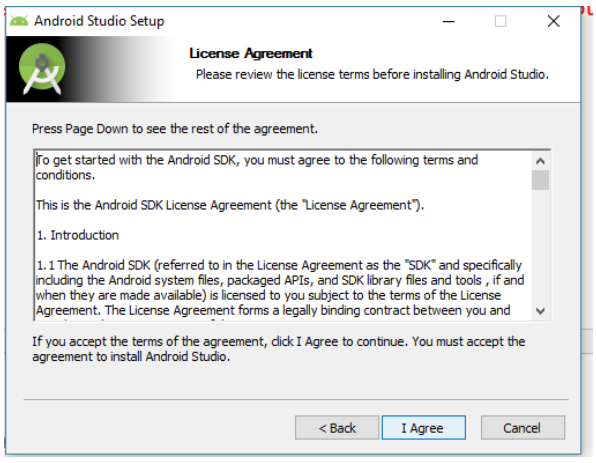
En la práctica no. 1 descargamos el IDE, abrir la carpeta donde se descargó el archivo y darle doble clic para ejecutar. Aparecerá un asistente de instalación, damos clic en el botón Next.



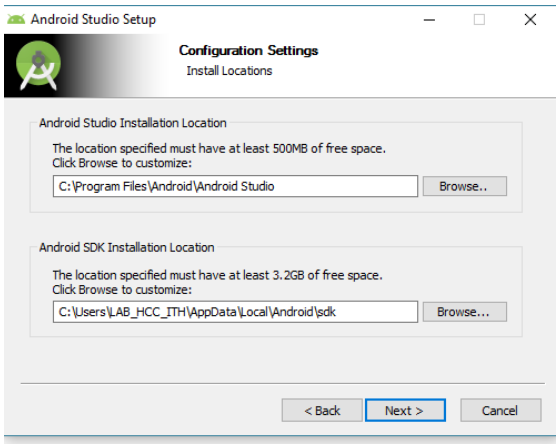
Seleccionar de la lista las opciones Android SDK y Android Virtual Device. Dar clic en el botón Next



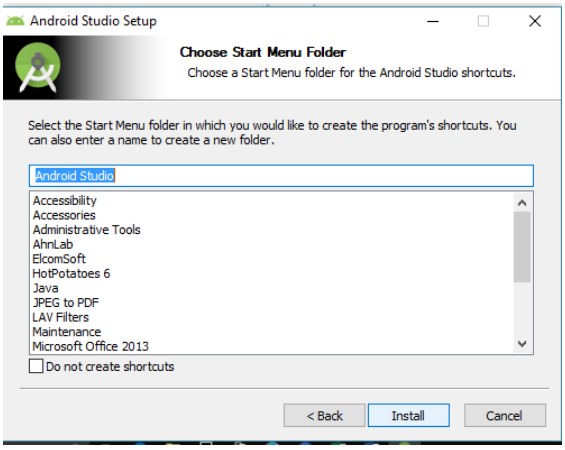
Leer y aceptar los términos y condiciones. Dar clic en el botón I Agree.



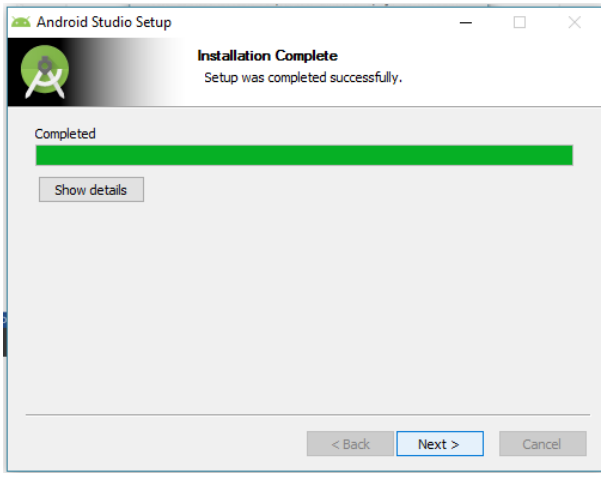
Seleccionar la ubicación de instalación de Android Studio y del SDK de Android (Se recomienda no cambiar la ubicación), dar clic en Next.



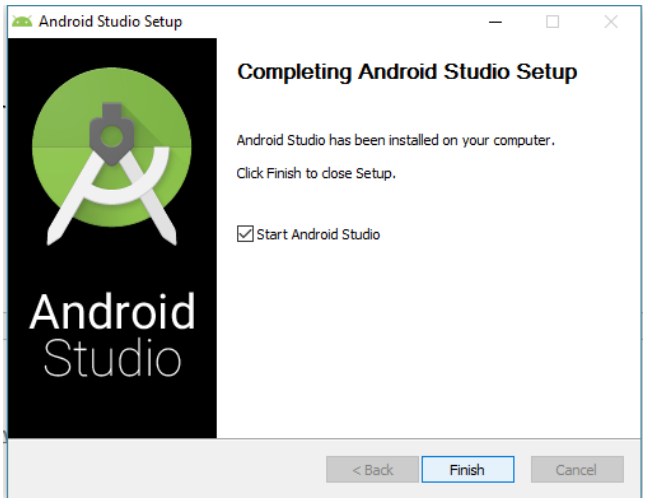
Una vez configurado lo anterior se procederá con la instalación. Dar clic en el botón Install.



Esperar a que la instalación se lleve a cabo, esto puede llevar varios minutos. Terminado el proceso dar clic en el botón Next.



Y finalizar el asistente



Requisitos del sistema

Windows:	- Microsoft Windows 10/8/7/Vista/2003 (32 o 64 bits). 2 GB en RAM, 4 GB recomendado. - 400 MB de espacio en disco duro para el IDE Android Studio - Al menos 1 GB de espacio en disco duro para el SDK de Android. - 1280 x 800 resolución de pantalla como mínimo. - Oracle Java Development Kit (JDK) 7 o superior
Mac OS X:	- Mac OS X 10.8.5 o superior. - 2 GB en RAM, 4 GB recomendado. - 400 MB de espacio en disco duro para el IDE Android Studio - Al menos 1 GB de espacio en disco duro para el SDK de Android. - 1280 x 800 resolución de pantalla como mínimo. - Oracle Java Runtime Environment (JRE) 6 o superior. - Oracle Java Development Kit (JDK) 7 o superior
Mac OS	Android Studio corre con Java Runtime Environment (JRE) 6 para la renderización de fuentes optimizado. El proyecto puede ser configurado para utilizar Java Development Kit (JDK) 6 o 7
Linux:	- GNOME o KDE Desktop - GNU C Library (glibc) 2.15 o superior - 2 GB en RAM, 4 GB recomendado. - 400 MB de espacio en disco duro para el IDE Android Studio - Al menos 1 GB de espacio en disco duro para el SDK de Android. - 1280 x 800 resolución de pantalla como mínimo. - Oracle Java Development Kit (JDK) 7 o superior
Ubuntu 14.04 o superior	- Probando

Sugerencias didácticas.

- Investigar como acelerar el emulador del SDK de android

Reporte del alumno (discusión de resultados y conclusiones).

En un documento PDF, envía las imágenes de los pasos realizado de las descargas de las Herramientas de software. Capturas de pantallas

Criterio para calificar	Sí	No
Variables de Entorno de Java	1 punto	0 punto
Ejecutar Andriod	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Omnius soluciones en tecnología. (2016). *Http://eduomniuscommx/*. Retrieved 4 August, 2017, from

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #3

CONFIGURAR S.O.

Competencia(s) a desarrollar.

Conocer y aplicar los diferentes sistemas operativos, arquitecturas y entornos de programación para el desarrollo de aplicaciones móviles**.**

Introducción

Una vez instalado el entorno de desarrollo se deben instalar y actualizar las plataformas para poder desarrollar plenamente las aplicaciones para Android. En esta práctica llevaremos a cabo esta actualización**.**

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.

Esta práctica está directamente relacionados con la unidad 2.

El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio

Material y equipo necesario

Equipo Android. Uno de los puntos importantes que se recomienda, es tener es un teléfono que corra con Android. Esto es porque de esta manera se tendrá una auténtica forma de probar cómo es que corren las aplicaciones en un ambiente real, además de que existen algunos ejemplos prácticos que por sus características es altamente recomendable ejecutarlos en un dispositivo físico

Metodología

Actualizar el IDE y las herramientas del SDK

En este documento:

1. Actualizar tu IDE y cambiar canales
2. Actualizar tus herramientas con SDK Manager
    a) Paquetes recomendados
    b) Editar o agregar sitios de herramientas del SDK

Una vez que instalas Android Studio, es fácil mantener actualizados el IDE de Android Studio y las herramientas de Android SDK con actualizaciones automáticas y Android SDK Manager.

Actualizar tu IDE y cambiar canales

Android Studio te notifica con un cuadro de diálogo pequeño cuando hay una actualización disponible para el IDE, pero puedes comprobar manualmente la

disponibilidad de actualizaciones haciendo clic en **Help** > **Check for Update** (en Mac, **Android Studio** > **Check for Updates**).

Las actualizaciones para Android Studio están disponibles a través de los siguientes canales para versiones:

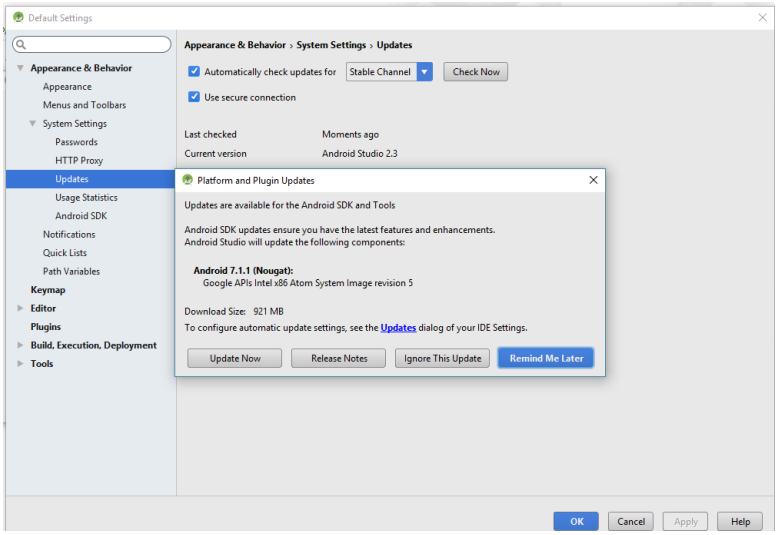
- **Canal Canary**: versiones de vanguardia que se actualizan semanalmente. Si bien estas compilaciones están sujetas a más errores, se someten a prueba y nuestro deseo es ofrecer acceso anticipado para que puedas probar nuevas funciones y proporcionar comentarios. Este canal **no se recomienda para desarrollo de producción**.
- **Canal Dev**: compilaciones de Canary seleccionadas que superaron una ronda completa de pruebas internas.
- **Canal Beta**: candidatos a lanzamiento basados en compilaciones de Canary estables que se lanzan para recibir comentarios antes de pasar al canal estable.
- **Canal estable**: versión estable oficial que está disponible para la descarga en developer.android.com/studio.

De forma predeterminada, Android Studio ofrece actualizaciones desde el canal estable. Si deseas probar una de las demás versiones de Android Studio, conocidas en conjunto como canales de vista previa, puedes optar por recibir actualizaciones desde uno de ellos.

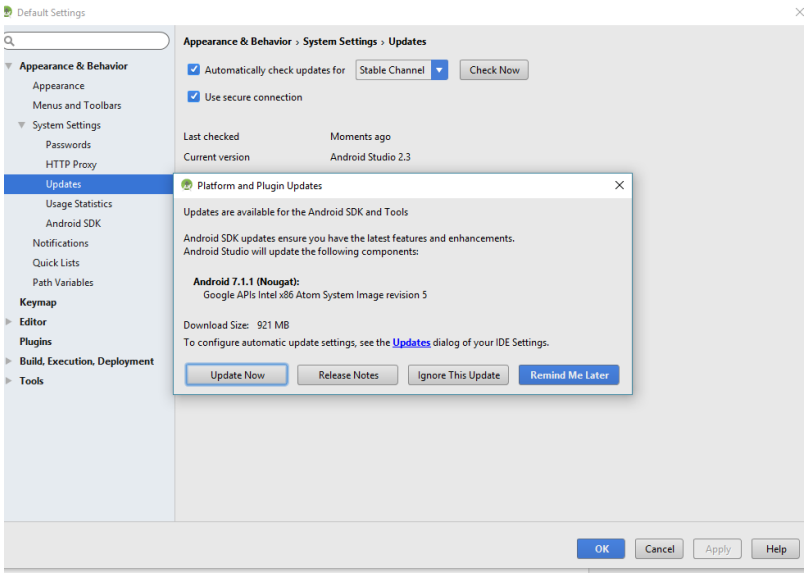
Para cambiar tu canal de actualización, haz lo siguiente:

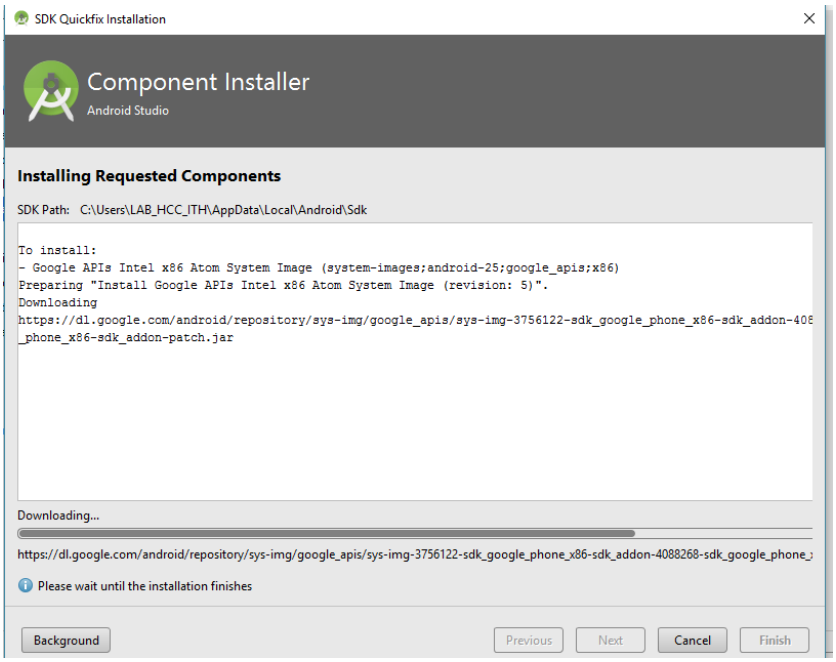
1. Abre la ventana **Preferences** haciendo clic en **File > Settings** (en Mac, **Android Studio > Preferences**).
2. En el panel izquierdo, haz clic en **Appearance & Behavior > System Settings > Updates**.
3. Asegúrate de que **Automatically check for updates** esté seleccionado y luego selecciona un canal en la lista desplegable (consulta la figura 1).
4. Haz clic en **Apply** u **OK**.

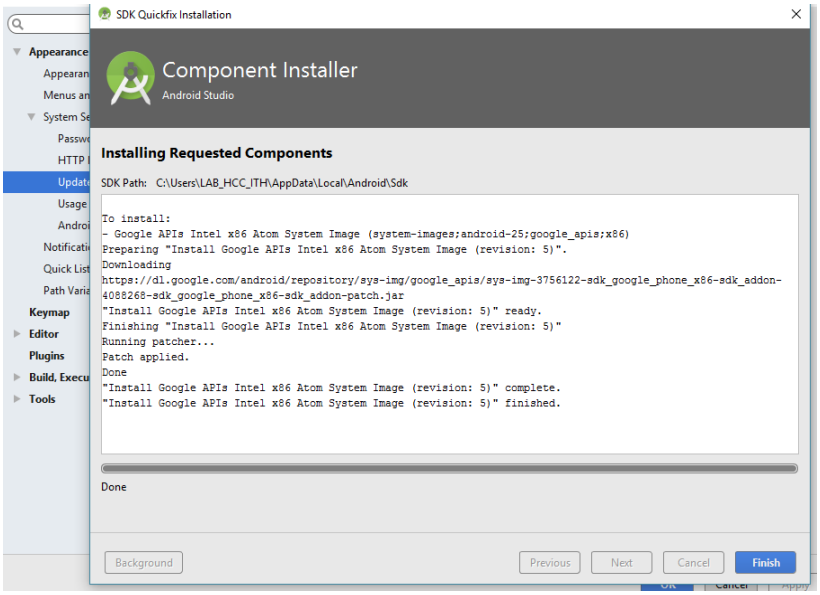
Figura 1: Preferencias de actualización de Android Studio.



Seleccionas Update Now



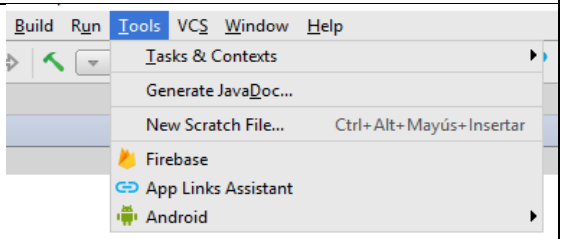
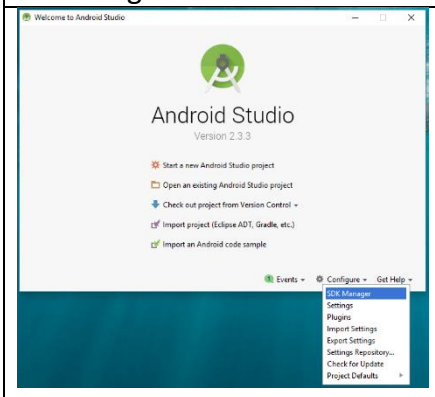
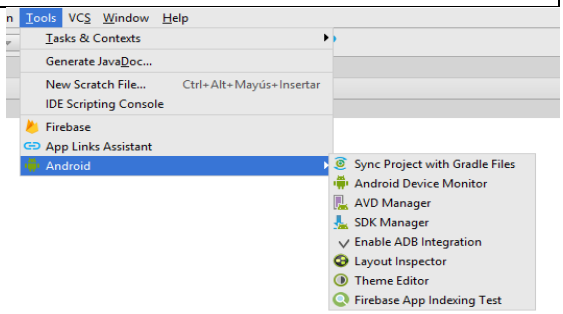




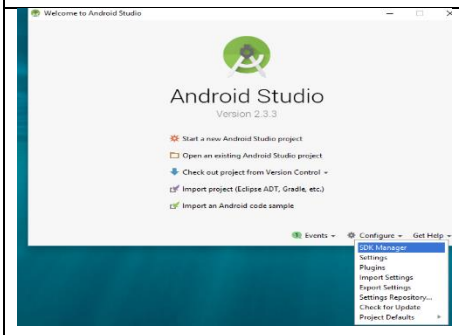
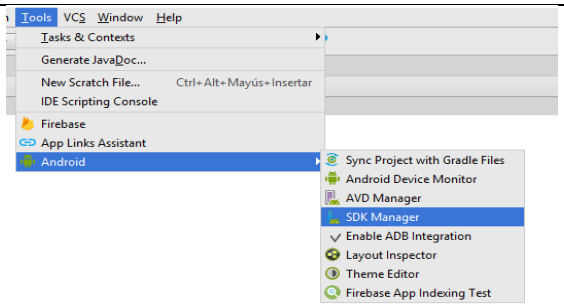
Si deseas probar uno de los canales de vista previa (Canary, Dev o Beta) mientras usas la compilación estable para tus proyectos de producción de Android, puedes instalar de forma segura una segunda versión de Android Studio descargando la compilación de vista previa desde tools.android.com.

Para configurar SDK manager:

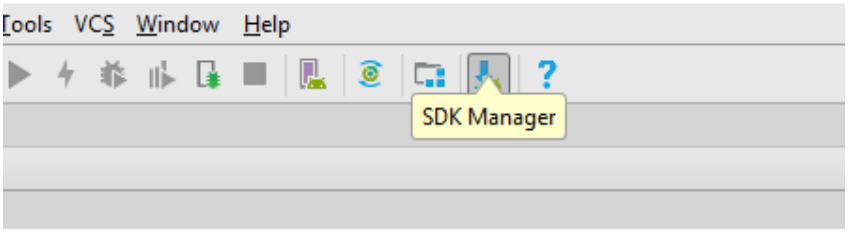
1. Darle click al icono Android Studio.
2. Aparecerá lo siguiente:

Opciones 1	Opciones 2
	
Opciones 1	Opciones 2
3. Le damos click a la opción de Configure:	3. Ir al apartado Android
	

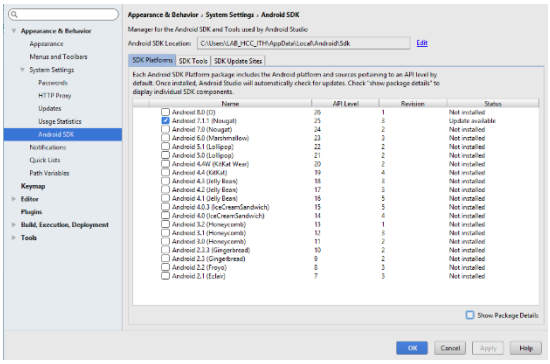
4. Seleccionamos SDK Manager

Opciones 1	Opciones 2
	

O También se puede acceder directamente al SDK Manager presionando el ícono que se encuentra en la barra de herramientas de Android Studio

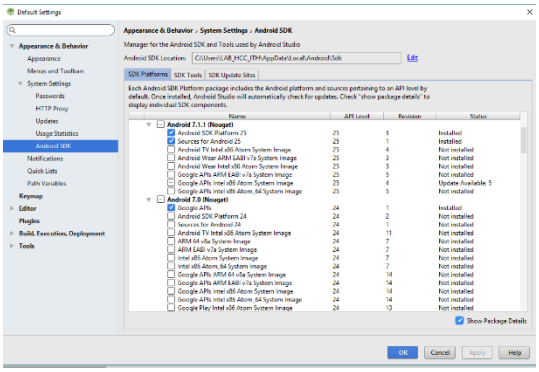


5. Al entrar al SDK Manager te muestra los paquetes instalados y los que puedes instalar o actualizar



En la lengüeta *SDK Plataforms* se muestran los paquetes de plataforma. Pulsa en *Show Package Details* para ver los diferentes paquetes. Siempre es conveniente que tengas instalados los siguientes paquetes de la última plataforma disponible:

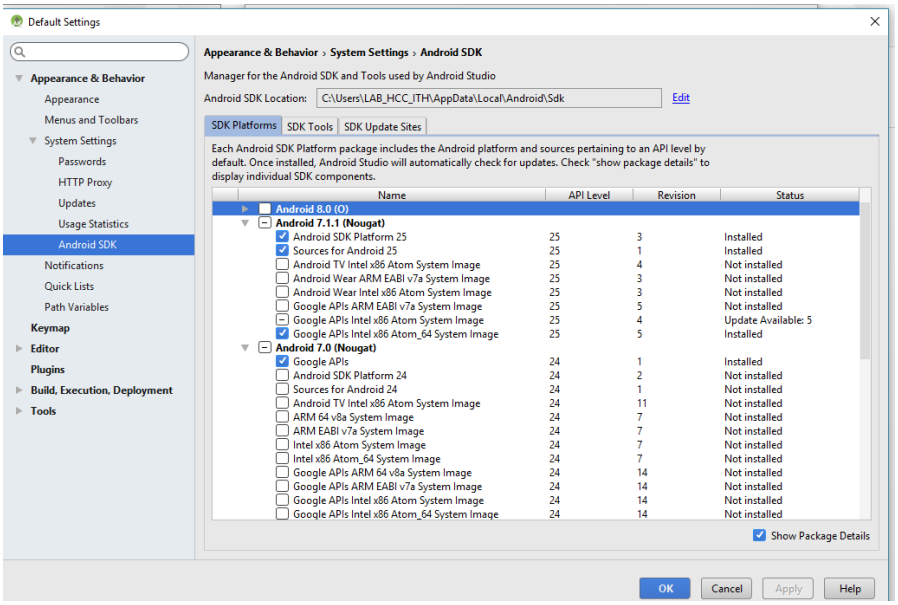
- *Android X.X Platform*
- *Google APIs, Android X*
- Alguno de los *System Image (para crear emuladores)*



En la lengüeta *SDK Tools* se muestran paquetes con herramientas de la plataforma. Siempre es conveniente que tengas actualizados los siguientes paquetes:

- *Android SDK Build-tools*
- *Android SDK Platform-tools*
- *Android SDK Tools*

- *Android Support Repository*
- *Google Play services*



Dependiendo del número de plataformas y componentes que se decida descargar, este proceso puede tardar varios minutos.

