Tecnológico Nacional de México

Reporte Final del Ejercicio Sabático

ELABORACION DE MATERIALES, RECURSOS O AUXILIARES DIDACTICOS.

Julio César Flores López

Manual de Prácticas de Desarrollo Aplicaciones Dispositivos Móviles

14 de Febrero de 2018 al 13 de Agosto de 2018

Introducción

La materia de Desarrollo de Aplicaciones de Dispositivos Móviles (DADM) es una continuación de la programación Java y la materia aporta al estudiante conocimientos adicionales a la programación orientada a objetos, incluyendo herramientas como la Interface Gráfica de Usuario (GUI) en particular se empleará Android.

El estilo de los ejercicios es para ser seguido paso a paso. Más que dar una explicación profunda sobre los conceptos se centra en la experiencia de ser maestro de los cursos de la materia DADM impartida en el salón de clases. Este documento está escrito para alumnos que tienen dominio de los conceptos básicos de Java, y docentes que puedan apoyar su curso empleando como guía los ejercicios mostrados.

Los ejercicios están organizados en función del programa oficial de la materia. Estas prácticas, incluyen una breve descripción de cada una de las unidades, en cada tema se muestran los ejercicios en forma individual. En total se muestran las 15 prácticas.

Las prácticas siguen el siguiente formato: Número de práctica, título, objetivo, correlación con temas y subtemas del programa de estudio, introducción, metodología, sugerencias didácticas y equipo necesario, procedimiento y bibliografía preliminar.

OBJETIVO GENERAL

Elaborar un Manual de prácticas que permita a los estudiantes de Desarrollo de Aplicaciones de Dispositivos Móviles crear aplicaciones computacionales integrando la programación Java por medio del entorno de desarrollo Android, utilizando herramientas de programación para el desarrollo de aplicaciones móviles.

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.	**No. 1. Instalar Interface Gráfica de Usuario Android.** Objetivo. Instalar la herramienta Interface Gráfica de Usuario Android. **No. 2. Configurar Android.** Objetivo. Establecer una configuración de desarrollo en Android. **No. 3. Creación y estructura de un Proyecto.** Objetivo. Crear una primera aplicación Android.
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.	**No. 4. Unidades y layouts.** Objetivo. Crear una aplicación Android con varias actividades. **No. 5. Usando Listas.** Objetivo. Empleo de la librería ListView. **No. 6. Listas personalizadas.** Objetivo. Conocer los elementos comunes en una interface gráfica. **No. 7. Elementos gráficos.** Objetivo. Crear una librería gráfica personal Círculo.

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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.
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.	**No. 13. Geolocalización.** Objetivo. Conocer las formas de almacenar información. **No. 14. JSON.** Objetivo. Conocer las formas de almacenar información. **No. 15. XML.** Objetivo. Conocer las formas de almacenar información.

No. 01. Instalar Interface Gráfica de Usuario Android.

OBJETIVO

Instalar una herramienta de desarrollo con Interface Gráfica de Usuario para Android.

TEMARIO

TEMA	NÚM. 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.	**No. 1. Instalar Interface Gráfica de Usuario Android.** Objetivo. Instalar la herramienta Interface Gráfica de Usuario Android. **No. 2. Configurar Android.** Objetivo. Establecer una configuración de desarrollo en Android. **No. 3. Creación y estructura de un Proyecto.** Objetivo. Crear una primera aplicación Android.

Introducción

Java es un lenguaje de programación orientado a objetos ampliamente utilizado. Las aplicaciones Java se ejecutan en Android, Windows, Mac OS X, Linux, Solaris etc.

Una interfaz gráfica para usuarios es una aplicación que proporciona varios servicios para facilitarle al programador el desarrollo de software [1].

El software requerido para instalar el Entorno de Desarrollo Integrado, en inglés Integrated Development Environment (IDE) es Android Studio y JDK (Ambiente de desarrollo de Java).

Android Studio es gratuito y se usa bajo licencia de Google. Java también es software gratuito y se usa bajo licencia "COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0"

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con los requisitos mínimos siguientes: [2]
   - Sistema Operativo: Microsoft Windows® 7/8/10
   - Procesador 64 bits, soporte en Intel VT-x, EM64T y XD. En AMD AMD-V y SSSE3
   - Memoria: 4 GB mínimo.
   - Espacio en disco: 4 GB libre sugerido.
   - Resolución de video 1280 x 800 mínimo.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso al recurso Wiki del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Conectarse a Internet para obtener el software de las páginas oficiales.
3. Seguir el procedimiento de la práctica.
4. Probar que el software esté funcionando.

Sugerencias didácticas

1. Realizar una investigación documental de los dispositivos Android.
2. Seleccionar a varios participantes para demostrar el funcionamiento de Android Studio.
3. Llenar el recurso de Wiki con las funciones más importantes de Android.

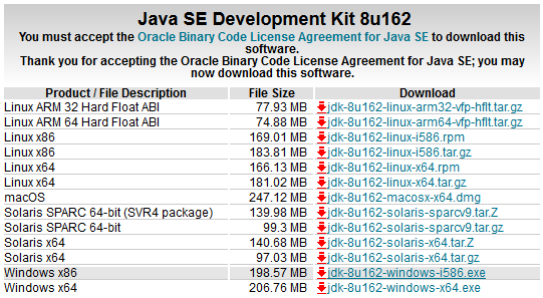
Reporte de los alumnos (resultados)

1. Elaborar ventajas de usar Netbeans con Android, Eclipse con Android, MIT App Inventor, Xamarin, Basic4android, Appcelerator Titanium, ItelliJ IDEA, Scripting Layer For Android, AppInventor. Elaborar una matriz de software para el desarrollo de aplicaciones en Android mostrando ventajas y desventajas.
2. Participar en Moodle en el glosario de términos incorporando software de desarrollo para Android.

Procedimiento:

1. Descargar e instalar el Software Java.
   a. Verificar tener instalado el JDK (Java Development Kit). Si es así ir al paso 2. En caso contrario seguir las instrucciones siguientes.
   b. En un navegador busque "java jdk download". Lo cual le mostrará varias ligas, de las cuales le mostrará una que diga "Descargas de Java SE - Oracle" y nos muestra una imagen como:

   c. Se selecciona Java Download. Damos aceptar la licencia.



Java SE Development Kit 8u162

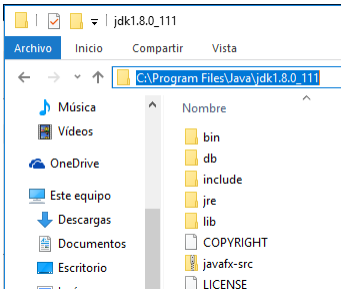
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

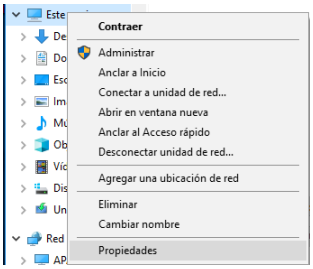
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.93 MB	jdk-8u162-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u162-linux-arm64-vfp-hflt.tar.gz
Linux x86	169.01 MB	jdk-8u162-linux-i586.rpm
Linux x86	183.81 MB	jdk-8u162-linux-i586.tar.gz
Linux x64	166.13 MB	jdk-8u162-linux-x64.rpm
Linux x64	181.02 MB	jdk-8u162-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u162-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.98 MB	jdk-8u162-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.3 MB	jdk-8u162-solaris-sparcv9.tar.gz
Solaris x64	140.68 MB	jdk-8u162-solaris-x64.tar.Z
Solaris x64	97.03 MB	jdk-8u162-solaris-x64.tar.gz
Windows x86	198.57 MB	jdk-8u162-windows-i586.exe
Windows x64	206.76 MB	jdk-8u162-windows-x64.exe

   d. Descargamos el software (verificar si su equipo soporta 32 o 64 bits) y lo ejecutamos y damos "Next" hasta que termine la instalación.

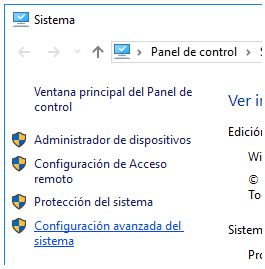
e. Android Studio requiere Java, y para poder localizar el JDK se requiere dar de alta una variable de entorno “JAVA_HOME”. Localizamos en donde está instalado el JDK. Usamos el explorador de archivos por lo general lo encontramos en “Archivos de programa” o “Program Files”. Realizamos una copia de la trayectoria.



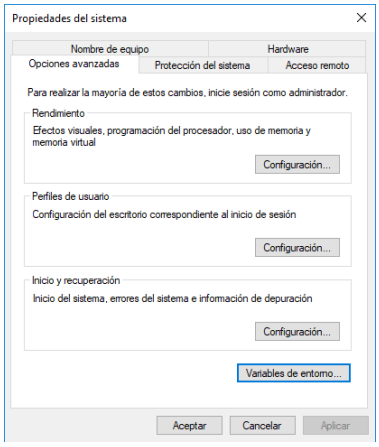
f. Una vez terminada vamos a dar de alta la variable de entorno “JAVA_HOME”. Usamos el explorador de archivos. Encontramos el icono “Este equipo” y con el botón derecho seleccionamos propiedades.



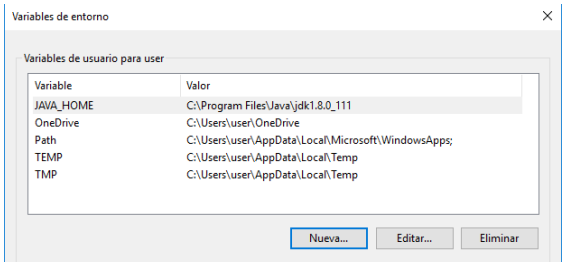
g. Una vez abierto propiedades. Seleccionamos “Configuración avanzada del sistema”.



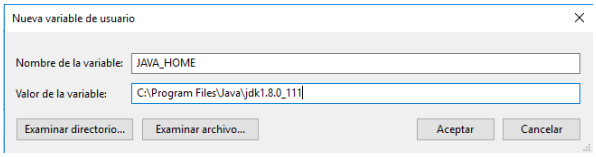
h. Y luego en la siguiente pantalla. Seleccionamos variables de entorno.



i. En la parte superior en donde dice: “Variables de usuarios para …” seleccionamos “Nueva”.

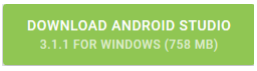


j. Del paso 1-e escribimos lo siguiente.

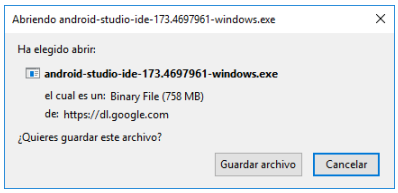


k. Y finalmente damos “Aceptar” y en la pantalla previa “Aceptar”. Y de nuevo “Aceptar”.

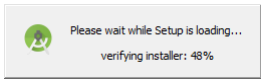
2. Descargar e instalar el software Android Studio.
   a. En un navegador busque “android studio download”. Nos va a llevar a la liga: https://developer.android.com/studio/index.html. Y la abrimos.
   b. Damos clic en la liga siguiente.



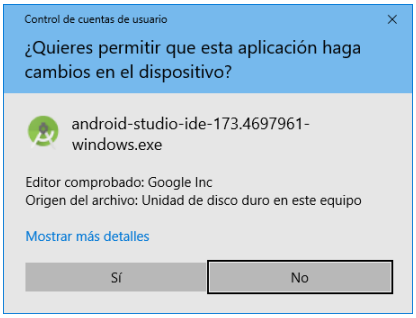
c. Al bajar el software obtenemos un archivo similar a “android-studio-ide-173.4697961-windows.exe”.



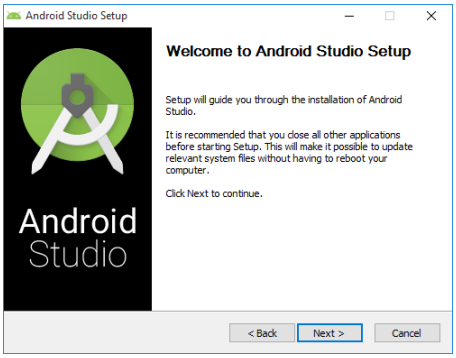
d. Lo ejecutamos. Se auto verifica.



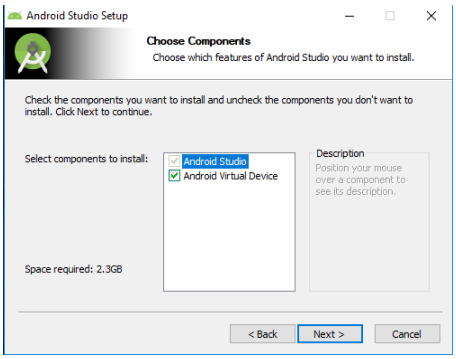
e. Nos pide permiso para ser instalado. Le indicamos que sí.



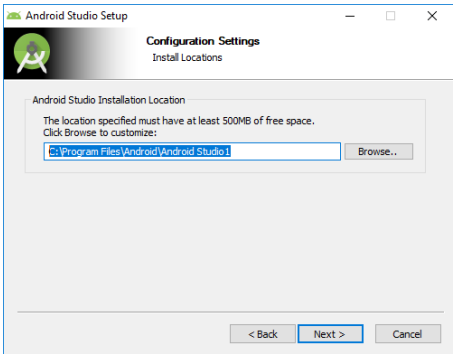
f. Luego se muestra la pantalla inicial. Damos “Next”.



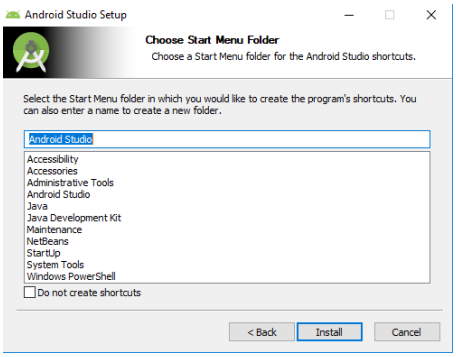
g. Luego que instale los componentes seleccionados por default y damos “Next”.



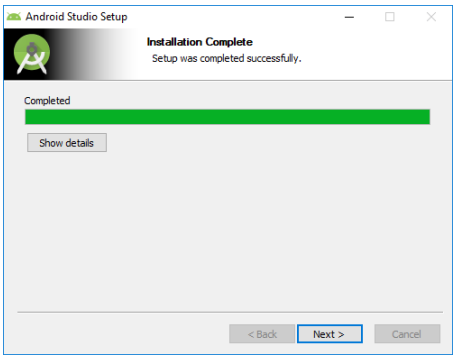
h. Luego nos muestra donde queremos instalar la aplicación. Damos “Next”.



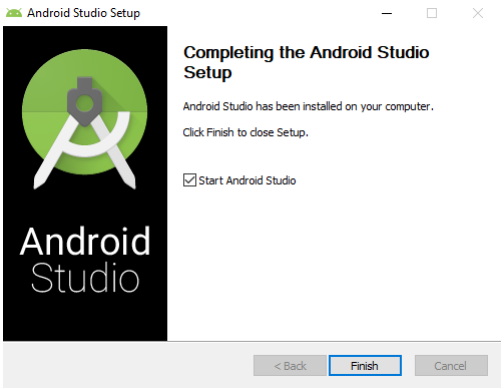
i. luego “Install”. Esperamos unos momentos a que la instalación termine.



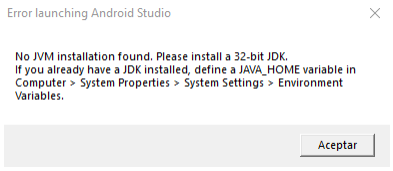
j. Nos muestra la instalación completa. Damos “Next”.



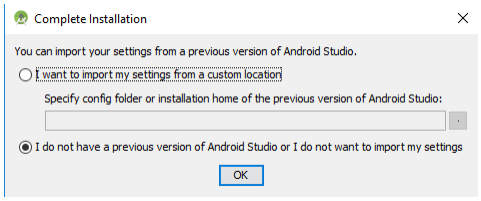
k. Al finalizar la instalación nos muestra una última pantalla. En este momento no pregunta si deseamos ejecutar Android Studio. Dejamos seleccionado la ejecución de Android Studio y damos “Finish”.



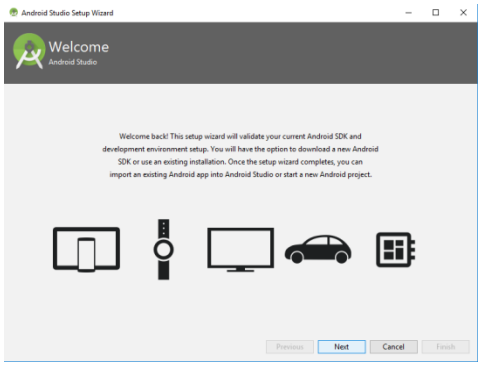
l. Al dar finalizar es posible que Android Studio no encuentre Java. Si nos da el siguiente error hacer el procedimiento del paso 1-e al 1-k.



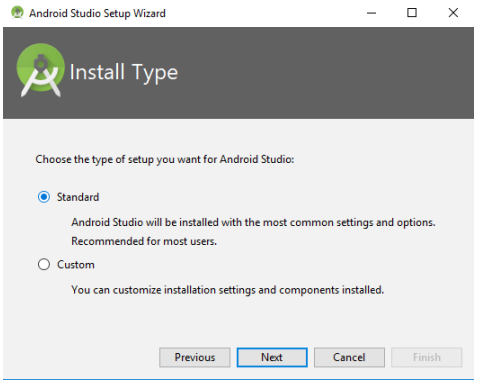
m. Lanzamos Android Studio. Y le indicamos no importar nada. Damos “OK”.



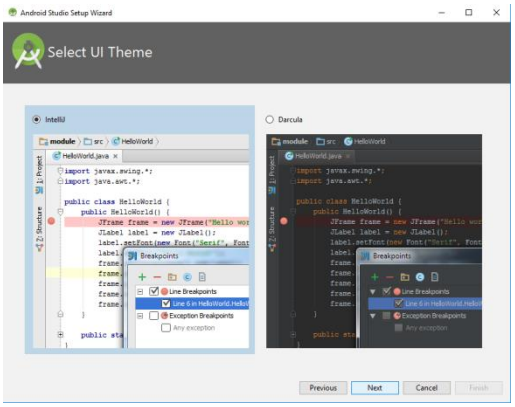
n. Nos muestra una pantalla donde se está cargando Android Studio. Luego sigue configurar el ambiente de desarrollo. Damos “Next”.



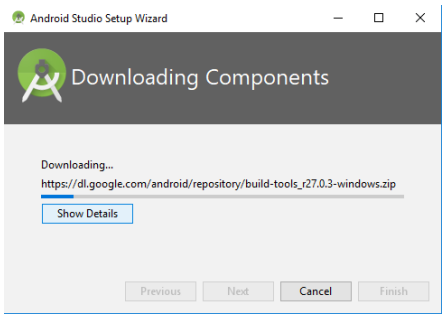
o. En las siguiente pantalla escogemos el valor default “Standard”.



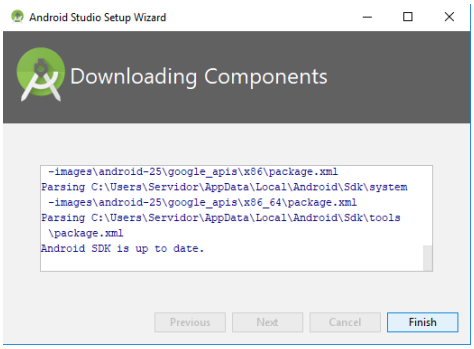
p. Escogemos “IntelliJ”.



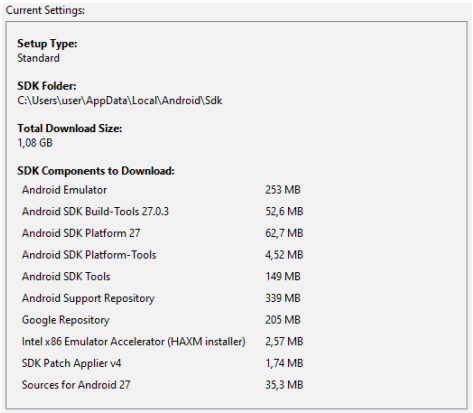
q. En seguida se pueden bajar algunos componentes adicionales.



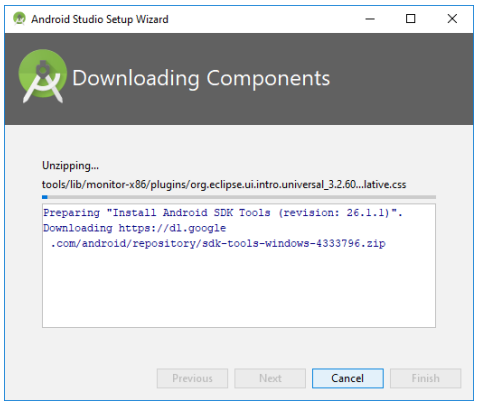
r. Se terminan de bajar y nos indica el SDK ha sido actualizado.



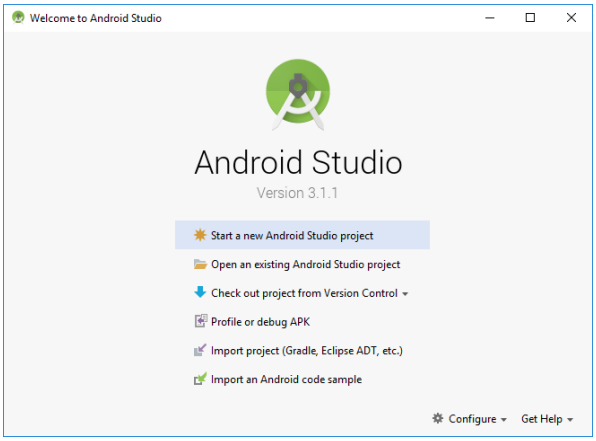
s. Ahora se muestra el resumen de las opciones escogidas. Damos “Finish”



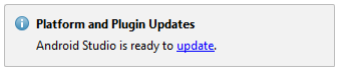
t. Baja otros archivos adicionales de acuerdo a nuestra selección.



u. Nos muestra ahora la pantalla principal de Android Studio.



3. La actualización de Android Studio puede invocarse cuando arrancamos nuestra IDE y/o abrimos un proyecto en la parte inferior derecha nos aparece un mensaje siguiente.



a. Antes de actualizar debemos de prever el tiempo puede ser considerable.
b. Podemos optar por que nos avise luego. Abrimos “Updates” y escogemos “Remind Me Later”.

4. Ejercicio finalizado ☺.

Bibliografía Preliminar

[1] Fundación Wikimedia, Inc, «Wikipedia,» 24 Octuble 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado. [Último acceso: 9 Noviembre 2017].

[2] Google, «Android Studio,» [En línea]. Available: https://developer.android.com/studio/index.html#Requirements. [Último acceso: 01 2018].

No. 02. Configurar Android Studio.

Objetivo

Configurar algunos parámetros del ambiente de desarrollo en Android Studio, permitiendo que el usuario esté en condiciones de crear aplicaciones empleando Android Studio.

TEMARIO

TEMA	NÚM. 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.	**No. 1. Instalar Interface Gráfica de Usuario Android.** Objetivo. Instalar una herramienta con Interface Gráfica de Usuario con Android Studio. **No. 2. Configurar Android Studio.** Objetivo. Establecer una configuración de desarrollo en Android. **No. 3. Creación y estructura de un Proyecto.** Objetivo. Crear una primera aplicación para Android.

Introducción

Android está ligado a los teléfonos inteligentes y muchos otros dispositivos como tabletas, cámaras, refrigeradores, televisores, estéreos etc. Android, es un sistema operativo (SO) basado en Linux desarrollado por Google. (1)

Anteriormente Google se apoyaba fuertemente en eclipse. Ahora el esfuerzo se ha trasladado a Android Studio presentado en 2014 como el IDE oficial para el desarrollo de aplicaciones de Android. (2)

Al llevar a cabo los pasos del ejercicio previo (numeral 1). Se ha instalado Java y el IDE Android Studio. Sin embargo, cuando se instala Android Studio por primera vez, sólo se instala la última versión del SDK. (3)

Java ha dejado de ser el lenguaje oficial para desarrollar aplicaciones Android. A partir de 2017 el lenguaje oficial es Kotlin. Significa que, aunque todavía se pueden desarrollar aplicaciones en Java, desde ahora Kotlin es totalmente compatible y Google se asegurará de que todas las nuevas funciones y bibliotecas funcionen perfectamente con el nuevo lenguaje. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Seguir los pasos de configuración de Android Studio.
3. Conectarse a Internet para obtener la información para buscar información sobre las versiones Android SDK, Intel x86 Emulator Accelerator (HAXM installer).
4. Investigar nombres básicos de herramientas en Android.
5. Seguir el procedimiento de la práctica.
6. Corroborar que Adroid Studio y el emulador estén corriendo con la nueva configuración.

Sugerencias didácticas.

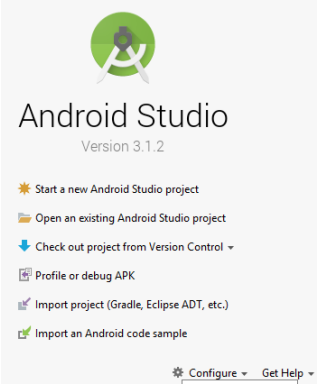
1. Realizar una investigación documental de: Android SDK Build-tools, Android SDK Tools, Android SDK Platform-tools, Android Support Repository, Android Support Library, Google Repository, Google USB Driver.
2. Seleccionar a varios participantes para exponer una configuración de los componentes configurables de Android Studio.
3. Seleccionar a voluntarios para configurar HAXM installer.

Reporte de los alumnos (resultados)

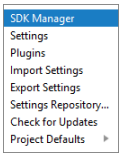
1. Configurar con una versión de SDK de Android Studio para el teléfono móvil del alumno.
2. Hacer un reporte incluyendo fotos para detectar la versión de Android en su teléfono y los pasos seguidos de acuerdo a la presente práctica.
3. Crear un dispositivo virtual elegido por el alumno que contemple la versión del móvil,
4. Documentar el paso anterior en un reporte incorporando en un documento los pasos seguidos. Y en caso necesario incluir fotos para justificar los pasos del proceso.

Procedimiento:

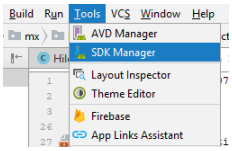
1. Instalación de actualizaciones de SDK. Después de instalar Android Studio es recomendable revisar las actualizaciones de SDK.
    a. Correr Android Studio si no tenemos un proyecto abierto. En la parte inferior damos un clic en configure.



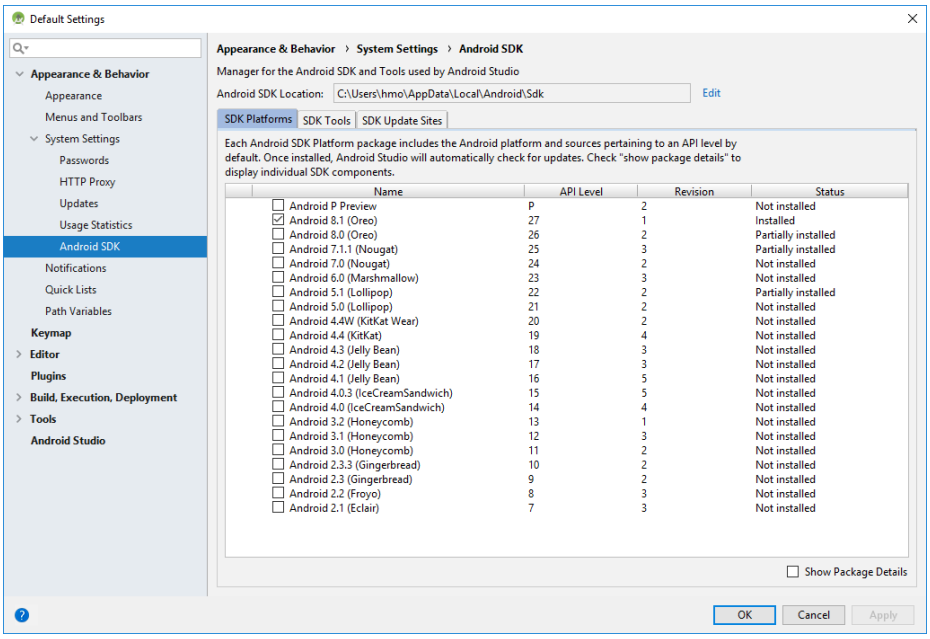
b. Seleccionamos “SDK manager”.



c. O si estamos en el menú de un proyecto abierto. Seleccionamos en “Tools” -> “SDK manager”.



d. Se abre el “SDK manager”. Podemos observar que sólo tiene instalado la versión 8.1. Si deseamos instalar una versión SDK aquí es el lugar para hacerlo.



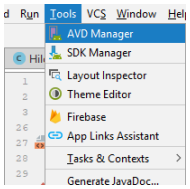
e. El menú izquierdo en “Updates” podemos ver si está actualizado la versión de “Android Studio”.

f. Cerramos el menú dando “Cancel”.

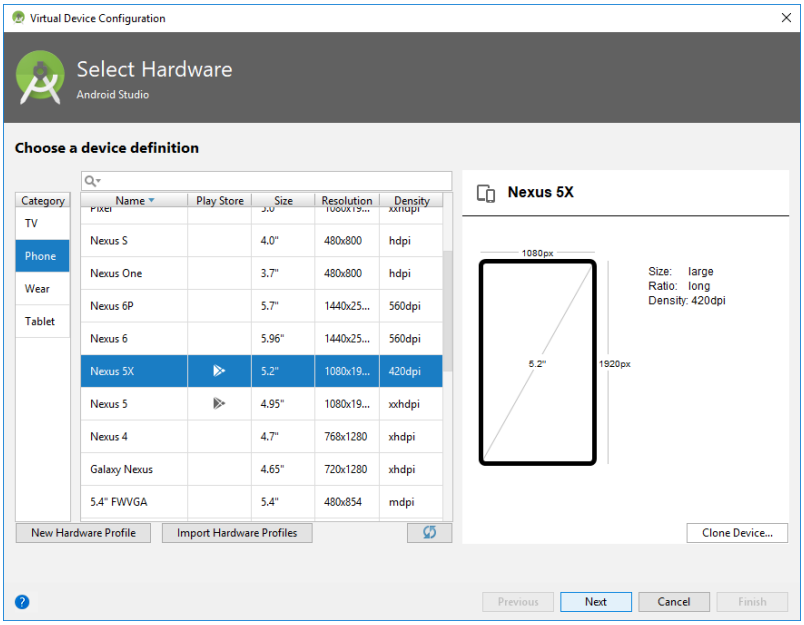
2. Una parte esencial en el desarrollo de aplicaciones móviles es la instalación de emuladores. Software que imita el comportamiento de dispositivos reales. Los

emuladores por sus siglas en inglés se denominan “AVDs” (Android Virtual Devices” en Android Studio.

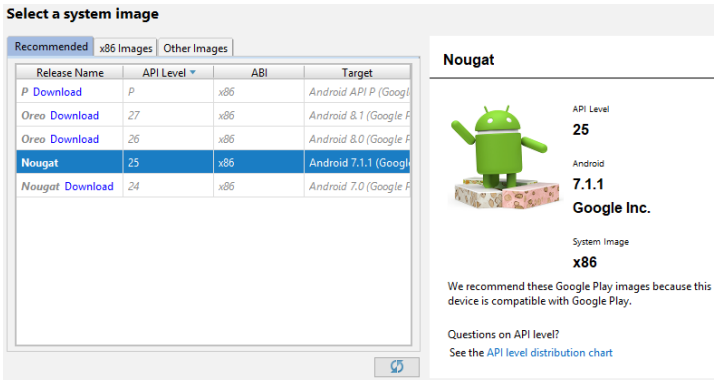
a. Del menú principal seleccionamos “Tools” -> “AVD Manager”.



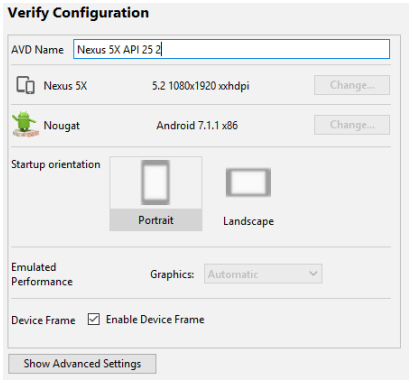
b. En un principio al mostrarse éste menú señala que no tenemos ninguna AVD creada. Seleccionamos “+ Create Virtual Device”. Nos lleva la la siguiente pantalla.



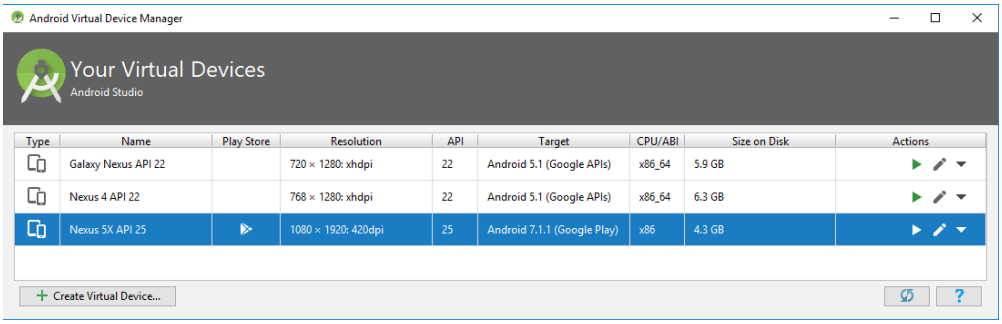
c. Aquí nos muestra diferentes tipos de dispositivos con pantallas de varios tamaños y otras características. Vemos que por default está seleccionado en “Category” Phone y Nexus 5X. Dejamos esos valores y damos “Next”.



d. Dejamos la sugerencia de usar Android 7.1.1 (Nougat). Enseguida presionamos “Next”. Y lo nombramos con un nombre arbitrario o dejamos el valor por default.



e. Presionamos “Finish” y se crea nuestra AVD. Podemos repetir a partir del paso 2-b la creación de otras AVDs con imágenes de acuerdo al procesador x86 o x86_64. Cuando se selecciona una imagen de una Api diferente a la sugerida, tiene marcado “download”. Será necesario bajar la imagen apropiada. Estas operaciones de descarga pueden ocasionar un tiempo considerable dependiendo de nuestra conexión de Internet. Las AVDs son imágenes creadas a partir de la versión 22 también conocida como Android 5.1.1 o Lollipop (obtenida del paso 3-b).



f. Mostraremos nuestro emulador de la configuración seleccionada dando ▶ o seleccionando doble clic en el AVD creado. Ahora ya estamos listos para correr nuestras aplicaciones en una máquina virtual. Sin embargo si deseamos correr una aplicación en un dispositivo real. Debemos de seguir los pasos del apartado 3 siguiente.

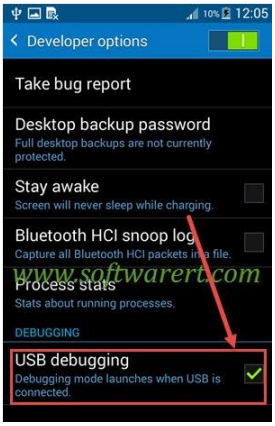
3. Activar el modo depuración en un teléfono celular. El modo de depuración es imprescindible para instalar y depurar aplicaciones usando Android Studio en un teléfono celular. Este modo damos permiso para que una computadora se conecte con nuestro teléfono y pueda instalar aplicaciones. Para cada marca y modelo varía la activación del modo de depuración.
    a. En particular tengo acceso a un teléfono móvil galaxy grand prime sm-g531h.

    b. Como éste celular está configurado para México está en español. Buscamos en “Ajustes” -> “Acerca del Dispositivo” -> “Número de compilación”. Si el celular estuviera en Inglés sería “Settings” -> “About Device” -> “Build number”. Note que la versión de la imagen es Android es 4.4.4 también conocida como “KitKat”. Sin embargo en el celular físico aparece la “Versión de Android” 5.1.1 con nombre “Lollipop”. Se recomienda crear una AVD para ésta versión.

    c. Estando ahí tocamos siete veces para activar el modo de depuración. Una vez activado nos muestra un mensaje “Ahora eres un desarrollador”.

d. Ahora debemos activar el modo debug por USB. Vamos a “Ajustes” -> “Sistema” -> “Opciones de desarrollador”. Y lo activamos.

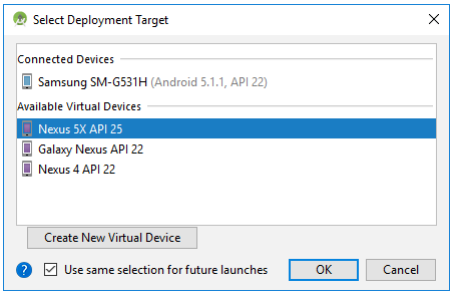
e. Seleccionamos “Depuración USB” y lo activamos.



f. Nos mostrará un mensaje sobre la instalación de programas y transferencia de datos entre la computadora y el teléfono. Le damos “Aceptar” para confirmar el modo depuración por USB.

g. El dispositivo está listo para recibir la instalación de aplicaciones generadas en Android Studio. Al conectar el teléfono también nos da acceso las imágenes y videos almacenados en nuestro celular de manera similar a una memoria USB.

4. Ya estamos en condiciones de correr nuestras aplicaciones en modo virtual o modo real. Al correr una aplicación se nos presentará una imagen como la siguiente para determinar el tipo de ambiente correrá nuestra aplicación.



5. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Wei-Meng, Lee.** *Beginning Android™ Application development.* Indianapolis, IN 4625 : Wiley Publishing, Inc., © 2011.
2. **Serhan, Yamacli.** *Beginner's Guide to Android App Development.* s.l. : Manchester Academic Publishers, 2017.
3. **Smyth, Neil.** *Android Studio Development Essentials.* s.l. : eBookFrenzy, © 2015.
4. **Leiva, Antonio.** *Kotlin for Android Developers.* s.l. : Leanpub, 2017-06-20.

No. 03. Creación y estructura de un proyecto Android.

Objetivo

Crear una primera aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una primera aplicación Android.

TEMARIO

TEMA	NÚM. 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.	**No. 1. Instalar Interface Gráfica de Usuario Android.** Objetivo. Instalar la herramienta Interface Gráfica de Usuario Android. **No. 2. Configurar Android.** Objetivo. Establecer una configuración de desarrollo en Android. **No. 3. Creación y estructura de un Proyecto.** Objetivo. Crear una primera aplicación Android.

Introducción

Una aplicación Android contiene pantallas, de manera similar en una ventana de Windows. Las pantallas en Android se llaman “Activities” o actividades. Las pantallas contienen “Layouts” o disposición y “widget” o controles. Los controles activan cierta lógica llamada “actions” o acciones. Los controles tienen propiedades como text, width, heigth. [1]

Una App o aplicación de Android consiste en código de Java, algunos documentos XML y otra información. [2]

Un “layout” define un grupo de objetos y su disposición en la pantalla. Un “layout” está hecho de definiciones en un archivo XML. Cada definición se emplea para crear un objeto a mostrar en la pantalla, como un botón o un texto. [3]

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre las aplicaciones Android.
4. Investigar ejemplos básicos en Android.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas

1. Realizar una investigación documental de la estructura de una “App Android”.
2. Seleccionar a varios participantes para exponer ejemplos de los componentes de una App Android.
3. Seleccionar a voluntarios para explicar el uso de Android Studio.
4. Llenar el recurso de Wiki con las definiciones de “Activities”, “Intents”, “Layouts”, “widgets”, “XML”.

Reporte de los alumnos (resultados)

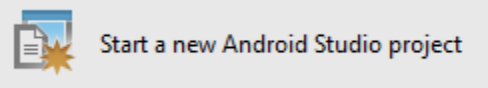
1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno incluyendo los componentes básicos.
3. Participar en Moodle con el glosario de términos incorporando Android, Kernel, Linux, SDK, nombres de versiones Android, IDE, Apps, Dispositivos,

Procedimiento:

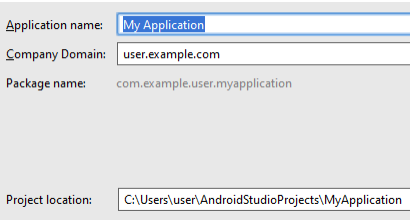
1. El tipo de problema a resolver se propone lo siguiente
   "Crear una primera aplicación Android mostrando el texto 'Hola Mundo' ".

   El algoritmo de solución aquí propuesto es: "Hacer una interface gráfica a través de Android Studio donde crea el texto se construye y ejecuta la aplicación".

2. Iniciar Android Studio. Al iniciar el programa se muestra una pantalla de inicio donde se selecciona "Start a new Android Studio Project".



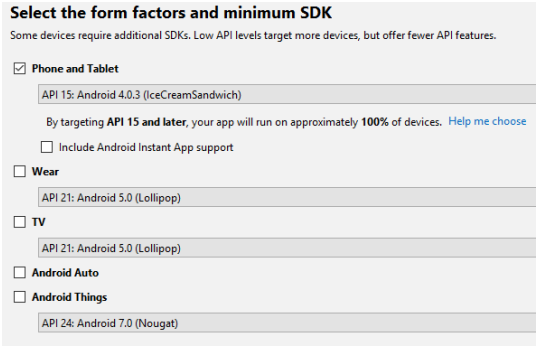
3. Una vez dado el inicio de un proyecto de Android se nos pide información inicial para nuestro proyecto.



   a. El nombre de nuestro proyecto en Android. En "Application name" escribiremos "Practica 03".
   b. El siguiente punto escribimos un dominio de nuestro lugar de trabajo y si no podemos inventar uno o dejar el preexistente. En éste caso pondremos "ith.mx".
   c. Y en "Project location" define el nombre del subdirectorio (de trabajos Android) en donde se crearán los archivos necesarios para el proyecto. Se crea en ese subdirectorio uno nuevo como subdirectorio del proyecto (puede ser el nombre del inciso a sin espacios). Damos "Next".

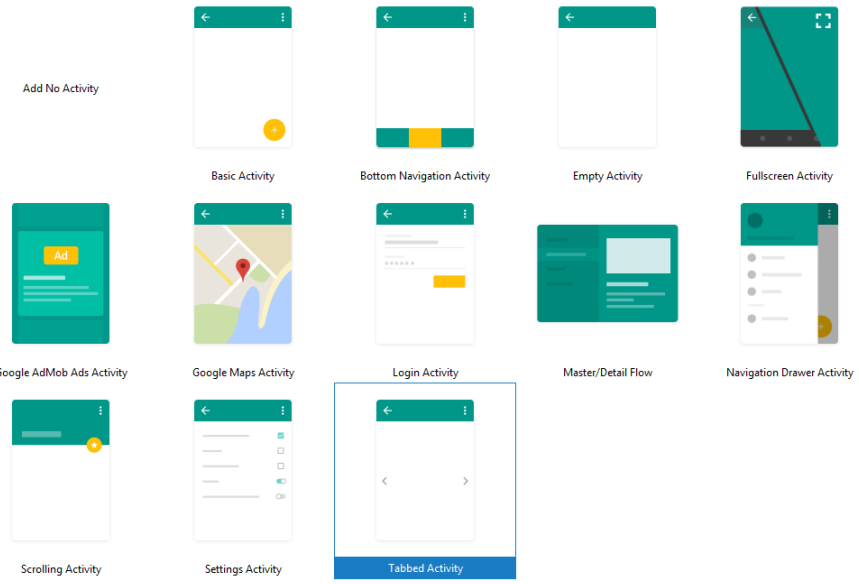
Ejemplo: **C:/Sabatico/Semestre2/Practicas/***Practica03*

4. Enseguida nos pide el SDK mínimo a usar.



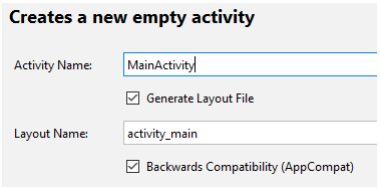
   a. Nos muestra el mercado disponible es 100%. Esto es, de todos los teléfonos existentes nuestra aplicación se va a poder instalar en la mayoría. Cuando damos "Help me choose" nos muestra una tabla en donde el compromiso va a estar en abarcar mayor mercado pero nuestra aplicación va a tener menos funcionalidad y viceversa.
   b. Dejaremos usar el mínimo SDK como "IceCreamSandwich".
   c. Seleccionamos "Next".

5. Nos presenta un menú para seleccionar el tipo de actividad a usar.



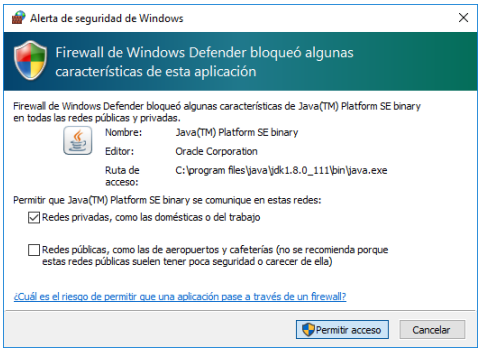
   a. Damos "Next". Es decir, aceptamos la opción default "Empty Activity".

6. Enseguida nos muestra las opciones para nuestra pantalla.

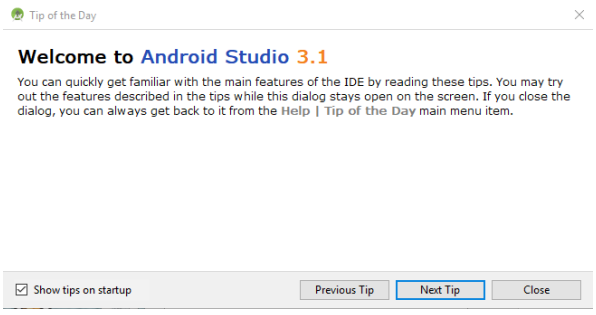


   a. Seleccionamos “Finish”. Significa: aceptar los valores por default.
   b. Si seleccionamos otra actividad (pantalla), damos “previous” para corregir nuestra selección.

7. El firewall de Windows nos puede prevenir de un acceso a redes privadas de Java. Damos “Permitir acceso”.



8. Ahora se nos presenta la interface de Android Studio con el “Tip of the Day” o consejo del día. Éste pequeño aviso nos puede ser útil para el manejo de Android Studio. Lo cerramos.

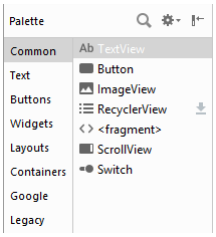


9. Nos aparece la interface con dos archivos uno llamado “`MainActivity.java`” con el siguiente contenido. El contenido es muy similar al siguiente.