Sugerencias didácticas.

Investiga sobre Sistema Operativo Android: Características Y Funcionalidad para Dispositivos Móviles

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Entrega de la investigación	1 punto	0 punto
El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto

Bibliografía

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #4

ENTORNO DE DESARROLLO EMULADORES A UTILIZAR

Competencia(s) a desarrollar.

Desarrollar aplicaciones móviles nativas, web e híbridas para atender las necesidades del entorno.

Introducción

Un dispositivo virtual Android (AVD) te va a permitir emular en tu ordenador diferentes tipos de dispositivos basados en Android. De esta forma podrás probar tus aplicaciones en una gran variedad de teléfonos, tabletas, relojes o TV con cualquier versión de Android, tamaño de pantalla o tipo de entrada.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

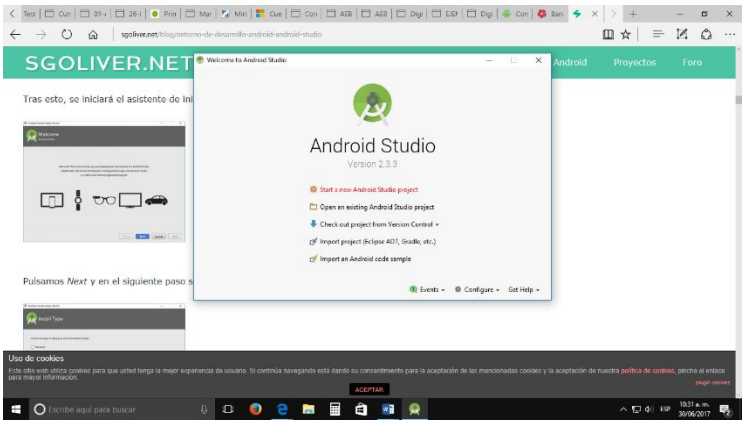
Esta práctica está directamente relacionados con la unidad 2.

El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio

Material y equipo necesario

PC o Laptop
Android instalado y configurado
Internet

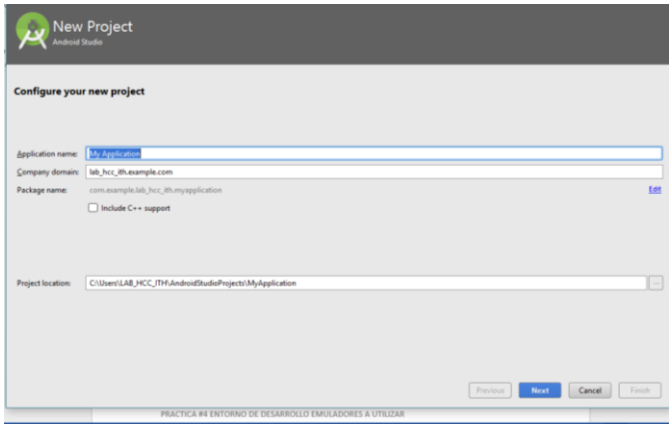
Metodología



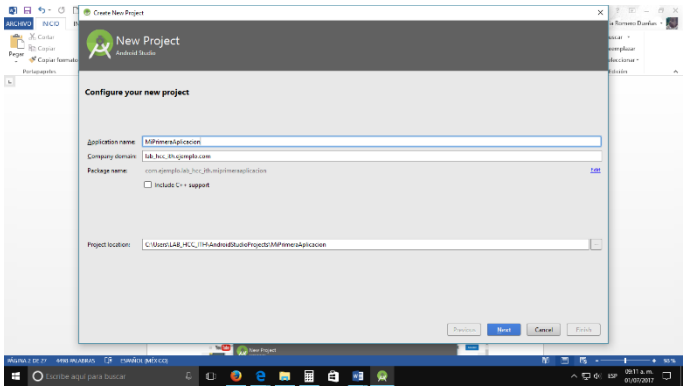
En la primera pantalla indicaremos, por este orden:

- El nombre de la aplicación,
- El dominio de la compañía
- La ruta donde crear el proyecto.

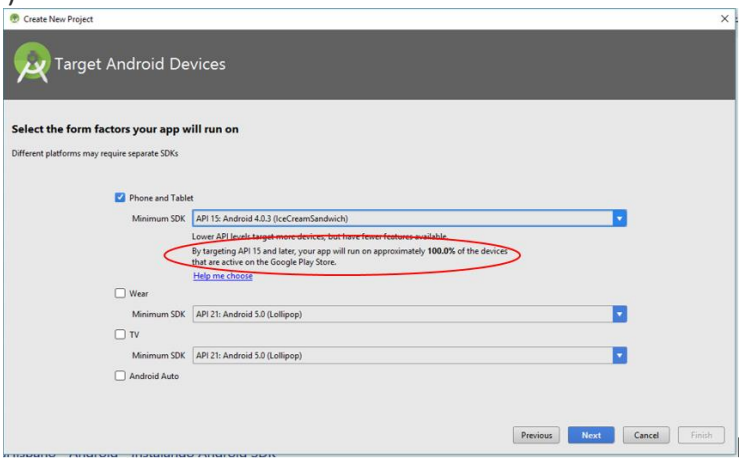
El segundo de los datos indicados tan sólo se utilizará como paquete de nuestras clases java. En el cual puedes utilizar cualquier otro dominio.



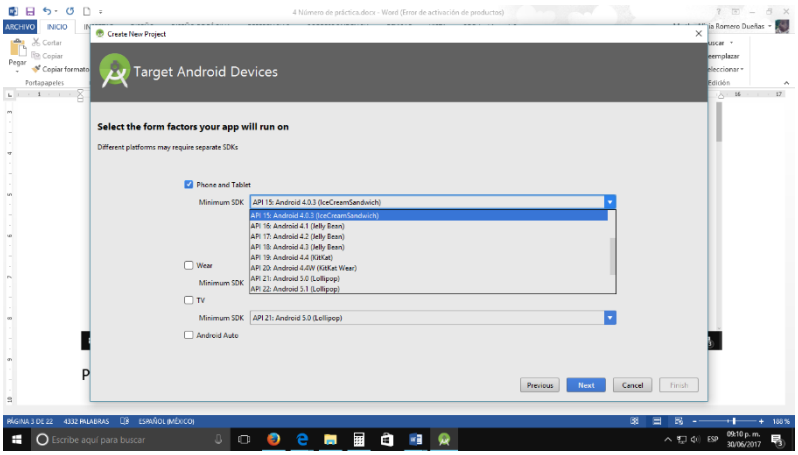
Por ejemplo:



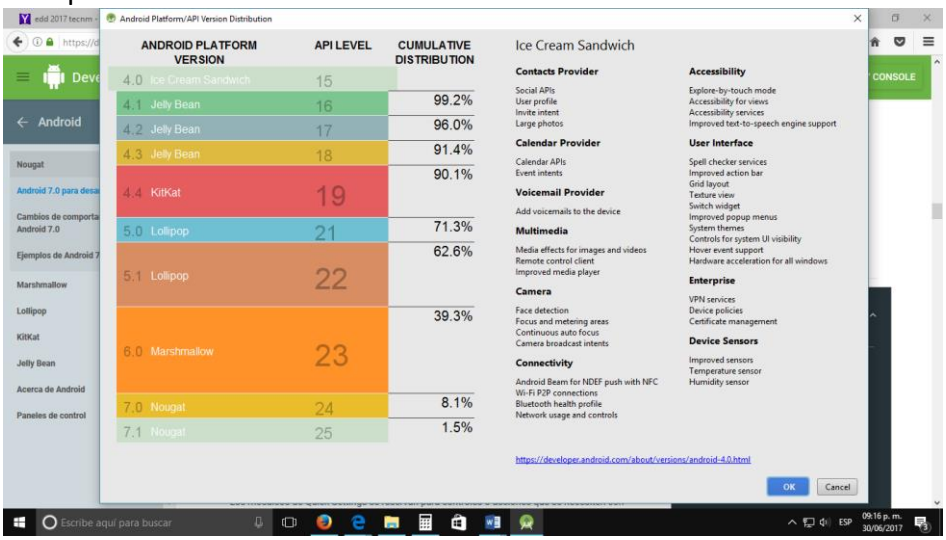
Crear un AVD Crear y configurar un Dispositivo Virtual Android (Android Virtual Device - AVD)



Al seleccionar la API 15 y posterior, tu aplicación se ejecutará en aproximadamente el 100% de los dispositivos que están activos en Google Play Store
Seleccionamos que APIs (las APIs son las versiones del sistema operativo de Android) o SDK vamos a utilizar



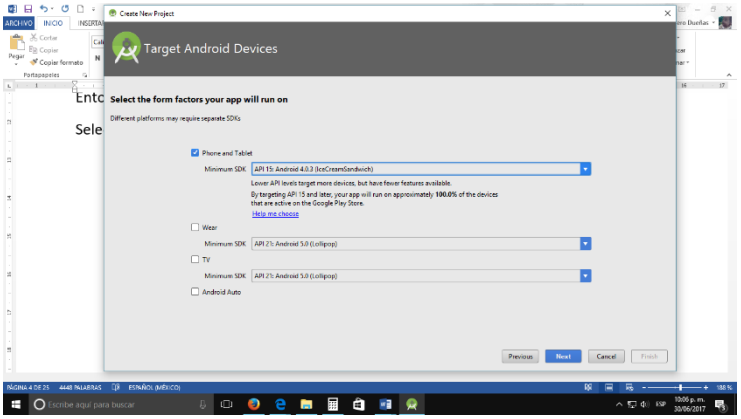
Podemos seleccionar Help me choose, si no sabemos que escoger y tendremos la siguiente pantalla:

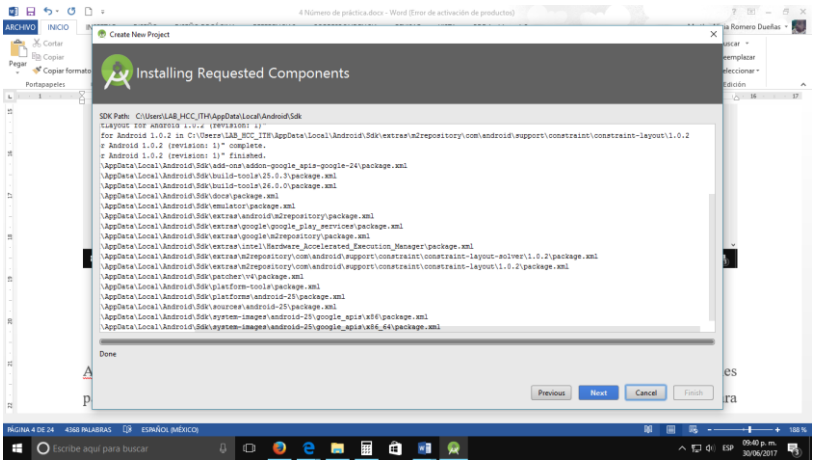


Si yo llegase a seleccionar el nivel de la API, 21 para la version 5.0 Lollipop únicamente estaría disponible para el 71.3% de las personas que usan Android Studio, de la versión 5.0 para adelante no hacia atrás.

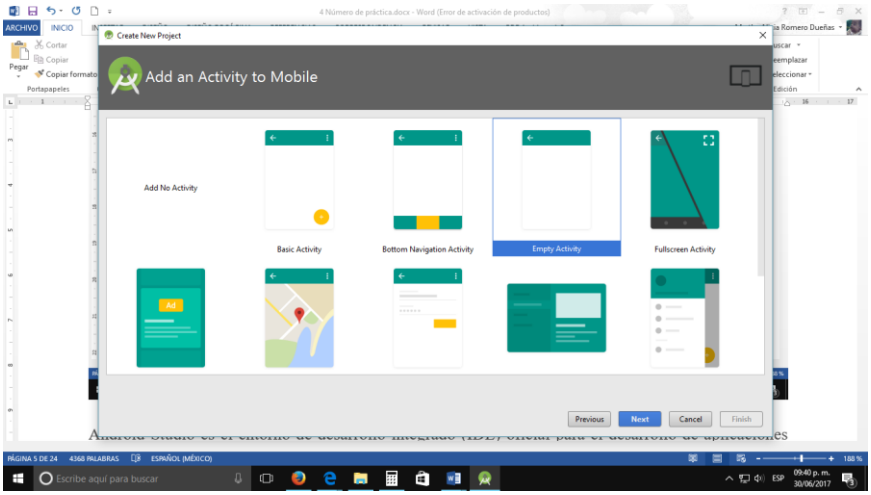
Entonces sería muy bueno escoger una nivel de API 15 o 16..

Seleccionamos en este caso seleccionaremos

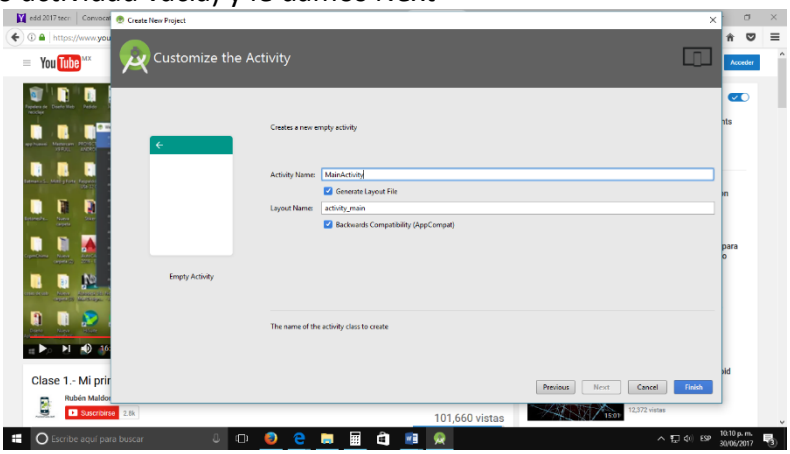




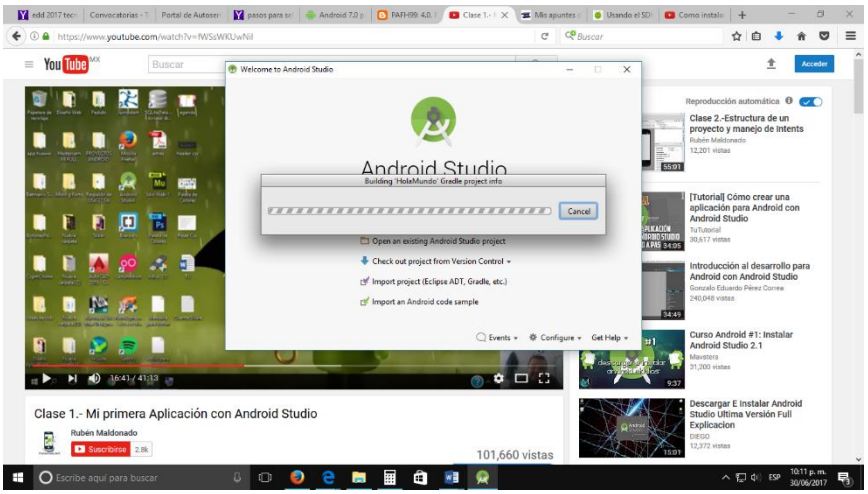
Y al terminar de instalar damos next



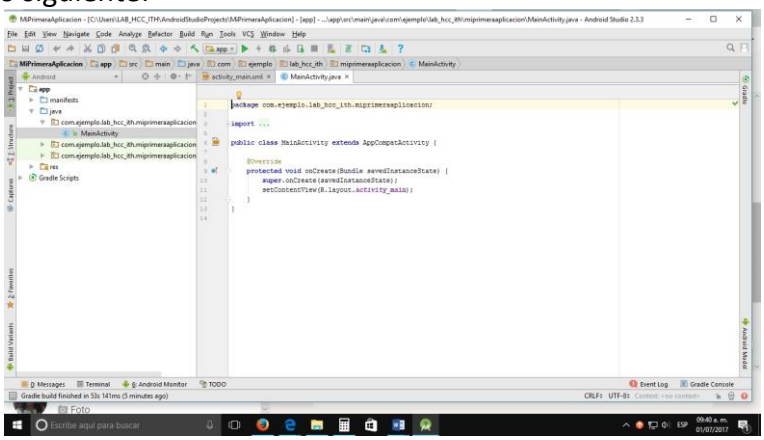
Nos aparecen varias aplicaciones de Activity (plantilla para la pantalla de tu móvil, que es el diseño) y seleccionamos la que deseemos en este caso selecionare Empty Activity (o actividad vacia) y le damos Next



Aquí puedes poner el Nombre de la Actividad y tu Layout por ahora lo dejaremos tal cual y seleccionamos Finish



Aquí los que está haciendo es Creando el Proyecto cuando ya haya realizado todo tenemos lo siguiente:



Cerramos la ventana del archivo de java

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA . Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

- Un sistema de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba

- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine

Estructura del proyecto

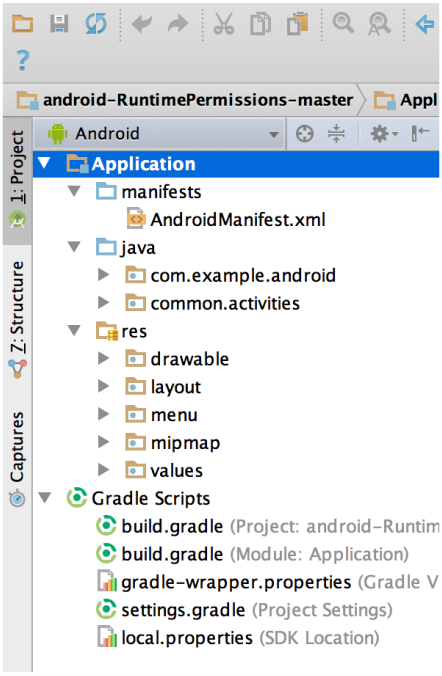


Figura 1. Archivos del proyecto en la vista de Android.

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- módulos de apps para Android
- módulos de bibliotecas
- módulos de Google App Engine

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android, como se muestra en la figura 1. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de **Secuencias de comando de Gradle** y cada módulo de la aplicación contiene las siguientes carpetas:

- **manifests**: contiene el archivo AndroidManifest.xml.
- **java**: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- **res**: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

La estructura del proyecto para Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, selecciona **Project** en la lista desplegable **Project** (en la figura 1 se muestra como **Android**).
También puedes personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu app. Por ejemplo, al seleccionar la vista **Problems** de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante para un elemento XML en un archivo de diseño.

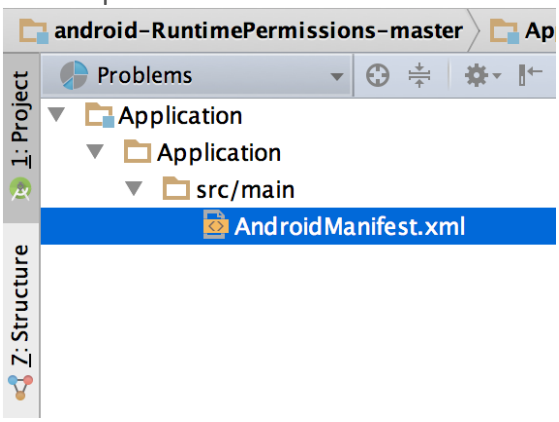


Figura 2. Archivos del proyecto en la vista Problems, en la que se muestra un archivo de diseño con un problema.

Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican en la figura 3.

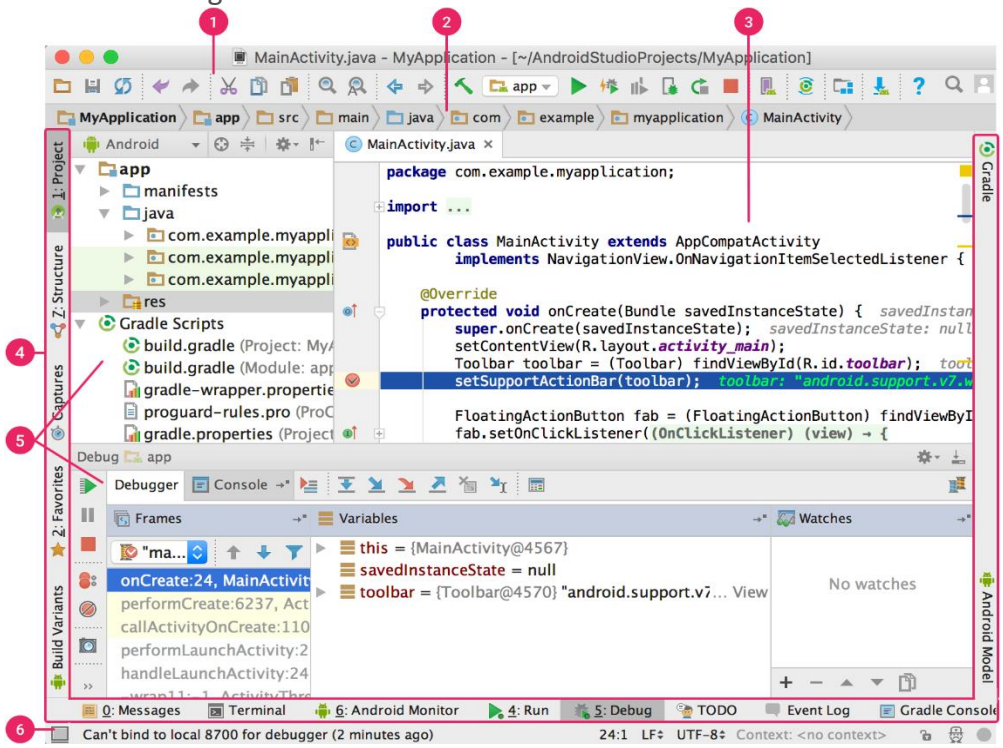


Figura 3. Ventana principal de Android Studio.