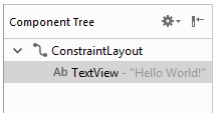
```
package mx.ith.practica03;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```
public class MainActivity extends AppCompatActivity {

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

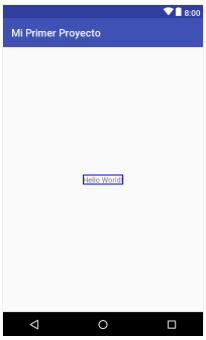
10. Damos un clic en la pestaña "`activity_main.xml`". Y nos muestra varia áreas.
    a. Un área donde están algunos componentes a escoger. Por default nos enseña los componentes comunes.



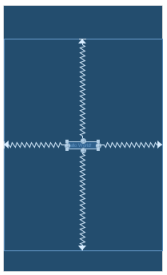
    b. Otra área se encuentra el árbol de los componentes agregados.



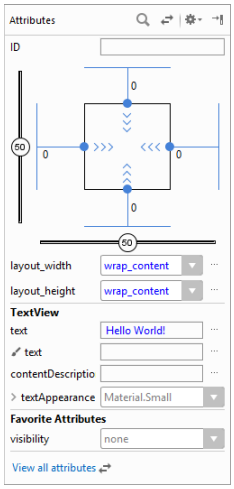
    c. Un área más nos muestra cómo se visualizarán los componentes agregados. Seleccionamos el componente donde dice "Hello World" con el ratón dando un clic.



d. Hay otra área muy parecida a la anterior y nos muestra donde un componente puede ser desplazado.



e. Existe un área adicional donde un componente puede ser modificado a través de sus atributos.



f. Cambiaremos el texto "`Hello World!`" por "`Hola Mundo!`".

g. En la parte inferior hay dos pestañas 

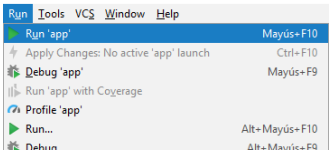
. Podemos cambiar ya sea en diseño o en texto "**`Hola Mundo!`**". Por cualquier método usado generaremos el mismo resultado. Esto es tanto en diseño como en texto se modifica el mismo archivo "`activity_main.xml`".

h. Si escogemos "Text" obtenemos el archivo "`activity_main.xml`" generado.

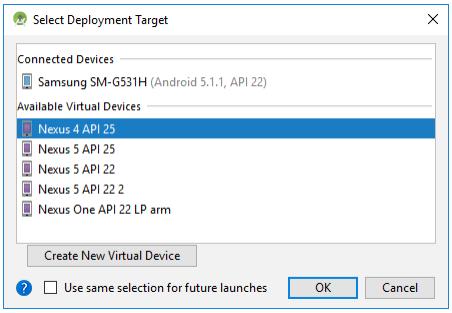
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hola Mundo!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

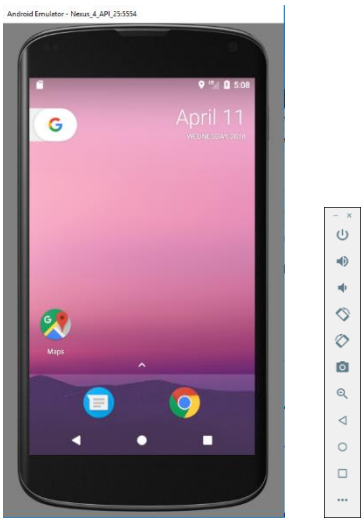
11. Ahora corremos el programa. En la parte superior seleccionamos “Run” y de nuevo “Run” o con una combinación de teclas sería <Mayus><F10>



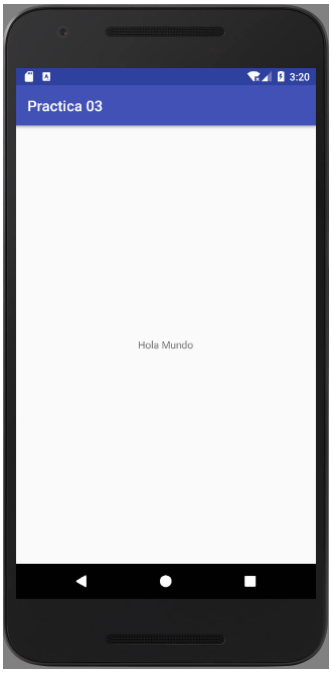
12. Al correr nos indica si deseamos correrlo en un dispositivo o en un emulador (resultados obtenidos de la práctica número 02) o en un dispositivo (resultados obtenidos de la práctica número 02). Si seleccionáramos el dispositivo celular Samsung debe estar configurado en modo debug conectado a una USB (debemos dar ‘OK’ en el celular en el diálogo ‘Allow USB Debugging’). Obtendríamos el mismo resultado del paso 14 en nuestro celular.



13. Escogeremos el primer dispositivo creado y escogemos “OK”. El dispositivo hace la emulación de un teléfono. Una vez “prendido” se carga en él la aplicación.



14. Y nos muestra nuestro “Hola Mundo”.



15. Ejercicio finalizado ☺.

Bibliografía Preliminar

[1] M. L. Murphy, The Busy Coder's Guide to Android Development, CommonsWare, LLC, 2008.

[2] B. Burd, Java® Programming for Android™ Developers For Dummies®, Hoboken, New Jersey: John Wiley & Sons, Inc., 2014.

[3] B. Phillips y B. Hardy, Android Programming: The Big Nerd Ranch Guide, Indianapolis, IN 46240 USA: Pearson Technology Group, 2013.

No. 4. Unidades y layouts.

Objetivo

Crear una segunda aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android con dos "Activities" distribuyendo los elementos usando un "layout" particular.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 4. Unidades y layouts.** Objetivo. Crear una aplicación Android con varias actividades. **No. 5. Usando Listas.** Objetivo. Empleo de la librería ListView. **No. 6. Listas personalizadas.** Objetivo. Conocer los elementos comunes en una interface gráfica. **No. 7. Elementos gráficos.** Objetivo. Crear una librería gráfica personal Círculo.

Introducción

Al crear una aplicación Android se llevan a cabo en varios pasos: (1)

1. Instalación de Android Studio y su configuración.
2. Construir una aplicación (app) más elaborada.
3. Correr la App en el emulador.
4. Cambiar la app con unos cambios adicionales del paso 2 y correrla de nuevo.

Los controles comunes de Android son: (2)

1. Los controles de Texto:
    a. TextView. Muestra texto, no se puede editar.
    b. EditText. Muestra texto y se puede modificar.
    c. AutoCompleteTextView. Texto editable con sugerencias.
    d. MultiAutoCompleteTextView. Texto editable con sugerencias para varias palabras.
2. Los controles de botón.
    a. Button. Se programa un evento con código de Java.
    b. ImageButton. Un botón (button) con imagen.
    c. ToggleButton. Un botón con dos estados prendido o apagado.
    d. CheckBox. Es como un ToggleButon pero se muestra con un "palomita".
3. Los controles ImageView. Es un área para mostrar una imagen.
4. Los controles de Date y Time. Muestran la fecha y la hora.
5. El control MapView. Junto con los servicios de "Google Play services" sirve para mostrar un mapa.

Las intenciones (Intents) encapsulan una solicitud, hecha a Android, para que alguna actividad u otro receptor haga algo. (3)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los controles comunes en las aplicaciones Android.

4. Investigar ejemplos básicos en Android usando controles comunes.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de controles comunes en una “App Android”.
2. Seleccionar a varios participantes para exponer ejemplos de los controles comunes.
3. Seleccionar a voluntarios para explicar el uso de controles comunes.
4. Llenar el recurso de Wiki con las funciones de “Text”, “Button”, “Activities”, “Button”, “ImageView”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno incluyendo los controles comunes.
3. Participar en Moodle con el glosario de términos incorporando: controles, eventos, el objeto “R”, funciones más importantes de “Android Software Stack”, Servicios, “Intents”.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente
   “Crear una primera aplicación con una ventana principal e invoque a otra ventana”.

   El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica a través de Android Studio donde crea una “activity” principal con un control se invoca a otra “activity” secundaria en donde se encuentra otro control para regresar a la pantalla principal”.

2. Iniciar Android Studio. Al iniciar el programa se muestra una pantalla de inicio donde se selecciona “Start a new Android Studio Project” o en su defecto en el menú seleccionar: “File”, “New”, “New Project”.

3. Una vez dado el inicio de un proyecto de Android se nos pide información inicial para nuestro proyecto.

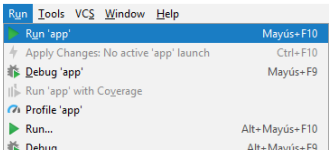
   a. El nombre de nuestro proyecto en Android. En “Application name” escribiremos “Practica 04”.
   b. El siguiente punto escribimos un dominio de nuestro lugar de trabajo y si no estamos liberando nuestra app no es importante y podemos inventar uno o dejar el preexistente. En éste caso pondremos “ith.mx”.
   c. Y en “Project location” define el nombre del subdirectorio (de trabajos Android) en donde se crearán los archivos necesarios para el proyecto. Se crea en ese subdirectorio uno nuevo como subdirectorio del proyecto (puede ser el nombre del inciso a sin espacios). Escribimos **C:\Sabatico\Semestre2\Practicas***Practica04* y damos “Next”.

4. Enseguida nos pide el SDK mínimo a usar.
   a. Por el momento no escogemos otro que no sea “Phone and Tablet”.
   b. Dejaremos usar el mínimo SDK como API 15: “IceCreamSandwich”.
   c. Seleccionamos “Next”.

5. Nos presenta un menú para seleccionar el tipo de actividad a usar.
   a. Damos “Next”. Es decir, aceptamos la opción default “Empty Activity”. Nos genera menos código y nos facilita agregar código.

6. Enseguida nos muestra las opciones para nuestra pantalla.
   a. Seleccionamos “Finish”. Significa: aceptar los valores por default. No nos preocupemos por los nombres, ya que los podemos cambiar después.
   b. Ahora Android Studio hace su trabajo.
   c. En el editor aparecen disponibles “`MainActivity`.java” y “`activity_main`.xml”.

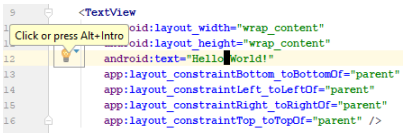
7. Si corremos el programa. En la parte superior seleccionamos “Run” y de nuevo “Run” o con una combinación de teclas sería <Mayus><F10> o la flecha verde



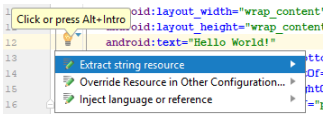
8. Y nos muestra el título la aplicación con título “Practica 04” y con el texto “Hello World!”.

9. Vamos a modificar "activity_main.xml". Al texto creado por default le asociaremos a un identificador constante.
   a. Seleccionamos "activity_main.xml" y lo abrimos en modo texto (dando clic en “Text” a un lado de “Design”).

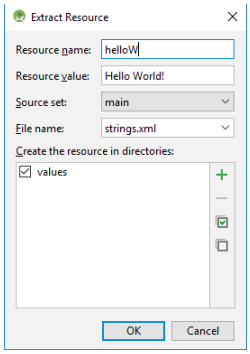
b. En la línea donde está "`android:text="Hello World!"`" damos clic sobre cualquier parte del texto `"Hello World!"`



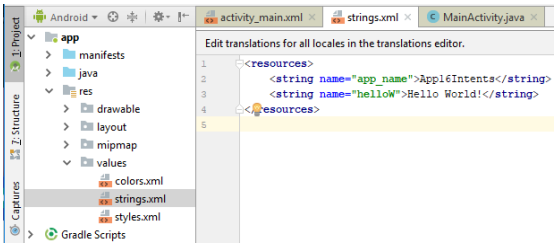
c. Se nos presenta un foco con la invitación a abrirlo. Lo abrimos con cualquiera de las opciones presentadas “clic” o “<Alt><Enter>”.



d. Seleccionamos la primer opción “Extract string resource”. Usamos el alias `"helloW"`



e. Estas constantes junto con otros recursos se guardan en varios archivos. En el proyecto abriendo “app” en un apartado “res” de recursos y en él está “values” que a su vez contiene el archivo “strings.xml” donde se almacenan los valores.



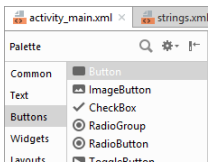
f. Y regresando al archivo fuente "activity_main.xml" vemos que queda modificada la línea "`android:text="Hello World!"`" cambia a "`android:text="@string/hellow"`".

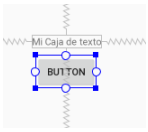
10. Modificar el texto "`android:text="@string/hellow"`"" (Su valor anterior era "`android:text="Hello World!"`") por un valor personal.
    a. En el archivo fuente "activity_main.xml" en la línea: "`android:text="@string/hellow"`"
    La cambiamos a:
    "`android:text="Mi Caja de texto"`"
    Hacemos un procedimiento similar al punto 9. Para que nuestro texto quede: "`android:text="@string/mi_caja_de_texto"`".

11. Como ya la constante previa "`hellow`" ya no es necesaria. Es opcional borrar la siguiente línea del archivo "strings.xml":
    "`<string name="helloW">Hello World!</string>`"

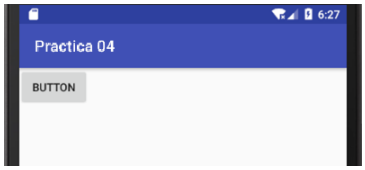
12. Agregar un botón. Observe el texto del archivo "activity_main.xml" al principio tiene alrededor de 18 líneas y al terminar los incisos c y d se incrementa y al terminar tiene cerca de 30 líneas.

    a. En diseño (presionamos en "Design" en la parte baja a un lado de "Text") en "activity_main.xml". Seleccionamos de la paleta de componentes en "Buttons" (damos clic) escogemos "Button" y lo arrastramos debajo de "Mi Caja de Texto".

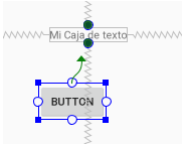




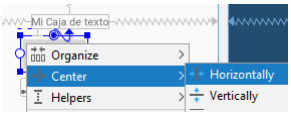
    b. Si corremos nuestro programa nos muestra el botón en la posición (0,0) esto es aparece en la esquina superior izquierda. Y no en la posición aplicada en diseño.



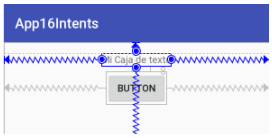
    c. Para que el botón se muestra debajo de nuestro texto "Mi Caja de Texto" debemos editar el botón. En diseño lo seleccionamos y conectamos la parte superior con la parte de debajo de nuestro texto.



d. Luego vamos a centrarlo horizontalmente. Con el botón seleccionado dando un clic derecho sobre el ratón le indicamos el centrado.



e. Ahora llevamos el texto hacia la parte superior justo debajo del título de nuestro proyecto.



f. Podemos observar: Cada vez, al realizar un cambio en diseño, se refleja los cambios en el archivo "activity_main.xml" (lo verificamos en modo texto). Lo corremos y ahora aparece el botón aparece centrado y debajo de nuestro texto y el texto en la parte superior. Así obtenemos el siguiente resultado.

13. Cada elemento agregado tiene mayor cantidad de propiedades que las mostradas por default. Una vez seleccionado un elemento podemos ver sus propiedades. Para ver mayor o menor cantidad de propiedades hacemos lo siguiente:

    En modo diseño, seleccionamos el texto o el botón, en la parte derecha abajo aparece un letrero "View all attributes". Lo presionamos y nos muestra todos los atributos. Para regresarnos posiblemente usemos la barra de deslizamiento (scroll) para ver en la parte inferior "View fewer attributes".

    Nota. Al cambiar el valor de un atributo su cambio se refleja cuando damos <Enter>. O también cuando seleccionamos otra propiedad.

14. Es recomendable asociar un elemento con un nombre representativo. Cada elemento tiene una propiedad de identificación denominada "ID". Cada elemento agregado tiene un nombre por default. Para que sea útil para el programador se le asigna un nombre ligado a la función del elemento.

    Nota. Hay ocasiones, al cambiar id, se nos muestre un letrero informando actualizar las referencias en el campo "Java R" le indicamos "Yes".

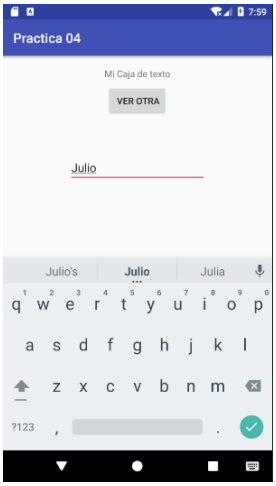
15. Cambiaremos el “ID” del botón. La función prevista para el botón es “Cuando se le dé clic va a mostrar la otra actividad y le va a enviar información”.

    a. Nos aseguramos que esté seleccionado el botón. En diseño en “activity_main” damos clic en “BUTTON”.
    b. En propiedades su “ID” tiene asignada el valor por default “button”. Lo cambiamos a “btnVerOtra”.
    c. En propiedad text dice “Button” lo cambiamos a “Ver Otra”.

16. Agregamos un “Plain Text”. Va estar centrado horizontalmente y pegado a la parte superior similar al punto 12-e.

    a. Seleccionamos de la “Palette” en “Text” el “Plain Text” y lo arrastramos debajo del botón. No olvide la alineación como el punto 12-e.
    b. Observe el nombre desplegado por default dice “Name”. Lo cambiamos a “Julio” (en la propiedad “text”).
    c. Modificamos el atributo “id”. Por default muestra “editText”. Lo cambiamos a “txtNombre”.
    d. Cambiamos la propiedad “hint” en lugar de “Name” escribimos “Nombre”.

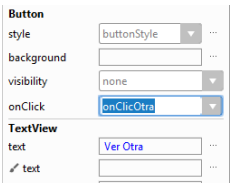
17. Corremos nuestro programa. Nos permite escribir y modificar “Julio”. Y el botón “VER OTRA” no hace ninguna función todavía. Y cuando borramos “Julio” aparece “Nombre” en color gris.



18. Programaremos el botón “VER OTRA”. Su función servirá para mostrar otra ventana y enviarle información.

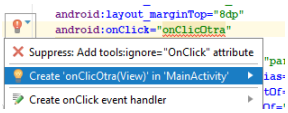
    a. Seleccionamos en diseño en “activity_main.xml” el botón “VER OTRA”.

b. Cambiamos la propiedad propiedad OnClick por default “” por el valor “onClicOtra”.



c. Sin embargo “onClicOtra” no está definido. Abrimos “activity_main.xml” en modo texto. Observar que del lado derecho aparece una marca en forma de línea roja y del lado izquierdo aparece un foco rojo.

d. En el foco rojo le indicamos generar el evento. (En caso de no aparecer el foco rojo dar un clic en la marca roja).



e. Una vez indicado la creación del método abrimos “MainActivity.java”. Ya está generado el método “`onClicOtra`”.

```
package mx.ith.practica04;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

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

    public void onClicOtra (View view) {
    }
}
```

f. En la clase agregamos el atributo “txtMiNombre” el cual hace referencia a “txtNombre”. Lo inicializamos en “`onCreate`”. Para definir una librería como “`EditText`” nos pide que presionemos <Alt><Enter>.

```
public class MainActivity extends AppCompatActivity {

 android.widget.EditText txtMiNombre;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtMiNombre = findViewById(R.id.txtNombre); //Línea agregada
}
```

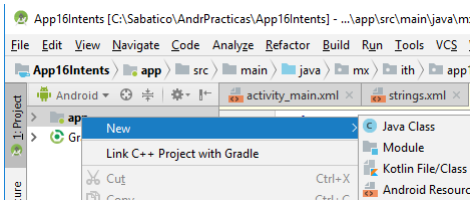
g. Agregamos el código siguiente a "`onClicOtra`". Importamos la librería de Intent (<Alt><Enter> sobre la línea de Intent). Todavía no existe "`OtraActivity.class`" y por lo tanto nos marca un error.

```
public void (android.view.View view) {
                                        //No existe OtraActivity
    Intent intent = new Intent(MainActivity.this, OtraActivity.class);
    Bundle bundle= new Bundle();
    bundle.putString("miNombre", txtMiNombre.getText().toString());
    intent.putExtras(bundle);
    startActivityForResult(intent, 1);
}
```

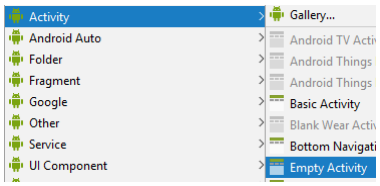
h. En la clase vamos a agregar un método "`onActivityResult`". Nos sirve cuando la otra actividad regresa a la actividad principal. En la clase con el botón derecho seleccionamos "Generate" luego "Override Methods" y vamos hasta el bloque "android.support.v4.pp.FragmentActivity" y seleccionamos el método. Y escribimos dentro de él lo siguiente.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1) {
        if(resultCode == android.app.Activity.RESULT_OK){
            String s = data.getStringExtra("miNombre");
            android.util.Log.e("X", "regreso "+ s );
            txtMiNombre.setText(s);
        }else
        if (resultCode == android.app.Activity.RESULT_CANCELED) {
            android.widget.Toast.makeText(this, "Cancelado",
            android.widget.Toast.LENGTH_LONG).show();
        }
    }
}
```

19. Agregamos una nueva actividad a nuestro proyecto.
    a. En nuestro proyecto con el botón derecho seleccionamos "New".



b. Luego buscamos “Activity” y “Empty activity”.



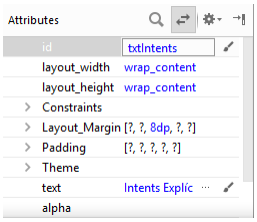
c. Nos presenta un menú con nombre de la actividad por default “Main2Activity” lo cambiamos con nombre “OtraActivity”. Y dejamos los valores por default. Vemos el nombre del layout como “activity_otra” y seleccionado “Backwards Compatibility (AppCompat)”. Escogemos “Finish”.

20. En la nueva actividad “activity_otra.xml”, agregamos un TextView. En los pasos siguientes cambiaremos sus propiedades id y text.

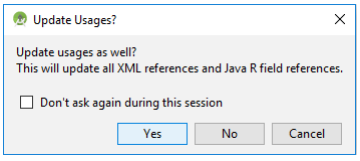
a. Abrimos en diseño “activity_otra.xml”. Seleccionamos de la paleta en “Common” un “TextView”.

b. Lo ajustamos centrado horizontalmente y pegado a la parte superior similar al punto 12-e.

c. Modificamos apropiadamente los atributos id y text. Cambiamos su id con “txtIntents” con text con “Intents explícitos”.

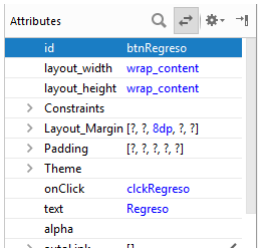


d. Al cambiar id nos puede mostrar un letrero informado actualizar las referencias en el campo “Java R” le indicamos “Yes”.

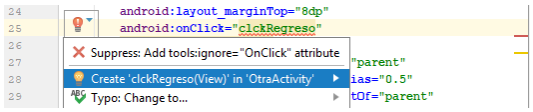


21. En la nueva actividad “activity_otra.xml”, agregamos un botón. Lo colocamos centrado horizontalmente debajo del texto “Intents Explícitos” similar los pasos 12-c y 12-d.

    a. Cambiamos la propiedad text por default “Button” por el valor “Regreso”. Y la propiedad OnClick por default “” por el valor “clckRegreso”. Y id por “btnRegreso”.



    b. Sin embargo “clckRegreso” no está definido. Abrimos “activity_otra.xml” en modo texto. Observar que del lado derecho aparece una marca en forma de línea roja y del lado izquierdo aparece un foco rojo. En el foco rojo le indicamos generar el evento. (En caso de no aparecer el foco rojo dar un clic la marca roja).



    c. Una vez indicado la creación del método abrimos “OtraActivity.java”. Ya está generado el método “`clckRegreso`”.

```
package mx.ith.practica04;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class OtraActivity extends AppCompatActivity {

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

    public void clckRegreso(View view) {
    }
}
```

    d. En la clase agregamos el atributo “*miTexto*” y lo inicializamos. Insertamos la librería “*EditText*” con <Alt><Enter>

```
public class OtraActivity extends AppCompatActivity {
```

```
  EditText miTexto;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_otra);
    miTexto = findViewById(R.id.txtNombre); //Agregado
}
```

e. Ahora programamos el botón de regreso.

```
public void clckRegreso(View view) {
    Intent intent = new Intent();
    intent.putExtra("miNombre", miTexto.getText().toString());
    setResult( android.app.Activity.RESULT_OK, intent);
    finish();
}
```

22. En la nueva actividad "activity_otra.xml", agregamos un "Plain Text". En los pasos siguientes cambiaremos sus propiedades id, hint y text.

    a. Lo ponemos abajo del botón y centrado.

    b. Cambiamos su id a "txtNombre" y su hint a "Otra" y text en "Intent".

23. Si ahora corremos el programa y damos clic en el botón "Ver Otra". Nos muestra la otra ventana. Observamos la falta de información actualizada de nuestra primera ventana. El texto dice "`Intent`" y no "Julio" que proviene de la primera ventana. Sin embargo si escribimos algo o dejamos "`Intent`" y presionamos el botón "Regreso" si se muestra el cambio en la primer ventana

    a. Para resolverlo, en "Otractivity.java", reprogramamos el método "`onResume`" de la clase OtraActivity de la siguiente forma.

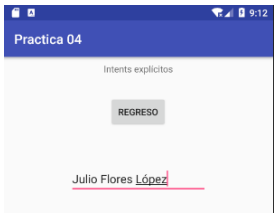
```
@Override
public void onResume() {
    super.onResume();

    String s= getIntent().getExtras().getString("miNombre");
    miTexto.setText(s);
}
```

24. Ahora ya podemos enviar y recibir información de una caja de texto de una actividad a otra actividad.

    a. Capturamos un nombre en la pantalla principal (O dejamos Julio).

    b. Éste nombre se recibe en la otra pantalla.

    c. Si lo modificamos de nuevo y nos regresamos. Obtenemos en principal los cambios.



25. Ejercicio finalizado ☺.

Bibliografía Preliminar
1. **Griffiths, Dawn y Griffiths, David.** *Head First Android Development.* 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly, 2015.
2. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* New York : Apress, 2015.
3. **Murphy, Mark L.** *The Busy Coder's Guide to Android Development.* United States of America : CommonsWare, LLC., © 2008-2017.

No. 5. Usando Listas.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para programar elementos tipo "ListView".

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 4. Unidades y layouts.** Objetivo. Crear una aplicación Android con varias actividades. **No. 5. Usando Listas.** Objetivo. Empleo de la librería ListView. **No. 6. Listas personalizadas.** Objetivo. Conocer los elementos comunes en una interface gráfica. **No. 7. Elementos gráficos.** Objetivo. Crear una librería gráfica personal Círculo.

Introducción

Cada instancia de Activity tiene un ciclo de vida. Durante este ciclo de vida, una actividad transita entre varios estados: al crearse, en ejecución, en pausa y detenido. Para cada transición, existen métodos para notificar la actividad del cambio en su estado. Estos métodos también denominados "callbacks" son: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, y `onDestroy()`. (1)

Los controles de lista se utilizan para mostrar colecciones de datos. Pero en lugar de usar un solo tipo de control para administrar tanto la pantalla como los datos, Android separa estas dos responsabilidades en controles de lista y adaptadores. Los controles de lista son: ListView, GridView, Spinner y Gallery. (2)

Un adaptador es una instancia de una clase que implementa la interfaz “Adapter”. Ejemplo: Si vamos a utilizar una instancia de ArrayAdapter <T>, es un adaptador que sabe cómo trabajar con datos en una matriz (o una ArrayList) de objetos de tipo T. (3)

Los fragmentos (Fragments) de diseño son una forma poderosa de reutilizar Actividades simplemente recargando una porción de la interface gráfica en la Actividad cargada actualmente. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre el ciclo de vida de las “Activities”, “List Views” y “Adapters”.
4. Obtener ejemplos básicos en Android usando configurando los “Callbacks”.
5. Obtener ejemplos básicos en Android usando “List Views”.
6. Seguir el procedimiento de la práctica.
7. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de controles comunes en una “App Android” empleando “List Views” o “RecyclerViews” con “Fragments”.
2. Seleccionar a varios participantes para exponer ejemplos de los controles comunes en “List Views”.
3. Seleccionar a voluntarios para explicar el ciclo de vida de “Activities”.

4. Seleccionar a voluntarios para explicar “Fragments”.
5. Seleccionar a voluntarios para explicar “Toast”.
6. Llenar el recurso de Wiki con las funciones de “RecyclerView”, “List Views”, de “Adapters”, de los “Callbacks” en Activities.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno incluyendo “RecyclerViews”.
3. Participar en Moodle con el glosario de términos incorporando: “RecyclerView”, “Activities”, “Adapter”, “Callback”, “Layout”, “Widget”, “Fragment”, “Toast”.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente
   “Crear una primera aplicación con una ventana principal con una lista de datos como elemento base para mostrar un diálogo para cada elemento de la lista”.

   El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde crea una pantalla principal con elementos responsivos a eventos del ratón y programados para mostrar diálogos informativos”. Se escogió hacerlo con una lista de libros.

2. Iniciar Android Studio. Al iniciar el programa se muestra una pantalla de inicio donde se selecciona “Start a new Android Studio Project” o en su defecto en el menú seleccionar: “File”, “New”, “New Project”.

3. Una vez dado el inicio de un proyecto de Android se nos pide información inicial para nuestro proyecto. El nombre de nuestro proyecto escribiremos “Practica05”.

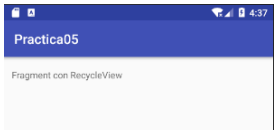
4. Enseguida nos pide el SDK mínimo a usar.
   a. Por el momento no escogemos otro que no sea “Phone and Tablet”.
   b. Dejaremos usar el mínimo SDK como API 15: “IceCreamSandwich”.
   c. Seleccionamos “Next”.

5. Nos presenta un menú para seleccionar el tipo de actividad a usar.
   a. Damos “Next”. Es decir, aceptamos la opción default “Empty Activity”. Nos genera menos código y nos facilita agregar código.

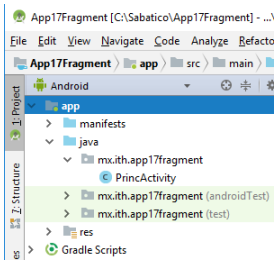
6. Enseguida nos muestra las opciones para nuestra pantalla.
   a. En “Activity Main” escribimos “PrincActivity” y seleccionamos “Finish”.

b. Ahora Android Studio hace su trabajo.
c. En el editor aparecen disponibles "`PrincActivity.java`" y "`activity_princ.xml`".

7. Vamos a modificar "activity_princ.xml".
   a. Lo seleccionamos.
   b. Nos aseguramos estar en diseño (Design).
   c. El texto "TextView" indicando "Hello Wold!" lo moveremos al tope y a la izquierda y lo cambiamos a "Fragment con RecycleView". Lo corremos:



8. Agregaremos un "fragment" con "RecycleView" a nuestro proyecto.
   a. Nos aseguramos de seleccionar nuestro proyecto.



   b. Seleccionamos "File" -> "New" -> "Fragment" -> Fragment (List).

   c. En ese momento nos indica el siguiente menú del cual en "Object Kind" escribimos "Libro". Y las demás opciones las dejamos por default. Y damos "finish".

   d. Observamos que se ha generado varias clases adicionales a la clase "PrincActivity". Estas son: "DummyContent" (dentro de dummy), "LibroFragment" y "MyLibroRecyclerViewAdapter.

9. Con el propósito de ser congruente con nuestro ejemplo "libro". Renombraremos los paquetes y los nombres de las clases.

   a. Nos posicionamos en el paquete "dummy" y con el botón derecho seleccionamos "Refactor" -> "Rename" o <Shift><F6>. En la ventana de diálogo escribimos "libro".

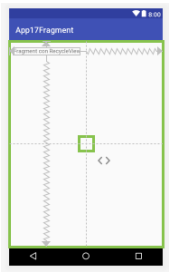
b. Con la clase “DummyContent” la modificamos con “refactor” a “GeneraLibros”.

c. Usando refactor cambiamos la clase “DummyItem” por “Libro” (La clase se encuentra al final de la recién renombrada clase GeneraLibros)

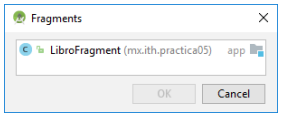
d. Cambiamos de la misma forma “createDummyItem” por “crearLibro”. Seleccionada la palabra “createDummyItem” y luego <Shift><F6>.

10. Al correr de nuevo nuestro programa no encontramos nada nuevo. Necesitamos incorporar el fragmento a nuestra actividad “PrincActivity”. Vamos a modificar "activity_princ.xml".

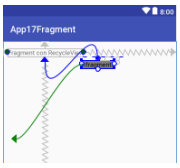
a. En diseño abrimos “activity_princ.xml”.
b. De la paleta escogemos “Containers” y luego “<fragment>”. Lo arrastramos y lo ubicamos en el centro.



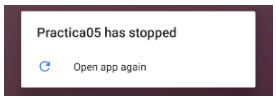
c. En ese momento nos muestra un diálogo para escoger el fragmento que hemos creado “LibroFragment”.



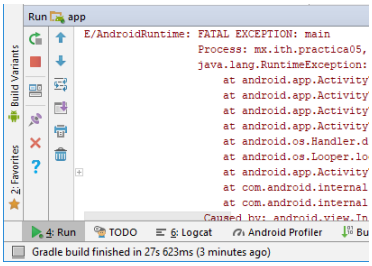
d. Lo seleccionamos y damos “OK”. Ahora ubicamos nuestro fragment abajo del texto y alineado a la parte inferior del texto y alineado a la izquierda.



11. Si corremos nuestro programa no funciona.
    a. Nos muestra un error en el emulador.



    b. Y en los mensajes de "Run" nos dice:



    c. Nos desplazamos hasta "Caused by:"

```
E/AndroidRuntime: FATAL EXCEPTION: main
…
Caused by: java.lang.RuntimeException: mx.ith.practica05.PrincActivity@43f7663
        must implement OnListFragmentInteractionListener
at mx.ith.practica05.LibroFragment.onAttach(LibroFragment.java:81)
…
```

    d. Seleccionamos la liga en donde nos señala el error, dándole clic. Nos lleva al código donde explica la actividad debe implementar: "**`OnListFragmentInteractionListener`**".