1. La **barra de herramientas** te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.
2. La **barra de navegación** te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana **Project**.
3. La **ventana del editor** es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.
4. La **barra de la ventana de herramientas** se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.
5. Las **ventanas de herramientas** te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.
6. En la **barra de estado**, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje.
7. Puedes organizar la ventana principal para tener más espacio en pantalla ocultando o desplazando barras y ventanas de herramientas. También puedes usar combinaciones de teclas para acceder a la mayoría de las funciones del IDE.
8. En cualquier momento, puedes realizar búsquedas en tu código fuente, bases de datos, acciones, elementos de la interfaz de usuario, etc., presionando dos veces la tecla Shift o haciendo clic en la lupa que se encuentra en la esquina superior derecha de la ventana de Android Studio. Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar.
9. Ventanas de herramientas

En lugar de usar perspectivas preestablecidas, Android Studio sigue tu contexto y te ofrece automáticamente ventanas de herramientas relevantes mientras trabajas. De forma predeterminada, las ventanas de herramientas usadas con mayor frecuencia se fijan en la barra de ventanas de herramientas en los bordes de la ventana de la aplicación.

- Para expandir o contraer una ventana de herramientas, haz clic en el nombre de la herramienta, en la barra de la ventana de herramientas. También puedes arrastrar, anclar, desanclar, adjuntar y ocultar ventanas de herramientas.
- Para volver al diseño predeterminado actual de la ventana de herramientas, haz clic en **Window > Restore Default Layout** o personaliza tu diseño predeterminado haciendo clic en **Window > Store Current Layout as Default**.

- Para mostrar u ocultar la barra de la ventana de herramientas completa, haz clic en el ícono de ventana  en la esquina inferior izquierda de la ventana de Android Studio.
- Para localizar una ventana de herramientas específica, coloca el puntero sobre el ícono de ventana y selecciona la ventana de herramientas en el menú.

También puedes usar combinaciones de teclas para abrir ventanas de herramientas. En la Tabla 1 se muestran las combinaciones de teclas para las ventanas más comunes.

Tabla 1. Combinaciones de teclas para algunas ventanas de herramientas útiles.

Ventana de herramientas	Windows y Linux	Mac
Project	**Alt+1**	**Comando+1**
Version Control	**Alt+9**	**Comando+9**
Run	**Mayús+F10**	**Control+R**
Debug	**Mayús+F9**	**Control+D**
Android Monitor	**Alt+6**	**Comando+6**
Return to Editor	**Esc**	**Esc**
Hide All Tool Windows	**Control+Mayús+F12**	**Comando+Mayús+F12**

Si quieres ocultar todas las barras de herramientas, ventanas de herramientas y pestañas del editor, haz clic en **View > Enter Distraction Free Mode**. Esto habilita el modo *Distraction Free Mode*. Para salir del modo Distraction Free Mode, haz clic en **View > Exit Distraction Free Mode**.

Puedes usar la *Búsqueda rápida* para buscar y aplicar filtros en la mayoría de las ventanas de herramientas en Android Studio. Para usar la búsqueda rápida, selecciona la ventana de herramientas y luego escribe el texto de tu búsqueda. Para obtener más sugerencias, consulta Combinaciones de teclas.

Compleción de código

Android Studio ofrece tres opciones para completar código, a las que puedes acceder usando combinaciones de teclas.

Tabla 2. Combinaciones de teclas para completar código.

Tipo	Descripción	Windows y Linux	Mac
Compleción básica	Muestra sugerencias básicas para variables, tipos, métodos y expresiones, entre otros. Si llamas a la compleción	**Control+Espacio**	**Control+Espacio**

	básica dos veces seguidas, verás más resultados. Entre otros, miembros privados y miembros estáticos sin importar.		
Compleción inteligente	Muestra opciones relevantes en función del contexto. La compleción inteligente reconoce el tipo y los flujos de datos previstos. Si llamas a la compleción inteligente dos veces seguidas, verás más resultados. Por ejemplo, cadenas.	**Control+Mayús +Espacio**	**Control+Mayús+Espacio**
Compleción de enunciados	Completa el enunciado actual por usted agregando elementos faltantes, como paréntesis, corchetes, llaves, formato, etc.	**Control+Mayús +Enter**	**Mayús+Comando+Enter**

También puedes realizar correcciones rápidas y mostrar acciones de intención presionando **Alt+Enter**.
Para obtener más información sobre la compleción de código, consulta

Compleción de código.

Búsqueda de ejemplos de código

El navegador de ejemplos de código de Android Studio te ayuda a buscar ejemplos de código de Android de alta calidad y proporcionados por Google según el símbolo actualmente destacado en tu proyecto. Para obtener más información, consulta

Búsqueda de ejemplos de código.

Navegación

Aquí te ofrecemos algunas sugerencias para ayudarte a recorrer Android Studio.

- Alterna entre los archivos a los que accediste recientemente mediante la acción *Recent Files* . Presiona **Control+E** (**Comando+E** en una Mac) para activar la acción Recent Files. De forma predeterminada, se selecciona el último archivo al que accediste. También puedes acceder a cualquier ventana de herramientas a través de la columna izquierda en esta acción.
- Visualiza la estructura del archivo actual utilizando la acción *File Structure*. Activa la acción File Structure presionando **Control+F12** (**Comando+F12** en una

Mac). Con esta acción, podrás navegar rápidamente hacia cualquier parte del archivo actual.

- Busca una clase específica en tu proyecto y navega hacia ella con la acción *Navigate to Class*. Activa la acción presionando **Control+N** (**Comando+O** en una Mac). Navigate to Class admite expresiones sofisticadas, como jorobas de camellos, rutas de acceso, la línea de navegar a y coincidencia de segundo nombre, entre otras. Si la llamas dos veces seguidas, te mostrará los resultados fuera de las clases de proyectos.
- Navega hasta un archivo o carpeta utilizando la acción *Navigate to File*. Activa la acción Navigate to File presionando **Control+Shift+N** (**Comando+Shift+O** en una Mac). Para buscar carpetas en lugar de archivos, agrega una "/" al final de la expresión.
- Navega hasta un método o campo por nombre utilizando la acción *Navigate to Symbol*. Activa la acción Navigate to Symbol presionando **Control+Shift+Alt+N** (**Comando+Shift+Alt+O** en una Mac).
- Busca todas las partes de código que hagan referencia a la clase, el método, el campo, el parámetro o el enunciado en la posición actual del cursor presionando **Alt+F7**.

Estilo y formato

Mientras editas, Android Studio aplica automáticamente formatos y estilos según lo especificado en tu configuración de estilo de código. Puedes personalizar la configuración de estilo de código programando el idioma, que incluye la especificación de convenciones para pestañas y sangrías, espacios, ajuste y llaves, y líneas en blanco. Para personalizar la configuración de estilo de tu código, haz clic en **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** en una Mac.)

Si bien el IDE aplica formato automáticamente mientras trabajas, también puedes llamar explícitamente a la acción *Reformat Code* presionando **Control+Alt+L** (**Opt+Comando+L** en una Mac), o aplicar sangrías automáticas a todas las líneas presionando **Control+Alt+I** (**Alt+Option+I** en una Mac).

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
        mActionBar.setDisplayHomeAsUpEnabled(true);
```

Figura 5. Código antes de la aplicación del formato.

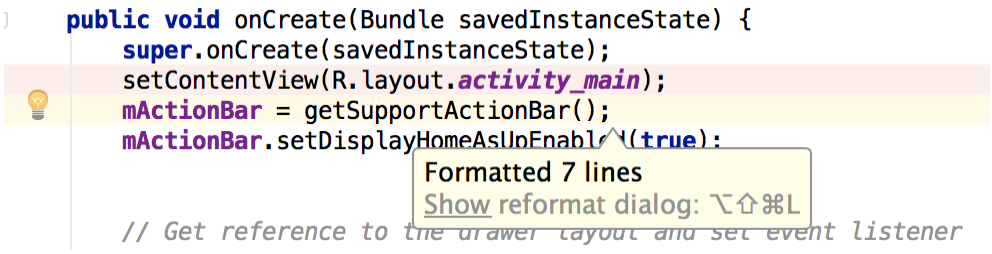


Figura 6. Código después de la aplicación del formato.

Aspectos básicos del control de versión

Android Studio admite diferentes sistemas de control de versión (VCS), incluidos Git, GitHub, CVS, Mercurial, Subversion y Google Cloud Source Repositories.

Después de importar tu app a Android Studio, usa las opciones del menú del VCS de Android Studio a fin de habilitar la compatibilidad con VCS para el sistema de control de versión deseado, crea un repositorio, importa los nuevos archivos al control de versión y realiza otras operaciones de control de versión:

1. En el menú **VCS** de Android Studio, haz clic en **Enable Version Control Integration**.
2. En el menú desplegable, selecciona un sistema de control de versión para asociarlo con la raíz del proyecto y luego haz clic en **OK**.
3. En el menú del VCS ahora se muestran diversas opciones de control de versión según el sistema que hayas seleccionado.

Nota: También puedes usar la opción del menú **File > Settings > Version Control** para configurar y modificar los ajustes de control de versión.

Sistema de compilación de Gradle

Android Studio usa Gradle como la base del sistema de compilación, con más capacidades específicas de Android a través del complemento de Android para Gradle. Este sistema de compilación se ejecuta en una herramienta integrada desde el menú de Android Studio, y lo hace independientemente de la línea de comandos. Puedes usar las funciones del sistema de compilación para lo siguiente:

- personalizar, configurar y extender el proceso de compilación;
- crear múltiples APK para tu app, con diferentes funciones utilizando el mismo proyecto y los mismos módulos;
- volver a usar códigos y recursos entre conjuntos de archivos de origen.

Recurriendo a la flexibilidad de Gradle, puedes lograr todo esto sin modificar los archivos de origen de tu app. Los archivos de compilación de Android Studio se denominan build.gradle. Son archivos de texto sin formato que usan la sintaxis Groovy para configurar la compilación con elementos proporcionados por el complemento de Android para Gradle. Cada proyecto tiene un archivo de compilación de nivel superior para todo el proyecto y archivos de compilación de

nivel de módulo independientes para cada módulo. Cuando importas un proyecto existente, Android Studio genera automáticamente los archivos de compilación necesarios.

Variantes de compilación
El sistema de compilación puede ayudarte a crear diferentes versiones de la misma aplicación a partir de un solo proyecto. Esto resulta útil cuando tienes una versión gratuita o una versión paga de tu app, o si quieres distribuir múltiples APK para diferentes configuraciones de dispositivos en Google Play.

Divisiones de APK
La división de APK te permite crear de forma eficiente varios APK en función de la densidad de la pantalla o ABI. Por ejemplo, la división de APK te permite crear versiones hdpi y mdpi independientes para una app sin dejar de considerarlas como una variante individual y permitiéndoles compartir la configuración de una app de prueba, javac, dx y ProGuard.

Reducción de recursos
La reducción de recursos en Android Studio elimina automáticamente los recursos sin usar del paquete de tu app y de las dependencias de bibliotecas. Por ejemplo, si en tu aplicación se usan servicios de Google Play para acceder a la funcionalidad de Google Drive y actualmente no usas Google Sign-In, la reducción de recursos puede eliminar los diferentes recursos de elemento de diseño de los botones SignInButton.
Nota: La reducción de recursos funciona con herramientas de reducción de código, como ProGuard.

Administración de dependencias
Las dependencias para tu proyecto se especifican por nombre en el archivo build.gradle. Gradle se ocupa de buscar tus dependencias y hacer que estén disponibles en tu compilación. Puedes declarar dependencias de módulos, dependencias binarias remotas y dependencias binarias locales en tu archivo build.gradle. Android Studio configura los proyectos para que usen el repositorio central de Maven de forma predeterminada. (Esta configuración está incluida en el archivo de compilación de nivel superior del proyecto).

Herramientas de depuración y perfil
Android Studio te ayuda a depurar y mejorar el rendimiento de tu código. Esto incluye herramientas integradas de depuración y análisis de rendimiento.
Depuración integrada
Usa la depuración integrada para mejorar las revisiones de código en la vista del depurador con verificación integrada de referencias, expresiones y valores de variables. La información de depuración integrada incluye:

+valores de variables integradas; + objetos que hacen referencia a un objeto seleccionado; + valores de retorno de métodos; + expresiones Lambda y de operador; + valores de información sobre la herramienta.

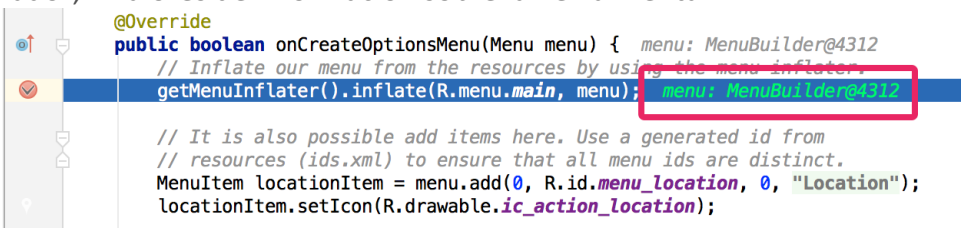


Figura 7. Valor de una variable integrada.

Para habilitar la depuración integrada, en la ventana **Debug**, haz clic en **Settings** y selecciona la casilla de verificación **Show Values Inline**.

Monitores de rendimiento

Android Studio proporciona monitores de rendimiento para que puedas realizar de manera más sencilla un seguimiento del uso de CPU y memoria de tu app, buscar objetos sin asignar, localizar pérdidas de memoria, optimizar el rendimiento de los gráficos y analizar solicitudes de la red. Con tu app ejecutándose en un dispositivo o emulador, abre la ventana de herramientas **Android Monitor** y haz clic en la pestaña **Monitors**.

Volcado de montón

Cuando controlas el uso de la memoria en Android Studio, puedes iniciar simultáneamente la recolección de elementos no usados y volcar el montón de Java a una captura instantánea del montón en un archivo de formato binario HPROF específico de Android. El visor de HPROF muestra las clases, las instancias de cada clase y un árbol de referencia para ayudarte a realizar el seguimiento del uso de la memoria y encontrar fugas de memoria.

Seguimiento de asignaciones

Android Studio te permite realizar un seguimiento de la asignación de memoria mientras controla el uso de esta. El seguimiento de la asignación de memoria te permite controlar dónde se asignan los objetos cuando realizas ciertas acciones. Conocer estas asignaciones te permite optimizar el rendimiento de tu app y el uso de la memoria ajustando las llamadas del método relacionadas con las acciones en cuestión.

Acceso a archivos de datos

Las herramientas del Android SDK, como Systrace, logcat y Traceview, , generan datos de rendimiento y depuración para un análisis detallado de la app.

Para ver los archivos de datos generados disponibles, abre la ventana de herramientas Captures. En la lista de los archivos generados, haz doble clic en uno para ver los datos. Haz clic con el botón secundario en cualquiera de los archivos .hprof para convertirlos al formato de archivo .hprof estándar.

Inspecciones de código

Cuando compilas tu programa, Android Studio ejecuta automáticamente inspecciones de Lint y otras inspecciones de IDE configuradas para ayudarte a identificar y corregir problemas fácilmente con respecto a la calidad estructural de tu código.

La herramienta Lint verifica los archivos de origen de tu proyecto Android para detectar posibles errores y mejoras de optimización en relación con la corrección, la seguridad, el rendimiento, el uso, la accesibilidad y la internacionalización.

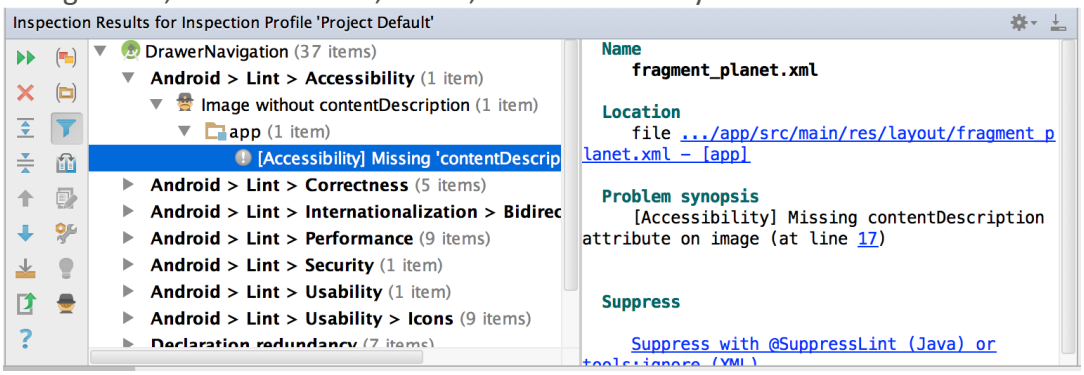


Figura 8. Resultados de una inspección de Lint en Android Studio.

Además de las verificaciones de Lint, Android Studio también realiza inspecciones de código de IntelliJ y valida anotaciones para simplificar tu flujo de trabajo de codificación.

Anotaciones en Android Studio

Android Studio admite anotaciones para variables, parámetros y valores de retorno para ayudarte a detectar errores, como excepciones de puntero nulo y conflictos de tipos de recurso. SDK Manager de Android empaqueta la biblioteca de compatibilidad-anotaciones en el Repositorio de compatibilidad de Android para usarla con Android Studio. Android Studio valida las anotaciones configuradas durante la inspección del código.

Mensajes de registro

Cuando compilas y ejecutas tu app con Android Studio, puedes ver mensajes adb de salida y mensajes de registro del dispositivo (logcat) haciendo clic en **Android Monitor** en la parte inferior de la ventana.

Si quieres depurar tu app con el Monitor de dispositivos Android, puedes iniciar el Monitor de dispositivos haciendo clic en **Tools > Android > Android Device Monitor**. En el Monitor de dispositivos encontrarás el conjunto completo de herramientas DDMS para perfilar tu app, controlar comportamientos del dispositivo y más. En este también se incluye la herramienta del Visor de jerarquía para ayudarte a optimizar tus diseños.

Sugerencias didácticas.

El **sistema operativo más popular para dispositivos móviles** en el mercado, no podía quedarse lejos por mucho tiempo del número uno **para computadoras: Windows**.

Es por ello que los millones de usuarios alrededor de **Android** del mundo, han puesto su atención desde hace un tiempo en herramientas que permitan seguir disfrutando de los emocionantes juegos y las prácticas aplicaciones disponibles para smartphones, con la **calidad de imagen y rendimiento de una computadora**.

Investiga y genera un cuadro sinóptico de los siguientes emuladores:

1. .Bluestacks
2. Genymotion
3. Andy OS
4. Droid4X
5. ARChon
6. YouWave
7. Manymo

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Entrega de la investigación	1 punto	0 punto
El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto

Bibliografía (emplear formato APA)

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #5

ENTORNO DE DESARROLLO LENGUAJE A UTILIZAR

Competencia(s) a desarrollar.

Desarrolla aplicaciones móviles nativas, web e híbridas para atender las necesidades del entorno**.**

Introducción

Hoy en día, muchos dispositivos basados en la red o con capacidad para red ejecutan un tipo de kernel Linux. Es una plataforma sólida: rentable para desplegar y soportar, y aceptada inmediatamente como un buen abordaje de diseño para la implementación. La UI para dichos dispositivos está a menudo basada en HTML y se puede acceder con un navegador de PC o Mac. Pero no todos los dispositivos necesitan ser controlados por un dispositivo general de computación. Considere un dispositivo convencional, como por ejemplo una cocina, un microondas o una máquina panificadora. ¿Qué pasaría si los aparatos electrodomésticos fueran controlados por Android y tuvieran una pantalla táctil a color? Con una UI Android en la parte superior de la cocina, el autor podría incluso cocinar algo.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 2.
El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de Android Studio

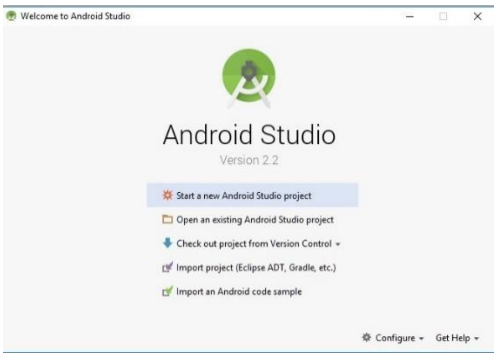
Material y equipo necesario

PC o Laptop, Android instalado y configurado
Internet

Metodología

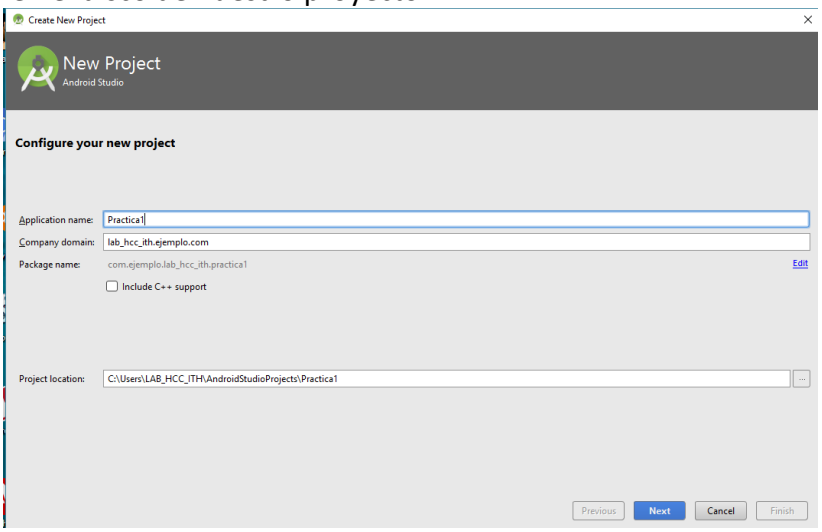
Pasos para crear el primer proyecto Android Studio

1. Una vez que iniciamos el entorno del Android Studio aparece el diálogo principal:

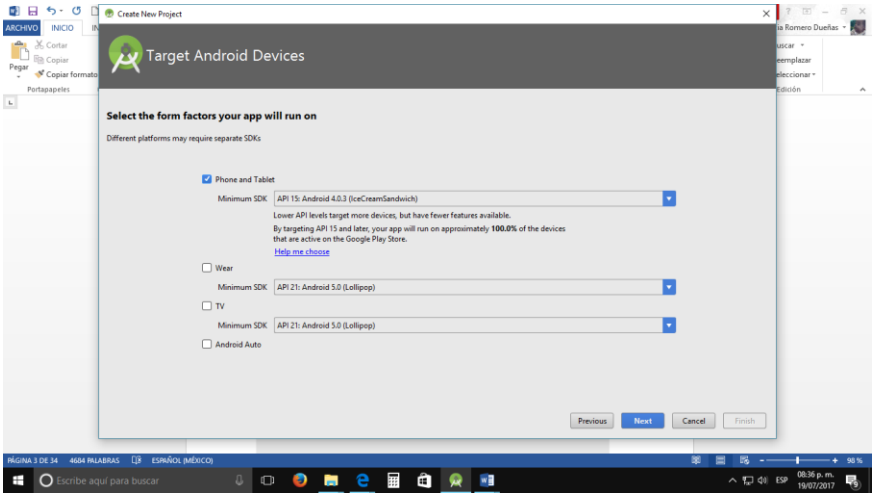


Elegimos la opción "Start a New Android Studio project

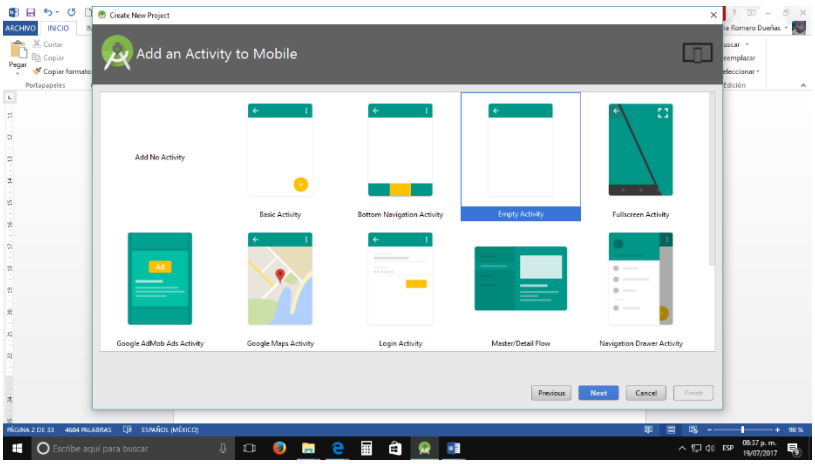
Ahora aparecerán una serie de ventanas para configurar el proyecto, el primer diálogo debemos especificar el Nombre de la aplicación, la url de nuestra empresa (que será el nombre del paquete que asigna java para los archivos fuentes) y la ubicación en el disco de nuestro proyecto:



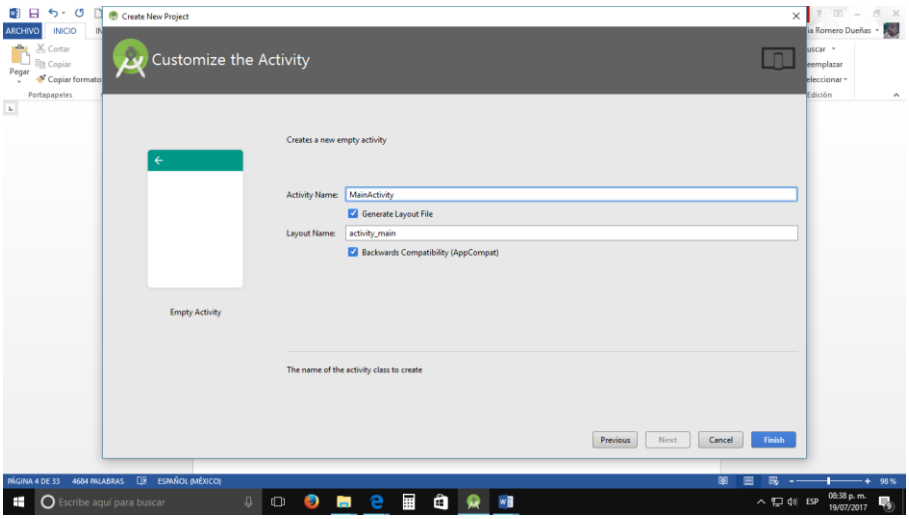
En el segundo diálogo procedemos a especificar la versión de Android mínima donde se ejecutará la aplicación que desarrollemos (dejaremos la versión 4.0.3):



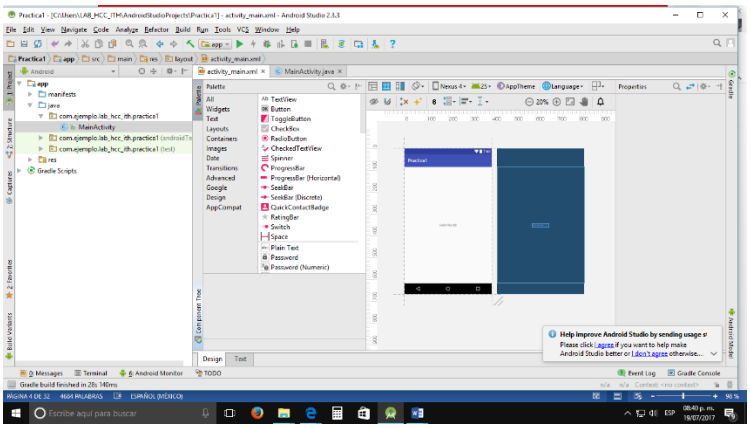
El tercer diálogo especificamos el esqueleto básico de nuestra aplicación, seleccionaremos "Empty Activity" si tenemos el Android Studio 2.2:



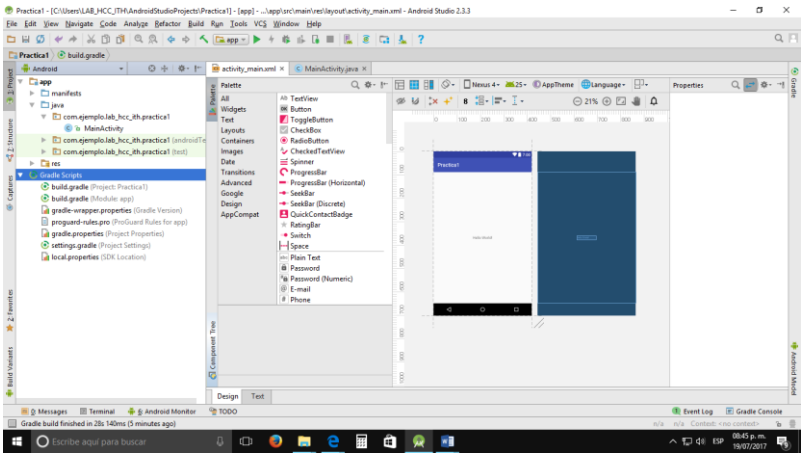
Finalmente el último diálogo tenemos que indicar el nombre de la ventana principal de la aplicación (Activity Name) y otros datos más que veremos a lo largo del curso (dejaremos con los nombres por defecto que propone Android Studio):



Tenemos finalmente creado nuestro primer proyecto en Android Studio y podemos ahora ver el entorno del Android Studio para codificar la aplicación:

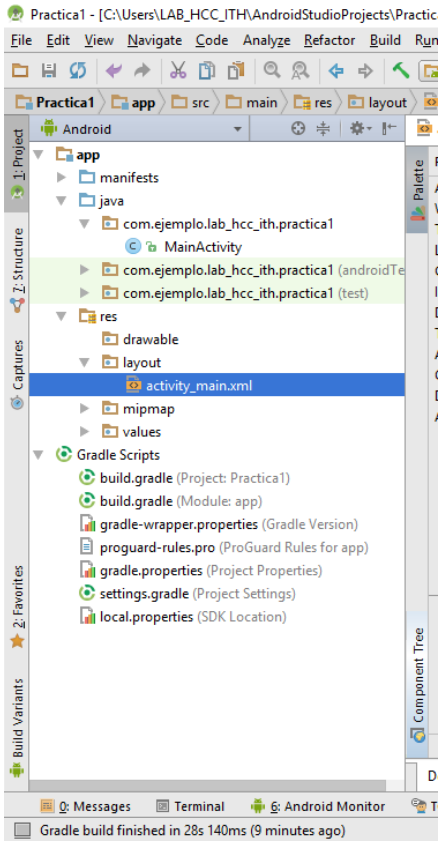


El Android Studio nos genera todos los directorios y archivos básicos para iniciar nuestro proyecto, los podemos ver en el lado izquierdo del entorno de desarrollo:



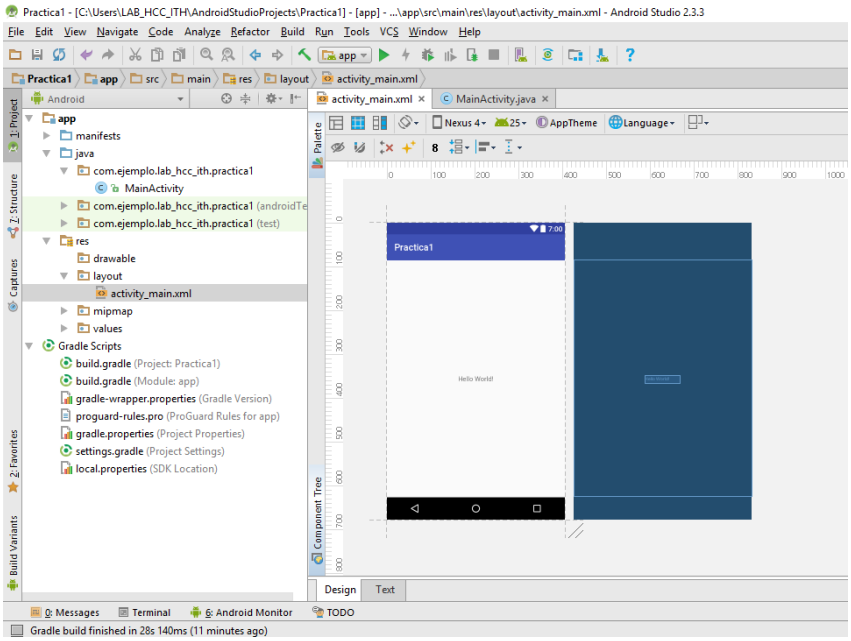
No haremos en este momento un análisis del significado y objetivo de cada uno de estas secciones y archivos generados, sino a medida que avancemos con este curso iremos viendo en forma puntual y profunda.

La interfaz visual de nuestro programa para Android se almacena en un archivo XML en la carpeta **res**, subcarpeta **layout** y el archivo se llama **activity_main.xml**. En esta carpeta tenemos creada nuestra primer pantalla.

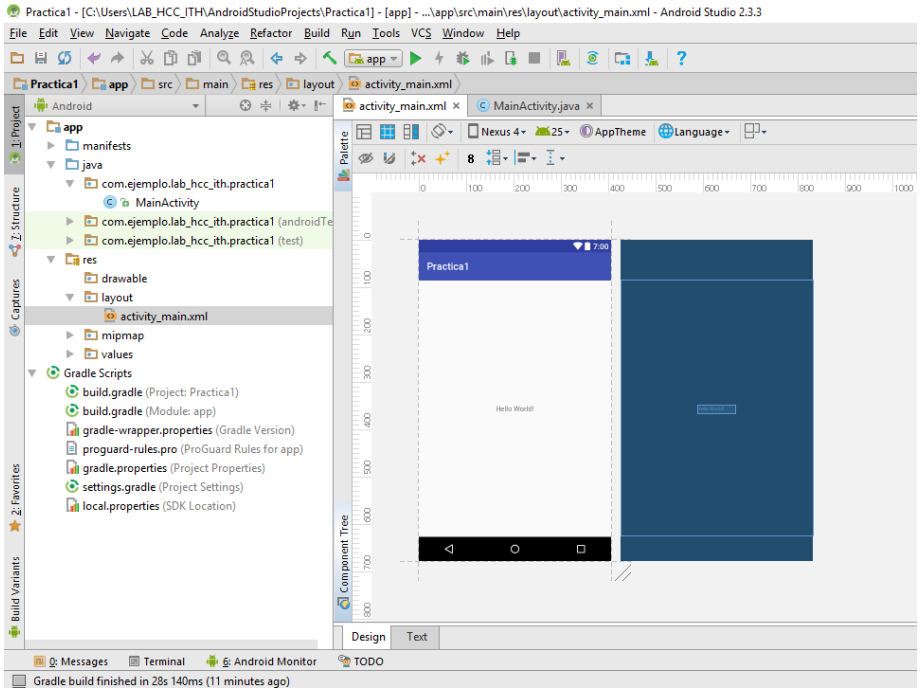


Al seleccionar este archivo el Android Studio nos permite visualizar el contenido en "Design" o "Text" (es decir en vista de diseño o en vista de código):

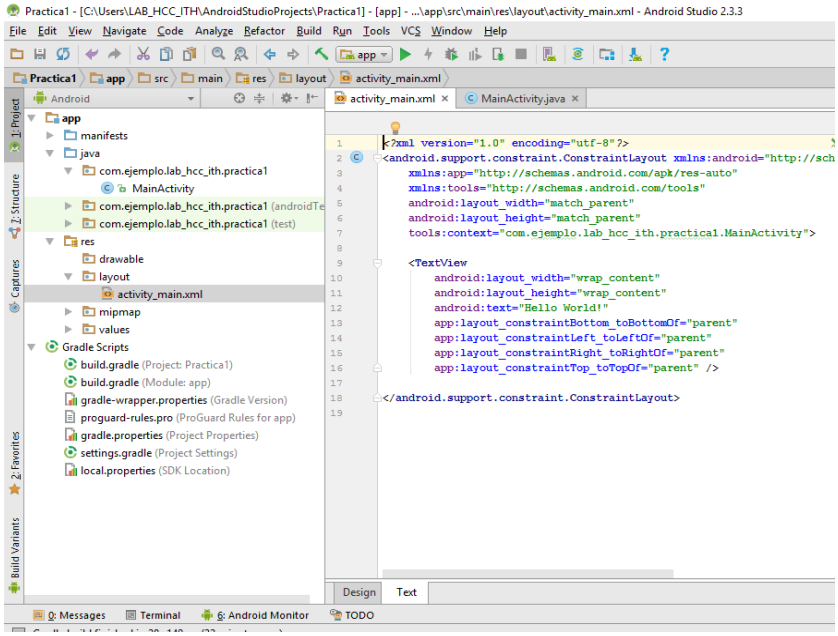
Vista de diseño:



A partir de la versión 2.2 del Android Studio tenemos la vista "blueprint" que nos nuestra una interfaz simplificada muy útil cuando tenemos pantallas complejas que veremos más adelante. Podemos ver solo la vista de diseño o "blueprint" seleccionando alguno de los botones que aparecen aquí:



Vista de código:

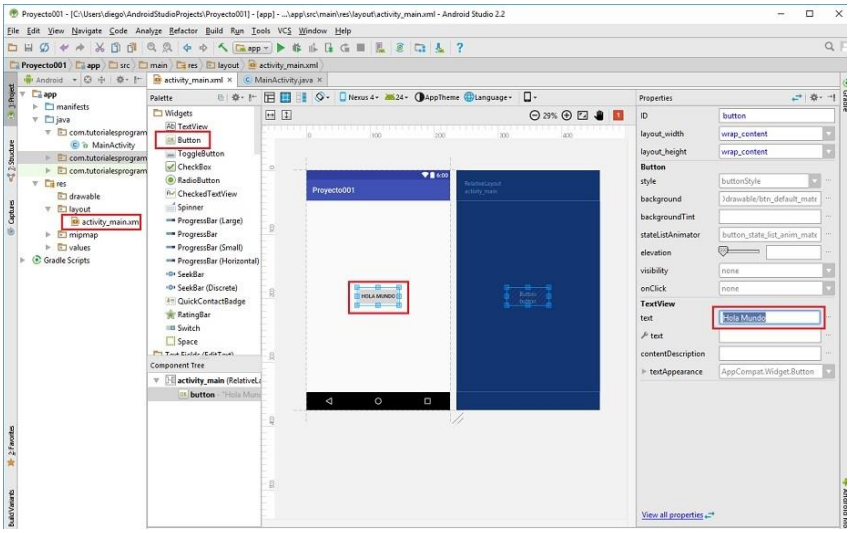


El Android Studio ya insertó un control de tipo RelativeLayout que permite ingresar controles visuales alineados a los bordes y a otros controles que haya en la ventana (más adelante analizaremos este layout)

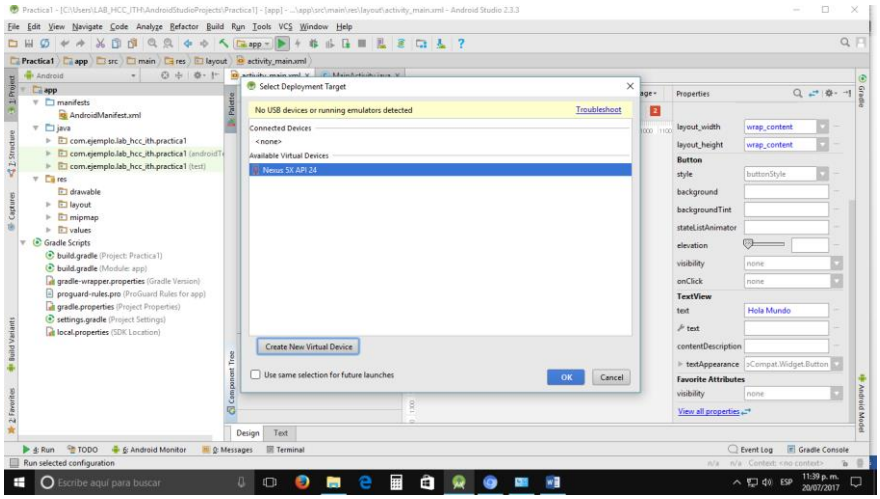
Ya veremos que podemos modificar todo este archivo para que se adapte a la aplicación que queremos desarrollar.

A lo largo de este curso iremos viendo los objetivos de cada una de las secciones que cuenta el Android Studio para implementar la interfaz, codificar en java las funcionalidades de la aplicación etc.

Antes de probar la aplicación en el emulador de un dispositivo Android procederemos a hacer un pequeño cambio a la interfaz que aparece en el celular: borraremos la label que dice "Hello World" (simplemente seleccionando con el mouse dicho elemento y presionando la tecla delete, podemos seleccionarla de cualquiera de las dos interfaces "Design" o "blueprint") y de la "Palette" arrastraremos un "Button" al centro del celular y en la ventana "Properties" estando seleccionado el "Button" cambiaremos la propiedad "text" por la cadena "Hola Mundo":

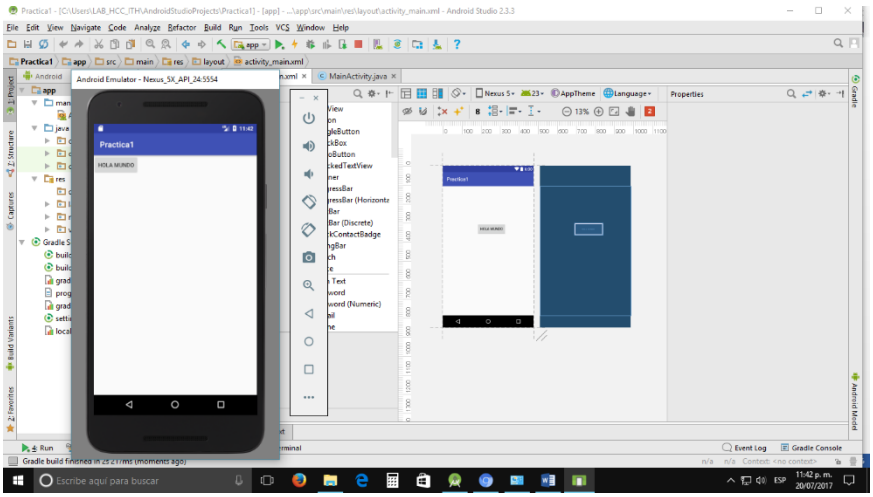


Para ejecutar la aplicación presionamos el triángulo verde o seleccionamos del menú de opciones "Run -> Run app" y en este diálogo procedemos a dejar seleccionado el emulador por defecto que aparece (Nexus 5X) y presionamos el botón "OK" (si no tiene ningún emulador puede crear uno):



Luego de un rato aparecerá el emulador de Android en pantalla (el arranque del emulador puede llevar más de un minuto), es IMPORTANTE tener en cuenta que una vez que el emulador se ha arrancado no lo debemos cerrar cada vez que hacemos cambios en nuestra aplicación o codificamos otras aplicaciones, sino que volvemos a ejecutar la aplicación con los cambios y al estar el emulador corriendo el tiempo que tarda hasta que aparece nuestro programa en el emulador es muy reducido.

Cuando terminó de cargarse el emulador debe aparecer nuestra aplicación ejecutándose:



Sugerencias didácticas.

Investigar cómo crear un entorno de desarrollo para Android

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Entrega de la investigación	1 punto	0 punto

El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía (emplear formato APA)

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #6

INSTALACION DE UN SISTEMA GESTOR DE BASE DE DATOS PARA MOVILES

Competencia(s) a desarrollar.

Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

Inicialmente las aplicaciones inalámbricas para dispositivos móviles eran programas totalmente desconectados de las empresas o sistemas de computación. Esto desde el punto de vista en tiempo real. Estas aplicaciones eran por lo general Palm Pilots limitadas a libreta de direcciones, horarios, etc. Esta pequeña base de datos estaba bien para ese momento, pero a medida que avanza la tecnología, las personas necesitan más.

Surgen las redes inalámbricas, pero existe el problema de disponibilidad de dispositivos inalámbricos y teléfonos WAP (Wireles Application Protocol) que se encuentran limitados en memoria y el tamaño de la pantalla para mostrar la información, además que introducir información en estos dispositivos móviles puede llegar a ser bastante difícil por su tamaño. De todas formas la tecnología es excelente, la habilidad que se tiene para conectarse al World Wide Web desde casi cualquier sitio en cualquier momento es una de las mayores ventajas que se tiene en la era de la información.

Con Internet inalámbrico, ya no es necesario estar físicamente frente al computador personas o dentro de las empresas para poderse conectar a las aplicaciones o las base de datos. Con las portabilidad de la tecnología inalámbrica, nos podemos conectar a Internet o la intranet de la empresa para tomar datos almacenados. Estos datos están estructurados y organizados en entidades y objetos que se encuentran disponibles para los usuarios como información. La mayor ventaja se encuentra en que se le da la información al usuario en el mismo momento que es solicitada.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 4.

El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD

Material y equipo necesario

PC o Laptop, Android instalado y configurado