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnListFragmentInteractionListener) {
        mListener = (OnListFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString() +
        " must implement OnListFragmentInteractionListener");
    }
}
```

12. Para corregir el error debemos implementar "**`OnListFragmentInteractionListener`**" en la clase "`PrincActivity`".

    a. Abrimos "PrincActivity.java".

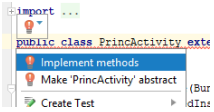
```
package mx.ith.app17fragment;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class PrincActivity extends AppCompatActivity {

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_princ);
    }
}
```

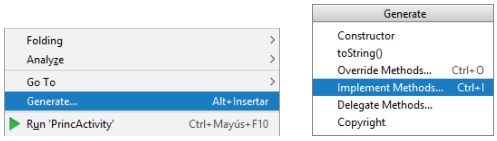
b. A la clase le indicamos que implemente la interface. Agregamos lo siguiente:

"`implements LibroFragment.OnListFragmentInteractionListener`"

c. Al escribir la implementación de la interface nos indica con un foco rojo las opciones para corregir el la agregación de la interface.



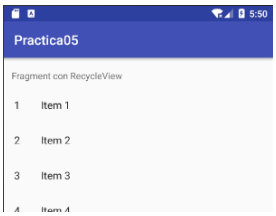
d. Una alternativa es seleccionar con el botón derecho y nos aparece el menú de contexto escoger "Generate" y luego "Implement Methods" y seleccionamos "`OnListFragmentInteractionListener`" y damos "OK"



13. Optamos por la primera opción de implementar los métodos. Y finalmente la clase queda como sigue.

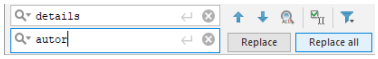
```
public class PrincActivity extends AppCompatActivity
    implements LibroFragment.OnListFragmentInteractionListener {

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

    @Override
    public void onListFragmentInteraction(GeneraLibros.Libro item) {

    }
}
```

14. Al correr nuestro programa nos muestra una pantalla similar a la siguiente:



15. Pero la información no tiene nada relacionada con libros. Vamos a modificar la clase “Libros” para tener dos campos el título y el autor.

    a. Abrimos la clase “GeneraLibros” ubicamos la clase “Libro” (al final de la clase “GeneraLibros”).
    b. Vamos a cambiar el atributo “content” por “nombre” y “details” por “autor” usando replace. El atributo “id” lo dejamos intacto. Seleccionamos primero “content” y damos <Ctrl>R y en la ventana escribimos “nombre”. Repetimos la operación seleccionamos “details” <Ctrl>R y lo reemplazamos por “autor”.



    c. Creamos los getters de los tres atributos (<Alt><Ins> “Getter” y <Shift>clic para incluir cada atributo y “OK”). Y los atributos los dejamos de paquete (borramos ”`public`”). La clase queda como sigue:

```
public static class Libro {
    final String id;
    final String nombre;
    final String autor;

    Libro(String id, String nombre, String autor) {
        this.id = id;
        this.nombre = nombre;
        this.autor = autor;
    }

    public String getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public String getAutor() {
        return autor;
    }

    @Override
    public String toString() {
        return nombre;
    }
}
```

16. Vamos a hacer alguno cambios a la clase “GeneraLibros”.
    a. Como vamos a tener una colección de libros cambiamos “ITEMS” por “LIBROS”.

```
public static final List<Libro> LIBROS = new ArrayList<>();
```

    b. Comentamos (o borramos) algunas líneas innecesarias para nuestro propósito. Se muestran en orden de aparición de arriba abajo. Éstas son:

```
/**
 * A map of sample (dummy) items, by ID.
public static final Map<String, Libro> ITEM_MAP = new HashMap<String,
Libro>();

private static final int COUNT = 25;

static {
    // Add some sample items.
    for (int i = 1; i <= COUNT; i++) {
        addItem(createDummyItem(i));
    }
}

private static void addItem(Libro item) {
    ITEMS.add(item);
    ITEM_MAP.put(item.id, item);
}

private static Libro createDummyItem(int position) {
    return new Libro(String.valueOf(position), "Item " + position,
makeautor(position));
}

private static String makeautor(int position) {
    StringBuilder builder = new StringBuilder();
    builder.append("autor about Item: ").append(position);
    for (int i = 0; i < position; i++) {
        builder.append("\nMore autor information here.");
    }
    return builder.toString();
}

/**
 * A dummy item representing a piece of nombre.
 */
```

c. Creamos los libros de muestra con las siguientes instrucciones.

```
static String[][] infoLibros = {
        {"De (casi) todo se aprende","Paula Gonu"},
        {"El Príncipe de la Niebla","Carlos Ruiz Zafón"},
        {"Detective Conan II n° 89","Meitantei Conan II"},
        {"Leiva. Toquemos juntos hasta que la muerte nos joda","Wilma
Lorenzo"},
        {"Las hijas del Capitán","María Dueñas"},
        {"Donde fuimos invencibles","María Oruña"},
        {"Patria","Fernando Aramburu Irigoyen"},
        {"Morder la manzana","Leticia Dolera"},
        {"El legado de los espías","John le Carré"},
        {"Las almas de Brandon","César Brandon Ndjocu"},
        {"El fuego invisible","Javier Sierra"},
        {"Maestros de la costura","Shine | CR TVE"},
        {"A comer se aprende","Álvaro Vargas"}};

static {
    int id=1;
    for( String[] si:infoLibros){
        Libro libro = new Libro(String.valueOf(id++), si[0], si[1]);
        LIBROS.add(libro);
    }
}
```

17. Los cambios van a afectar a las demás clases. Vamos a corregir la clase “MyLibroRecyclerAcapter”.

a. Abrimos el archivo “MyLibroRecyclerAcapter.java”.
b. Localizamos abajo la clase “`ViewHolder`” y cambiamos el atributo “mitem” por “libro” (<Ctl>R).

```
public class ViewHolder extends RecyclerView.ViewHolder {
    final View mView;
    final TextView mIdView;
    final TextView mContentView;
    public Libro libro;
```

c. Cambiamos en el método “`onBindViewHolder`” las primeras líneas por:

```
@Override
public void onBindViewHolder(@NonNull final ViewHolder holder, int
position) {
    holder.libro = mValues.get(position);
    holder.mIdView.setText(holder.libro.getId());
    holder.mContentView.setText(holder.libro.getNombre());
```

18. Otra clase a corregir es “LibroFragment”. Modificamos “ITEMS” por “LIBROS”.

```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_libro_list, container, false);

    // Set the adapter
    if (view instanceof RecyclerView) {
        Context context = view.getContext();
        RecyclerView recyclerView = (RecyclerView) view;
        if (mColumnCount <= 1) {
            recyclerView.setLayoutManager(new LinearLayoutManager(context));
        } else {
            recyclerView.setLayoutManager(new GridLayoutManager(context,
mColumnCount));
        }
        recyclerView.setAdapter(new MyLibroRecyclerViewAdapter
                (GeneraLibros.LIBROS, mListener));
    }
    return view;
}
```

19. Ejecutamos nuestro programa y obtenemos un resultado siguiente:

20. Ahora le vamos a agregar una función. Ésta función consiste en que cuando le demos clic en un elemento de la lista nos muestre el autor. Modificaremos la clase “PrincActivity”.

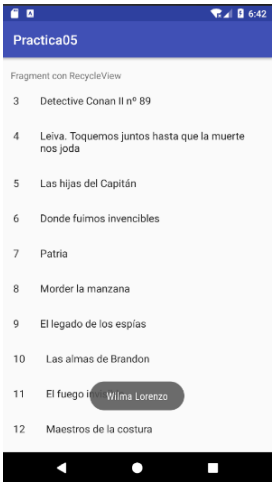
    a. Abrimos la clase “PrincActivity.java”.
    b. Crearemos un mensaje tipo “Toast” en el método:

    “`onListFragmentInteraction`”.

    c. El método modificado se muestra a continuación:

```
@Override
public void onListFragmentInteraction(GeneraLibros.Libro libro) {
    Toast.makeText(this, libro.getAutor(),
            Toast.LENGTH_SHORT).show();
}
```

21. Al correr el programa si se selecciona un nombre de libro “*Leiva. Toquemos juntos hasta que la muerte nos joda*” nos muestra al autor:



22. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Google.** Activities. *The Activity Life Cycle.* [En línea] https://developer.android.com/guide/components/activities/activity-lifecycle.html.
2. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* New York : Apress, 2015.
3. **Phillips, Bill y Hardy, Brian.** *Android Programming: The Big Nerd Ranch Guide.* Indianapolis, IN 46240 USA : Pearson Technology Group, 2013.
4. **Deutsch, Roger.** *Launch Your Android App.* s.l. : Kindle Edition , © 2016.

No. 6. Usando Listas personalizadas.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para programar listas personalizadas.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 4. Unidades y layouts.** Objetivo. Crear una aplicación Android con varias actividades. **No. 5. Usando Listas.** Objetivo. Empleo de la librería ListView. **No. 6. Listas personalizadas.** Objetivo. Conocer los elementos comunes en una interface gráfica. **No. 7. Elementos gráficos.** Objetivo. Crear una librería gráfica personal Círculo.

Introducción

Para crear una pantalla, se crea una clase Java que extiende de la clase base “Activity”. La actividad carga los componentes gráficos (UI) usando un archivo XML definido en el folder *res/layout*. Cada actividad debe ser declarada en el archivo *AndroidManifest.xml*. (1)

Las intenciones (Intents) permite a diferentes actividades de varias aplicaciones trabajen juntas sin problemas como si pertenecieran a una sola aplicación. Una intención encapsula una solicitud para que alguna actividad u otro receptor hagan algo. (2)

Un adaptador es un puente entre los datos de su modelo y la representación visual de esos datos. Si requerimos sobreponernos a las limitaciones de los adaptadores estándar. Android proporciona una clase abstracta llamada "BaseAdapter". Derivamos la clase para crear un adaptador personalizado. (3)

Sin embargo ListView es considerada una clase obsoleta (en Android Studio 3.12 en paleta en legacy se encuentra ListView). Debemos usar RecyclerView para desplegar una lista de elementos. RecyclerView en un widget más avanzado y flexible que ListView. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los "RecyclerView"
4. Obtener ejemplos básicos en Android usando "RecyclerView" personalizados.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de RecyclerView.
2. Seleccionar a varios participantes para exponer ejemplos de los controles comunes en "RecyclerView" personalizados.
3. Seleccionar a voluntarios para explicar "La clase modelo".

4. Llenar el recurso de Wiki con las funciones de “RecyclerView”: “ViewHolder, onCreateViewHolder, onBindViewHolder, getItemCount”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno incluyendo “RecyclerView” personalizada, la clase modelo, “Row layout”, una clase derivada de RecyclerView.
3. Participar en Moodle con el glosario de términos incorporando: tipos de dispositivos Android, Sistemas Operativos (SO), Emulador, versiones de SO y niveles de API.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente
   “Crear una aplicación con una ventana principal con una lista de datos como elemento base para mostrar un diálogo para cada elemento de la lista personalizada”.

   El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica con imágenes donde crea una pantalla principal con elementos definidos por renglón y programados para mostrar diálogos informativos para cada uno”. Se escogió hacerlo con una lista de libros personalizada.

2. Iniciar Android Studio. Al iniciar el programa se muestra una pantalla de inicio donde se selecciona “Start a new Android Studio Project” o en su defecto en el menú seleccionar: “File”, “New”, “New Project”.

3. Una vez dado el inicio de un proyecto de Android se nos pide información inicial para nuestro proyecto. El nombre de nuestro proyecto escribiremos “Practica06”.

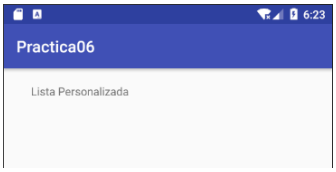
4. Enseguida nos pide el SDK mínimo a usar.
   a. Por el momento no escogemos otro que no sea “Phone and Tablet”.
   b. Dejaremos usar el mínimo SDK como API 15: “IceCreamSandwich”.
   c. Seleccionamos “Next”.

5. Nos presenta un menú para seleccionar el tipo de actividad a usar.
   a. Damos “Next”. Es decir, aceptamos la opción default “Empty Activity”. Nos genera menos código y nos facilita agregar código.

6. Enseguida nos muestra las opciones para nuestra pantalla.
   a. En “Activity Main” escribimos “PrincActivity” y seleccionamos “Finish”.

b. Ahora Android Studio hace su trabajo.
c. En el editor aparecen disponibles "`PrincActivity.java`" y "`activity_princ.xml`".

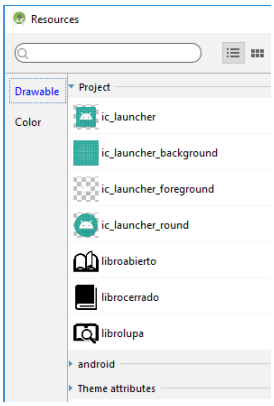
7. Vamos a modificar "activity_princ.xml".
   a. Lo seleccionamos.
   b. Nos aseguramos estar en diseño (Design).
   c. El texto "TextView" indicando "Hello Wold!" lo moveremos al tope y a la izquierda y lo cambiamos a "Lista Personalizada". Lo corremos:



   d. Vamos a incorporar algunas imágenes a nuestro proyecto. Buscamos imágenes de menos de unos 50x50 pixeles en formato png y cuyo nombre esté sólo en letras minúsculas. Para nuestro propósito encontramos las siguientes imágenes: libroabierto.png, librolupa.png, librocerrado.png.

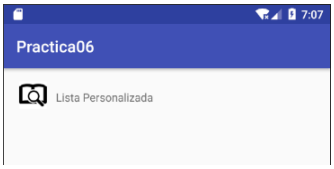
   e. Incorporamos éstas imágenes en nuestro proyecto. Hacemos una copia de ellas al el subdirectorio "…\Practica06\app\src\main\res\drawable".

   f. En diseño agregamos de la paleta en common. Seleccionamos un ImageView y lo agregamos justificado a la izquierda. Nos aparece un menú donde seleccionamos de Drawable -> Project -> librolupa y damos "OK".

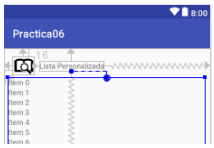


g. Ajustamos nuestro texto al lado derecho de la imagen. Y el tamaño del icono se reduce hasta aparecer cercano al tamaño de la altura del texto.

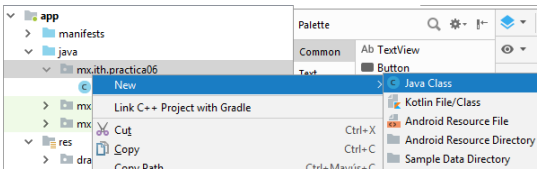
h. Lo corremos y obtenemos lo siguiente:



i. Incorporamos de paleta->commons->RecyclerView lo ajustamos a los márgenes para abarcar toda el área visible debajo de "Lista Personalizada".



8. Crear una clase (MiAdaptador) para la clase RecycleView.
   a. Seleccionamos el paquete de nuestro proyecto. Y con el botón derecho escogemos New-> Java Class



   b. El nombre será "MiAdaptador" y damos "OK". Escribimos el siguiente código para derivar la clase.

```
public class MiAcaptador extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
```

   c. Importamos la librería "RecycleView". Y nos pide implementar los métodos y resulta en lo siguiente:

```
package mx.ith.practica06;

import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;
```

```
public class MiAcaptador extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder,
int position) {

    }

    @Override
    public int getItemCount() {
        return 0;
    }
}
```

d. Dentro de la clase creamos nuestra lista de libros. Creamos la clase ViewHolder. Reescribimos los métodos generados en el paso anterior como sigue:

```
package mx.ith.practica06;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;

public class MiAdaptador extends
RecyclerView.Adapter<MiAdaptador.ViewHolder> {

    List<Libro> listaLibros;
    public MiAdaptador(List<Libro> listaLibros) {
        this.listaLibros = listaLibros;
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        public final TextView tvLibro, tvAutor;
        public ViewHolder(View view) {
            super(view);
            this.tvLibro = view.findViewById(R.id.tvLibro);
            this.tvAutor = view.findViewById(R.id.tvAutor);
        }
    }

    @NonNull
    @Override
    public MiAdaptador.ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        //R.layout.renglon no existe todavía (hasta punto 10)
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.renglon, parent, false);
        return new ViewHolder(view);
    }
```

```
    @Override
    public void onBindViewHolder(@NonNull MiAdaptador.ViewHolder holder,
int position) {
        Libro libro = listaLibros.get(position);
        holder.tvAutor.setText(libro.autor);
        holder.tvLibro.setText(libro.nombre);
    }

    @Override
    public int getItemCount() {
        return listaLibros.size();
    }
}
```

9. Ahora crearemos una clase “Libros” en el mismo paquete anterior. De la siguiente forma.

```
package mx.ith.practica06;

import java.util.ArrayList;
import java.util.List;

public class Libro {
    String nombre,autor;

    public Libro(String nombre, String autor) {
        this.nombre = nombre;
        this.autor = autor;
    }

    public String getNombre() {
        return nombre;
    }

    public String getAutor() {
        return autor;
    }

    public static final List<Libro> LIBROS = new ArrayList<Libro>();

    static String[][] infoLibros = {
            {"De (casi) todo se aprende","Paula Gonu"},
            {"El Príncipe de la Niebla","Carlos Ruiz Zafón"},
            {"Detective Conan II n° 89","Meitantei Conan II"},
            {"Leiva. Toquemos juntos hasta que la muerte nos joda","Wilma
Lorenzo"},
            {"Las hijas del Capitán","María Dueñas"},
            {"Donde fuimos invencibles","María Oruña"},
            {"Patria","Fernando Aramburu Irigoyen"},
            {"Morder la manzana","Leticia Dolera"},
            {"El legado de los espías","John le Carré"},
            {"Las almas de Brandon","César Brandon Ndjocu"},
            {"El fuego invisible","Javier Sierra"},
            {"Maestros de la costura","Shine | CR TVE"},
            {"A comer se aprende","Álvaro Vargas"}};

    static {
        for( String[] si:infoLibros){
            LIBROS.add(new Libro(si[0], si[1]));
        }
    }
}
```

10. Agregaremos como queremos desplegar cada renglón.

    a. En el proyecto app -> res -> layout con el botón derecho escogemos: New -> XML -> Layout XML File.

    b. Escogemos el nombre de renglon (en lugar de layout) y dejamos LinearLayout y damos “Finish”.

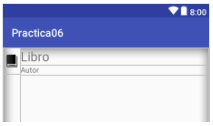
    c. Nos cambiamos a Diseño (Design). Agregamos de common un ImageView y seleccionamos librocerrado.

    d. Ahora incorporamos del lado derecho del libro un LinearLayout(vertical).

    e. Dentro del LinearLayout agregamos dos TextView. Lo que hemos agregado se refleja en lo siguiente.

    f. Cambiamos la propiedad id de cada elemento. Para el libro id=imLibro y padding -> all=5dp, para el Textview superior id=tvLibro, text=Libro y textsize=24sp, y finalmente el Textview inferior id=tvAutor, text=Autor y textsize=14sp.

    g. El tamaño del libro lo ajustamos para que se vea de la siguiente forma.



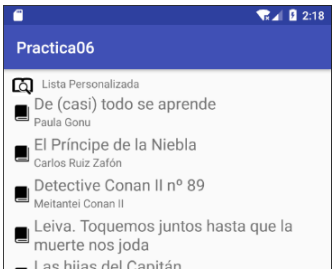
    h. Verificamos que el primer LinearLayout el horizontal tenga el atributo layout_heigth=wrap_content. Si no lo hacemos corremos el riesgo de mostrar un solo libro.

11. Modificaremos la clase principal “PrincActivity.java” para mostrar nuestra lista personalizada de libros. Creamos un atributo como referencia a nuestra vista de tipo RecyclerView y otra referencia a mi adaptador.

```
package mx.ith.practica06;

import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PrincActivity extends AppCompatActivity {

    RecyclerView rvVista;
    MiAdaptador miAdaptador;

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

        rvVista = findViewById(R.id.vista);
        rvVista.setHasFixedSize(true);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(this);
        rvVista.setLayoutManager(mLayoutManager);

        miAdaptador = new MiAdaptador(Libro.LIBROS);
        rvVista.setAdapter(miAdaptador);
    }
}
```

12. Corremos nuestro programa.



13. Ahora vamos a implementar el darle clic a uno de los elementos que nos muestre un mensaje de cual libro hemos seleccionado y que el libro se muestre abierto.

    a. Modificamos la clase principal que implemente "`View.OnClickListener`" y nos pide generar el método "`onClick`". El llamado al constructor de "`MiAdaptador(Libro.LIBROS, this)`" ha sido modificado. El resultado es el siguiente.

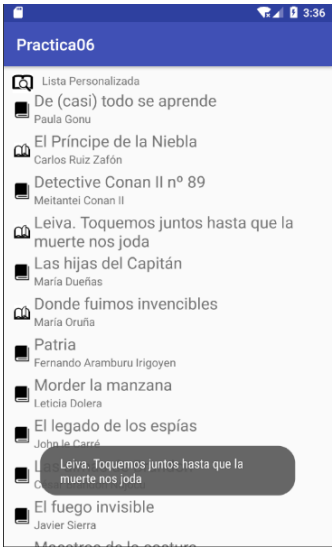
```
public class PrincActivity extends AppCompatActivity implements
View.OnClickListener {

    RecyclerView rvVista;
    MiAdaptador miAdaptador;

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

        rvVista = findViewById(R.id.vista);
        rvVista.setHasFixedSize(true);
```

```
        LinearLayoutManager mLayoutManager = new
LinearLayoutManager(this);
        rvVista.setLayoutManager(mLayoutManager);

        miAdaptador = new MiAdaptador(Libro.LIBROS, this);
        rvVista.setAdapter(miAdaptador);
    }

    @Override
    public void onClick(View v) {
        int itemPosition = rvVista.getChildLayoutPosition(v);
        Libro libro = Libro.LIBROS.get(itemPosition);
        Toast.makeText(this, libro.getNombre(), Toast.LENGTH_LONG).show();
        ImageView icon=(ImageView)v.findViewById(R.id.imLibro);
        icon.setImageResource(R.drawable.libroabierto);
    }
}
```

b. Modificamos la clase "`MiAdaptador`" para incorporar una referencia al método "`onClick(View v)`". Modificamos el constructor de "`ViewHolder`" para reaccionar al evento. Se muestra el resultado de la parte modificada.

```
List<Libro> listaLibros;
static View.OnClickListener clickListener;

public MiAdaptador(List<Libro> listaLibros, View.OnClickListener
clickListener) {
    this.listaLibros = listaLibros;
    this.clickListener = clickListener;
}

public static class ViewHolder extends RecyclerView.ViewHolder{
    public final TextView tvLibro, tvAutor;
    public ViewHolder(View view) {
        super(view);
        view.setOnClickListener(MiAdaptador.clickListener);
        this.tvLibro = view.findViewById(R.id.tvLibro);
        this.tvAutor = view.findViewById(R.id.tvAutor);
    }
}
```

14. Verificamos los cambios y corremos el programa. Si damos clic sobre algunos libros se mostrará el siguiente resultado.



15. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Wei-Meng, Lee.** *Beginning Androi 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
2. **Murphy, Mark L.** *The Busy Coder's Guide to Android Development.* United States of America : CommonsWare, LLC., © 2008-2017.
3. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* New York : Apress, 2015.
4. **Android.** Create a List with RecyclerView. *Developers Documentation.* [En línea] Android, Mar de 2018. https://developer.android.com/guide/topics/ui/layout/recyclerview.

No. 7. Elementos gráficos.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para usar imágenes y animación.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 4. Unidades y layouts.** Objetivo. Crear una aplicación Android con varias actividades. **No. 5. Usando Listas.** Objetivo. Empleo de la librería ListView. **No. 6. Listas personalizadas.** Objetivo. Conocer los elementos comunes en una interface gráfica. **No. 7. Elementos gráficos.** Objetivo. Crear una librería gráfica personal Círculo.

Introducción

La animación permite que un objeto en una pantalla cambie su color, posición, tamaño u orientación a lo largo del tiempo. Las capacidades de animación en Android son prácticas, divertidas y simples. (1)

Las herramientas de compilación nos proporcionan un icono por default para usar en el iniciador de la aplicación. La imagen utilizada varía según la versión de herramientas de Android. Esta imagen la podemos cambiar. (2)

Las transiciones permiten que los cambios realizados en el diseño y la apariencia de las vistas en una interfaz de usuario se activen durante el tiempo de ejecución de la aplicación. (3)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los “TransitionManager”, “Handler”.
4. Obtener ejemplos básicos en Android usando “Handler” y/o “TransitionManager”.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de “Handler” y “ViewGroup”.
2. Seleccionar a varios participantes para exponer ejemplos de las funciones comunes en “TransitionManager”.
3. Seleccionar a voluntarios para explicar “Handler”.
4. Llenar el recurso de Wiki con las funciones de “Image Asset”: “strings.xml”, “ViewGroup”, “Threads”, “Runnable”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.

2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de agregar dos dados. Contendrá “Image Asset” y con nombre personalizado del proyecto usando strings.xml.
3. Participar en Moodle con el glosario de términos incorporando:
    a. La estructura de un proyecto.
        i. Descripción del folder manifest.
        ii. Descripción del folder java.
        iii. Descripción del folder res.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal para simular el lanzamiento de un dado”.
    b. Adicionalmente la aplicación tendrá una imagen distintiva con un nombre de proyecto personalizado.

    El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica con imágenes donde crea una pantalla principal con gráficos mostrando en un hilo una secuencia valores al azar entre 1-6 para mostrar imágenes informativos para cada valor el resultado se muestra en una foto animada en escala”.

2. Crear un proyecto en Android Studio.

    a. Iniciar Android Studio. Seleccionamos del menú: “File”, “New”, “New Project”.

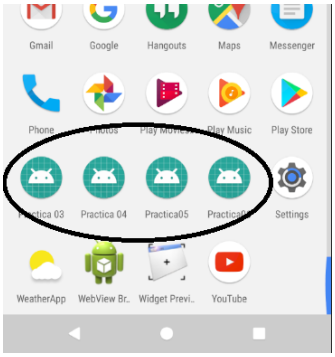
    b. Una vez dado el inicio de un proyecto nos pide información inicial para nuestro proyecto escribiremos “Practica07”.

    c. Enseguida nos pide el SDK mínimo a usar. Usamos “Phone and Tablet” y SDK como API 15.

    d. Del menú seleccionamos “Empty Activity”.

    e. En “Activity Main” escribimos “PrincActivity”.

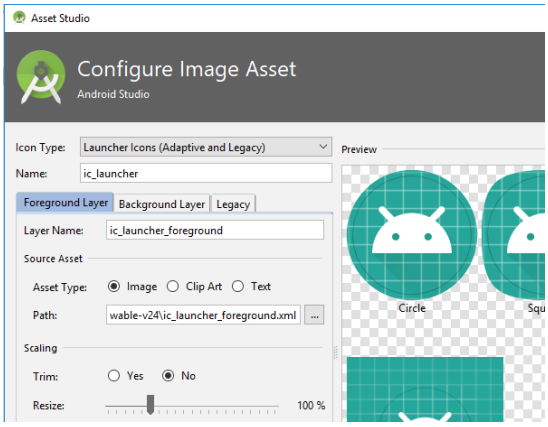
3. Iniciaremos resolviendo el punto 1.b. Cuando una aplicación se crea tiene un distintivo gráfico dependiendo de la versión de Android Studio y del recurso gráfico en curso. De los ejercicios generados hasta hoy todos tienen un mismo ícono  . El nombre del proyecto se asocia con el nombre del proyecto.



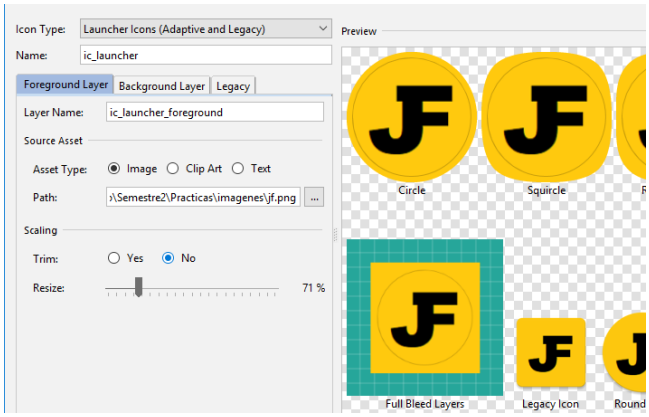
4. Cambiaremos el ícono por default del proyecto.

   a. Se requiere una imagen. Para nuestro propósito empleamos la siguiente imagen.

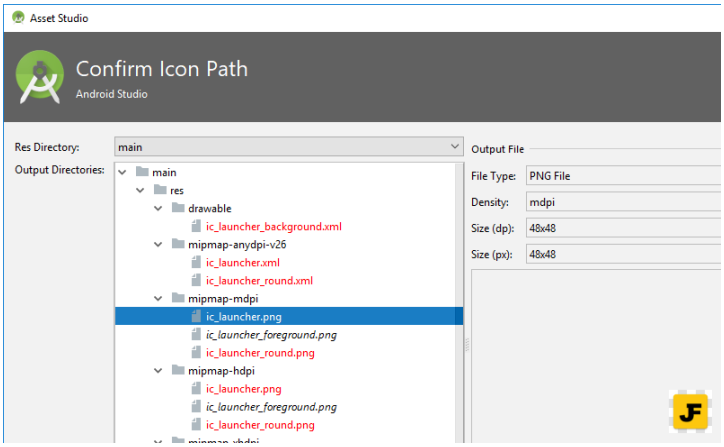
   b. En el proyecto en el subdirectorio “res” con el botón derecho seleccionamos “new” -> “Image Asset” y nos muestra el siguiente menú (mostrado una parte).



c. En “Path” ubicamos la trayectoria de la imagen 4-a. Al darle aceptar la imagen el menú cambia a:



d. Modificamos “Resize” hasta que muestre la imagen incluida dentro del círculo. Escogemos “Next” y nos lleva al siguiente menú.



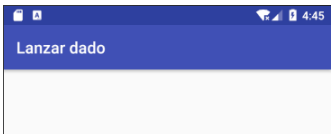
e. En este menú nos muestra las características de los archivos nuevos en reemplaza de los mostrados por default. Escogemos “Finish”.

5. Cambiaremos el nombre por default del proyecto.
   a. Abrimos el archivo “strings.xml” ubicado en “app”->”res”->”values”.

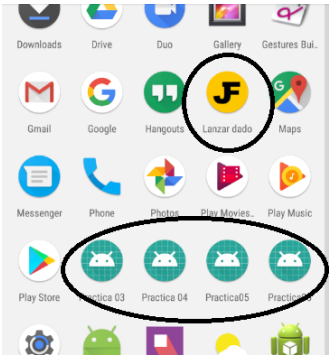
```
<resources>
    <string name="app_name">Lanzar dado</string>
</resources>
```

   b. Cambiamos “**Practica07**” por “**Lanzar dado**”.

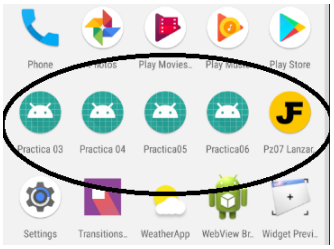
6. Corremos nuestro programa. El resultado se observa al correr la aplicación. Se muestra el nuevo nombre del proyecto.



Si observamos cómo quedó instalada nuestra aplicación. El nombre del proyecto y la imagen han cambiado también a diferencia de los ejercicios previos.



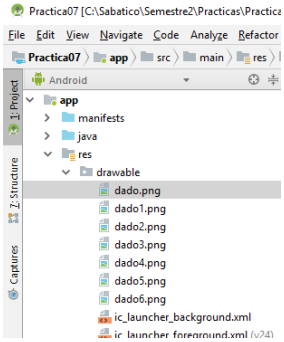
Sin embargo como un estilo propio de continuar en secuencia con los ejercicios. Renombramos el proyecto a “Pz07 Lanzar dado”. Ahora nuestros proyectos aparecen en secuencia por el orden alfabético.



7. Para resolver el problema 1-a usamos una colección de imágenes. Como se trata de lanzar un dado recurrimos a las seis caras de un dado y una imagen adicional para mostrar el estado previo al “lanzamiento”. Todas las imágenes son del mismo tamaño de 74x74 pixeles. Y sus nombres de archivo son: dado1.png, dado2.png … dado6.png y dado.png.

   a. Colocar las imágenes en:
   “…Practicas\Practica07\app\src\main\res\drawable”.

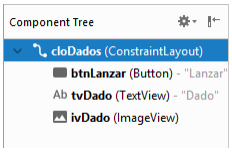
b. Verificar en el proyecto en app -> res -> drawable la incorporación de las imágenes.



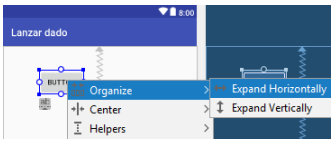
8. Incorporaremos un botón para iniciar las acciones. Se modificará el texto por default “Hello World!”. Y se agregará un “ImageView” donde se mostrará el avance de un “lanzamiento de dado”. Estos cambios se observan en el archivo “activity_princ.xml”.

   a. Abrimos el archivo “activity_princ.xml” y establecemos en modo diseño.

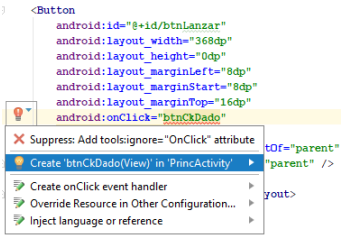
   b. Seleccionamos “ConstraintLayout” y cambiamos su id por “cloDados”.



   c. En la parte superior incorporamos un “Button” con texto “Lanzar”. Le indicamos “Expand Horizontally”. Lo ligamos al tope y al inicio de la ventana contenedora. Su id con “btnLanzar”.



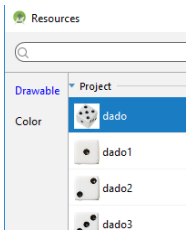
   d. Modificamos el atributo onClick y escribimos “btnCkDado”. Nos vamos a modo texto y generamos el código del evento onClick en PrincActivity.java.



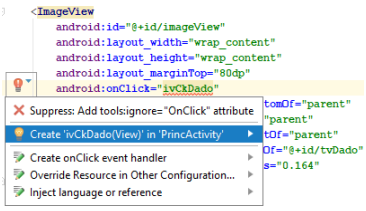
e. Nos lleva al código de PrincActivity.java. Por el momento lo dejamos pendiente. Nos regresamos a activity_princ.xml en modo diseño.

f. Modificamos el “TextView” a “Dado” y su id con “tvDado”. El propósito de éste texto será meramente informativo sobre cuál era el valor anterior del dado y cuál es el actual.

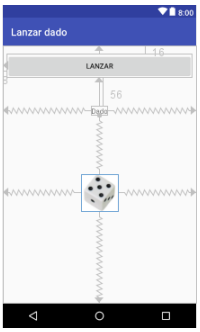
g. Ahora insertaremos un elemento tipo “ImageView”. Nos sirve para mostar el avance del “lanzamiento”. Y de Project escogemos dado.



h. Cambiamos la propiedad id=”ivDado”. Creamos el evento onClick=”ivCkDado”. Nos vamos a modo texto y generamos el código del evento onClick en PrincActivity.java.



i. Acomodamos los elementos para obtener una configuración similar a la siguiente.



9. Creamos los siguientes atributos para la clase “PrincActivity.java”. Al agregarlos nos pide la confirmación de importar las clases correspondientes. Los ubicamos al inicio de la clase.

```
//número de lanzamientos previos al valor final
private static final int NVECES = 20;

//Mensajes para informar al Manejador
private static final int INICIAR = 100;
private static final int SIGDADO = 101;
private static final int TERMINAR = 102;

ImageView imDado;  // La imagen cambiante del dado
TextView tvDado;   // Texto informativo
Manejador m;       // Clase tipo handler usando hilos en Android
int nVeces;        // contador de 0-NVECES
Random rnd;        // generador aleatorio
int valorAnterior; // valor del dado previo.
int valorActual;   // valor actual del dado.


//Arreglo de imágenes representado a los dados del 1-6
int[] resIm = {R.drawable.dado1, R.drawable.dado2, R.drawable.dado3,
        R.drawable.dado4, R.drawable.dado5, R.drawable.dado6};

ViewGroup tContainer; // El contenedor del botón, texto e imagen
```

Al agregar los atributos nos indica un error en la inexistencia de la clase “Manejador” la cual vamos a crear.

10. La clase “Manejador” es una clase derivada de la clase “Handler”. La función de la clase “Handler” proporciona el manejo de Hilos de acuerdo a Android para modificar los elementos gráficos en forma “segura”. La clase la incorporamos como clase interna a la la clase “PrincActivity.java”. La ubicaremos al final de la clase.
Tiene un solo método reescrito de ésta clase (PrincActivity). Cuando recibe un mensaje de INICIAR crea un hilo y si no actualiza un mensaje y la imagen del dado. En caso de recibir un mensaje TERMINAR muestra el valor final y agranda la imagen del dado.

```
class Manejador extends Handler {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.what == INICIAR) {
            new Thread(PrincActivity.this).start();
        } else {
            imDado.setImageResource(resIm[msg.arg1]);
            valorActual = msg.arg1 +1;
            tvDado.setText("Anterior="+valorAnterior+" Actual=" + valorActual +
               " Suma="+ (valorActual+valorAnterior));
            if (msg.what == TERMINAR) {
                Toast.makeText(PrincActivity.this,
                        "Valor=" + valorActual, Toast.LENGTH_LONG).show();
                escalarImagen(true);
                valorAnterior = valorActual;
            }
        }
```

```
    }
}
```

Ahora tenemos dos errores. No está definido el constructor para la clase Thread en el primer subrayado del código anterior. Tampoco se encuentra "escalarImagen".

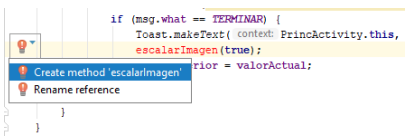
Para resolver el primer error implementamos la interface Runnable y le indicamos que implemente el método run.

```
public class PrincActivity extends AppCompatActivity
    implements Runnable{
…

@Override
public void run() {

}
…
```

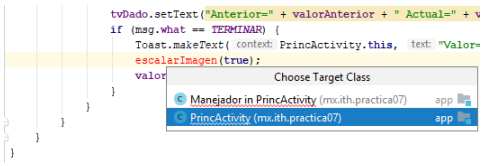
Una vez generado el método podemos escribir el siguiente código. Creamos la variable "faltan" para determinar cuántos elementos se muestran antes de terminar. Otra variable "sigue" indica mostrar un nuevo valor de dado. Para mostrar los valores más rápido al principio (75 ms) o más lento al final (500 ms).

```
@Override
public void run() {
    int faltan = NVECES - nVeces;
    boolean sigue = nVeces++ < NVECES;
    try {
        Message msg = new Message();
        msg.arg1 = rnd.nextInt(6); //0-5
        if (sigue) {
            msg.what = SIGDADO;
        } else {
            msg.what = TERMINAR;
        }
        m.sendMessage(msg);
    } catch (Exception ignored) {
    } finally {
        int delay;
        if (sigue) {
            delay = faltan < 5 ? 500:75;
            m.postDelayed(this, delay);
        }
    }
}
```

El otro problema es la inexistencia del método "`escalarImagen(true);`". Damos clic en el método (Ctl <Enter>) para generarlo.



Lo generamos y nos pregunta si lo generamos en la clase “Manejador” o “PrincActivity”. Seleccionamos la segunda opción.



Una vez generado insertamos el siguiente código el cual hace que la imagen ocupe la mayor parte del despliegue a través de una transición.

```
@TargetApi(21)
private void escalarImagen(boolean grande) {
    private void escalarImagen(boolean grande) {
        TransitionManager.beginDelayedTransition(tContainer,
                new TransitionSet().addTransition(new
ChangeBounds()).addTransition(new ChangeImageTransform()));

        ViewGroup.LayoutParams params = imDado.getLayoutParams();
        if (grande) {
            params.height = ViewGroup.LayoutParams.MATCH_PARENT;
            params.width = ViewGroup.LayoutParams.MATCH_PARENT;
        } else {
            params.height = ViewGroup.LayoutParams.WRAP_CONTENT;
            params.width = ViewGroup.LayoutParams.WRAP_CONTENT;
        }
        imDado.setLayoutParams(params);
    }
}
```

11. Nos falta programar que hacemos con los eventos “btnCkDado” y “ivCkDado”. Cuando demos un clic en el botón y cuando demos clic en la imagen. Al presionar el botón si no está en proceso hacer la imagen pequeña y mandar el mensaje de INICIAR en caso contrario mostrar el mensaje. Si en la imagen se la da un clic la imagen se muestra pequeña.

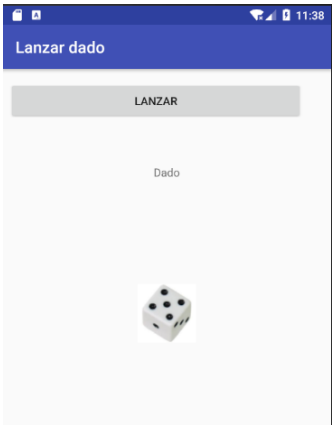
```
public void btnCkDado(View view) {
    if (nVeces >= NVECES) {
        escalarImagen(false);
        nVeces = 0;
        m.sendEmptyMessage(INICIAR);
    } else {
        Toast.makeText(PrincActivity.this,
                "Espere", Toast.LENGTH_LONG).show();
    }
}

public void ivCkDado(View view) {
    escalarImagen(false);
}
```

12. Falta relacionar las referencias con el archivo “activity_princ.xml" e inicializar variables. Esto lo hacemos dentro del método “onCreate”.

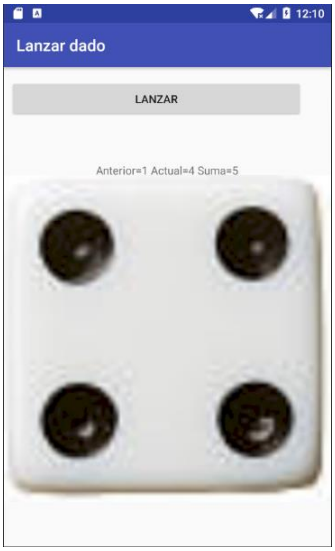
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        //Asocia las referencias con las definidas en activity_princ.xml
        tvDado = findViewById(R.id.tvDado);
        imDado = findViewById(R.id.ivDado);
        tContainer = findViewById(R.id.loTDados);

        // Inicializar m, un generador de números e indicar sin proceso
        m = new Manejador();
        rnd = new Random();
        nVeces = NVECES;
    }
```

13. Corremos el programa. Obtenemos la siguiente pantalla inicial.



14. Al presionar el botón “Lanzar” muestra en forma cambiante varios valores del dado en imágenes hasta mostrar el valor final.



15. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* 2015.
2. **Murphy, Mark L.** *The Busy Coder's Guide to Android Development.* United States of America : CommonsWare, LLC., © 2008-2017.
3. **Smyth, Neil.** *Android Studio Development Essentials.* 2015.

No. 8. Animaciones.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para usar animación de Listas.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.

Introducción

La animación permite que un objeto en una pantalla cambie su color, posición, tamaño u orientación a lo largo del tiempo. Las capacidades de animación en Android son prácticas, divertidas y simples. (1)

La diferencia entre una imagen regular y un Sprite es que un Sprite encapsulará los datos de la imagen y los métodos necesarios para manipularla. (2)

La clase ExecutorService simplifican la programación concurrente manejando los objetos de tipo Thread por nosotros. (3)

Ejemplo basado en “Programación de Juegos Android” y los recursos de igual manera. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre “Graphics architecture”, “Surface”, “SurfaceHolder”.
4. Obtener ejemplos básicos en Android usando “SurfaceView”.
5. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de “Graphics architecture” en Android.
2. Seleccionar a varios participantes para exponer ejemplos de las funciones comunes en “Graphics architecture” en Android.
3. Seleccionar a voluntarios para explicar “Surface”, “SurfaceHolder”.

4. Llenar el recurso de Wiki con las definiciones de “Sprite”. Las funciones de las interfaces “Runnable”, “SurfaceHolder.Callback”, “View.OnTouchListener” y los métodos por implementar.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de incorporar su propia animación.
3. Participar en Moodle con el glosario de términos incorporando:
    a. Dentro de la estructura de un proyecto.
        i. Descripción del folder res.drawable.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal para simular la animación de varios Sprites y acabarlos al seleccionar a cada uno dejando una marca”.

    El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde crea una pantalla principal con gráficos mostrando en un hilo una secuencia valores de posición al azar y mostrar un cuadro a la vez de un bitmap conteniendo varios cuadros. Se escogerán hasta 12 bitmaps”.

2. Crear un proyecto en Android Studio.

    a. Iniciar Android Studio. Seleccionamos del menú: “File”, “New”, “New Project”.

    b. Una vez dado el inicio de un proyecto nos pide información inicial para nuestro proyecto escribiremos “Practica08”.

    c. Enseguida nos pide el SDK mínimo a usar. Usamos “Phone and Tablet” y SDK como API 15.

    d. Del menú seleccionamos “Empty Activity”.

    e. En “Activity Main” escribimos “PrincActivity”.

Tomamos una imagen para iniciar el proceso de construcción de nuestro modelo. Para esta parte insertamos la siguiente imagen en el subdirectorio res.drawable llamada "jf.png".

3. Formamos una primera aproximación al resultado.
    a. Creamos una clase GameView que derive de View.
    b. La clase tiene un objeto de tipo Bitmap llamado bmp. En un constructor creamos el objeto a partir de la imagen.
    c. Reescribimos el método onDraw. Para ello borramos el área de dibujo también llamada Canvas y sobre ella escribimos la imagen representada por el objeto guardado en bmp.
    d. La clase queda como sigue.

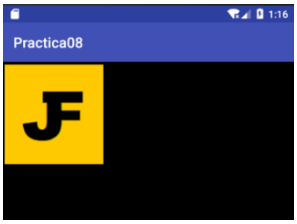
```
package mx.ith.anima;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.View;

public class GameView extends View {
    private Bitmap bmp;

    public GameView(Context context) {
        super(context);
        bmp = BitmapFactory.decodeResource(getResources(),
R.drawable.droid );
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawColor(Color.BLACK);
        canvas.drawBitmap(bmp, 10, 10, null);
    }
}
```

    e. Abrimos la clase PrincActivity.java y la modificamos para presentar como su área de despliegue un objeto de la clase GameView. Comentamos el llamado al método actual setContentview y lo reemplazamos. La clase modificada queda como sigue.

```
package mx.ith.anima;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;

public class MainActivity extends AppCompatActivity {

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

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(new GameView(this));
    }
}
```

f. Corremos el programa y obtenemos el siguiente resultado.



4. Sin embargo la clase View es del nivel apropiado. Requerimos hacer movimientos continuos de las imágenes. Para un control más apropiado usaremos en lugar de View una clase de bajo nivel como lo es SurfaceView.
   Junto con la clase SurfaceView está asociada la clase SurfaceHolder. La clase SurfaceHolder nos permite tomar el control del Canvas (área de dibujo). Adicionalmente nos informa por medio de SurfaceHolder.Callback si está creada el área de dibujo o si ha cambiado o ha sido destruida.
   Necesitamos crear un hilo independiente del hilo de GUI. Para ello incorporamos la clase Executors y programamos la clase GameView para la creación de un hilo a través del método scheduleWithFixedDelay y hacemos la implementación de Runnable para crear el método run donde se lleva a cabo el ciclo continuo de animación y a su vez manda llamar continuamente a Dibuja. Pero si no está disponible el área de dibujo establecemos “running” a falso de otra forma es verdadero. En Dibuja se plasma el bitmap en el Canvas y se va incrementando la posición x.
   La clase nos queda:

```
package mx.ith.practica08;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.view.SurfaceHolder;
```

```
import android.view.SurfaceView;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class GameView extends SurfaceView implements Runnable,
SurfaceHolder.Callback {
    final int FPS = 20; //Cuadros por segundo desplegados
    final long tiempoEspera = 1000 / FPS; // Convercion a tiempo en milisegundos
    ScheduledExecutorService executor; // Clase que genera un hilo
    final int VELOCIDAD = 3;
    private boolean running = false; // En caso de suspención de nuestra App

    private Bitmap bmp; // El dibujo a mostrar
    private SurfaceHolder holder;

    private int x = 0;  // variable que nos señala la primer posición
    private int xVel = 10; // incremento de x

    public GameView(Context context) {
        super(context);
        holder = getHolder();
        holder.addCallback(this);

        bmp = BitmapFactory.decodeResource(getResources(), R.drawable.jf);

        executor = Executors.newScheduledThreadPool(1);
        executor.scheduleWithFixedDelay(this, 0, tiempoEspera,
TimeUnit.MILLISECONDS);
    }

    protected void Dibuja(Canvas canvas) {
        if (x >= getWidth() - bmp.getWidth()) {
            xVel = -VELOCIDAD;
        }
        if (x <= 0) {
            xVel = VELOCIDAD;
        }
        x = x + xVel;
        canvas.drawColor(Color.BLACK);
        canvas.drawBitmap(bmp, x, 10, null);
    }

    public void setRunning(boolean run) {
        running = run;
    }

    @Override
    public void run() {
        if (running) {
            Canvas c = null;
            try {
                c = holder.lockCanvas();
                synchronized (getHolder()) {
                    if (c != null){
                        Dibuja(c);
                    }
                }
            } finally {
                if (c != null) {
                    getHolder().unlockCanvasAndPost(c);
                }
            }
        }
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        setRunning(true);
```

```
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int
height) {
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        setRunning(false);
    }
}
```

Al correr el programa vemos el resultado. Es una imagen moviéndose de repetidamente de izquierda a derecha y viceversa.

5. Ahora crearemos la funcionalidad la imagen en una clase llamada Sprite. Cuyo propósito es encapsular varias funciones. Entre ellas es guardar la posición (x,y) y la velocidad de desplazamiento. Una imagen contiene varios cuadros y será responsabilidad de la clase escoger cual cuadro será desplegado.
   a. La imagen o Bitmap almacenado es similar a la siguiente imagen. La cual contiene cuatro renglones por tres columnas. En cada renglón de arriba a abajo representa una secuencia abajo, izquierda, derecha, arriba. Y cada secuencia es de tres movimientos correspondiente a cada columna. Agregamos la imagen good1.png a res.drawable.

   b. Cuando la superficie está disponible o ha cambiado a través del método surfacechanged, guardamos en dos variables globales ancho y alto del área de dibujo (Canvas).
   c. Las clases quedan como sigue:

```
public class GameView extends SurfaceView implements Runnable,
SurfaceHolder.Callback {
    final int FPS = 20; //Cuadros por segundo desplegados
    final long tiempoEspera = 1000 / FPS; // Conversión a milisegundos
    ScheduledExecutorService executor; // Clase que genera un hilo
    private boolean running = false; // En caso de suspensión
    private SurfaceHolder holder;  // Referencia a la clase Canvas

    static int gblAncho, gblAlto;
    private Sprite sprite; //Nuestro dibujo

    public GameView(Context context) {
        super(context);
        holder = getHolder();
        holder.addCallback(this);
        executor = Executors.newScheduledThreadPool(1);
        executor.scheduleWithFixedDelay(this, 0, tiempoEspera,
```

```
TimeUnit.MILLISECONDS);

        Bitmap bmp = BitmapFactory.decodeResource(
            getResources(), R.drawable.good1); // Cambio a la nueva imagen
        sprite = new Sprite(bmp);
    }

    public void setRunning(boolean run) {
        running = run;
    }

    public void Dibuja(Canvas canvas) {
        sprite.Dibuja(canvas);
    }

    @Override
    public void run() {
        if (running) {
            Canvas c = null;
            try {
                c = holder.lockCanvas();
                synchronized (getHolder()) {
                    if (c != null) {
                        Dibuja(c);
                    }
                }
            } finally {
                if (c != null) {
                    getHolder().unlockCanvasAndPost(c);
                }
            }
        }
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        setRunning(true);
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {
        gblAncho = width;
        gblAlto = height;
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        setRunning(false);
    }
}



public class Sprite {

    private final int VELOCIDAD = 3;
    private int x = 0;  // variable que nos señala la primer posición
    private int xVel = 10; // incremento de x
    private Bitmap bmp; // El dibujo a mostrar

    public Sprite(Bitmap bmp) {
        this.bmp = bmp;
    }

    protected void Dibuja(Canvas canvas) {
        if (x >= GameView.gblAncho - bmp.getWidth()) {
            xVel = -VELOCIDAD;
        }
```

```
            if (x <= 0) {
                xVel = VELOCIDAD;
            }
            x = x + xVel;
            canvas.drawColor(Color.BLACK);
            canvas.drawBitmap(bmp, x, 10, null);
        }
    }
```

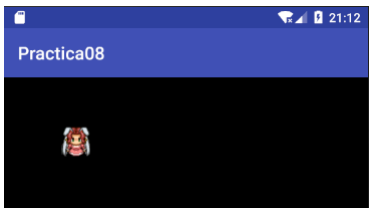
d. Ahora corremos nuestro programa y vemos el resultado. Este resultado es similar al punto 4. La diferencia aparente es mostrar una imagen diferente.

6. Ahora sólo trabajaremos con la clase Sprite. Como todas las imágenes de éste ejemplo se muestran en 4x3 definimos las constantes REN y COL. Vamos a definir Ancho y Alto para cada cuadro de nuestra imagen. Y la velocidad será variable entre (-5 y 5) y crearemos una variable Random. Asimismo una función getRenglon para obtener el renglón de la animación entre 0..3 dependiendo de la velocidad de cambio DX y DY. Si el incremento en DX es mayor que DY se mueve a la izquierda o la derecha. En caso contrario se mueve de arriba o abajo. Para ambos casos se determina el caso final dependiendo del signo de DY. La clase Sprite queda:

```
public class Sprite {
    private final static Random RND = new Random();

    private int x;  // Posición (x,y)
    private int y;
    private int xVel;   // incremento de (x,y)
    private int yVel;

    private Bitmap bmp; // El dibujo a mostrar
    private final int RENS = 4; // tiene 4*3 cuadros
    private final int COLS = 3;
    private final int ANCHO; // Ancho del cuadro a mostrar
    private final int ALTO;  // Alto del cuadro a mostrar
    private int mostrandoCuadro; // 0..2 cuadros

    private int renglon;

    public Sprite(Bitmap bmp) {
        this.bmp = bmp;
        ANCHO = bmp.getWidth() / COLS;
        ALTO = bmp.getHeight() / RENS;
        xVel = RND.nextInt(10) - 5;
        yVel = RND.nextInt(10) - 5;
        renglon = getRenglon(xVel, yVel);
    }

    private void Actualiza() {
        if (x > GameView.gblAncho - ANCHO - xVel || x + xVel < 0) {
            xVel = -xVel;
            renglon = getRenglon(xVel, yVel);
        }
        x = x + xVel;
        if (y > GameView.gblAlto - ALTO - yVel || y + yVel < 0) {
            yVel = -yVel;
            renglon = getRenglon(xVel, yVel);
        }
```

```
            y = y + yVel;
            mostrandoCuadro = ++mostrandoCuadro % COLS;
        }

        protected void Dibuja(Canvas canvas) {
            Actualiza();
            int srcX = mostrandoCuadro * ANCHO;
            int srcY = renglon * ALTO;
            Rect src = new Rect(srcX, srcY, srcX + ANCHO, srcY + ALTO);
            Rect dst = new Rect(x, y, x + ANCHO, y + ALTO);
            canvas.drawBitmap(bmp, src, dst, null);
        }

        private int getRenglon(int dx, int dy) {
            boolean negy;
            if (dy < 0) {
                negy = true;
                dy = -dy;
            } else {
                negy = false;
            }
            if (dx > 0) {
                return dx > dy ? 2 : (negy ? 3 : 0);
            } else if (dx < 0) {
                return -dx > dy ? 1 : (negy ? 3 : 0);
            } else {
                return negy ? 3 : 0;
            }
        }
    }
```

Al correr la nueva versión modificada vemos nuestra animación desplazándose sobre el área de dibujo.