Internet

XAMPP

Metodología

1. Una vez presentados los distintos SGBD en la publicación anterior, elegiremos el **gestor MySQL** como **base de datos externa de nuestra Aplicación Android** con la que estableceremos una **conexión remota**.

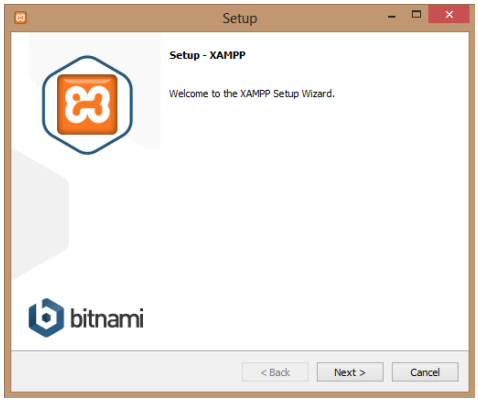
2. Contando con tener ya instalado nuestro IDE Eclipse con el SDK Android, el primer paso será **instalar un servidor MySQL**, que es lo que veremos en este breve tutorial, y a partir de ahí, ya podremos definir nuestra base de datos y conectarnos remotamente a ella cómo explicaremos en la próxima publicación y video de esta serie.

3. Para este primer objetivo vamos a utilizar el servidor de plataforma libre **XAMPP**, gratuito bajo licencia GNU , que entre otras características nos proporciona un **servidor Apache** y un **servidor de Base de Datos MySQL**. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X y es un entorno muy popular para el desarrollo de páginas web dinámicas con PHP.

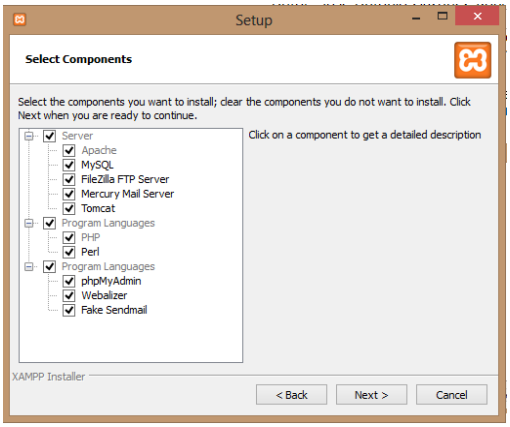
4. Para descargarlo, accederemos a la web del equipo de proyecto que lo desarrolló y mantiene, **Apache Friends**: https://www.apachefriends.org/es/index.html

5. Una vez descargado (se ha utilizado la versión 1.8.3 para Windows) se instalará siguiendo esta serie de pasos :

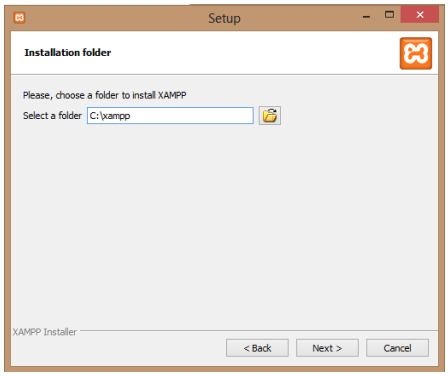
6. Al ejecutar el archivo de instalación, se mostrará la ventana de inicio del setup:



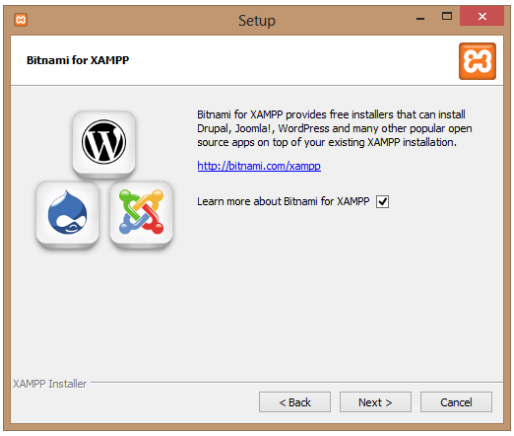
7. Tras pulsar a continuar (‘next’), indicaremos los componentes que deseamos instalar (en nuestro caso se dejarán los componentes seleccionados por defecto):



8. También dejaremos la ruta indicada por defecto para su instalación, en la ventana siguiente:

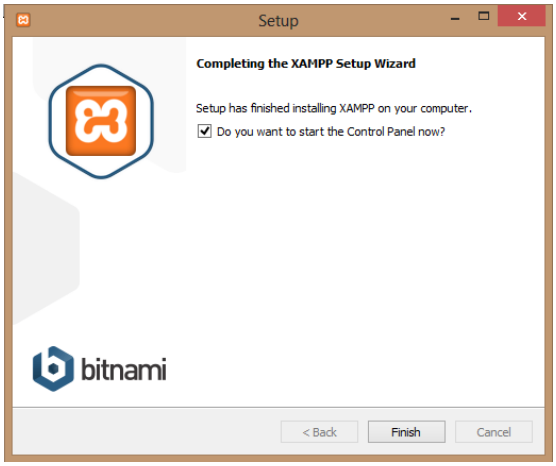


9. A continuación, nos indicará si deseamos instalar Bitnami, que nos posibilita la instalación de varios CMS y muchas otras Aplicaciones Open Source:

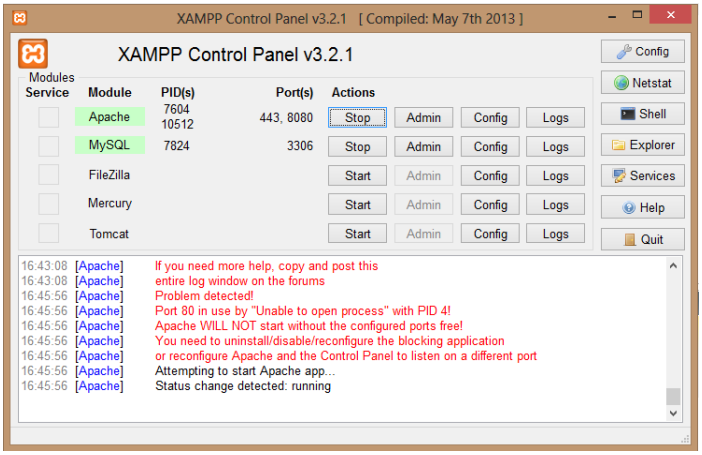


10. Tras esa pantalla, nos informará de que ya está preparado para comenzar la instalación. Pulsamos en “Next” e iniciará dicho proceso:

11. Una vez finalizada la instalación, se marcará la casilla para que se inicie el Panel de Control y pulsamos en “Finish”:



12. Al tener marcada aquella casilla, se mostrará el Panel de Control, y se deberán seleccionar las opciones de Apache y MySQL pulsando sobre los botones “**Start**” correspondientes a cada módulo en la columna “**Actions**“:

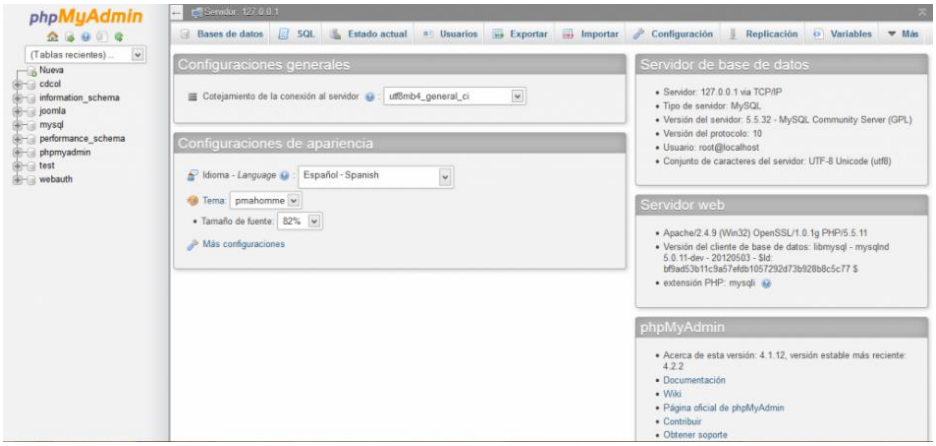


13. Nota: En caso de error al iniciar el módulo de Apache, bastará con cambiar el puerto pulsando en la opción "**Config**" de Apache y modificar las líneas **Listen 80** y **Listen XX.XX.XX.XX:80** por un puerto que no haya sido asignado (en el ejemplo se utiliza el puerto 8080). El archivo a modificar se llama "**httpd.conf**".

14. Tras iniciar los servidores, se abrirá una ventana del navegador y en la dirección introduciremos: http://localhost:<Nº Puerto> , y seleccionaremos el idioma. Finalmente nos mostrará el administrador XAMPP, y por último debemos buscar y pulsar sobre una opción en el menú de navegación situado a la derecha llamada "**phpMyAdmin**", dentro de las opciones de "**Tools**":

15. phpMyAdmin es una **herramienta** que nos facilita la **administración de MySQL**: gestionar base de datos, tablas, columnas, usuarios, permisos, etc. También nos permite **ejecutar sentencias SQL directamente** a través de su interfaz de usuario.

16. Tras pulsar en "**phpMyAdmin**", ya se podrá acceder a su interfaz para interaccionar con el sistema de gestión de base de datos MySQL:



Sugerencias didácticas.

Investigar sobre las ventajas e inconvenientes de las bases de datos móviles

Reporte del alumno

Criterio para calificar	Sí	No
Entrega de la investigación	1 punto	0 punto
El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso.

"Ejemplo de aplicación Android y SQLite | jc-Mouse.net." N.p. Web. 25 Junio 2017. http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

PRÁCTICA #7

DESARROLLO DE APLICACIONES MOVILES NATIVAS QUE RESUELVAN PROBLEMATICAS DIVERSAS

Competencia(s) a desarrollar.

Conocer y aplicar tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

Las aplicaciones nativas son las que se encuentran dentro de un sistema operativo o plataforma por default, es decir que pertenece al mismo sistema y son desarrollados por la misma compañía, por lo que presentan gran compatibilidad, por lo que se reduce el porcentaje de error que se puede presentar dentro de las operaciones que realiza.
Sin embargo presenta como desventaja que muchas veces sólo se aplica a un sistema operativo sin importar si es de la misma empresa, pues al llegar las nuevas versiones del sistema y se requiere del programa, se necesita de un nuevo desarrollo del mismo para la ejecución sin problemas de compilación.
El código con el que se desarrollan las aplicaciones nativas es el mismo o un derivado del que se utiliza para la realización de los comandos del sistema operativo o plataforma en que se ejecutan, es la diferencia que se tienen con las aplicaciones web o las híbridas, pues en estas se utiliza un sistema de codificación diferente para realizar las instrucciones, pero mediante un compilador y un emulador se pueden ejecutar en el sistema que se desee.

Por ejemplo, las aplicaciones con HTML, HTML5, Java, SAP, JOOMLA, ABAP, CSS y otros sistemas de codificación no son nativas, sino que son híbridas o Web, por lo que no siempre funcionan en todas las plataformas base.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 3.

Diseñar, exportar e implementar bases de datos para su manejo en aplicaciones que gestionan bases de datos desde dispositivos móviles y presentar reporte de funcionamiento

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Microsoft Visual Studio* 2012 o 2013 (ediciones Professional o Ultimate). No se admiten las ediciones Express.

Versión de 32 bits de JDK 7 o superior.

Metodología

DESARROLLANDO APLICACIONES NATIVAS PARA ANDROID CON C#

Alejandro Tamayo Castillo

RESUMEN

En esta entrega se mostrará la utilización de *Mono for Android*, componente que permite el desarrollo de aplicaciones para Android utilizando C# y Visual Studio. *Mono for Android*, desarrollado por Xamarin (compañía fundada por Miguel de Icaza, patrocinadora de Mono) permite crear, depurar y empaquetar una aplicación en un .apk y utilizarla directamente en un teléfono Android. A diferencia de otros productos, *Mono for Android* permite el desarrollo de aplicaciones nativas, significando que se integra al ecosistema Android e interactúa con las aplicaciones nativas creadas en Java, utiliza la interfaz (UI) nativa del sistema operativo y sigue el modelo de desarrollo de Android.

LA APUESTA POR .NET Y C#

Si bien es atractivo programar para Android en C#, existen tres razones adicionales por lo cual es conveniente apostar por .NET y C# para el desarrollo de aplicaciones móviles:

1. Desarrollo Multiplataforma
2. Reutilización de bibliotecas de código existentes
3. Interfaz de usuario nativa y rendimiento nativo

El mercado de dispositivos móviles está dominado fundamentalmente por tres plataformas: Android de Google, iOS de Apple y Windows Phone de Microsoft, que a pesar de ser minoritario muestra un discreto crecimiento. Para hacer una aplicación que ejecute en estas plataformas de manera nativa, habría que programar en lenguajes y herramientas de desarrollo diferentes: Java en Android, Objective-C en iOS y C# en Windows Phone. Para una empresa esto puede significar un reto, ya que tendría que mantener y darle soporte a tres códigos fuentes diferentes, triplicando los esfuerzos. Es por ello que las alternativas multi-plataforma (como *PhoneGap* o *Appcelerator Titanium*) han cobrado fuerza ya que ayudan a abaratar los costos y simplificar el esfuerzo. En el caso de Xamarin, empresa fundada por Miguel de Icaza, se brinda una solución integral (IDE+Plataforma) que posibilita la creación de aplicaciones para iOS, Android, Mac, Linux y Windows ya que cuenta con versiones de Mono así como los compiladores y las herramientas de desarrollo necesarias para convertir de C#/.NET al código nativo que ejecuta en cada una de estas plataformas.

Contar con C# y .NET nos permite reutilizar las bibliotecas de código existentes que se hayan desarrollado previamente (incluso para otras aplicaciones .NET) y explotar los algoritmos escritos en C# por la comunidad. Utilizando la característica **Portable Class Library** (PCL) de .NET, es posible incluso desarrollar la lógica de la aplicación una sola vez y desarrollar para cada plataforma sólo el código específico que se requiera. De esta forma se simplifica el mantenimiento de la aplicación.

Al obtener un resultado que compila directamente e interactúa con las características nativas del dispositivo, se obtienen beneficios de rendimiento con respecto a las aplicaciones HTML5 que requieren capas de abstracción más complejas para su ejecución. Además, la experiencia de usuario (UI) se mantiene intacta, ya que se utilizan los mismos componentes nativos de la plataforma para el desarrollo de la interfaz de usuario.
Para valorar la apuesta por .NET/C#, también hay que tener en cuenta que las herramientas de desarrollo que brinda Xamarin tienen un carácter comercial, y el costo de la licencia varía entre $299 y $999 en dependencia de si es una licencia individual o empresarial. Existe también una variante gratuita limitada a 32Kb de código IL y una licencia un 90% más barata para propósitos académicos ($99 la versión empresarial que es la más completa e incluye la integración con Visual Studio). Por otra parte hay que tener en cuenta que la aplicación compilada (en el caso de Android, un .apk) contendrá el código IL que se compilará Just-In-Time (JIT) al ejecutar la aplicación, por lo que se gastará un poco más de memoria (tanto en espacio en disco como en RAM) para poder soportar el proceso de compilación JIT.
Finalmente, hay que decir que las herramientas de Xamarin se actualizan unos meses (2-3) después que sale la versión del sistema operativo y las herramientas de desarrollo nativas, que es un tiempo relativamente rápido en que se incorporan las nuevas características a la plataforma Mono.
Recientemente Gartner reconoció a Xamarin como Visionaria en el campo del desarrollo de aplicaciones móviles (http://blog.xamarin.com/gartnermq), y resaltó el rendimiento de la plataforma, la posibilidad de creación de una interfaz de usuario (UI) completamente nativa y la facilidad de acceso a las características nativas de los dispositivos.

ARQUITECTURA DE MONO FOR ANDROID

Mono es la implementación de código abierto (*open source*) de la plataforma .NET sobre Windows y Linux. En el caso de Android, Mono ejecuta sobre el Kernel de Linux, a la par de Dalvik, que es la máquina virtual de Java que soporta todo el ecosistema Android. Lo interesante en este caso (Figura 1) es la coexistencia de Mono con Dalvik. Primeramente hay que destacar los *Android Callable Wrappers* (ACW) que empaquetan la funcionalidad de Dalvik y la hace disponible como parte de la API de Mono, de forma tal que una aplicación escrita en C# pueda realizar llamadas a componentes escritos en Java que ejecuten en Dalvik. De manera inversa, están los *Managed Callable Wrappers* que son proxies generados en Java que le permiten a las aplicaciones ejecutando en Dalvik, poder acceder a los componentes desarrollados en .NET que ejecutan en Mono. Gracias a esta coexistencia, aplicaciones nativas desarrolladas en Java pueden interactuar con las desarrolladas en .NET de manera natural a nivel de API.

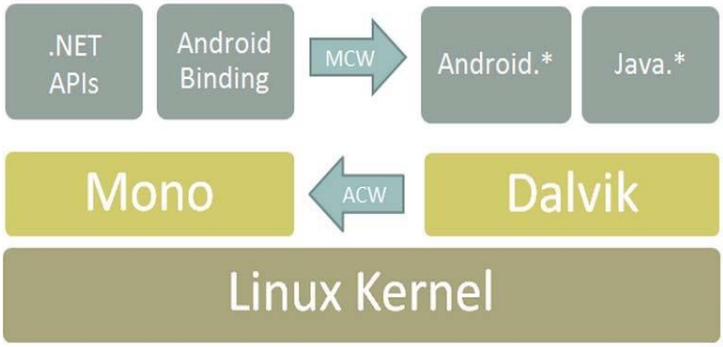


FIGURA 1 ARQUITECTURA DE MONO FOR ANDROID. TOMADO DEL SITIO DE XAMARIN.

De manera experimental, Xamarin portó el código de Android, de Java a C#, eliminando la dependencia con Dalvik y logrando un Android 100% en C# (proyecto XobotOS). Como resultado se obtuvo un incremento notable en el rendimiento del sistema. Este resultado se le atribuye a características únicas que tiene la plataforma .NET con respecto a Java, tales como la genericidad real, el uso de *structs* y las llamadas P/Invoke. La herramienta de conversión de código de Java a C#, desarrollada por Xamarin para este experimento, le permite a esta compañía mejorar *Mono for Android* en tanto se puedan ir eliminando las dependencias con Dalvik posibilitando la ejecución directa del código sobre el Kernel de Linux.

MI PRIMERA APLICACIÓN CON XAMARIN STUDIO O VISUAL STUDIO

Para empezar a desarrollar aplicaciones para Android tenemos dos posibles entornos de desarrollo (IDE):

- Xamarin Studio 4
- Visual Studio 2010/2012

Xamarin Studio es una versión personalizada para desarrollo móvil de MonoDevelop, que es el IDE oficial de Mono. Mono for Android, cuenta con una extensión para Visual Studio 2010/2012 que habilita la compilación y la depuración de aplicaciones Android. Lo interesante es que el mismo proyecto/solución sirve tanto en Visual Studio como Xamarin Studio y se puede utilizar indistintamente tanto uno como el otro, ya que MonoDevelop, en el cual se basa Xamarin Studio, utiliza el mismo formato (.sln y .csproj) que Visual Studio.

FIGURA 2 ANDROID SDK MANAGER. SE MUESTRA SOMBREADO LA IMAGEN DE ANDROID 4.2.2 COMPATIBLE CON LA ARQUITECTURA X86 DE INTEL

Antes de empezar, se tiene que tener instalado el **Android SDK** y una máquina virtual con Android que es donde se probará y depurará la aplicación. Utilizando el **Android SDK Manager** (Figura 2) se pueden descargar de Internet las imágenes (máquinas virtuales) de Android tanto para ARM como para x86. El desarrollo para Android debe realizarse sobre un procesador Intel con *Virtualization Technology* (VT-x) e instalar el componente **Intel HAXM**, que permitirá la ejecución nativa de Android en el PC. En caso de que se tenga un procesador AMD o se utilice una imagen de Android con arquitectura ARM, la ejecución y depuración de la aplicación será un poco tortuosa producto de la lentitud de conversión de instrucciones ARM a x86 o la carencia de VT-x. En tal caso (AMD) se recomienda el uso de VirtualBox con una imagen x86 de Android conectada a través de la red utilizando adb. A pesar de que esta variante es un *hack* y se pierde la integración con el IDE, es preferible producto de la ganancia en velocidad.
Aquellos que no estén familiarizados con un entorno de desarrollo para Android, pueden descargar el **Xamarin Installer** desde el sitio web oficial de Xamarin, y éste se encargará de bajar de internet todos los componentes necesarios, instalarlos y configurarlos, dejando lista la PC para empezar a desarrollar.

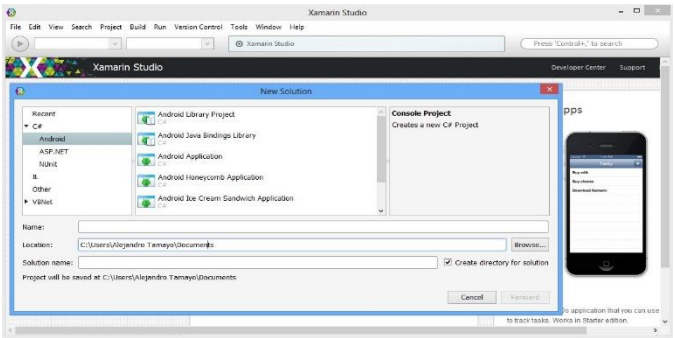


FIGURA 3 NUEVO PROYECTO DE ANDROID UTILIZANDO XAMARIN STUDIO

Existen diferentes tipos de proyectos de Android. Estos tipos de proyectos están disponibles tanto desde Xamarin Studio (Figura 3) como desde Visual Studio 2012 (Figura 4):

- **Android Application (incluyendo las versiones para Tablets, HC y ICS):** Es la plantilla de proyectos predeterminada. Permite crear una aplicación de Android con interfaz de usuario (UI) y genera manera predeterminada un Activity de ejemplo.
- **Android OpenGL Application:** Permite crear una aplicación OpenGL para Android. Usualmente Juegos y multimedias.
- **Android Class Library:** Es el equivalente al proyecto Class Library de .NET lo que utilizando las referencias de Mono for Android.
- **Java Binding Library:** Proyecto que permite importar un .jar de Java al ecosistema .NET. Similar a los Wrappers que en .Net se le hacen a los objetos COM, pero en este caso son objetos Java. Útil por ejemplo para referenciar componentes como *Google Maps library* que están disponibles solo para Java.

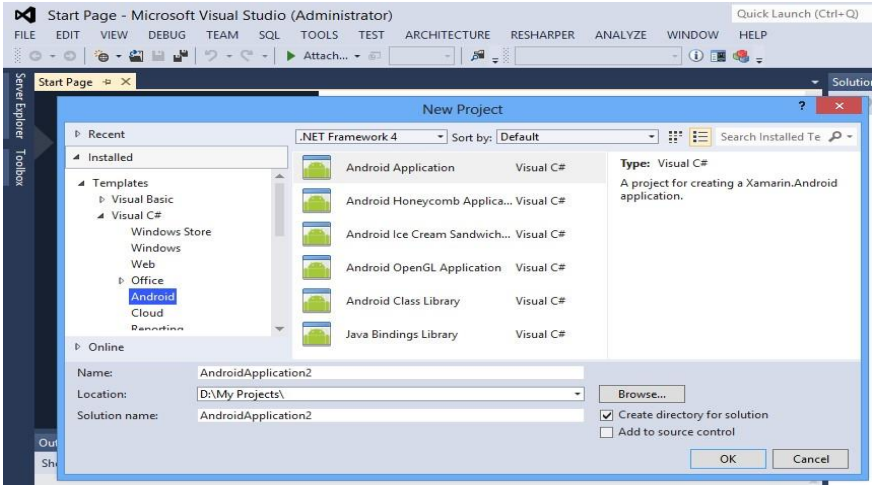


FIGURA 4 NUEVO PROYECTO DE ANDROID UTILIZANDO VISUAL STUDIO 2012

En el caso de la plantilla **Android Application**, el proyecto que se genera se organiza igual a como se hace en Java. De hecho, se respetan los convenios propuestos por Android en cuanto a nombre de carpetas y elementos a almacenar. Existen dos carpetas importantes, **Assets** y **Resources**. En la carpeta *Assets* se almacena cualquier fichero que se quiera tener disponible como recurso (un fichero html, texto, un binario, etc). En la carpeta *Resources*, se almacenan recursos especializados, que pueden ser **Layouts** (interfaz gráfica UI), **Drawables** (imágenes) y **Values** (recursos de texto). Estos aspectos son específicos de Android.

Lo interesante de esta organización es que Mono no re-inventa los componentes de UI ni la disposición del proyecto, por lo que el conocimiento de cómo diseñar una interfaz para Android, se puede obtener de cualquier documentación orientada a Java y el código AXML (lenguaje XML en que se especifica la UI en Android) se puede reutilizar indistintamente.

Tanto Xamarin Studio (Figura 5) como Visual Studio (Figura 6) cuentan con las mismas herramientas de desarrollo. Es posible diseñar una interfaz de usuario gráficamente arrastrando controles desde la barra de herramienta y asignándole acciones a los eventos (OnClick por ejemplo) que cada componente define.

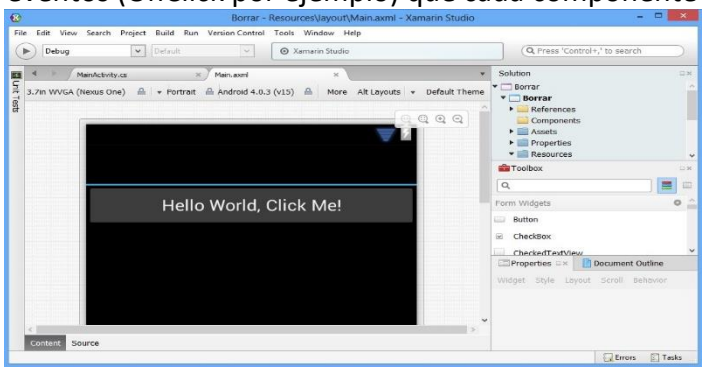


FIGURA 5 APLICACIÓN DE EJEMPLO EN XAMARIN STUDIO. SE MUESTRA EL DISEÑADOR GRÁFICO DE LA INTERFAZ DE USUARIO

El código del *Layout* (Interfaz de Usuario) de la aplicación de ejemplo, se muestra en el Listado 1. Aquellos que hayan trabajado con XAML/WPF en .NET notarán cierta semejanza con el AXML de Android para definir la interfaz de usuario. En este caso, se define un **LinearLayout** de raíz, que es un esquema que posiciona los controles interiores de forma lineal con orientación horizontal o vertical según se especifique. Los controles anidados, en este caso el *Button*, pueden definir el ajuste de ancho y alto con respecto al layout padre utilizando las propiedades *layout_width* y *layout_height*.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <Button
    android:id="@+id/MyButton"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/Hello" />
</LinearLayout>
```