7. Una modificación importante es el tiempo de creación de los Sprites. Si deseamos definir una posición aleatoria en función de las variables globales "*gblAncho*, *gblAlto*". Estas variables no se encuentran disponibles hasta ser creada la superficie de dibujo.
    a. Si incorporamos en el constructor de Sprite las siguientes instrucciones nuestro programa va a tener problemas al ejecutarlo.

```
x = RND.nextInt(GameView.gblAncho - ANCHO);
y = RND.nextInt(GameView.gblAlto - ALTO);
```

    b. Para resolverlo en cambiamos la construcción del sprite ubicado en el constructor de GameView a Surface created.

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    gblAncho = getWidth();
    gblAlto = getHeight();
    Bitmap bmp = BitmapFactory.decodeResource(
        getResources(), R.drawable.good1);
    sprite = new Sprite(bmp);
    setRunning(true);
}
```

c. Ahora al ejecutarse no tiene problemas. Y nuestro Sprite inicia en cualquier posición del área de dibujo.

8. Vamos a incorporar ahora la creación de varios Sprites y los incorporaremos en una lista.
   a. Creamos un lista de sprites dentro de los atributos de la clase GameView. Y eliminamos el objeto sprite.

```
private List<Sprite> listaSprites = new ArrayList<>();
```

b. Agregamos las imagines a R.drawable desde bad1..bad6 y good1..good6.

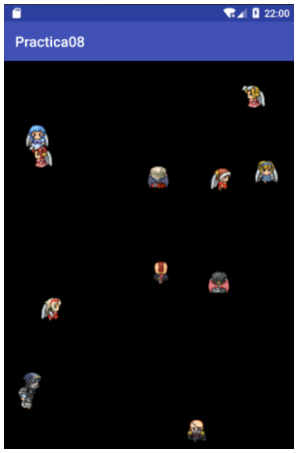
c. Incorporamos los métodos siguientes junto con la modificación a surfaceCreated.

```
private Sprite crearSprite(int resouce) {
    Bitmap bmp = BitmapFactory.decodeResource(getResources(), resouce);
    return new Sprite(bmp);
}

private void crearSprites() {
    listaSprites.add(crearSprite(R.drawable.bad1));
    listaSprites.add(crearSprite(R.drawable.bad2));
    listaSprites.add(crearSprite(R.drawable.bad3));
    listaSprites.add(crearSprite(R.drawable.bad4));
    listaSprites.add(crearSprite(R.drawable.bad5));
    listaSprites.add(crearSprite(R.drawable.bad6));
    listaSprites.add(crearSprite(R.drawable.good1));
    listaSprites.add(crearSprite(R.drawable.good2));
    listaSprites.add(crearSprite(R.drawable.good3));
    listaSprites.add(crearSprite(R.drawable.good4));
    listaSprites.add(crearSprite(R.drawable.good5));
    listaSprites.add(crearSprite(R.drawable.good6));
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    gblAncho = getWidth();
    gblAlto = getHeight();
    crearSprites();
    setRunning(true);
}
```

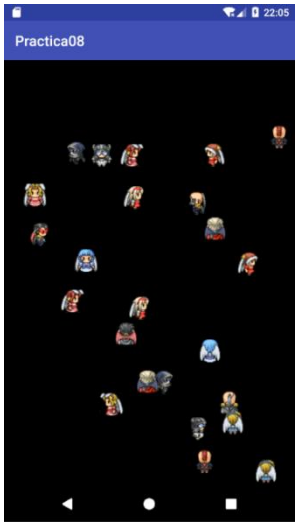
d. Ahora para desplegar la lista de Sprites modificamos el método Dibuja.

```
public void Dibuja(Canvas canvas) {
    canvas.drawColor(Color.BLACK);
    for( Sprite si:listaSprites){
        si.Dibuja(canvas);
    }
}
```

e. Si corremos nuestra aplicación obtenemos los 12 sprites haciendo su recorrido por el área de dibujo.



f. Pero si pausamos la aplicación y la reabrimos de nuevo resultan 24 sprites moviéndose por la pantalla.



g. Como hemos cambiado la construcción de los sprites al método surfaceCreated el comportamiento es diferente. Al pausar la aplicación se

destruye el área de dibujo. Al reiniciar la aplicación se vuelve a construir una nueva superficie de dibujo con la creación de nuevos Sprites. Para evitar éste comportamiento agregamos una variable denominada primera vez inicializada con true. Al llamar a surfaceCreated se actualiza con false. Y prevenimos la creación de nuevos elementos al pausar la aplicación.

```
private boolean primeravez = true;

@Override
public void surfaceCreated(SurfaceHolder holder) {
    gblAncho = getWidth();
    gblAlto = getHeight();
    if(primeravez) {
        crearSprites();
        primeravez=false;
    }
    setRunning(true);
}
```

9. En ésta ocasión vamos a eliminar un Sprite al “tocarlo”.
    a. Vamos a implementar en la clase la interface View.OnTouchListener. Nos pide generar el método onTouch y aceptamos.

```
public class GameView extends SurfaceView implements Runnable,
        SurfaceHolder.Callback, View.OnTouchListener{

...

@Override
public boolean onTouch(View v, MotionEvent event) {
    return false;
}
```

    b. En el constructor agregamos.

```
setOnTouchListener(this);
```

    c. Una de las mejores prácticas para programas en Android, sugiere no mezclar hilos con el hilo de la UI. Por ello dejaremos indicado cuando un evento de tipo Touch lo lleve a cabo dentro del hilo del método run. Para éste propósito lo dejamos indicado en la variable booleana touched y la posición en (x_touch,y_touch).

```
private boolean touched = false;
private float x_touch;
private float y_touch;
```

d. Antes de la creación del método Touched. Requerimos obtener de un sprite si su área insersecta un punto (x_touch,y_touch). Dentro de sprite creamos el método.

```
public boolean seIntersecta(float x2, float y2) {
    return x2 > x && x2 < x + ANCHO && y2 > y && y2 < y + ALTO;
}
```

e. Creamos un método Touched para ser invocado en el método run. Y si hay un punto dentro del área del Sprite significa la desaparición de él. Se hace un recorrido de la lista y si hay coincidencia con un Sprite éste es removido de la lista.

```
protected void Touched() {
    for (int i = sprites.size() - 1; i >= 0; i--) {
        Sprite sprite = sprites.get(i);
        if (sprite.seIntersecta(x_touch, y_touch)) {
            sprites.remove(sprite);// remueve un solo sprite
            break;
        }
    }
}
```

f. Modificamos el método run para llamar a Touched en caso cuando la variable touched sea verdadera.

```
@Override
public void run() {
    if (running) {
        Canvas c = null;
        try {
            c = holder.lockCanvas();
            synchronized (getHolder()) {
                if (c != null) {
                    if (touched) {
                        touched = false;
                        Touched();
                    }
                    Draw(c);
                }
            }
        } finally {
            if (c != null) {
                getHolder().unlockCanvasAndPost(c);
            }
        }
    }
}
```

g. Sólo nos falta modificar el método onTouch donde dejamos indicado a través de los atributos previos si se ha invocado a éste método para posteriormente llamar a Touched.

```
@Override
public boolean onTouch(View v, MotionEvent event) {
    x_touch = event.getX(); //Touch no puede estar en el hilo UI
    y_touch = event.getY();
    touched = true;
    return true;
}
```

h. Al correr el programa nos permite desaparecer un sprite seleccionado.

10. Como punto final agregaremos un sprite temporal al desaparecer un sprite dejaremos una marca. Y ésta desaparecerá momentos después.
    a. Incorporamos la imagen “blood1.png”. Insertamos el atributo conteniendo el bitmap. Y lo inicializamos en el constructor.

```
private Bitmap bmpBlood;

...

bmpBlood = BitmapFactory.decodeResource
                (getResources(), R.drawable.blood1);
```

b. Creamos una clase llamada TempSprite. La clase muestra una imagen por un tiempo predeterminado y luego ya no la dibuja.

```
public class TempSprite {
    private float x;
    private float y;
    private Bitmap bmp;

    private long tiempoArranque;
    private long duracion;
    private boolean timeUp;

    public TempSprite(float x, float y, Bitmap bmp, long duracion) {
        this.x = Math.min(Math.max(x - bmp.getWidth() / 2, 0),
GameView.gblAncho - bmp.getWidth());
        this.y = Math.min(Math.max(y - bmp.getHeight() / 2, 0),
GameView.gblAlto - bmp.getHeight());
        this.bmp = bmp;

        timeUp = false;
        this.duracion = duracion;
        tiempoArranque = System.currentTimeMillis();
    }

    public void Dibuja(Canvas canvas) {
        Actualiza();
        canvas.drawBitmap(bmp, x, y, null);
```

```
    }

    public boolean isTimeUp() {
        return timeUp;
    }

    private void Actualiza() {
        if (timeUp) {
            return;
        }
        long transcurrido = System.currentTimeMillis() - tiempoArranque;
        if (transcurrido >= duracion) {
            timeUp = true;
        }
    }
}
```

c. En la clase GameView creamos una lista de sprites temporales.

```
private List<TempSprite> listaTmpSprites = new ArrayList<>();
```

d. Para las imágenes fijas todas van a tener un mismo bmp.
e. Agregamos un sprite temporal al eliminar un sprite normal. Incorporamos el elemento a la lista de la siguiente forma.

```
protected void Touched() {
    for (int i = sprites.size() - 1; i >= 0; i--) {
        Sprite sprite = sprites.get(i);
        if (sprite.isCollition(x_touch, y_touch)) {
            sprites.remove(sprite);// remueve un solo sprite

   tmp = new TempSprite(x_touch, y_touch, bmpBlood, 150);
   listaTmpSprites.add(tmp);

            break;
        }
    }
}
```

f. Modificamos el método Draw. Hacemos un barrido de la lista de listaTmpSprites y si ya pasó el tiempo de despliegue lo quitamos. El método Dibuja se muestra.

```
public void Dibuja(Canvas canvas) {
    canvas.drawColor(Color.BLACK);

    for (int i = listaTmpSprites.size() - 1; i >= 0; i--) {
        TempSprite sprite = listaTmpSprites.get(i);
        if (sprite.isTimeUp()) {
            listaTmpSprites.remove(sprite);
        } else {
            sprite.Dibuja(canvas);
        }
    }
```

```
        for (Sprite si : listaSprites) {
            si.Dibuja(canvas);
        }
    }
```

g. Al correr nuestro programa y eliminamos una animación resulta en lo siguiente.

11. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* 2015.
2. **Harbour, J. S.** *Beginning Java SE 6 Grame Programming.* Boston : Course Technology, 2012.
3. **Eckel, B.** *Piensa en Java.* Madrid : Prentice Hall, 2003.
4. **edu4JAVA.** Juegos en Android. *Introducción y panorama general.* [En línea] [Citado el:] http://edu4java.com/es/androidgame/androidgame1.html.

No. 9. Menús.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para crear y usar menús.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.

Introducción

Una barra de acción es una parte de la pantalla, generalmente en la parte superior, que es persistente en varias vistas. Proporciona algunas funciones clave: Marca de la aplicación, área de iconos, área de título, área de acción clave, área de menú. (1)

El área superior derecha de la barra de herramientas (action bar) está reservada para el menú. El menú consta de elementos y pueden realizar una acción. (2)

Las API de Menú presenta al usuario tres tipos fundamentales de menús: Menú de opciones, menú contextual y menú emergente. (3)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre “Action Bar”, “Menú de opciones”.
4. Obtener ejemplos básicos en Android usando “Menú de opciones”.
5. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de “Tipos de menús” en Android.
2. Seleccionar a varios participantes para exponer ejemplos de los tipos comunes de “Menús” en Android.
3. Seleccionar a voluntarios para explicar “Action bar”, “Tipos de menús”.
4. Llenar el recurso de Wiki con las definiciones de “Action Bar”. Las funciones de “onCreateOptionsMenu”, “OnOptionsItemSelected”, “MenuItem”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de incorporar su propio menú.
3. Participar en Moodle con el glosario de términos incorporando:
    a. Dentro de la estructura de un proyecto.
        i. Descripción del folder res.menu.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal mostrando dos tipos de menús indicando una acción al tocar cada MenuItem”.

    El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde crea una pantalla principal introduciendo dos de los tipos de menús y al activar una acción crear un mensaje tipo Toast con un texto indicativo”.

2. Crear un proyecto en Android Studio.

    a. Iniciar Android Studio. Seleccionamos del menú: “File”, “New”, “New Project”.

    b. Una vez dado el inicio de un proyecto nos pide información inicial para nuestro proyecto escribiremos “Practica09”.

    c. Enseguida nos pide el SDK mínimo a usar. Usamos “Phone and Tablet” y SDK como API 15.

    d. Del menú seleccionamos “Empty Activity”.

    e. En “Activity Main” escribimos “PrincActivity”.

3. Crear un menú de opciones.

    a. Crear el subdirectorio de recursos “res/menu”. Si no existe éste subdirectorio nos ubicamos sobre el directorio res y con el botón derecho seleccionamos “New” -> “Android Resource Directory” y escribimos como nombre “menu” y de tipo “menu”.

b. Creamos el archivo mimenu.xml. Sobre el directorio “res/menu” con el botón derecho seleccionamos “Menu Resource File” y escribimos “minenu”.

c. Nos vamos a diseño y agregamos de la paleta dos “Menu Item”. Modificamos el archivo de tal forma que nos quede (en string.xml se incorporaron “Agregar” y “Eliminar”):

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/action_agrega"
        android:icon="@android:drawable/ic_menu_add"
        android:orderInCategory="0"
        android:title="@string/st_menu_agregar"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/action_elimina"
        android:icon="@android:drawable/ic_menu_delete"
        android:orderInCategory="0"
        android:title="@string/st_menu_eliminar"
        app:showAsAction="ifRoom" />

</menu>
```

d. Abrimos PrincActivity.java. En la clase presionamos <Ctl><O> y buscamos onCreateOptionsMenu y lo generamos. Reescribimos el código por la creación de nuestro menú.

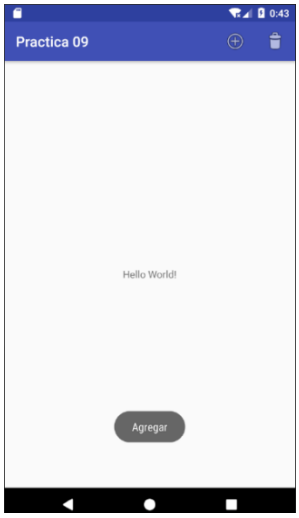
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.mimenu, menu);
    return true;
}
```

e. Realizamos una operación similar a la anterior con onOptionsItemSelected. Aquí definimos la acción a seguir mostrada con un objeto tipo Toast.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_agrega:
            Toast.makeText(this, "Agregar", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.action_elimina:
            Toast.makeText(this, "Eliminar", Toast.LENGTH_SHORT).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
```

```
}
```

f. Corremos nuestro programa y obtenemos el siguiente resultado al dar clic sobre .



4. Crear un menú emergente (PopUp). Para crear un menú emergente sobre una lista tipo RecyclerView creamos la lista de apoyo y continuamos en el paso 5.

a. En el área de diseño borramos el texto “Hello world!” que viene por default.

b. Agregamos la librería RecyclerView dando un clic en el símbolo de descarga RecyclerView y confirmamos la inclusión de la librería “com.android.support:recyclerview-v7”.

c. Ahora agregamos un elemento RecyclerView en lugar de “Hello Wold!” con un id = “rvlista”

d. Requerimos como queremos mostrar cada elemento de la lista “rvlista” y creamos un layout llamado lista_numeros.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/tv_numero"
```

```
        style="@style/TextAppearance.AppCompat.Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|start"
        app:fontFamily="monospace"
        android:textSize="42sp"
        tools:text="#15" />

    <TextView
        android:id="@+id/tv_holder"
        style="@style/TextAppearance.AppCompat.Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:fontFamily="monospace"
        android:padding="5dp"
        android:textColor="#000"
        app:layout_constraintStart_toEndOf="@+id/tv_numero"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_constraintBaseline_toBaselineOf="@+id/tv_numero"
        tools:text="Instancia ViewHolder: 3" />
</android.support.constraint.ConstraintLayout>
```

e. Asociamos en una nueva clase java “AdaptadorLista” la lista de números.

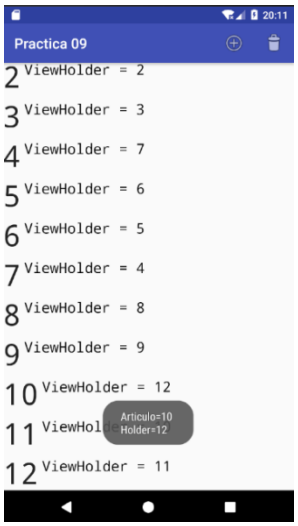
```
public class AdaptadorLista extends
RecyclerView.Adapter<AdaptadorLista.ViewHolderNumero> {

    private static int CuentaVHolder;
    private int num_articulos;

    final private ArticuloClic articuloClic;

    public interface ArticuloClic {
        void onClic(int numArticulo, int numHolder);
    }

    public AdaptadorLista(int num_articulos, ArticuloClic articuloClic) {
        this.num_articulos = num_articulos;
        CuentaVHolder = 0;
        this.articuloClic = articuloClic;
    }

    @NonNull
    @Override
    public AdaptadorLista.ViewHolderNumero onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        View view = inflater.inflate(R.layout.lista_numeros, parent,
false);
        ViewHolderNumero viewHolderNumero = new ViewHolderNumero(view,
CuentaVHolder++);
        return viewHolderNumero;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolderNumero holder, int
position) {
        holder.enlaza(position);
    }

    @Override
    public int getItemCount() {
        return num_articulos;
    }
```

```
    class ViewHolderNumero extends RecyclerView.ViewHolder implements
View.OnClickListener {
        int numHolder;
        TextView tvNumero;
        TextView tvHolder;

        public ViewHolderNumero(View itemView, int i) {
            super(itemView);
            this.tvHolder = itemView.findViewById(R.id.tv_holder);
            this.tvNumero = itemView.findViewById(R.id.tv_numero);

            numHolder = i;
            tvHolder.setText("ViewHolder = " + numHolder);
            itemView.setOnClickListener(this);
        }

        public void enlaza(int position) {
            tvNumero.setText(String.valueOf(position));
        }

        @Override
        public void onClick(View v) {
            articuloClic.onClic(getAdapterPosition(), numHolder);
        }
    }
}
```

f. En la clase PricActivity descendemos de la interface personalizada de la clase AdaptadorLista.ArticuloClic e implementamos onClick.

```
public class PrincActivity extends AppCompatActivity implements
AdaptadorLista.ArticuloClic {
    private static final int NUM_ARTICULOS = 50;
    AdaptadorLista adaptadorLista;
    RecyclerView listaNumeros;
    private Toast mToast;

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

        listaNumeros = findViewById(R.id.rvlista);
        LinearLayoutManager layoutM = new LinearLayoutManager(this);
        listaNumeros.setLayoutManager(layoutM);
        listaNumeros.setHasFixedSize(true);

        adaptadorLista = new AdaptadorLista(NUM_ARTICULOS, this);
        listaNumeros.setAdapter(adaptadorLista);
    }

…


@Override
    public void onClic(int numArticulo, int numHolder) {
        if (mToast != null) {
            mToast.cancel();
        }
        String msg = "Articulo=" + numArticulo + "\nHolder=" + numHolder;
        mToast = Toast.makeText(this, msg, Toast.LENGTH_LONG);
        mToast.show();
    }
}
```

g. Corremos el programa y nos muestra:



5. Ahora continuamos con el menú emergente propiamente.
   a. Creamos en res.menu un menú llamado menucontexto.xml con los siguientes valores.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/menu1"
        android:title="Editar" />
    <item
        android:id="@+id/menu2"
        android:title="Borrar" />
</menu>
```

   b. Modificamos el método clic de la clase interna ViewHolderNumero para crear un menú PopUp.

```
@Override
public void onClick(final View v) {
    final int numArticulo = getAdapterPosition();
    PopupMenu popup = new PopupMenu(v.getContext(), v);
    popup.inflate(R.menu.menucontexto);
    popup.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {
            String msg;
            switch (item.getItemId()) {
                case R.id.menu1:
                    msg = "Editar=" + numArticulo + "\nHolder=" +
numHolder;
                    Toast.makeText(v.getContext(), msg,
```

```
Toast.LENGTH_LONG).show();
                        break;
                    case R.id.menu2:
                        msg = "Borrar=" + numArticulo + "\nHolder=" +
numHolder;
                        Toast.makeText(v.getContext(), msg,
Toast.LENGTH_LONG).show();
                        break;
                }
                return false;
            }
        });
        popup.show();
    }
```

c. Corremos el programa y vemos el menú pop-up.

6. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Azzola, Francesco.** *Android UI Design.* s.l. : Exelixis Media Ltd., 2014.
2. **Phillips, Bill y Hardy, Brian.** *Android Programming: The Big Nerd Ranch Guide.* Indianapolis, IN 46240 USA : Pearson Technology Group, 2013.
3. **google.** Menús. *Documentación.* [En línea] https://developer.android.com/guide/topics/ui/menus.

No. 10. Diálogos.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para crear y usar diálogos.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.

Introducción

Los cuadros de diálogo exigen atención y aportes del usuario. Son útiles para presentar una elección o información importante. (1)

Con solo un par de líneas de trabajo, puede presentar un diálogo al usuario, y también se puede personalizar ese diálogo para que se ajuste mejor a su aplicación. (2)

Un diálogo no ocupa toda la pantalla y generalmente se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar. (3)

Los diálogos deliberadamente interrumpen el curso del programa, por lo que deben usarse con moderación. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre “Cuadros de diálogo”.
4. Obtener ejemplos básicos en Android usando “Diálogos”.
5. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de “Cuadros de diálogo” en Android.
2. Seleccionar a varios participantes para exponer ejemplos de los tipos comunes de “Diálogos” en Android.
3. Seleccionar a voluntarios para explicar “Diálogos”.
4. Llenar el recurso de Wiki con las definiciones de “Alert dialog”, “Simple dialog”, “Confirmation dialog” y “Full-screen dialog”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de incorporar sus propio diálogos.
3. Participar en Moodle con el glosario de términos incorporando: “Alert dialog”, “Simple dialog”, “Confirmation dialog” y “Full-screen dialog”

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal mostrando el tipo de diálogo simple indicando una acción al tocar cada opción de diálogo”.

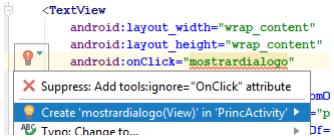
    El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde crea una pantalla principal introduciendo dos de los tipos de diálogos y al activar una acción crear un mensaje tipo Toast con un texto indicativo”.

2. Crear un proyecto en Android Studio.

    a. En Android Studio creamos un proyecto “Practica10”. Usamos “Phone and Tablet” y SDK como API 15 con “Empty Activity” y la clase principal “PrincActivity”.

3. Creamos una liga para mostrar un diálogo simple.

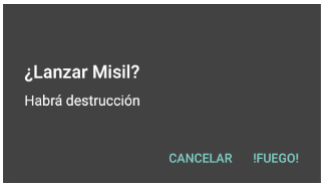
    a. Abrimos en diseño activity_princ.xml y seleccionamos el TextView “Hello World!” cambiamos la propiedad text a “Diálogo Simple” y la propiedad “onClick” a “mostrardialogo” y en modo texto generamos el método.



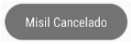
b. Modificamos la clase principal para crear el diálogo simple.

```
package mx.ith.practica10;

import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

public class PrincActivity extends AppCompatActivity {

    Toast toast;
    AlertDialog.Builder dialogo;

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

        dialogo = new AlertDialog.Builder(this,
android.R.style.Theme_Material_Dialog_Alert);
        dialogo.setTitle("¿Lanzar Misil?").setCancelable(false);
        dialogo.setMessage("Habrá
destrucción").setPositiveButton("!Fuego!", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                toast = Toast.makeText(PrincActivity.this, "Enviando
Misil", Toast.LENGTH_SHORT);
                toast.show();
            }
        }).setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                toast = Toast.makeText(PrincActivity.this, "Misil
Cancelado", Toast.LENGTH_SHORT);
                toast.show();
            }
        });
    }

    public void mostrardialogo(View view) {
        if (toast != null) {
            toast.cancel();
        }
        dialogo.show();
    }

}
```

c. Corremos nuestro programa. Damos un clic en el texto “Diálogo Simple”.



d. Y al dar clic en “Cancelar”. Aparece un mensaje Toast indicando la acción.



e. Y al dar clic en “!Fuego!”. Aparece un mensaje Toast indicando la acción.

4. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Phillips, Bill y Hardy, Brian.** *Android Programming: The Big Nerd Ranch Guide.* Indianapolis, IN 46240 USA : Pearson Technology Group, 2013.
2. **Clifton, Ian G.** *Android™ User Interface Design.* New Jersey : Addison-Wesley, 2013.
3. **google.** Developers. *Documentación.* [En línea] https://developer.android.com/guide/topics/ui/dialogs.
4. **Google.** Material Design. *Dialogs.* [En línea] https://material.io/design/components/dialogs.html.

No. 11. Archivos.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, permitiendo que el usuario cree una aplicación Android para almacenar texto en archivos.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.

Introducción

Guardar y cargar archivos en el almacenamiento interno son privados para la aplicación y otras aplicaciones no tendrán acceso a estos archivos. Cuando el usuario desinstala las aplicaciones, los archivos internos almacenados asociados con la aplicación también se eliminan. (1)

El almacenamiento interno se refiere a la porción de su aplicación del almacenamiento en memoria flash, siempre disponible. El almacenamiento externo se refiere al espacio de almacenamiento que puede ser agregado por el usuario. (2)

Es posible que prefiera utilizar el sistema de archivos tradicional para almacenar sus datos. Por ejemplo, es posible que desee almacenar el texto de los poemas que desea mostrar en sus aplicaciones. (3)

Android SDK reconoce un patrón y ha abstraído el controlador y los detalles de hilos en una clase de utilidad denominada AsyncTask. Puede usar AsyncTask para ejecutar tareas que tarden más de cinco segundos en el contexto de la interfaz de usuario. (4)

Aunque la aplicación de ejemplo no tarda más de 5 segundos. Es un buen pretexto para correrla con el uso de hilos.

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre “Almacenar datos en archivos”.
4. Obtener ejemplos básicos en Android usando “Files”.
5. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de “Archivos” en Android.
2. Seleccionar a varios participantes para exponer ejemplos de los tipos comunes de almacenamiento “Interno”, “Externo”, “Standard”, “Cache”, en Android.
3. Seleccionar a voluntarios para explicar “Almacenaje de datos con Archivos”.
4. Llenar el recurso de Wiki con las definiciones de “Internal Storage”, “External Storage”, “Permisos” y “Cache”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de almacenaje de datos en un archivo.
3. Participar en Moodle con el glosario de términos incorporando: “Almacenaje”, “Archivos”, “Almacenaje Interno” y “Almacenaje externo”

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal mostrando información guardada previamente y con opciones de modificar y recuperar de nuevo”.

    El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde crea una pantalla principal introduciendo un texto simple con opciones para guardar y leer el texto”.

2. Crear un proyecto en Android Studio.

    a. En Android Studio creamos un proyecto “Practica11”. Usamos “Phone and Tablet” y SDK como API 15 con “Empty Activity” y la clase principal “PrincActivity”.

3. Modificamos Activity_Princ.xml.

    a. Agregamos: Un texto, un área de texto (multilínea) y dos botones para leer y guardar.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PrincActivity">

    <TextView
        android:id="@+id/tvTexto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Escribir un poema"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etTexto"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:gravity="top|left"
        android:inputType="textMultiLine"
        android:scrollbars="vertical"
        app:layout_constraintBottom_toTopOf="@+id/btnLee"
        app:layout_constraintTop_toBottomOf="@+id/tvTexto" />

    <Button
        android:id="@+id/btnLee"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:onClick="onClickLee"
        android:text="Leer"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/btnGuarda" />

    <Button
        android:id="@+id/btnGuarda"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClickGuarda"
        android:text="Guardar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</android.support.constraint.ConstraintLayout>
```

b. Generamos los métodos onClickLee y onClickGuarda.

```
package mx.ith.practica11;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class PrincActivity extends AppCompatActivity {

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

    public void onClickLee(View view) {
    }
```

```
    public void onClickGuarda(View view) {
    }
}
```

4. Creamos varios atributos y generamos dos clases para Leer y Guardar.

    a. Una constante con el nombre del archivo a leer y guardar.

    b. Una referencia Toast y otras dos para los botones de leer y guardar y una más para el texto escrito por el usuario.

    c. Creamos dos clases llamadas Leer y Salvar que extiendan de AsyncTask. La clase Leer regresa un String. La clase Salvar recibe un String y nos regresa un Boolean.

    d. La clase nos queda como sigue:

```
public class PrincActivity extends AppCompatActivity {

    private static final String ARCHIVO = "archivo.txt";

    Toast toast;

    Button btnLee;
    Button btnGuarda;
    EditText etTexto;

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

        btnLee = findViewById(R.id.btnLee);
        btnGuarda = findViewById(R.id.btnGuarda);
        etTexto = findViewById(R.id.etTexto);
    }

    public void onClickLee(View view) {
    }

    public void onClickGuarda(View view) {
    }

    private class Leer extends AsyncTask<Void, Void, String> {
        @Override
        protected String doInBackground(Void... voids) {
            return null;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
        }
    }


    private class Guardar extends AsyncTask<String, Void, Boolean> {
        @Override
        protected Boolean doInBackground(String... strings) {
```

```
            return null;
        }

        @Override
        protected void onPostExecute(Boolean b) {
            super.onPostExecute(b);
        }
    }

}
```

5. Completamos la clase Leer con su métodos.

    a. Básicamente el método doInBackground hace la lectura del archivo y nos regresa en un String el contenido del archivo. Si el proceso es fallido obtenemos null.

    b. El método onPostExecute muestra el String obtenido en el paso anterior. Y si el String no es null muestra el texto del valor obtenido desplegando un mensaje y habilita los botones.

    c. La clase Leer queda como sigue.

```
private class Leer extends AsyncTask<Void, Void, String> {
    @Override
    protected String doInBackground(Void... voids) {
        String sret = null;
        String s;
        try {
            InputStream is = openFileInput(ARCHIVO);

            if (is != null) {
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader bufferedReader = new BufferedReader(isr);

                StringBuilder stringBuilder = new StringBuilder();

                while ((s = bufferedReader.readLine()) != null) {
                    stringBuilder.append(s).append("\n");
                }

                is.close();
                sret = stringBuilder.toString();
            }
        } catch (FileNotFoundException e) {
            Log.e("Leer", "File not found: " + e.toString());
        } catch (IOException e) {
            Log.e("Leer", "Can not read file: " + e.toString());
        }

        return sret;
    }

    @Override
    protected void onPostExecute(String s) {
        String msg;
        if (toast != null) {
            toast.cancel();
        }
```

```
        if (s != null) {
            etTexto.setText(s);
            msg = "Datos leídos";
        } else {
            msg = "Error al leer";
        }
        toast = Toast.makeText(PrincActivity.this, msg,
Toast.LENGTH_SHORT);
        toast.show();

        btnGuarda.setEnabled(true);
        btnLee.setEnabled(true);
    }
}
```

6. Completamos la clase Guardar.

   a. El método doInBackground hace la escritura del archivo le proporcionamos en un String el contenido del archivo y nos informa en una variable Boolean si el proceso fue exitoso.

   b. El método onPostExecute muestra un mensaje de acuerdo a la variable Boolean obtenido en el paso anterior y habilita los botones.

   c. La clase nos queda como sigue.

```
private class Guardar extends AsyncTask<String, Void, Boolean> {

    @Override
    protected Boolean doInBackground(String... strings) {
        String s;
        s = strings[0];
        if (!(s == null || s.isEmpty())) {
            try {
                FileOutputStream fos = openFileOutput(ARCHIVO,
MODE_PRIVATE);
                OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(fos);
                outputStreamWriter.write(s);
                outputStreamWriter.close();
                return true;
            } catch (IOException e) {
                Log.e("Exception", "Error al guardar: " + e.toString());
            }
        }
        return false;
    }

    @Override
    protected void onPostExecute(Boolean b) {
        String msg;
        if (toast != null) {
            toast.cancel();
        }
        msg = b ? "Datos guardados" : "Error al guardar";
        toast = Toast.makeText(PrincActivity.this, msg,
Toast.LENGTH_SHORT);
        toast.show();

        btnGuarda.setEnabled(true);
```

```
            btnLee.setEnabled(true);
        }
    }
```

7. El siguiente paso es generar las instancias de las clases Leer y Guardar. También deshabilitar los botones de leer y guardar. Y lo más relevante es arrancar los hilos de las clases Leer y Guardar. Para lograrlo modificamos los métodos onClickLee y onClickGuarda.

```
public void onClickLee(View view) {
    btnGuarda.setEnabled(false);
    btnLee.setEnabled(false);

    Leer leer = new Leer();
    leer.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
}

public void onClickGuarda(View view) {
    btnGuarda.setEnabled(false);
    btnLee.setEnabled(false);

    String s = etTexto.getText().toString();
    Guardar guardar = new Guardar();
    guardar.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, s);
}
```

8. Corremos nuestro programa y necesitamos observar varias consideraciones.

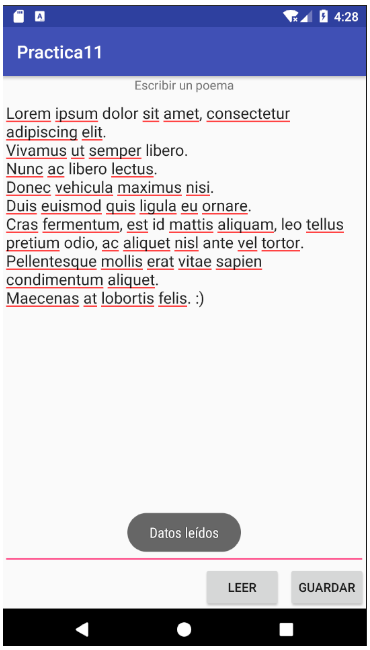
    a. La primera vez estará vacío el área de texto y no existirá el archivo. Por ello si damos clic en el botón de leer nos mostrará un mensaje de error “Error al leer”.

    b. Si el área de texto está vacía o con texto y damos clic en el botón guardar se creará el archivo de texto.

    c. Si el archivo contiene información y damos clic en el botón leer. La información será mostrada en el área de texto.

    d. Una vez cerrada nuestra aplicación y vuelta a abrir se mostrará vacío el texto mostrado. Si deseamos recuperar el texto lo podemos lograr con la opción de dar clic al botón de leer.

e. Una corrida será como sigue.



9. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Chugh, Anupam.** Android Internal Storage Example Tutorial. *JournalDev.* [En línea] 2 de April de 2018. https://www.journaldev.com/9383/android-internal-storage-example-tutorial.
2. **Murphy, Mark L.** *The Busy Coder's Guide to Android Development.* United States of America : CommonsWare, LLC., © 2008-2017.
3. **Lee, Wei Meng.** *Beginning Android™ 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
4. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* 2015.

No. 12. Almacenamiento en base de datos.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, Mostrar la interacción con el usuario para almacenar información en una base de datos.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 8. Animaciones.** Objetivo. Crear una un hilo para mostrar la concurrencia. **No. 9. Menús.** Objetivo. Crear una un hilo para mostrar la concurrencia y sincronización. **No. 10. Diálogos.** Objetivo. Crear varios hilos para mostrar interacción. **No. 11. Almacenamiento en archivos.** Objetivo. Crear una aplicación con menús de librería y gráficos. **No. 12. Almacenamiento en base de datos.** Objetivo. Crear una aplicación con menús de librería y gráficos y tener características de persistencia.

Introducción

Hay varias formas de salvar el estado de una aplicación con el SDK de Android. Algunas de formas son: Preferencias compartidas, archivos internos, archivos externos, SQLite, proveedores de contenido, herramientas de mapeo O/R y almacenamiento de red en la nube. (1)

Android usa el sistema de base de datos SQLite. El uso de bases de datos le permite hacer cumplir la integridad de los datos al especificar las relaciones entre diferentes conjuntos de datos. La base de datos creada para una aplicación es solo de acceso local; otras aplicaciones no tienen acceso. (2)

Room proporciona una capa de abstracción sobre SQLite para permitir el acceso fluido a la base de datos y, al mismo tiempo, aprovechar toda la potencia de SQLite. (3)

Ejemplo basado en “Android Room with a View”. (4)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los “SQLite”, “Room”.
4. Obtener ejemplos básicos en Android usando “SQLite” y/o “Room”.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de “SQLite” y “Room”.

2. Seleccionar a varios participantes para exponer ejemplos de las funciones comunes en “Base de Datos Room”.
3. Seleccionar a voluntarios para explicar “Base de datos”, “Entity”, “Dao”.
4. Llenar el recurso de Wiki con las definiciones de: “Room DataBase”, “PrimaryKey”, “ColumnInfo”, “Dao”, “Query”, “Repository”, “ViewModel”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una variante de la práctica inventada por el alumno. Contendrá “Room” y/o “SQLite”.
3. Participar en Moodle con el glosario de términos incorporando: “Repository”, “LiveData”, “ViewModel”, “Room database”.

1. El tipo de problema a resolver se propone lo siguiente:

    a. “Crear una aplicación con una ventana principal para realizar Altas, Bajas y Cambios de varios libros”.

    El algoritmo de solución aquí propuesto es: “Hacer una interface mostrando los libros actuales en una lista con un ícono dar de alta un nuevo libro y con un menú de contexto seleccionar Bajas y Cambios empleando Room”.

2. Crear un proyecto en Android Studio.

    a. En Android Studio creamos un proyecto “Practica12”. Usamos “Phone and Tablet” y SDK como API 15 con “**Basic Activity**” y la clase principal “PrincActivity”.

3. Actualizamos los archivos de gradle con las librerías Room.

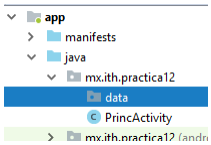
    a. Agregamos lo siguiente a “build.gradle (Module: app)” al final de “dependencies” (dentro de las llaves).

```
// Room components
implementation
"android.arch.persistence.room:runtime:$rootProject.roomVersion"
annotationProcessor
"android.arch.persistence.room:compiler:$rootProject.roomVersion"
androidTestImplementation
"android.arch.persistence.room:testing:$rootProject.roomVersion"

// Lifecycle components
implementation
"android.arch.lifecycle:extensions:$rootProject.archLifecycleVersion"
annotationProcessor
"android.arch.lifecycle:compiler:$rootProject.archLifecycleVersion"
```

b. En el archivo “build.gradle (Project: Practica12)”, agregamos el siguiente código al final.

```
ext {
    roomVersion = '1.0.0'
    archLifecycleVersion = '1.1.0'
}
```

4. Creamos la entidad.
   a. Creamos un paquete “data” para agrupar la lógica de la base de datos Room.



   b. Creamos la clase Libro dentro de data con dos atributos: título y autor y generamos los getters. Y un atributo más como un int id.

```
public class Libro {
    private int id;
    String titulo;
    String autor;

    public Libro(String titulo, String autor) {
        this.titulo = titulo;
        this.autor = autor;
    }

    public int getId() {
        return id;
    }

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }
}
```

c. La clase requiere ser manejada por Room y se requiere agregar anotaciones para relacionarla con la estructura de una Base de Datos.

```
@Entity(tableName = "libro")
public class Libro {

    @PrimaryKey(autoGenerate = true)
    private int id;

    @ColumnInfo(name = "titulo")
    @NonNull
    String titulo;

    @ColumnInfo(name = "autor")
    @NonNull
    String autor;

    @Ignore
    public Libro() {
    }
...
```

5. Ahora crearemos una interface tipo DAO. Aquí se insertan los querys para listar o modificar la tabla Libro.
   a. En data la creamos y le llamamos LibroDao.

```
package mx.ith.practica12.data;

public interface LibroDao {
}
```

   b. Para el manejo de Room agregamos anotaciones.

```
@Dao
public interface LibroDao {

    @Query("SELECT * FROM libros ORDER BY titulo")
    LiveData<List<Libro>> getAllbyTitulo();

    @Query("SELECT * FROM libros where id = :id")
    LiveData<Libro> findById(int id);

    @Query("SELECT COUNT(*) from libros")
    int getCount();

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertaVarios(Libro... libros);

    @Delete
    void elimina(Libro libro);

    @Query("DELETE FROM libros")
    void eliminaTodo();
}
```

6. El siguiente paso es crear la base de datos Room. Creamos una clase abstracta derivada de RoomDatabase. Y se usa con el patrón singleton.

```
package mx.ith.practica12.data;

import android.arch.persistence.room.Database;
import android.arch.persistence.room.Room;
import android.arch.persistence.room.RoomDatabase;
import android.content.Context;

@Database(entities = {Libro.class}, version = 1)
public abstract class LibroRoomDB extends RoomDatabase {
    public abstract LibroDao libroDao();

    private static LibroRoomDB INSTANCIA;

    static LibroRoomDB getDatabase(final Context context) {
        if (INSTANCIA == null) {
            synchronized (LibroRoomDB.class) {
                if (INSTANCIA == null) {
                    INSTANCIA =
Room.databaseBuilder(context.getApplicationContext(),
                            LibroRoomDB.class, "word_database").build();
                }
            }
        }
        return INSTANCIA;
    }
}
```

7. Agregamos un repositorio para manejar asincrónicamente la actualización de la tabla.

```
public class LibroRepository {
    private LibroDao mLibroDao;
    private LiveData<List<Libro>> mLibros;

    LibroRepository(Application application) {
        LibroRoomDB db = LibroRoomDB.getDatabase(application);
        mLibroDao = db.libroDao();
        mLibros = mLibroDao.getAllbyTitulo();
    }

    LiveData<List<Libro>> getLibros() {
        return mLibros;
    }

    public void insert (Libro libro) {
        new insertAsyncTask(mLibroDao).execute(libro);
    }

    private static class insertAsyncTask extends AsyncTask<Libro, Void, Void> {

        private LibroDao mAsyncLibroDao;

        insertAsyncTask(LibroDao dao) {
            mAsyncLibroDao = dao;
        }

        @Override
        protected Void doInBackground(Libro... libros) {
            mAsyncLibroDao.insertaVarios(libros);
```

```
            return null;
        }
    }
}
```

8. Ahora necesitamos una clase derivada de AndroidViewModel. Separa la interface gráfica de los procesos relativos a los datos.

```
package mx.ith.practica12.data;

import android.app.Application;
import android.arch.lifecycle.AndroidViewModel;
import android.arch.lifecycle.LiveData;

import java.util.List;

public class LibroViewModel extends AndroidViewModel {
    private LibroRepository mRepository;

    private LiveData<List<Libro>> mLibros;

    public LibroViewModel(Application application) {
        super(application);
        mRepository = new LibroRepository(application);
        mLibros = mRepository.getLibros();
    }

    LiveData<List<Libro>> getLibros() {
        return mLibros;
    }

    public void insert(Libro libro) {
        mRepository.insert(libro);
    }
}
```

9. Creamos los layouts XML para la lista y para cada renglón.

    a. Agregamos en layout/renglón_lista.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvTitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:fontFamily="sans-serif-medium"
        android:textAppearance="?android:textAppearanceMedium"
        android:textColor="#2B3D4D"  />

    <TextView
        android:id="@+id/tvAutor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:fontFamily="sans-serif"
        android:textAppearance="?android:textAppearanceSmall"
```

```
            android:textColor="#AEB6BD" />

    </LinearLayout>
```

b. En layout/content_main.xml, reemplazamos el TextView con un RecyclerView:

```
    <android.support.v7.widget.RecyclerView
        android:id="@+id/rvLista"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/darker_gray"
        tools:listitem="@layout/renglon_lista" />
```

c. En el archive layout/activity_princ.xml en FloatingActionButton Cambiamos la línea:

```
        app:srcCompat="@android:drawable/ic_dialog_email"
```

por:

```
        app:srcCompat="@android:drawable/ic_input_add"
```

10. Creamos la clase adaptadora para el RecycleView (Lista). Donde mostraremos los dos valores título y autor para cada renglón.

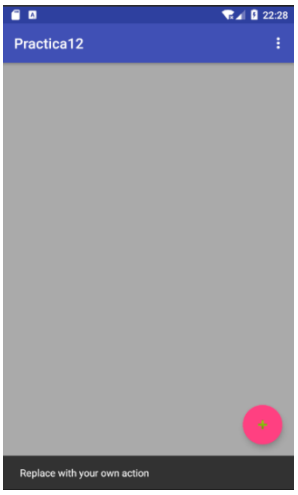
```
public class LibroAdaptador extends
RecyclerView.Adapter<LibroAdaptador.LibroViewHolder> {

    class LibroViewHolder extends RecyclerView.ViewHolder {
        private final TextView tvTitulo;
        private final TextView tvAutor;

        private LibroViewHolder(View itemView) {
            super(itemView);
            tvTitulo = itemView.findViewById(R.id.tvTitulo);
            tvAutor = itemView.findViewById(R.id.tvAutor);
        }
    }

    private final LayoutInflater mInflater;
    private List<Libro> mLibros; // Cached copy of words

    LibroAdaptador(Context context) {
        mInflater = LayoutInflater.from(context);
    }

    @Override
    public LibroViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
        View itemView = mInflater.inflate(R.layout.renglon_lista, parent,
false);
        return new LibroViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(LibroViewHolder holder, int position) {
        if (mLibros != null) {
            Libro current = mLibros.get(position);
            holder.tvAutor.setText(current.getAutor());
```

```
            holder.tvTitulo.setText(current.getTitulo());
        } else {
            holder.tvAutor.setText("Sin Libro");
            holder.tvTitulo.setText("Sin Libro");
        }
    }

    void setLibros(List<Libro> libros) {
        mLibros= libros;
        notifyDataSetChanged();
    }

    @Override
    public int getItemCount() {
        if (mLibros != null) return mLibros.size();
        else return 0;
    }
}
```

11. Dentro de PrincActivity.java agregamos en el método onCreate lo siguiente.

```
RecyclerView rvLista = findViewById(R.id.rvLista);
final LibroAdaptador adapter = new LibroAdaptador(this);
rvLista.setAdapter(adapter);
rvLista.setLayoutManager(new LinearLayoutManager(this));
```

12. Corremos nuestro programa. Veremos todavía no se muestran datos.



13. Lo siguiente es agregar información a la base de datos.
    a. Los datos no se recomienda agregarlos dentro del hilo de la UI los agregaremos por medio de AsyncTask.
    b. Abrimos el archivo LibroRoomDB.java y agregamos el atributo al abrir la base de datos y llame al llenado de datos con la clase interna siguiente.

```
private static RoomDatabase.Callback sLibroDatabaseCallback =
        new RoomDatabase.Callback(){

            @Override
            public void onOpen (@NonNull SupportSQLiteDatabase db){
```

```
                super.onOpen(db);
                new PopulateDbAsync(INSTANCIA).execute();
            }
        };

    private static class PopulateDbAsync extends AsyncTask<Void, Void, Void> {

        private final LibroDao mDao;

        PopulateDbAsync(LibroRoomDB db) {
            mDao = db.libroDao();
        }

        @Override
        protected Void doInBackground(final Void... params) {
            mDao.eliminaTodo();
            for( String[] si:infoLibros){
                mDao.insertaVarios(new Libro(si[0], si[1]));
            }
            return null;
        }
    }

    static String[][] infoLibros = {
            {"De (casi) todo se aprende","Paula Gonu"},
            {"El Príncipe de la Niebla","Carlos Ruiz Zafón"},
            {"Detective Conan II n° 89","Meitantei Conan II"},
            {"Leiva. Toquemos juntos hasta que la muerte nos joda","Wilma
Lorenzo"},
            {"Las hijas del Capitán","María Dueñas"},
            {"Donde fuimos invencibles","María Oruña"},
            {"Patria","Fernando Aramburu Irigoyen"},
            {"Morder la manzana","Leticia Dolera"},
            {"El legado de los espías","John le Carré"},
            {"Las almas de Brandon","César Brandon Ndjocu"},
            {"El fuego invisible","Javier Sierra"},
            {"Maestros de la costura","Shine | CR TVE"},
            {"A comer se aprende","Álvaro Vargas"}};
```

c. Modificamos la creación de la instancia a la base de datos.

```
    INSTANCIA = Room.databaseBuilder(context.getApplicationContext(),
            LibroRoomDB.class, "libro_db").
            addCallback(sLibroDatabaseCallback).
            build();
```

14. Ahora agregaremos la opción de insertar un nuevo libro.
    a. Agregamos una nueva actividad con “emptyActivity” llamada NuevoLibroActivity.
    b. En activity_nuevo_libro.xml reemplazamos por el siguiente texto.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <EditText
        android:id="@+id/etTitulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:layout_marginBottom="16dp"
            android:layout_marginTop="16dp"
            android:hint="Titulo"
            android:inputType="textAutoComplete"
            android:padding="6dp"
            android:textSize="18sp"
            app:fontFamily="sans-serif-light" />

        <EditText
            android:id="@+id/etAutor"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:layout_marginTop="16dp"
            android:hint="Autor"
            android:inputType="textAutoComplete"
            android:padding="6dp"
            android:textSize="18sp"
            app:fontFamily="sans-serif-light" />

        <Button
            android:id="@+id/button_save"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:onClick="onClicSave"
            android:text="Salvar"
            android:textColor="#d3d3d3" />

    </LinearLayout>
```

15. En el código de la clase NuevoLibroActivity.java lo modificamos para responder cuando se llenan los valores de título y autor.

```
public class NuevoLibroActivity extends AppCompatActivity {

    public static final String EXTRA_TITULO = "Titulo";
    public static final String EXTRA_AUTOR = "Autor";
    EditText etAutor;
    EditText etTitulo;

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

        etTitulo = findViewById(R.id.etTitulo);
        etAutor = findViewById(R.id.etAutor);
    }

    public void onClicSave(View view) {
        Intent replyIntent = new Intent();
        if (TextUtils.isEmpty(etTitulo.getText())) {
            setResult(RESULT_CANCELED, replyIntent);
        } else {
            String titulo = etTitulo.getText().toString();
            String autor = etAutor.getText().toString();
            replyIntent.putExtra(EXTRA_TITULO, titulo);
            replyIntent.putExtra(EXTRA_AUTOR, autor);
            setResult(RESULT_OK, replyIntent);
        }
        finish();
    }
}
```

16. Ahora conectaremos la lista con los datos de la base de datos.
    a. Creamos un atributo en la clase PincActivity.

```
private LibroViewModel mLibroViewModel;
```

    b. Obtenemos en OnCreate un ViewModel de ViewModelProvider.

```
mLibroViewModel =
ViewModelProviders.of(this).get(LibroViewModel.class);
```

    c. También en OnCreate agregamos un observador para LiveData generada por getLibros.

```
mLibroViewModel.getLibros().observe(this, new Observer<List<Libro>>()
{
    @Override
    public void onChanged(@Nullable final List<Libro> libros) {
        // Actualiza la copia en cache de libros
        adapter.setLibros(libros);
    }
});
```

    d. Creamos el método onActivityResult para recibir la información de un nuevo libro.

```
public static final int LIBRO_REQUEST_CODE = 1;

public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == LIBRO_REQUEST_CODE && resultCode == RESULT_OK) {
        Libro libro = new Libro(
                data.getStringExtra(NuevoLibroActivity.EXTRA_TITULO),
                data.getStringExtra(NuevoLibroActivity.EXTRA_AUTOR));
        mLibroViewModel.insert(libro);
    } else {
        Toast.makeText(getApplicationContext(), "Sin título no se salva",
Toast.LENGTH_LONG).show();
    }
}
```

    e. Reemplazamos en onClickListener del objeto fab por la invocación al llamado de la actividad para agregar un nuevo libro.

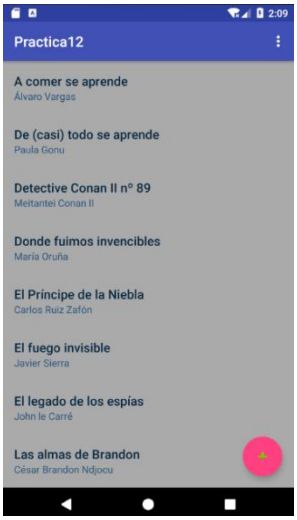
Reemplazamos:

```
Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG).setAction("Action", null).show();
```

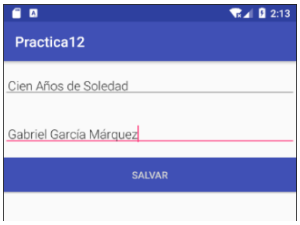
Por:

```
Intent intent = new Intent(PrincActivity.this, NuevoLibroActivity.class);
startActivityForResult(intent, LIBRO_REQUEST_CODE);
```

17. Podemos realizar una corrida.
    a. Vemos el listado de los libros agregados por default. Cada vez que se corre el programa se borran los valores previos y se vuelven a insertar.



    b. Si insertamos un nuevo elemento.



    c. Queda insertado en la segunda posición.

18. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* New York : Apress, 2015.
2. **Wei-Meng, Lee.** *Beginning Androi 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
3. **Google.** Save data in a local database using Room . *Developers Documentation.* [En línea] https://developer.android.com/training/data-storage/room/.
4. **Google.** Android Room with a View. *Developer codelabs.* [En línea] https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0.

No. 13. Geo localización.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, Mostrar la interacción con el usuario para mostrar información en un mapa.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 13. Geo localización.** Objetivo. Crear un mapa usando google maps. **No. 14. JSON.** Objetivo. Conocer las formas de almacenar información. **No. 15. XML.** Objetivo. Conocer las formas de almacenar información.

Introducción

Las aplicaciones basadas en la ubicación y auxiliadas por mapas pueden ofrecer servicios tales como la ubicación de servicios cercanos, así como ofrecer sugerencias para la planificación de rutas, y más. (1)

Las aplicaciones de geo localización y navegación son populares en las plataformas móviles y son auxiliadas por receptores GPS. Estos receptores toman señales de radio de satélites de posicionamiento global. (2)

El SDK de Android proporciona una API para que los desarrolladores de aplicaciones puedan visualizar y manipular mapas, obtener información de ubicación del dispositivo en tiempo real y aprovechar otras características interesantes. (3)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los “Location-Based Services”, “API Key”.
4. Obtener ejemplos básicos en Android usando “Mapas”.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de “Maps Api” y “Habilitar la Api”.
2. Seleccionar a varios participantes para exponer ejemplos de las funciones comunes en “Maps Api”.
3. Seleccionar a voluntarios para explicar “Tipos de Mapas”.
4. Llenar el recurso de Wiki con las funciones de “Road Maps”: “Satellite Maps”, “Hibrid Maps”, “Terrain Maps”

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá una variante de agregar marcadores para ubicarlos en un mapa.

Procedimiento:

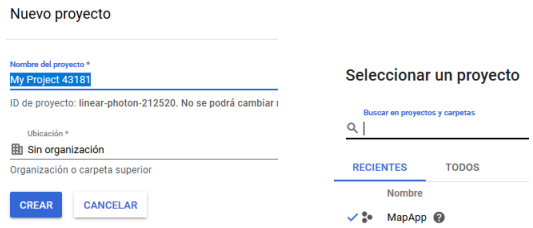
1. El tipo de problema a resolver se propone lo siguiente:

   “Crear una aplicación con una ventana principal para representar un marcador de posicionamiento global”.

   El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica con un mapa empleando Google Maps donde crea una pantalla principal. Almacenar coordenadas en una objeto y mostrarlas en el mapa”.

2. Antes de iniciar un proyecto Google pide tener o generar una cuenta de desarrollador y con ella obtener una llave (API key) para acceder a los recursos.

   a. Visitar https://console.developers.google.com
   b. Inicie sesión con su cuenta Google, algo así como, SuNombre@gmail.com. Si no tiene una, puede ir a accounts.google.com para crear una gratis.
   c. Seleccionamos un proyecto (si no tenemos creamos uno con una descripción adecuada).



   d. Dar clic en Crear y luego clic en "Google Maps JavaScript API" debajo de "Google Maps APIs" y después habilitar “Google Maps Android API”, “Google Places API for Android”. Si la intención es usar esta llave en aplicaciones WEB probablemente haya que seleccionar otros recursos.
   e. Dar clic en “Credenciales” y seleccionamos “Crear credenciales” y generamos una “Clave de API” y si deseamos la podemos restringir a “Apps de Android”.
   f. La llave nos queda algo así como:

      AIzaSyBGZr2ajky7teaP8gmorX_XxXxXxXxXxXx

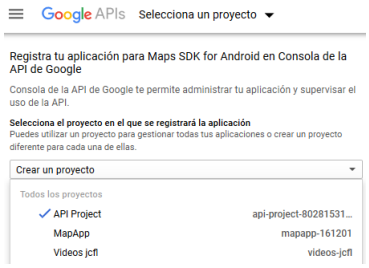
3. Crear un proyecto en Android Studio.

a. En Android Studio creamos un proyecto “Practica13”. Usamos “Phone and Tablet” y SDK como API 15 con “**Google Maps Activity**” y la clase principal “PrincActivity” y título “Practica13”.

b. Una vez creado el proyecto nos lleva directamente a un archivo llamado google_maps_api.xml donde nos pide la llave. Por facilidad ya está prefabricada una liga donde asocia nuestra aplicación con una huella digital (en hexadecimal) y el nombre del paquete.

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=1E:96:58:F3:35:32:0C:92:93:0C:F0:D6:D6:E4:3A:12:02:B4:1B:48%3Bmx.ith.practica13

c. Nos lleva directamente a registrar nuestro proyecto.



d. Donde podemos generar una nueva llave o usar la generada en el paso anterior.

e. Regresando al proyecto android en google_maps_api.xml:

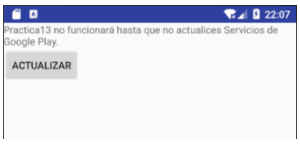
Reemplazamos:

```
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">YOUR_KEY_HERE</string>
```

Por nuestra llave generada:

```
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false"> AIzaSyBGZr2ajky7teaP8gmorX_XxXxXxXxXxXx
</string>
```

f. Si corremos nuestra aplicación en un emulador nos marca lo siguiente:



g. Sin embargo en mi configuración con el emulador no me permite actualizar “Google Play”. Verifico mi emulador no tiene acceso a Internet.

Para resolver éste problema cerramos el emulador y lo lanzamos manualmente. El error es que el emulador no lee el DNS. En una ventana de MS-DOS daremos los siguientes comandos.