LISTADO 1 LAYOUT QUE MUESTRA UN BOTÓN CON EL TEXTO HELLO WORLD, CLICK ME!

Hay que resaltar que existe una separación entre el diseño de la interfaz y la funcionalidad. En el AXML sólo se define la forma. Android sigue una filosofía Modelo-Vista-Controlador (MVC) que potencia la separación de responsabilidades (SoC). En este caso, el *Layout* sería la Vista y el controlador sería el *Activity*. En Android, cada pantalla visible es un *Activity*. El Activity es quien ensambla el funcionamiento con la vista (Layout) y se escribe en código C# (Listado 2).

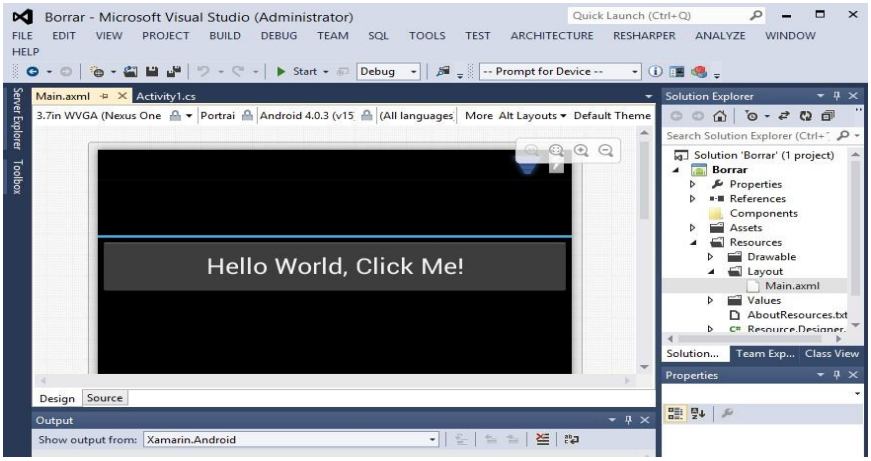


FIGURA 6 APLICACIÓN DE EJEMPLO EN VISUAL STUDIO. SE MUESTRA EL DISEÑADOR GRÁFICO DE LA INTERFAZ DE USUARIO

Como se puede observar, en C# un *Activity* hereda de la clase Activity e implementa al menos el método OnCreate que se ejecuta cada vez que se lance el *Activity*. Aquí se utiliza el método SetContentView para asociar la vista, que en este caso se liga con el Layout Main.axml y luego se localizan los controles dado su ID y se le asocia funcionalidad mediante un delegado.

```
[Activity(Label = "Borrar", MainLauncher = true, Icon =
"@drawable/icon")] public class Activity1 : Activity
{
  int count = 1;

  protected override void OnCreate(Bundle bundle)
  {

base.OnCreate(
bundle);
    // Set our view from the "main" layout resource

SetContentView(Resource.Layo
ut.Main);
    // Get our button from the layout resource,
    // and attach an event to it
    Button button = FindViewById<Button>(Resource.Id.MyButton);
    button.Click += delegate {button.Text = string.Format("{0} clicks!",
count++);};
  }
}
```

LISTADO 2 CÓDIGO FUENTE EN C# DE UN ACTIVITY

El *Activity* principal, aquel que se lanza por primera vez cuando se carga la aplicación de Android, se identifica utilizando el parámetro MainLauncher del atributo [Activity] con que se decora la clase.
Es interesante notar cómo el código del *Activity* se beneficia de las características del lenguaje C# y la plataforma .NET. Los desarrolladores de Xamarin han realizado un buen trabajo incorporando, ya que no han portado directamente el código de Java a C#, sino que han mejorado la sintaxis utilizando características como delegados, genericidad y propiedades.
En java, el código equivalente a la localización del botón y la asignación del click sería el siguiente:

```
final Button myButton = (Button)findViewById(R.id.myButton);
myButton.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View arg0)
    {
        myButton.setText(String.format("{0} clicks!", count++));
    }
};
```

Como se puede observar, los cast (Button) se reemplazaron por genericidad <Button>, los *Listeners* de Java por eventos de C# y el accesor .setText del tipo Button se remplazó por la propiedad Text, haciendo el código más legible y corto. Adicionalmente se reescribió la API para cumplir convenios de calidad de código (por ejemplo, findViewById se llevó a FindViewById para seguir el convenido de la mayúscula en métodos).

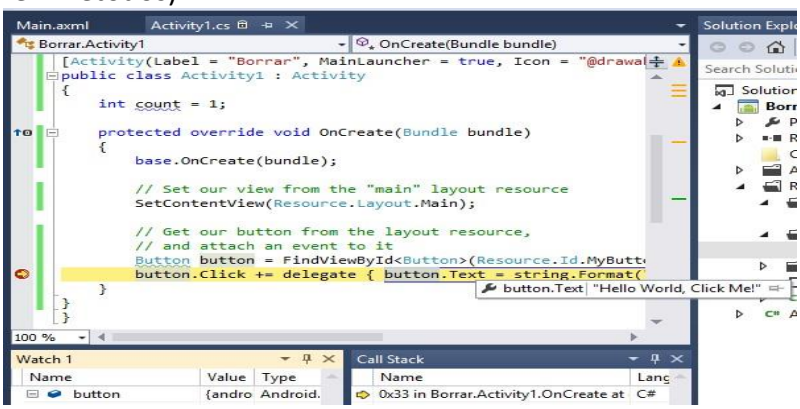


FIGURA 7 DEPURANDO LA APLICACIÓN DE ANDROID EN VISUAL STUDIO

Uno de los aspectos más interesantes es la posibilidad de depurar la aplicación y poder acceder a los valores de las variables (Figura 1). Esto es significativo, ya que la ejecución de la aplicación está sucediendo dentro de la máquina virtual de Android en el emulador.

CONCLUSIONES:

Java is not the only way to build native apps on Android. In fact, it's not even the best way

Miguel de Icaza
La combinación de C# y .NET, gracias a los esfuerzos invertidos en el proyecto Mono, ha llegado a otras plataformas como Linux, Mac y ahora, en un ámbito móvil, hasta iOS y Android. Con la fundación de Xamarin, Mono se ha convertido en una plataforma de clase empresarial, que cuenta con un amplio soporte y una abundante documentación. Desde http://docs.xamarin.com se puede acceder a tutoriales, guías y ejemplos que facilitan el desarrollo de aplicaciones tanto para iOS como para Android utilizando C# y .NET.
Utilizando ya sea Xamarin Studio o Visual Studio, se pueden desarrollar, depurar y empaquetar aplicaciones, que ejecutarán de manera nativa con un rendimiento notable en estas plataformas móviles.
En una próxima entrega, veremos cómo crear una aplicación multi-plataforma en C#/.NET que ejecute en diferentes plataformas, utilizando tecnología Xamarin.

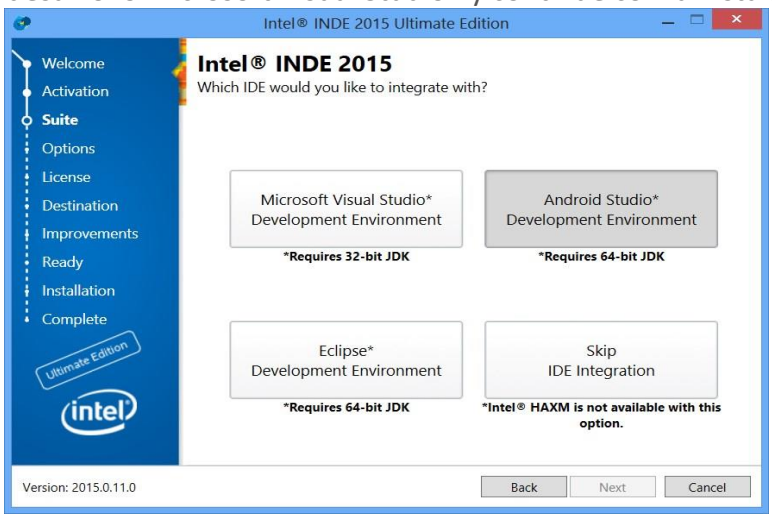
Introducción
Este artículo sirve de guía para escribir una aplicación Android* nativa de nombre "Hello World" por medio de la funcionalidad IDE Integration (integración de IDE) de Intel® INDE 2015.
Acerca de Intel® INDE
La Experiencia Integrada de Desarrollo Nativo de Intel® (Intel® INDE) es un conjunto de bibliotecas y herramientas de productividad para C++ y Java que acelera el desarrollo de aplicaciones para móviles y PC mediante la reutilización de código, para lograr código nativo sensible al rendimiento y soporte de flujo de trabajo integrado. Intel INDE hace posible crear aplicaciones Windows en la Arquitectura Intel® y aplicaciones Android en ARM y la Arquitectura Intel®. Los desarrolladores tienen la libertad de usar Intel INDE dentro del entorno de desarrollo integrado (IDE) que prefieran, incluidos Microsoft Visual Studio*, Google Android Studio* y Eclipse*. Intel® INDE también brinda a los desarrolladores acceso a funcionalidades de plataforma avanzadas, tales como aceleración de medios, detección de contexto, OpenCL™ 2.0 y bibliotecas de subprocesos, con un grupo selecto de compiladores y herramientas de análisis y depuración. Intel® INDE viene en tres ediciones: Starter, Professional y Ultimate. Se puede encontrar más información en el blog de anuncio de Intel INDE, dentro de la Zona Intel® de Desarrolladores.

Acerca de la integración con Visual Studio* en Intel® INDE 2015
Intel® INDE 2015 ha integrado el complemento vs-android disponible para Visual Studio* con una plantilla especial llamada "Android X86 Native Project" en Visual C++. INDE 2015 viene también con Debugger Extension para vs-android, con el fin de ayudar a depurar las aplicaciones. Veamos cómo compilar e implementar una aplicación nativa de ejemplo por medio de esta funcionalidad.

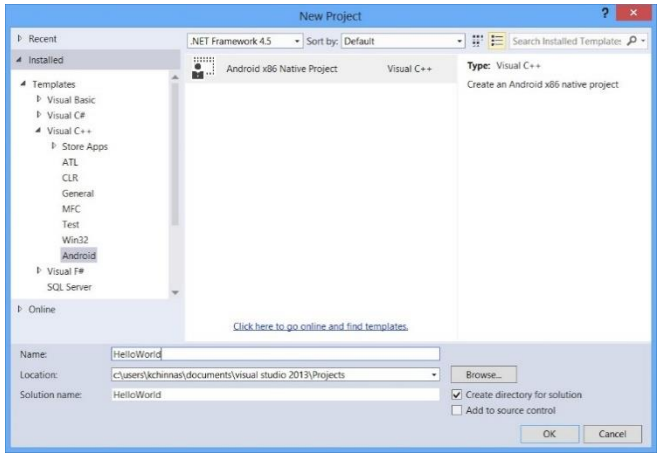
Prerrequisitos:

Microsoft Visual Studio* 2012 o 2013 (ediciones Professional o Ultimate). No se admiten las ediciones Express.
Versión de 32 bits de JDK 7 o superior.

Configuración de INDE 2015:
Descargue Intel® INDE 2015 e inicie la instalación. IDE Integration se incluye en todas las ediciones del producto. Escoja la versión que prefiera instalar y se abrirá la pantalla correspondiente a su opción de integración de IDE. Seleccione el entorno de desarrollo Microsoft Visual Studio* y continúe con la instalación.

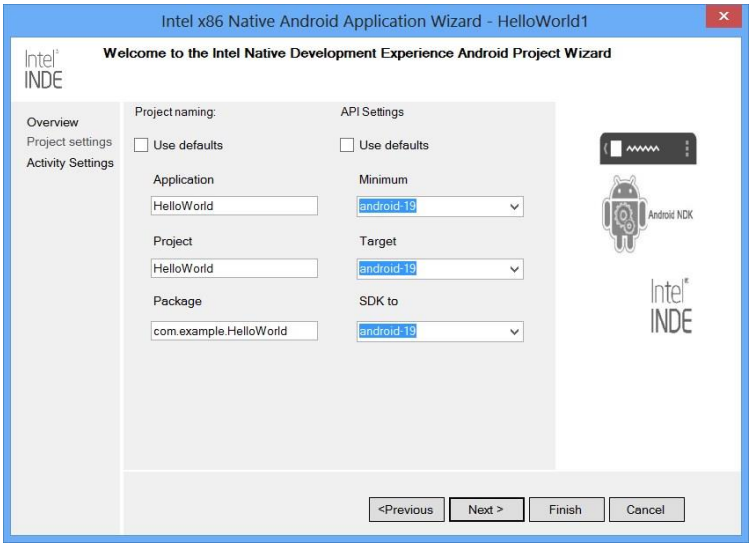


Se descargarán e instalarán todas las herramientas necesarias, incluidas: SDK de Android*, NDK, ANT, complementos de ADT, vs-android, etc.
Cómo crear su primera aplicación Android* nativa con INDE 2015:
Abra Visual Studio* y haga clic en *FILE -> New -> Project*.
Se inicia el asistente para nuevos proyectos. En *Installed -> Visual C++ -> Store Apps -> Android*, verá la plantilla "Android X86 Native Project". Cambie el nombre del proyecto a "Hello World".



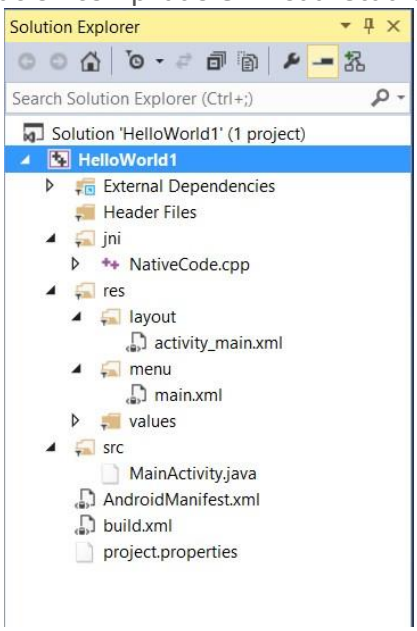
Se inicia el asistente de proyectos para Android* de Intel X86 Native Development Experience, que permite elegir la configuración del proyecto. Elija la misma

configuración de API que está establecida en el emulador o el dispositivo de destino.

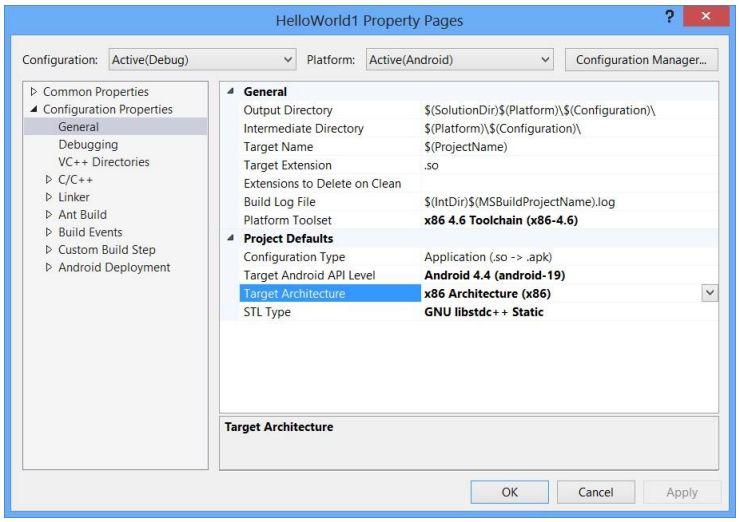


Seleccione los valores predeterminados en la página siguiente “Activity Settings” y termine.

Ahora verá el archivo solución compilado en Visual Studio*.



Haga clic con el botón derecho en el archivo solución “Hello World1” y luego haga clic en *Properties*. Las propiedades importantes ya están resaltadas. Escoja el nivel de API correcto, que debería coincidir con el emulador que vaya a ejecutar. Tiene la opción de escoger un destino ARM desde la arquitectura de destino. De forma predeterminada, se selecciona la arquitectura x86. El conjunto de herramientas para plataforma se selecciona como x86-4.6, que apunta a GCC. También puede elegir ICC.



Repasemos rápidamente algunos archivos importantes del explorador de soluciones Solution Explorer.

Jni/NativeCode.cpp tiene el código nativo C++ que puede acceder a todos los encabezados y las bibliotecas nativas.

Abra res/layout/activity_main.xml desde el explorador de soluciones. Este archivo define el diseño de interfaz de usuario de su aplicación. Observe que todavía no hay vista de diseño para esto.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="com.nativeproject.ChucksProjectApp.MainActivity">

  <TextView
      android:id="@+id/my_textview"
      android:text="@string/hello_world"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</RelativeLayout>
```

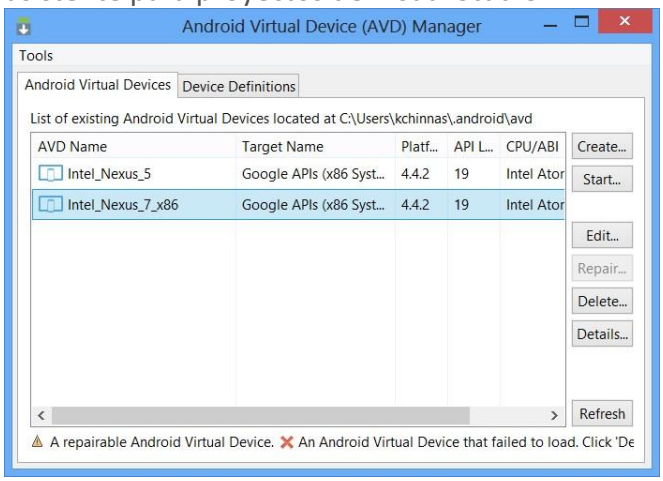
Abra src/MainActivity.java desde el explorador de soluciones. Este archivo define los controladores de eventos para su aplicación y llama al método nativo desde

aquí. Por ejemplo, el método “getStringFromNative()” es la llamada de interfaz a la función definida en jni/NativeCode.cpp.

MainActivity.java* NativeCode.cpp*

```
import android.widget.TextView;
import android.content.res.AssetManager;

public class MainActivity extends Activity
{
    static
    {
        System.loadLibrary("HelloWorld1");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView tv = (TextView) findViewById(R.id.my_textview);
        tv.setText(getStringFromNative());
    }
```

Antes de compilar e implementar este ejemplo, debe iniciar el emulador. Para ello, vaya a *<inde_install_directory>\INDE\IDEintegration\SDK* e inicie *AVD Manager.exe*. Inicie el emulador Intel_Nexus_7_x86 que viene de manera predeterminada con IDE Integration. Asegúrese de que el nivel de API sea el mismo que eligió en el asistente para proyectos de Visual Studio*.



Ahora, para compilar e implementar, elija *BUILD -> Build Solution desde Visual Studio**. Cuando la compilación haya sido exitosa, elija *BUILD -> Deploy Solution*. Debería ver la aplicación “HelloWorld” instalada en el emulador.

Haga clic en la aplicación “HelloWorld” y debería poder ver su primera aplicación en funcionamiento.

¡Felicitaciones por este primer paso tan importante!

Consejos para resolver problemas

- Para acelerar el emulador, instale Intel® HAXM. Tenga presente que debe activar Intel® VT en su BIOS y desinstalar Hyper-V si se encuentra en su máquina.
- Asegúrese de que JAVA_HOME esté configurado con el JDK de 32 bits más reciente que haya instalado.
- Es posible que vea esto “*Error occurred during initialization of VM, Could not reserve enough space for object heap, Could not create the Java virtual machine”* ("Se produjo un error durante la inicialización de VM. No se pudo reservar espacio suficiente para el montón de objeto. No se pudo crear la máquina virtual de Java"). La corrección será aumentar el tamaño de montón

MAX mediante el agregado de –Xmx512M (podría ser cualquier número grande) a la variable de entorno _JAVA_OPTIONS

Sugerencias didácticas.

Enunciar las tecnologías y herramientas asociadas a los dispositivos móviles y elaborar un cuadro sinóptico

Reporte del alumno

Criterio para calificar	Sí	No
Enviar cuadro sinóptico	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Intelcom. (2017). *Intelcom*. Retrieved 4 August, 2017, from https://software.intel.com/es-es/articles/c-mo-desarrollar-aplicaciones-android-nativas-con-ide-integration-de-intel-inde-2015-y

Unidedumx. (2017). *Unidedumx*. Retrieved 4 August, 2017, from http://moodle2.unid.edu.mx/dts_cursos_mdl/pos/TI/MN/S02/MN02_Lectura.pdf

Hermosaprogramacioncom. (2014). *Hermosa Programación: +50 Tutoriales Desarrollo Android*. Retrieved 4 August, 2017, from http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

Enrique Piñana. (2017). *Aprendeandroidcom*. Retrieved 4 August, 2017, from http://www.aprendeandroid.com

Neoteocom. (2013). *NeoTeo*. Retrieved 4 August, 2017, from http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/

Riunetupves. (2017). *Riunetupves*. Retrieved 4 August, 2017, from https://riunet.upv.es/bitstream/handle/10251/53940/Memoria.pdf;sequence=2

PRÁCTICA #8

DISEÑO DE UNA ESTRUCTURA DE APLICACIÓN EN UN AMBIENTE CLIENTE-SERVIDOR

Competencia(s) a desarrollar.

Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

La arquitectura cliente/servidor

Las aplicaciones en Internet suelen seguir la arquitectura cliente/servidor. Esta arquitectura se caracteriza por descomponer el trabajo en dos partes (es decir dos programas): el servidor, que centraliza el servicio, y el cliente, que controla la interacción con el usuario. El servidor ha de ofrecer sus servicios a través de una dirección conocida. Algunos ejemplos de aplicaciones basadas en la arquitectura cliente/servidor son WWW o el correo electrónico. Se suelen seguir las siguientes pautas de comportamiento en esta arquitectura:

Cliente	Servidor
1. Se conecta al servidor. 2. Solicita alguna información al servidor. 3. Recibe la respuesta. 4. Ir al punto 2. 5. Cierra la conexión	1. A la espera de que algún cliente se conecte. 2. Recibe solicitud. 3. Envía respuesta. 4. Ir al punto 2. 5. Cierra la conexión. 6. Ir al punto 1

¿Qué es un socket?

Cada una de las diferentes aplicaciones en Internet (web, correo electrónico,...) ha de poder intercambiar información entre programas situados en diferentes ordenadores o dispositivos. Con este propósito, se va a hacer uso del nivel de transporte de la pila de protocolos TCP/IP, cuyo objetivo final es permitir el intercambio de información a través de la red de forma fiable y transparente.

Poli[Media]: El Interfaz Socket.

La interfaz socket define las reglas que un programa ha de seguir para utilizar los servicios del nivel de transporte en una red TCP/IP. Esta interfaz se basa en el concepto de socket. Un socket es el punto final de una comunicación bidireccional entre dos programas que intercambian información a través de Internet (socket se traduce literalmente como enchufe).

Dado que en un mismo dispositivo/ordenador podemos estar ejecutando de forma simultánea diferentes aplicaciones que utilizan Internet para comunicarse, resulta imprescindible identificar cada socket con una dirección diferente. Un socket se va a identificar por la dirección IP del dispositivo donde está, mas un número de puerto (de 16 bits). En Internet se suele asociar a cada aplicación un número de puerto concreto (por ejemplo: 80 para la web, 25 para el correo electrónico, 7 para ECHO o 4661 para eDonkey). Una conexión está determinada por un par de sockets, que son los extremos de la conexión.

Existen dos tipos de socket, socket stream y socket datagram. Veamos en qué se diferencian:

Sockets stream (TCP)

Los sockets stream ofrecen un servicio orientado a conexión, donde los datos se transfieren como un flujo continuo sin encuadrarlos en registros o bloques. Este tipo de socket se basa en el protocolo TCP, que es un protocolo orientado a conexión. Esto implica que, antes de transmitir información, hay que establecer una conexión entre los dos sockets. Mientras uno de los sockets atiende peticiones de conexión (servidor), el otro solicita la conexión (cliente). Una vez que los dos sockets están conectados, ya se puede transmitir datos en ambas direcciones. El protocolo incorpora al programador, de forma transparente, la corrección de errores. Es decir, si detecta que parte de la información no llegó a su destino correctamente, ésta volverá a ser trasmitida. Además, no limita el tamaño máximo de información a transmitir.

Sockets datagram (UDP)

Los sockets datagram se basan en el protocolo UDP y ofrecen un servicio de transporte sin conexión. Es decir, podemos mandar información a un destino sin necesidad de realizar una conexión previa. El protocolo UDP es más eficiente que TCP, pero tiene el inconveniente que no se garantiza la fiabilidad. Además, los datos se envían y reciben en datagramas (paquetes de información) de tamaño limitado. La entrega de un datagrama no está garantizada: estos pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió.

La gran ventaja de este tipo de sockets es que apenas introduce sobrecarga sobre la información transmitida. Además, los retrasos introducidos son mínimos, lo que los hace especialmente interesantes para aplicaciones en tiempo real, como la transmisión de audio y vídeo sobre Internet. Sin embargo, presenta muchos inconvenientes al programador: cuando transmitimos un datagrama no tenemos la certeza de que este llegue a su destino, por lo que, si fuera necesario, tendríamos que implementar nuestro propio mecanismo de control de errores. Otro inconveniente es el hecho de que existe un tamaño máximo de datagrama, unos

1.500 bytes dependiendo de la implementación. Si la información a enviar es mayor, tendríamos que fraccionarla y enviar varios datagramas independientes. En el destino tendríamos que concatenarlos en el orden correcto.

En conclusión, si deseas una comunicación libre de errores y sin preocupaciones para el programador es más conveniente que utilices *sockets stream*. Es el tipo de *socket* que utilizaremos en los siguientes ejemplos.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 4.
El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Metodología

Un ejemplo de un cliente/servidor de ECHO

El servicio ECHO suele estar instalado en el puerto 7 de máquinas Unix y permite comprobar que la máquina está operativa y que se puede establecer una conexión con ella.

El funcionamiento de un servidor ECHO es muy sencillo: cuando alguien se conecta espera que le envíe algo y le responde exactamente con la misma información recibida. El cliente actúa de forma contraria; envía datos al servidor y luego comprueba que los datos recibidos son idénticos a los transmitidos.

Practica del Diseño: Un cliente de ECHO.

El siguiente ejemplo muestra cómo podrías desarrollar un cliente de ECHO que utiliza un *socket stream*.

NOTA: Para que el ejemplo funcione en la dirección IP 158.42.146.12 7 ha de haber un servidor de ECHO en funcionamiento. En caso de no ser así, puedes remplazar la IP por la de un servidor de ECHO en funcionamiento o realizar el siguiente ejercicio.

1. Crea la aplicación *ClienteECHO*, donde se cree una actividad con nombre

   ClienteECKO.

2. Remplaza el código por:

   public class ClienteECHO extends Activity {

```
      private extView output;
      @0ver.ride public void onCreate(Bundle savedlnstanceState) {
      super.onCreate(savedlnstanceState);
      setContentView (R. layout.rnain) ;
      output = (TextVi ew) f indViewById (R .i d .Text Vi ewOl) ;
      ejecutadiente () ;
      }
private void ejecutaCliente() {
String ip=": 158.42.146.127" ;
int puerto=7;
log(“ socket " + ip + " " + puerto);
try {
      Socket sk = new Socket(ip,puerto);
      BufferedReader entrada = new BufferedReader(new
InputStreamReader(sk.getlnputStream()));
      PrintWriter salida = new PrintWriter( new
      OutputStreamWriter(sk.getOutputStream()),true);
      log("enviando…");
      salida.println("Hola Mundo");
      log("recibiendo ... " + entrada.readLine());
      sk.close();
    } catch (Exception e) {
        log("error: " + e.toString());
      }
    }
  private void log(String string) {
   output. append (string + " \n" ) ;
  }
}
```

La parte interesante se encuentra en el método ejecutarcliente(). En primer lugar, todo cliente ha de conocer la dirección del socket del servidor; en este caso los valores se indican en el par de variables ip y puerto. Nunca tenemos la certeza de que el servidor admita la conexión, por lo que es obligatorio utilizar una sección try/catch. La conexión propiamente dicha se realiza con el constructor de la clase Socket. Siempre hay que tener previsto que ocurra algún problema, en tal caso, se creará la excepción y se pasará a la sección catch. En caso contrario, continuaremos obteniendo el InputStream y el OutputStream asociado al *socket*. Lo cual nos permitirá obtener las variables, entrada y *salida* mediante las que podremos recibir y transmitir información. El programa transmite la cadena "Hola Mundo", tras lo que visualiza la información recibida. Si todo es correcto, ha de coincidir con lo transmitido.

3. Solicita en la aplicación el permiso INTERNET.

4. Ejecuta la aplicación y verifica si el servidor responde. Recuerda que es posible que no se haya arrancado este servicio en el servidor.

Ejercicio paso a paso: Un servidor de ECHO.

Vamos a implementar un servidor de ECHO en tu ordenador.

1. Crea un nuevo proyecto Java (no para Android) con nombre *ServidorECHO*.
2. Crea en este proyecto la clase ServidorECHO con el siguiente código:

```
public class ServidorECHO {
public static void main(String args[]) {
  try{
     ServerSocket sk = new ServerSocket(7);
     while (true) {
        Socket cliente = sk.accept();
        BufferedReader entrada = new BufferedReader( new
InputStreamReader(cliente.getInputStream() ) ) ;
        PrintWriter salida = new PrintWriter( new
OutputStreamariter(cliente.getOutputStream()),true)
         String datos = entrada.readLine();
         salida.println(datos);
        cliente.close();
         }
   } catch (IOException e) {
      System.out.println(e);
     }
  }
}
```

   En este caso utilizaremos la clase ServerSocket asociada al puerto 7 para crear un *socket* que acepta conexiones. Luego, se introduce un bucle infinito para que el servidor esté perpetuamente en servicio. En la siguiente línea el servidor no se detendrá hasta que un cliente se conecte. Cuando ocurra esto, todo el intercambio de información se realizará a través de un nuevo socket creado con este propósito, el *socket* cliente. El resto del código es similar al cliente, aunque en este caso primero se recibe y luego se transmite lo mismo que ha recibido.
3. Sustituye la dirección IP en *ClienteECHO* por la IP de tu ordenador. El comando ipconfig (Windows) te permite averiguar la dirección IP de tu ordenador. No utilices como IP 127.0.0.1 (*localhost*) dado que, aunque se ejecuten en la misma máquina, la IP del emulador es diferente a la del PC.
4. Ejecuta primero el servidor y luego el cliente para verificar que funciona.

NOTA: Si la IP de tu ordenador es privada no podrás crear un servidor accesible desde cualquier parte de Internet. En este caso, utiliza el emulador o un dispositivo Android

real que se conecte por WiFi a la misma red de tu ordenador. Si no, el terminal no encontrará el servidor.

Sugerencias didácticas.

- Identificar mediante una investigación documental los diferentes sistemas gestores de Bases de datos para móviles, así como sus características y mostrarlas en un cuadro comparativo.
- Diseñar la instalación y uso de un SGBD a través de la elaboración de un tutorial en video.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Subir cuadro comparativo	1 punto	0 punto
El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Coreacuk. (2017). *Coreacuk*. Retrieved 4 August, 2017, from https://core.ac.uk/download/pdf/29406891.pdf

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso

Reggiani, F. (2017). *Crear una base de datos para aplicaciones móviles*. [online] NeoTeo. Available at: http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/ [Accessed 3 Aug. 2017].

App.desarrolloweb.com. (2017). *DesarrolloWeb App*. [online] Available at: https://app.desarrolloweb.com/categorias/android [Accessed 3 Aug. 2017].

Jc-mouse.net. (2017). *Ejemplo de aplicación Android y SQLite*. [online] Available at: http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite [Accessed 3 Aug. 2017].

Amazonawscom. (2017). *Amazonawscom*. Retrieved 4 August, 2017, from https://cenfotec.s3-us-west-2.amazonaws.com/prod/wpattchs/2013/12/desarrollo-de-aplicaciones-android-fundamentos.pdf

PRÁCTICA #9

DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MOVILES CON ENFOQUE CLIENTE Y SERVIDOR

Competencia(s) a desarrollar.

Conocer y aplicar tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

Los teléfonos Android suelen disponer de conexión a Internet. Esto nos permite no solo almacenar los datos en nuestro dispositivo, si no compartirlos con otros usuarios. En el primer punto del capítulo trataremos de resolver el problema de comunicar dos aplicaciones en Internet mediante la herramienta básica: los *sockets*. Existen otras alternativas de más alto nivel, como el uso del protocolo HTTP, que será estudiada en el segundo punto del capítulo. Para terminar se tratará una tercera alternativa, todavía de más alto nivel, los servicios web.
En este capítulo implementáramos el mismo ejemplo que en el capítulo anterior, es decir, almacenaremos las puntuaciones obtenidas en Asteroides, pero ahora en un servidor de Internet. Utilizaremos las tres alternativas descritas en el párrafo anterior. No obstante, has de tener claro que estos mecanismos están relacionados entre sí. Por ejemplo, si utilizas servicios web, internamente se utilizará el protocolo HTTP y además este protocolo utiliza *sockets* para establecer la comunicación.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 4.

El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD

Material y equipo necesario

PC o Laptop, Android instalado y configurado e Internet

Metodología

PRÁCTICA de un servidor por sockets para las puntuaciones

Siguiendo la estructura básica de un cliente y un servidor TCP que acabamos de ver, va a resultar muy sencillo implementar la interfaz AlmacenPuntuaciones

para que varios clientes con dispositivos Asteroides puedan conectarse a un servidor para intercambiar puntuaciones.

El primer lugar tenemos que diseñar un protocolo que permita realizar las dos operaciones que ha de implementar la interfaz. Un posible ejemplo de interacción cliente/servidor se muestra a continuación:

Para consultar puntuaciones:

Cliente: PUNTUACIONES\ n

Servidor: 19000 Pedro Perez\n

17500 María Suarez\n

13000 Juan García\n

Para almacenar puntuaciones:

Cliente: 32000 Eva Gutierrez\n

Servidor: 0K\n

En segundo lugar, hay que elegir un número de puerto para realizar la comunicación; por ejemplo el 1234.

Ejercicio paso a paso: Almacenando las puntuaciones mediante un protocolo basado en sockets.

1. Crea un nuevo proyecto Java para el servidor (Archivo/Nuevo/Proyecto.../ Proyecto Java) con nombre ServidorPuntuacionesSocket.
2. Pulsa con el botón derecho en la carpeta *src* de este proyecto y selecciona *Nueva/Clase*. Introduce como nombre *ServidorPuntuaciones*.
3. Remplaza en código por el que se muestra a continuación:

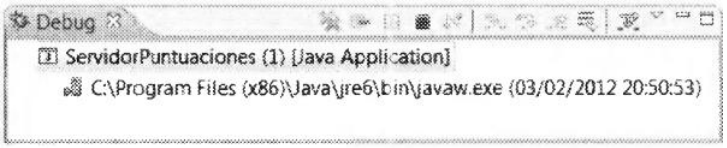
```
public class ServidorPuntuaciones {
  public static void main (String args[]) {
    Vector<String> puntuaciones = new Vector<String>();
   try { ServerSocket s = new ServerSocket(1234);
   System, out. println ( "Esperando conexiones. . . , ") ;
   while (true) {
     Socket cliente = s.acceptO;
     BufferedReader entrada = new BufferedReader(
     new InputStreamReader(cliente.getInputStream()));
     PrintWriter salida = new PrintWriter( new
     OutputStreamWriter(cliente.getOutputStream()),true);
     String datos = entrada.readLine();
     if (datos.equals("PUNTUACIONES")) {
```

```
          for (int n = 0; n<puntuaciones.size() ; n++ ){
          salida.println(puntuaciones.get(n));
            }
          } else {
              puntuaciones.add(0, datos);
              salida. println ("OK");
             } cliente.close ();
          }
        } catch (IOException e) {
        System.out.println(e);
      }
  }
}
```

4. Ejecuta el proyecto como Aplicación Java.

5. Verifica que en la vista Consola aparece: "Esperando conexiones..."

6. Cambia la perspectiva a modo Debug y Observa la vista de la esquina superior izquierda.



Desde esta vista podrás detener la aplicación pulsando el cuadro rojo.

NOTA: Si ejecutas de nuevo la aplicación dará un error. Esto es debido a que la aplicación ya lanzada no es detenida y esta aplicación tiene asociado el puerto 1234. El sistema no permitirá que una nueva aplicación escuche este puerto.

7. Abre el proyecto Asteroides y crea la siguiente clase:

```
public class AlmacenPuntuacionesSocket implemente AlmacenPuntuaciones{
  public void guardarPuntuacion(int puntos, String nombre, long fecha);
  try {
      Socket sk = new Socket("X.X.X.X", 1234);
      BufferedReader entrada = new BufferedReader( new
      InputStreamReader(sk.getInputStream()));
      PrintWriter salida = new PrintWriter( new
      OutputStreamWriter(sk.getOutputStream()),tnie);
      sal ida. println (puntos + ,f !I + nombre);
      String respuesta = entrada.readLine();
      if (!respuesta.equals("OKM) ) {
          Log.e("Asteroides", "Error: respuesta de servidor incorrecta");
          } sk.cióse () ;
```

```
      } catch (Exception e) {
      Log.e("Asteroides", e .toString(), e);
  }
}
public Vector<String> listaPuntuaciones(int cantidad) {

Vector<String> result = new Vector<String>();
try {
   Socket sk = new Socket("X.X.X.X", 1234);
   BufferedReader entrada = new BufferedReader( new
   InputStreamReader(sk.getInputStream()));
   PrintWriter salida = new PrintWriter( new
   OutputStreamWriter(sk.getOutputStreamO),true)
   salida.println("PUNTUACIONES"); int n = 0;
   String respuesta;
   do {
     respuesta = entrada.readLine();
     if (respuesta != nuil) {
     result.add(respuesta);
     n+ + ;
     }
   } while (n < cantidad && respuesta != nuil);
   sk.cióse();
   } catch (Exception e) {
       Log.e("Asteroides", e .toString(), e);
       } return result;
   }
}
```

8. Has de sustituir las dos apariciones de "x.x.x.x" por la dirección IP donde esté ejecutándose el servidor.
   NOTA: El comando ipconfig (Windows) te permite averiguar la dirección IP de tu ordenador. No utilices como IP 127.0.0.1 (localhost) dado que, aunque se ejecuten en la misma máquina, la IP del emulador es diferente a la del PC.

9. También has de recordar que ahora la aplicación Asteroides necesita el permiso INTERNET.
10. Ejecuta Asteroides y accede a visualizar la lista de puntuaciones y luego echa una partida nueva.
11. Verifica que en la vista Consola aparecen las consultas.
12. Para terminar, reemplaza la IP por la siguiente "158.42.140.127" para conectarte a un servidor compartido.

    NOTA: Es posible que este servicio no haya sido iniciado.

13. Comprueba si otros usuarios han accedido a este servidor y aparecen sus puntuaciones.

Sugerencias didácticas.

- Identificar mediante una investigación documental los diferentes sistemas gestores de Bases de datos para móviles, así como sus características y mostrarlas en un cuadro comparativo.
- Mostrar la instalación y uso de un SGBD a través de la elaboración de un tutorial en video.

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Subir a la nube el video generado	1 punto	0 punto
El documento de la investigación está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso

Reggiani, F. (2017). *Crear una base de datos para aplicaciones móviles*. [online] NeoTeo. Available at: http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/ [Accessed 3 Aug. 2017].

App.desarrolloweb.com. (2017). *DesarrolloWeb App*. [online] Available at: https://app.desarrolloweb.com/categorias/android [Accessed 3 Aug. 2017].

Jc-mouse.net. (2017). *Ejemplo de aplicación Android y SQLite*. [online] Available at: http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite [Accessed 3 Aug. 2017].

Amazonawscom. (2017). *Amazonawscom*. Retrieved 4 August, 2017, from https://cenfotec.s3-us-west-2.amazonaws.com/prod/wpattchs/2013/12/desarrollo-de-aplicaciones-android-fundamentos.pdf

PRÁCTICA #10

DISEÑO DE UNA APLICACION MOVIL QUE INTEGRE BASES DE DATOS

Competencia(s) a desarrollar.

Conoce y aplica tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias. Sin embargo, en este primer artículo sobre bases de datos en Android no vamos a entrar en mucho detalle con esta API. Por el momento nos limitaremos a ver el código necesario para crear una base de datos, insertaremos algún dato de prueba, y veremos cómo podemos comprobar que todo funciona correctamente.

Correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 4 de la unidad.

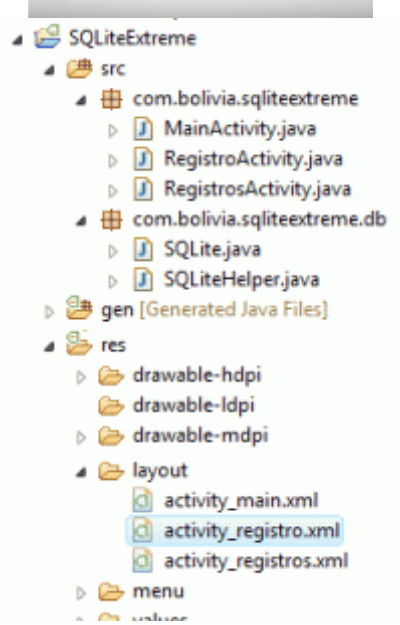
El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD

Material y equipo necesario

PC o Laptop, Android instalado y configurado

Metodología

Esta práctica se realiza de una aplicación que hace uso de **base de datos *SQLite***. En este proyecto, se cuenta con las opciones, de *inserción*, *consulta* y *eliminación* de registros, ademas se hace uso de paso de parámetros entre activities, llenado de registros en un ***ListView***, ***Toast***, **DatePicker** y **Dialog**.



La aplicación, hace uso de una sola tabla **UNIVERSITARIO** que consta de 6 campos, donde la llave primaria es auto incrementable de tipo entero.

CREATE TABLE "Universitario"("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE, “Nombre" TEXT, "FechaNac" DATETIME, "Pais" TEXT, "Sexo" TEXT, "Ingles" TEXT)

El proyecto hace uso de tres *layout* siendo **activity_main** el primero en mostrarse y donde se registran nuevos alumnos, si el registro tuvo éxito, se mostrara el layout **activity_registro**, este *layout* permite volver al layout anterior para agregar nuevos registros o eliminar el registro que este visible, también cuenta con un botón para ver la lista de registros en el *layout* **activity_registros** donde se llenaran en un ListView, cuando se realice un clic en cualquier item, se mostraran sus datos en el layout activity_registro.

Sugerencias didácticas.

- Diseño de una aplicación que gestionan bases de datos desde dispositivos móviles y presentar reporte de funcionamiento.
- Realizar el diseño de la aplicación web orientada a dispositivos móviles con conexión a base de datos, mostrar resultados de las operaciones realizadas

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Presentación del Prototipo Aplicación Móvil	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso

Reggiani, F. (2017). *Crear una base de datos para aplicaciones móviles*. [online] NeoTeo. Available at: http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/ [Accessed 3 Aug. 2017].

App.desarrolloweb.com. (2017). *DesarrolloWeb App*. [online] Available at: https://app.desarrolloweb.com/categorias/android [Accessed 3 Aug. 2017].

Jc-mouse.net. (2017). *Ejemplo de aplicación Android y SQLite*. [online] Available at: http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite [Accessed 3 Aug. 2017].

PRÁCTICA #11

DESARROLLO DE UNA APLICACIÓN MOVEL QUE INTEGRE BASES DE DATOS

Objetivo

Conocer y aplicar tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias. Sin embargo, en este primer artículo sobre bases de datos en Android no vamos a entrar en mucho detalle con esta API. Por el momento nos limitaremos a ver el código necesario para crear una base de datos, insertaremos algún dato de prueba, y veremos cómo podemos comprobar que todo funciona correctamente.

Correlación con el o los temas y subtemas del programa de estudio vigente.

Esta práctica está directamente relacionados con la unidad 4 de la unidad.

El propósito es que los estudiantes tengan una idea clara sobre la instalación y uso de un SGBD

Material y equipo necesario

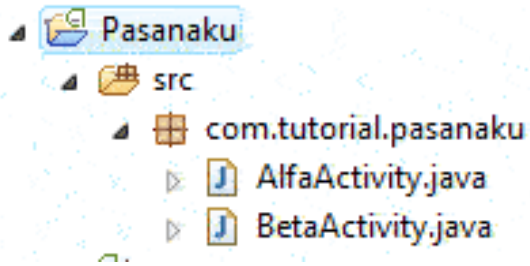
PC o Laptop, Android instalado y configurado

Metodología

En esta práctica veremos cómo pasar parámetros de un **activity** a otro **activity**, no hay mucho que decir así que manos a la obra.

1. Crea un nuevo **Application Android Project** con la siguiente configuración:
   Application Name: Pasanaku
   Project Name: Pasanaku
   Package Name: com.tutorial.Pasanaku
   Activity: Blank Activity
   Activity Name: AlfaActivity.java
   Layout Name: activity_alfa.xml
2. Agrega un segundo activity al proyecto.
   NEW -> OTHER -> ANDROID ACTIVITY
   Activity Name: BetaActivity.java
   Layout Name: activity_beta.xml

Nuestra aplicación esta ahora formado por 2 actividades, ***Alfa*** la primera será de donde se pasen los parámetros y la activity **Beta**, sera quien los reciba y muestre en pantalla.



3. El código XML de **alfa_activity** es:

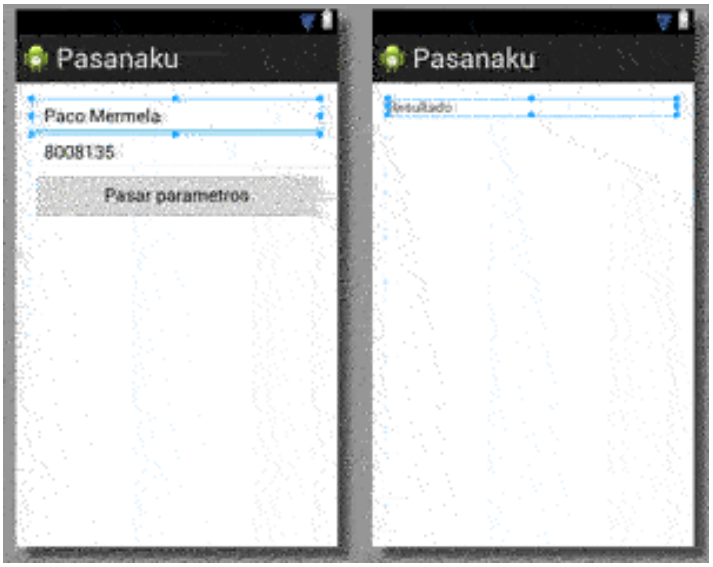
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".AlfaActivity" >

    <EditText
        android:id="@+id/txtName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="@string/strName" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/txtNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"

    <Button
        android:id="@+id/btnGo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/strButton" />

</LinearLayout>
```

4. El código XML de **beta_activity** es:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".BetaActivity" >

    <TextView
        android:id="@+id/txtResultado"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/strResultado" />

</RelativeLayout>
```

5. Finalmente nuestro archivo **String.xml** queda de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Pasanaku</string>
    <string name="action_settings">Settings</string>
    <string name="strName">Paco Mermela</string>
    <string name="strNumber">8008135</string>
    <string name="strButton">Pasar parametros</string>

    <string name="title_activity_beta">Pasanaku 2</string>
 <string name="strResultado">Resultado</string>

</resources>
```



6. El código de la clase **AlfaActivity.java**, es quien captura los datos de la interfaz y prepara todo para abrir el segundo layout pasando los parámetros.

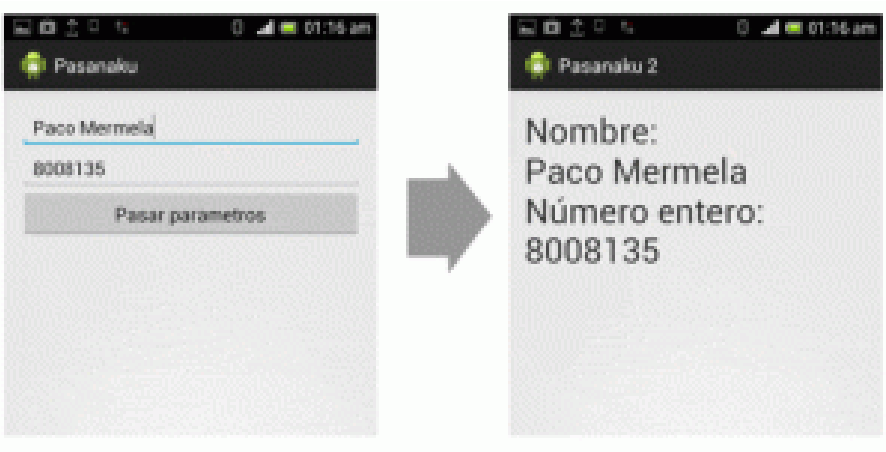
```
package com.tutorial.pasanaku;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AlfaActivity extends Activity implements OnClickListener{

 //variables utilizadas en la aplicacion
 private EditText txtName;
 private EditText txtNumber;
 private Button btnGo;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_alfa);

  //Referencia a los objetos del layout
  txtName = (EditText) findViewById( R.id.txtName );
  txtNumber = (EditText) findViewById( R.id.txtNumber );
  btnGo = (Button) findViewById( R.id.btnGo );
  btnGo.setOnClickListener( this );
  //+++
 }


@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.alfa, menu);
 return true;
}
```

```
@Override
public void onClick(View v) {

 //Si las cadenas no estan vacias
 if( txtName.length()>0 && txtNumber.length()>0 )
 {
  //bundle nos permite almacenar valores de la siguiente forma
  // bundle.putString( clave, valor );
  // pudiendo BUNDLE alamcenar valores de todo tipo
  Bundle bundle = new Bundle();
  bundle.putString("Nombre" , txtName.getText().toString() );
  try
  {
   int numero = Integer.valueOf(txtNumber.getText().toString());
   bundle.putInt( "Numero" , numero );
  }
  catch ( NumberFormatException ex ){
   //Si el valor ingresado no es un numero INT asigna 0
   bundle.putInt( "Numero" , 0 );
  }
  //Intent nos permite enlazar dos actividades
  Intent intent = new Intent( AlfaActivity.this, BetaActivity.class );
  //añadir parametros
  intent.putExtras( bundle );
  //ejuta intent
  startActivity( intent );
 }
 else
 {
  //si no existen valores en los objetos EditText muestra un mensaje
  Toast toast = Toast.makeText(getBaseContext(),"Debes escribir un Nombre y un número en
tero",Toast.LENGTH_SHORT );
  toast.show();
 }

}

}
```

- Finalmente la clase **BetaActivity.java** que corresponde al segundo layout y donde se muestran los parámetros pasados desde el primer layout.

```
package com.tutorial.pasanaku;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.widget.TextView;

public class BetaActivity extends Activity {

 //Variable utilizadas en la clase
 private TextView textView;
 private StringBuilder mensaje = new StringBuilder();

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_beta);
 //Referencia a los objetos del layout
 textView = (TextView) findViewById( R.id.txtResultado );
 //Recupera parametros y los muestra en el TextView
 Intent intent = getIntent();
 Bundle bundle = intent.getExtras();
  if ( bundle != null ) {
   mensaje.append("Nombre: \r\n");
   mensaje.append( bundle.getString("Nombre") + "\r\n" );
   mensaje.append("Número entero: \r\n");
   mensaje.append( bundle.getInt("Numero") );
  }
  textView.setTextSize(33);
  textView.setText( mensaje );
 //++++++++

 }

 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu; this adds items to the action bar if it is present.
  getMenuInflater().inflate(R.menu.beta, menu);
  return true;
 }

}
```

Ejecutando nuestra aplicación:



Android hace uso de la base de datos **SQLite** para el manejo de registros en las aplicaciones. Según Santa Wikipedia define SQLite como:

"SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. "

"SQLite usa un sistema de tipos inusual. En lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales. Por ejemplo, se puede insertar un string en una columna de tipo entero (a pesar de que SQLite tratará en primera instancia de convertir la cadena en un entero). "

Para hacer uso de una base de datos SQLite, android dispone de la clase **SQLiteOpenHelper** que tiene los métodos para la creación de base de datos y el control de versiones. Veremos como hacer uso de esa clase en un sencillo proyecto a continuación.

Utilizaremos Eclipse Indigo con Android instalado y configurado.

Proyecto de manejo a base de datos SQLite.

1. Crea un nuevo **Application Android Project** con la siguiente configuración:

   Application Name: SQLite Example

   Project Name: SQLiteExample

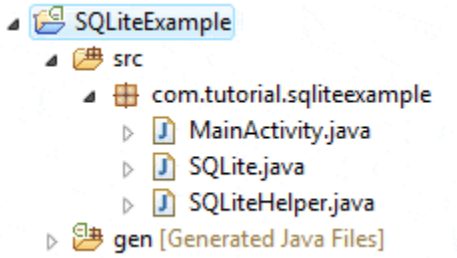
   Package Name: com.tutorial.sqliteexample

   Activity: Blank Activity

   Activity Name: MainActivity.java

Layout Name: activity_main.xml

2. Crearemos dos clases en el paquete **com.tutorial.sqliteexample**, estas se llamaran, "**SQLiteHelper.java**" y "**SQLite.java**"



3. La clase **SQLiteHelper**, se extiende de **SQLiteOpenHelper** y se sobreescriben los métodos onCreate() y onUpgrade().

   El método **oncreate()** sirve para crear la base de datos, android verifica si la aplicación no tiene base de datos, entonces ejecuta este método.

   El método **onUpgrade()**, es utilizada para el manejo y control de versiones, cuando se distribuye una nueva versión de un programa y se desea modificar la base de datos, ya sea con más o menos tablas y campos, quiere decir que se desea pasar a una **"nueva versión"** de base de datos, esto se maneja internamente con una variable de tipo entero "versión" y es aquí donde el método *onUpgrade()* se ejecuta y verifica si una nueva versión de nuestra aplicación hace uso de una nueva "**versión**" de base de datos o de lo contrario sigue usando la misma.

   Clase SQLiteHelper

```
package com.tutorial.sqliteexample;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class SQLiteHelper extends SQLiteOpenHelper {

 //nombre de la base de datos
 private static final String __database__ = "dbTest";
 //versión de la base de datos
 private static final int __version__ = 1;
 //Instrucción SQL para crear las tablas
 private String sql = "CREATE TABLE mi_tabla ( id INTEGER, nombre TEXT, apellido TEXT )";

 /**
 * Constructor de clase
 * */
 public SQLiteHelper(Context context) {
  super( context, __database__, null, __version__ );
 }
```

```
 @Override
 public void onCreate(SQLiteDatabase db) {
   db.execSQL( sql );
   Log.i("SQLite", "Se crea la base de datos " + __database__ + " version " + __version__
 );
 }

 @Override
 public void onUpgrade( SQLiteDatabase db,  int oldVersion, int newVersion ) {
  Log.i("SQLite", "Control de versiones: Old Version=" + oldVersion + " New Version= " +
newVersion  );
  if ( newVersion > oldVersion )
  {
   //elimina tabla
   db.execSQL( "DROP TABLE IF EXISTS mi_tabla" );
   //y luego creamos la nueva tabla
   db.execSQL( sql );
   Log.i("SQLite", "Se actualiza versión de la base de datos, New version= " + newVersion
  );
  }
 }

}
```

Nuestra base de datos se llama **dbTest** y consta de una sola tabla con dos campos, es más que suficiente para lo que deseamos hacer en esta oportunidad.

Vemos que se hace uso de una variable "***__version__***" de tipo entero, esta indica la versión de nuestra base de datos, más adelante veremos como pasar a una segunda versión modificando esta variable.

Nuestro método onUpgrade(), cuando android detecte que existe una nueva versión de base de datos, eliminará la tabla antigua y creara una nueva, esto claro es un ejemplo simple, en la práctica, dependiendo del tamaño de la base de datos y de los cambios que deseemos hacer este método será mucho más trabajado.

4. Una vez que tenemos nuestra clase **SQLiteHepler**, debemos crear otra clase más que haga uso de la anterior, esta clase es **SQLite.java**

SQLite.java

```
package com.tutorial.sqliteexample;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

public class SQLite {

 SQLiteHelper sqliteHelper;
 SQLiteDatabase db;

 /** Constructor de clase */
 public SQLite(Context context)
 {
  sqliteHelper = new SQLiteHelper( context );
 }
```

```
 /** Abre conexion a base de datos */
 public void abrir(){
  Log.i("SQLite", "Se abre conexion a la base de datos " + sqliteHelper.getDatabaseName()
 );
  db = sqliteHelper.getReadableDatabase(); // solo lectura
 }

 /** Cierra conexion a la base de datos */
 public void cerrar()
 {
  Log.i("SQLite", "Se cierra conexion a la base de datos " + sqliteHelper.getDatabaseName
() );
  sqliteHelper.close();
 }
}
```

La clase ***SQLite.java*** cuenta con 2 métodos para abrir y cerrar conexiones a la base de datos, por el momento es todo lo que necesitamos. A medida que vayamos necesitando mas funcionalidad en la aplicación ya sea para *insertar*, *eliminar* o *actualizar* registros, estos deberán ser colocados en esta clase.

Tanto en la clase *SQLiteHelper* y *SQLite*, hacemos uso de Log.i(), estos logs nos permiten mostrar *información*, *errores*, *warnings*, etc en el LogCat de Eclipse y así poder depurar nuestra aplicación. en este caso **Log.i** nos permite mostrar información.

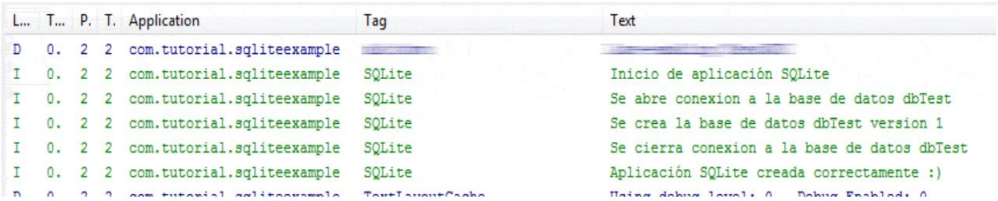
5. Para terminar abrimos nuestra clase ***MainActivity.java*** e implementamos la clase **SQLite.java** de la siguiente manera.

```
package com.tutorial.sqliteexample;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.Menu;

public class MainActivity extends Activity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  //
  Log.i("SQLite", "Inicio de aplicación SQLite" );
  SQLite sqlite = new SQLite( this );
  sqlite.abrir();
  sqlite.cerrar();
  Log.i("SQLite", "Aplicación SQLite creada correctamente ☺ " );
  //
 }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
}

}
```

En esta proyecto, no hacemos uso de interfaces XML porque nuestra aplicación lo único que hará es crear la base de datos, conectarse y desconectarse, en la interfaz del teléfono no veremos nada. Nuestros ojos deberán estar atentos al **LogCat**.

Ejecutamos por **PRIMERA** vez y tenemos:



L...	T...	P.	T.	Application	Tag	Text
D	0.	2	2	com.tutorial.sqliteexample	[redacted]	[redacted]
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Inicio de aplicación SQLite
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se abre conexion a la base de datos dbTest
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se crea la base de datos dbTest version 1
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se cierra conexion a la base de datos dbTest
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Aplicación SQLite creada correctamente :)

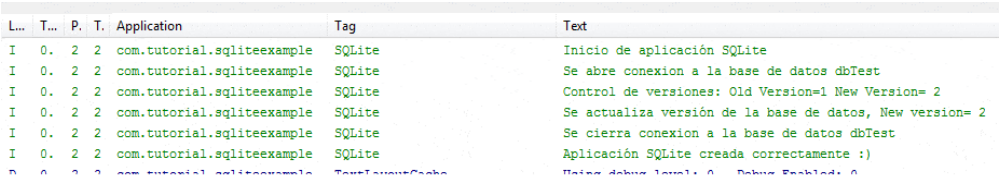
Como es la primera vez que ejecutamos nuestra aplicación, la base de datos **NO EXISTE**, por tanto se ejecuta el método **onCreate()** para crear la base de datos.

Ejecutamos por **SEGUNDA VEZ**, pero antes, debemos cambiar a **"2"** la variable ***__version__*** de la clase **SQLiteHelper**.

```
private static final int __version__ = 2;
```

En el ***LogCat*** tendremos:

Observamos como esta vez, la base de datos ya existe, por tanto no ejecuta el método **onCreate()**, pero como modificamos la versión de la base de datos a 2, se ejecuta el método **onUpgrade()**, actualiza a la nueva versión y cierra la conexión.



L...	T...	P.	T.	Application	Tag	Text
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Inicio de aplicación SQLite
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se abre conexion a la base de datos dbTest
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Control de versiones: Old Version=1 New Version= 2
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se actualiza versión de la base de datos, New version= 2
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Se cierra conexion a la base de datos dbTest
I	0.	2	2	com.tutorial.sqliteexample	SQLite	Aplicación SQLite creada correctamente :)

Eso es todo para comenzar con base de datos de android, más adelante iremos implementados los diferentes métodos de gestión como ser **INSERT**, **DELETE**, **UPDATE** Y **SELECT**.

Sugerencias didácticas.

- Desarrollar e implementar una aplicacion que gestionan bases de datos desde dispositivos móviles y presentar reporte de funcionamiento.
- Realizar el desarrollo de una aplicación web orientada a dispositivos móviles con conexión a base de datos, mostrar resultados de las operaciones realizadas

Reporte del alumno (discusión de resultados y conclusiones).

Criterio para calificar	Sí	No
Presentacion de la Aplicación Movil	1 punto	0 punto
Reporte de Funcionamiento	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso

Reggiani, F. (2017). *Crear una base de datos para aplicaciones móviles*. [online] NeoTeo. Available at: http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/ [Accessed 3 Aug. 2017].

App.desarrolloweb.com. (2017). *DesarrolloWeb App*. [online] Available at: https://app.desarrolloweb.com/categorias/android [Accessed 3 Aug. 2017].

Jc-mouse.net. (2017). *Ejemplo de aplicación Android y SQLite*. [online] Available at: http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite [Accessed 3 Aug. 2017].

PRÁCTICA #12

PROYECTO INTEGRADOR

Competencia(s) a desarrollar.

Conocer y aplicar tecnologías de conectividad a bases de datos actuales y emergentes para el desarrollo de aplicaciones móviles.

Introducción

El alumno generara un proyecto de aplicación móvil, con base de datos como proyecto final para evaluar el curso.

Especificar la correlación con el o los temas y subtemas del programa de estudio vigente.

Todas las unidades.

El propósito es que los estudiantes haya dominada el diseño y desarrollo de las aplicaciones móviles

Material y equipo necesario

PC o Laptop
Android instalado y configurado
XAMPP
Internet

Metodología

El Alumno elaborará una aplicación móvil integrando cada una de las competencias logradas durante el curso en la cual:

- Se evidencia comprensión total del problema
- Incluye todos los elementos requeridos en la actividad

Sugerencias didácticas.

Elaboración de proyectos donde el estudiante resuelva un problema de su entorno mediante la programación para dispositivos móviles:

- Interfaz
- Base de Datos
- Alcance de la Aplicación

Reporte del alumno

El proyecto de la Aplicación Móvil:

Criterio para calificar	Sí	No
Presentación de la Aplicación Móvil	1 punto	0 punto
Incluye todos los elementos	1 punto	0 punto
Reporte de Funcionamiento	1 punto	0 punto
El documento está en formato pdf	1 punto	0 punto
El documento tiene portada	1 punto	0 punto
Entrego en la fecha programada	1 punto	0 punto

Bibliografía (emplear formato APA)

Amaro Soriano, José Enrique. *Android: Programación de dispositivos móviles a través de ejemplos.* n.p.: Marcombo, S.A., 2011. Impreso

Reggiani, F. (2017). *Crear una base de datos para aplicaciones móviles*. [online] NeoTeo. Available at: http://www.neoteo.com/crear-una-base-de-datos-para-aplicaciones-moviles/ [Accessed 3 Aug. 2017].

App.desarrolloweb.com. (2017). *DesarrolloWeb App*. [online] Available at: https://app.desarrolloweb.com/categorias/android [Accessed 3 Aug. 2017].

Jc-mouse.net. (2017). *Ejemplo de aplicación Android y SQLite*. [online] Available at: http://www.jc-mouse.net/proyectos/ejemplo-de-aplicacion-android-y-sqlite [Accessed 3 Aug. 2017].