```
C:\Users\hmo>cd AppData\Local\Android\Sdk\tools
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -list-avds
Galaxy_Nexus_API_22
Nexus_4_API_22
Nexus_5X_API_25
Nexus_One_API_26
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -avd
Nexus_5X_API_25 -dns-server 8.8.8.8
```

h. Corro la aplicación en el emulador "`Nexus_5X_API_25`". Y nos manda directo a Australia. (Nota: Aunque el programa corre existe un error en build.gradle).

i. Otro detalle en mi configuración es el siguiente error cuando se abre el archivo "build.gradle (Module:app)". (Es muy posible que en futuras versiones de Android Studio sea corregido). El error lo marca en la línea:

"`implementation 'com.android.support:appcompat-v7:27.1.1'`"

Con el mensaje de error:

```
All com.android.support libraries must use the exact same version
specification (mixing versions can lead to runtime crashes). Found
versions 27.1.1, 26.1.0. Examples include com.android.support:animated-
vector-drawable:27.1.1 and com.android.support:support-media-compat:26.1.0
```

Para corregirlo cambiamos compileSdkVersion 26, targetSdkVersion 26 y la línea también.

"`implementation 'com.android.support:appcompat-v7:26.1.0'`"

4. Si analizamos el código de PrincActivity.java.
   a. Vemos que el código nos marca la latitud y longitud (-34,151) que es donde se encuentra Sydney.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in
Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

   b. Si nos interesara una nueva ubicación usemos:

"https://google-developers.appspot.com/maps/documentation/utils/geocoder"

   c. Localizamos nuevas coordenadas como la ubicación del ITH (o su lugar preferido): "29.097333,-110.996959" y las reemplazamos en nuestro programa y lo volvemos a correr. Dando doble clic y desplazando la pantalla podemos llegar a ver algo similar a lo siguiente:

5. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **Wei-Meng, Lee.** *Beginning Androi 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
2. **Yamacli, Serhan.** *Beginner's Guide to Android App Development.* s.l. : Manchester Academic Publishers.
3. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* New York : Apress, 2015.

No. 14. JSON.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, Mostrar la interacción con el usuario para mostrar información a través de la red usando objetos JSON.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 13. Geo localización.** Objetivo. Crear un mapa usando google maps. **No. 14. JSON.** Objetivo. Conocer las formas de almacenar información. **No. 15. XML.** Objetivo. Conocer las formas de almacenar información.

Introducción

Un servicio web es un sistema de software diseñado para admitir la interacción interoperable de máquina a máquina a través de una red. Generalmente se transmiten utilizando HTTP con una serialización XML junto con otros estándares relacionados con la Web. (1)

Las aplicaciones móviles ofrecen una funcionalidad muy rica en un dispositivo tan pequeño es que extraen información de varias fuentes. Una estrategia de integración común es usar HTTP a través de servicios web en Internet. (2)

JSON (JavaScript Object Notation) es un formato liviano de intercambio de datos. Es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones que son familiares para los programadores de la familia C de idiomas, incluida C , C ++, C #, Java, JavaScript, Perl, Python y muchos otros. (3)

La manipulación de documentos XML es computacionalmente operación costosa para dispositivos móviles. Los documentos XML son grandes y significa que su dispositivo tiene que usar más ancho de banda para descargarlo y son más difíciles de procesar usan más memoria y CPU a diferencia del formato JSON. (4)

JSON es un formato de intercambio de datos independiente y es la mejor alternativa para XML. Android proporciona cuatro clases diferentes para manipular datos JSON. Estas clases son JSONArray, JSONObject, JSONStringer y JSONTokenizer. (5)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los “Servicios Web”, “API Key”, “formato JSON”.
4. Obtener ejemplos básicos en Android usando “JSON”.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de “You Tube Api” y “Habilitar la Api”.
2. Seleccionar a varios participantes para exponer ejemplos de conversión de “JSON a Java”.
3. Seleccionar a varios voluntarios para obtener ejemplos de páginas usando JSON.

4. Seleccionar a voluntarios para explicar otros tipos de protocolos como “Soap”.
5. Llenar el recurso de Wiki con las funciones de “mensajes Soap”, “JSONArray”, “JSONObject”, “JSONStringer y JSONTokenizer”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá su propia página de acceso JSON.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

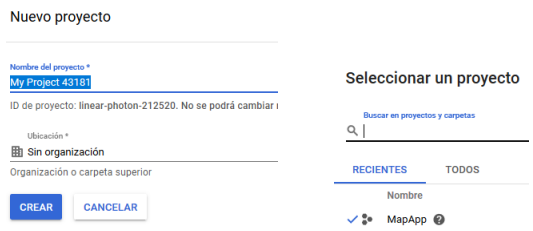
   > “Crear una aplicación con una ventana principal para representar la información de videos de YouTube™ a una solicitud del usuario”.

   El algoritmo de solución aquí propuesto es: “Hacer una interface gráfica donde se el usuario escribe parte del nombre del video. Posteriormente consultar YouTube™ a través de la Api y generar una lista de resultados”.

   Nota: Aunque Android tiene una api para manejar directamente los objetos en java de YouTube ésta práctica lo hace al nivel de Objetos JSON.

2. Antes de iniciar un proyecto Google pide tener o generar una cuenta de desarrollador y con ella obtener una llave (API key) para acceder a los recursos.

   a. Visitar https://console.developers.google.com
   b. Inicie sesión con su cuenta Google, algo así como, SuNombre@gmail.com. Si no tiene una, puede ir a accounts.google.com para crear una gratis.
   c. Seleccionamos un proyecto (si no tenemos creamos uno con una descripción adecuada).



d. Dar clic en Crear y luego clic en "Google Maps JavaScript API" debajo de "Google Maps APIs" y después habilitar “Google Maps Android API”, “Google Places API for Android”. Si la intención es usar esta llave en aplicaciones WEB probablemente haya que seleccionar otros recursos.
e. Dar clic en “Credenciales” y seleccionamos “Crear credenciales” y generamos una “Clave de API” y si deseamos la podemos restringir a “Apps de Android”.
f. La llave nos queda algo así como:

   AIzaSyBGZr2ajky7teaP8gmorX_XxXxXxXxXxXx

3. Crear un proyecto para Android Studio a partir de otro proyecto.

   a. Con el explorador de archivos localizamos la “Practica06” y hacemos una copia a “Practica14”.

4. Abrimos el emulador y verificamos el acceso a Internet.
   a. Abrimos un Navegador y buscamos una página.
   b. Si el emulador no tiene acceso a Internet probaremos lo siguiente. Cerramos el emulador y lo lanzamos manualmente. Probablemente el error sea que el emulador no lee el DNS. En una ventana de MS-DOS daremos los siguientes comandos.

```
C:\Users\hmo>cd AppData\Local\Android\Sdk\tools
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -list-avds
Galaxy Nexus API 22
Nexus_4_API_22
Nexus 5X API 25
Nexus_One_API_26
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -avd
Nexus_5X_API_25 -dns-server 8.8.8.8
```

5. Abrir el proyecto Practica14. Al correr el proyecto vemos varios detalles.

   a. El nombre proyecto indica Practica06. Abrimos res.values.strings.xml y corregimos a “Practica14”.

   b. El paquete principal de java indica “mx.ith.practica06”. Lo seleccionamos y presionamos <Shift><F6>. Renombramos el paquete a “mx.ith.practica14” y damos clic en “Do Refactor”.

6. Crearemos dos clases.
   a. Una clase llamada MiVideo. Donde almacenaremos el título del video, un url a una imagen y una identificación del video.

```
public class MiVideo {
    String titulo;
    String id;
```

```
    String imgurl;

    public MiVideo(String titulo, String id, String imgurl) {
        this.titulo = titulo;
        this.id = id;
        this.imgurl = imgurl;
    }

    public String getTitulo() {
        return titulo;
    }

    public String getId() {
        return id;
    }

    public String getImgUrl() {
        return imgurl;
    }

    @Override
    public String toString() {
        return "MiVideo{" + "titulo='" + titulo + '\'' + '}';
    }
}
```

b. Otra clase para generar el URL para consultar los servicios de YouTube llamada “GetYTAsyncTask”. Posteriormente implementaremos AsyncTask por lo tanto solo convertiremos a partir de una consulta un URL. En ésta clase incorporamos la llave generada en el paso 2 en “*YT_KEY*="Llave””. La liga de documentación de la api de YouTube se encuentra en él código en “//ver: https://...” .

```
package mx.ith.practica14.red;

import android.net.Uri;
import android.util.Log;

import java.net.MalformedURLException;
import java.net.URL;

public class GetYTAsyncTask {
    private static final String TAG =
GetYTAsyncTask.class.getSimpleName();

    private static final String
YT_KEY="AIzaSyBGZr2ajky7teaP8gmorX_XxXxXxXxXxXx";

    //ver: https://developers.google.com/youtube/v3/docs/search
    private static final String BASE_YT_URL =
"https://www.googleapis.com/youtube/v3/search";

    private static final String YT_PART = "snippet";
    private static final String YT_MAX_RESULTS = "10";
    private static final String YT_FIELDS =
"items(id(videoId),snippet(title,thumbnails(medium)))";
    private static final String YT_TYPE = "video";

    private static final String PARAM_PART = "part";
    private static final String PARAM_KEY = "key";
    private static final String PARAM_Q = "q";
    private static final String PARAM_MAX_RESULTS = "maxResults";
    private static final String PARAM_FIELDS = "fields";
    private static final String PARAM_TYPE = "video";
```

```
    public static URL creaUrlconQuery(String query){
        Uri youtTubeQueryUri = Uri.parse(BASE_YT_URL).buildUpon()
                .appendQueryParameter(PARAM_PART, YT_PART)
                .appendQueryParameter(PARAM_KEY, YT_KEY)
                .appendQueryParameter(PARAM_Q, query)
                .appendQueryParameter(PARAM_MAX_RESULTS,
YT_MAX_RESULTS)
                .appendQueryParameter(PARAM_FIELDS, YT_FIELDS)
                .appendQueryParameter(PARAM_TYPE, YT_TYPE)
                .build();

        try {
            URL youtTubeUrl = new URL(youtTubeQueryUri.toString());
            Log.v(TAG, "URL: " + youtTubeUrl);
            return youtTubeUrl;
        } catch (MalformedURLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

c. Dar el permiso de Internet en el archivo “AndroidManifest.xml”. Antes del tag “<application” insertamos el siguiente permiso.

```
<uses-permission android:name="android.permission.INTERNET" />
```

También vamos a incorporar la librería “picaso” la cual nos va a permitir leer de un URL una imagen. Abrimos el archivo “build.gradle (Module:app)” y la parte de dependencias nos queda (le indicamos que lo sincronize):

a. Es posible que nos marque un error “…Found version 27.1.1 y 27.1.0…”
b. Cambiamos donde diga 27.1.1 por 27.1.0 y sincronizamos de nuevo.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.android.support:recyclerview-v7:27.1.0'

    implementation 'com.squareup.picasso:picasso:2.71828'
}
```

7. Para darle la funcionalidad de “Hilo” a la clase “GetYTAsyncTask” vamos a implementar de AsyncTask generando el método doInBackGround. Y un método que nos proporcione en un String el objeto JSON sin procesar y otro método que lo procese y nos regrese una lista de MiVideo.

```
public class GetYTAsyncTask extends AsyncTask<String, Void, List<MiVideo>>

    …

@Override
protected List<MiVideo> doInBackground(String... strings) {
    String query;
```

```
        query = strings.length > 0 ? strings[0] : "";
        String stringJson = null;
        List<MiVideo> misVideos = null;

        try {
            URL url = creaUrlconQuery(query);
            stringJson = getJsonString(url);
            misVideos = getListaVideos(stringJson);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return misVideos;
    }

    public static String getJsonString(URL url) throws IOException {
        HttpURLConnection urlConnection = (HttpURLConnection)
    url.openConnection();
        try {
            InputStream in = urlConnection.getInputStream();

            Scanner scanner = new Scanner(in);
            scanner.useDelimiter("\\A");

            boolean hasInput = scanner.hasNext();
            String response = null;
            if (hasInput) {
                response = scanner.next();
            }
            scanner.close();
            return response;
        } finally {
            urlConnection.disconnect();
        }
    }

    private static final String ITEMS = "items";
    private static final String ID = "id";
    private static final String VIDEOID = "videoId";
    private static final String TITLE = "title";
    private static final String SNIPPET = "snippet";
    private static final String THUMBNAILS = "thumbnails";
    private static final String MEDIUM = "medium";
    private static final String URL = "url";

    public static List<MiVideo> getListaVideos(String stringJson) {
        String sid = null;
        String stitle = null;
        String surl = null;

        Log.v("Jason String", stringJson);

        List<MiVideo> misVideos = new ArrayList<>();
        JSONObject videosJson = null;
        try {
            videosJson = new JSONObject(stringJson);

            JSONArray arrayItems = videosJson.getJSONArray(ITEMS);

            for (int i = 0; i < arrayItems.length(); i++) {
                JSONObject item = arrayItems.getJSONObject(i);
                if (!item.has(ID)) {
                    continue;
                }
                JSONObject id = item.getJSONObject(ID);

                sid = id.getString(VIDEOID);
```

```
                JSONObject snippet = item.getJSONObject(SNIPPET);
                stitle = snippet.getString(TITLE);

                surl =
snippet.getJSONObject(THUMBNAILS).getJSONObject(MEDIUM).getString(URL);

                misVideos.add(new MiVideo(stitle, sid, surl));
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

        return misVideos;
    }
```

8. Ya no necesitamos la clase Libro. La borramos con “Delete Anyway”. Y Ocasiona varios errores. Los iremos arreglando paso a paso.

9. Abrimos renglón.xml. Y modificamos su contenido como sigue.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imVideo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="5dp"
        app:srcCompat="@drawable/youtube" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvID"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Id"
            android:textSize="15sp"
            tools:text="Libro" />

        <TextView
            android:id="@+id/tvTitulo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Titulo"
            android:textSize="12sp"
            tools:text="Autor" />
    </LinearLayout>
</LinearLayout>
```

10. Abrimos activity_princ.xml y cambiamos los nombres de las variable y agregamos un botón y un progressbar como sigue:

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

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PrincActivity">

    <Button
        android:id="@+id/btnBusca"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:onClick="ClickBusca"
        android:text="Buscar"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etBusca"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:inputType="text"
        android:text="Bose"
        app:layout_constraintEnd_toStartOf="@id/progressBar"
        app:layout_constraintStart_toEndOf="@id/btnBusca"
        app:layout_constraintTop_toTopOf="@id/btnBusca" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:visibility="invisible"
        app:layout_constraintEnd_toEndOf="parent" />

    <android.support.v7.widget.RecyclerView
        android:id="@+id/vista"
        android:layout_width="match_parent"
        android:layout_height="400dp"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnBusca" />

</android.support.constraint.ConstraintLayout>
```

11. En la clase “MiAdaptador.java” incorporamos MiVideo de la siguiente forma:

```
public class MiAdaptador extends
RecyclerView.Adapter<MiAdaptador.ViewHolder> {

    private List<MiVideo> listaVideos;
    static View.OnClickListener clickListener;
```

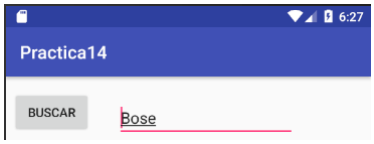
```
    MiAdaptador(List<MiVideo> listaVideos, View.OnClickListener
clickListener) {
        this.listaVideos = listaVideos;
        MiAdaptador.clickListener = clickListener;
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView tvTitulo, tvId;
        private final ImageView imVideo;

        ViewHolder(View view) {
            super(view);
            view.setOnClickListener(MiAdaptador.clickListener);
            this.tvTitulo = view.findViewById(R.id.tvTitulo);
            this.tvId = view.findViewById(R.id.tvID);
            this.imVideo = view.findViewById(R.id.imVideo);
        }
    }

    @NonNull
    @Override
    public MiAdaptador.ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.renglon,
parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MiAdaptador.ViewHolder
holder, int position) {
        MiVideo video = listaVideos.get(position);
        holder.tvTitulo.setText(video.titulo);
        holder.tvId.setText(video.id);
        Picasso.get().load(video.imgurl).into(holder.imVideo);
    }

    @Override
    public int getItemCount() {
        return listaVideos.size();
    }

    public void NewData(List<MiVideo> listaVideos) {
        this.listaVideos = listaVideos;
        notifyDataSetChanged();
    }
}
```

12. En la clase PrincActivity.java agregamos la siguiente clase derivada de GetYTAsyncTask. Nos mostrará un botón para buscar una canción o un cantante con video.

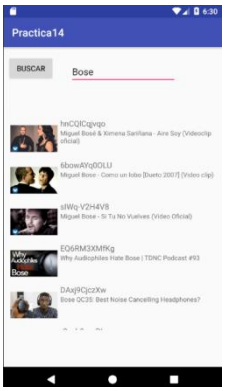
```
public class PrincActivity extends AppCompatActivity implements
View.OnClickListener {

    RecyclerView rvVista;
    MiAdaptador miAdaptador;

    EditText etBusca;
    ProgressBar progressBar;
    boolean buscando;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_princ);

        rvVista = findViewById(R.id.vista);
        rvVista.setHasFixedSize(true);

        LinearLayoutManager mLayoutManager = new LinearLayoutManager(this);
        rvVista.setLayoutManager(mLayoutManager);

        miAdaptador = new MiAdaptador(new ArrayList<MiVideo>(), this);
        rvVista.setAdapter(miAdaptador);

        etBusca = findViewById(R.id.etBusca);
        progressBar = findViewById(R.id.progressBar);
    }

    @Override
    public void onClick(View v) {
        String surl;
        TextView tvTitulo = v.findViewById(R.id.tvTitulo);
        TextView tvId = v.findViewById(R.id.tvID);
        Toast.makeText(this, tvTitulo.getText(), Toast.LENGTH_LONG).show();
        surl = "http://www.youtube.com/watch?v=" + tvId.getText();
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(surl)));
    }

    public void ClickBusca(View view) {
        EditText tvBusca = findViewById(R.id.etBusca);
        if (buscando) {
            Toast.makeText(this, "Buscando", Toast.LENGTH_LONG).show();
        } else {
            buscando = true;
            progressBar.setVisibility(View.VISIBLE);
            new ListaYTAsyncTask().execute(tvBusca.getText().toString());
        }
    }

    class ListaYTAsyncTask extends GetYTAsyncTask {

        @Override
        protected List<MiVideo> doInBackground(String... strings) {
            //Log.v("Princ", strings[0]);
            return super.doInBackground(strings);
        }

        @Override
        protected void onPostExecute(List<MiVideo> misVideos) {
            miAdaptador.NewData(misVideos);
            buscando = false;
            progressBar.setVisibility(View.INVISIBLE);
        }
    }
}
```

13. Al correr el programa muestra una interface gráfica para hacer una búsqueda en la cual por default está Bose.



14. Si presionamos buscar. Nos muestra lo siguiente.



15. Y si damos clic en una canción presenta el video de la canción escogida.

16. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **World Wide Web Consortium.** W3C Working Group Note. *Web Services Glossary.* [En línea] https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice.
2. **MacLean, Dave, Komatineni, Satya y Allen, Grant.** *Pro Android 5.* 2015.
3. Introducing JSON. *JSON.* [En línea] https://www.json.org/.
4. **Wei-Meng, Lee.** *Beginning Androi 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
5. **tutotials point.** Android - JSON Parser. *Learn Android.* [En línea] http://www.tutorialspoint.com/android/android_json_parser.htm.

No. 15. XML.

Objetivo

Crear una aplicación en entorno visual. Analizar y comprender el funcionamiento de una Interface Gráfica de Usuario con Android, Mostrar la interacción con el usuario para mostrar información a través de la red usando objetos XML.

TEMARIO

TEMA	NÚM. PRÁCTICA
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.	**No. 13. Geo localización.** Objetivo. Crear un mapa usando google maps. **No. 14. JSON.** Objetivo. Conocer las formas de almacenar información. **No. 15. XML.** Objetivo. Conocer las formas de almacenar información.

Introducción

Un servicio web es un sistema de software diseñado para admitir la interacción interoperable de máquina a máquina a través de una red. Generalmente se transmiten utilizando HTTP con una serialización XML junto con otros estándares relacionados con la Web. (1)

Usando el protocolo HTTP, puede realizar una amplia variedad de tareas, como descargar páginas web de un servidor web, descargando datos binarios, y más. (2)

XML significa lenguaje de marcado extensible, fue diseñado para almacenar y transportar datos, para distribuir datos a través de Internet y para ser comprendido por humanos y máquinas. (3)

Material y equipo necesario.

El material necesario para la práctica es el siguiente:

1. Tener una computadora con Android Studio instalado.
2. Tener acceso a Internet.
3. Este manual de prácticas.
4. Utilizar hojas y lápiz.
5. Tener acceso a los recursos del Moodle.

Metodología:

Las prácticas están pensadas para emplear los algoritmos de la materia Tópicos Avanzado de Programación en una computadora y hacer programas tipo.

1. Trabajo individual.
2. Analizar el problema tipo propuesto. Proponer un algoritmo de solución.
3. Conectarse a Internet para obtener la información para buscar información sobre los “XML”.
4. Obtener ejemplos básicos en Android usando “XML como transporte en Internet”.
5. Seguir el procedimiento de la práctica.
6. Probar que su aplicación este corriendo y otorgando resultados correctos.

Sugerencias didácticas.

1. Realizar una investigación documental de la implementación de “XML”.
2. Seleccionar a varios participantes para exponer ejemplos de conversión de páginas XML y obtener objetos Java.
3. Seleccionar a varios voluntarios para obtener ejemplos de páginas usando XML como resultado de petición de servicio.
4. Seleccionar a voluntarios para explicar servicios como WSDL, SOAP, RDF y RSS.
5. Llenar el recurso de Wiki con las funciones de “XmlPullParser”, “Tipos de TAG”.

Reporte de los alumnos (resultados).

1. Hacer el programa con el ejemplo de la práctica. Imprimir las pantallas, así como su programa fuente. Entregar con su nombre, materia, fecha, nombre del maestro.
2. Crear una aplicación por el alumno para usar una ejemplo inventada por el alumno contendrá su propia página de acceso XML.

Procedimiento:

1. El tipo de problema a resolver se propone lo siguiente:

   > "Crear una aplicación con una ventana principal para presentar la definición de palabras en un servicio web a una solicitud del usuario".

   El algoritmo de solución aquí propuesto es:
   "Hacer una interface gráfica donde se el usuario escribe la palabra a definir. Posteriormente consultar el servicio a través de una URL y generar una lista de resultados".

2. Crear un proyecto en Android Studio.

   a. En Android Studio creamos un proyecto "Practica15". Usamos "Phone and Tablet" y SDK como API 15 con "**Basic Activity**" y la clase principal "PrincActivity".

   b. Dar el permiso de Internet en el archivo "AndroidManifest.xml". Antes del tag "<application" insertamos el siguiente permiso.

```
<uses-permission android:name="android.permission.INTERNET" />
```

3. Abrimos el emulador y verificamos el acceso a Internet.
   a. Abrimos un Navegador y buscamos una página.
   b. Si el emulador no tiene acceso a Internet probaremos lo siguiente. Cerramos el emulador y lo lanzamos manualmente. Probablemente el error sea que el emulador no lee el DNS. En una ventana de MS-DOS daremos los siguientes comandos.

```
C:\Users\hmo>cd AppData\Local\Android\Sdk\tools
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -list-avds
Galaxy Nexus API 22
Nexus_4_API_22
Nexus_5X_API_25
Nexus_One_API_26
C:\Users\hmo\AppData\Local\Android\Sdk\tools>emulator -avd
Nexus_5X_API_25 -dns-server 8.8.8.8
```

4. Creamos una clase para obtener un String con todo un documento XML extraído de una URL. La dirección usada en esta práctica está asociada con un servicio web. Éste servicio obtiene de un diccionario la definición de una palabra dada.

   http://services.aonaware.com/DictService/DictService.asmx?op=Define

Y tiene un método GET:

http://services.aonaware.com/DictService/DictService.asmx/Define?**word**=string

5. Crearemos una clase “GetXmlDefWord” derivada de “AsyncTask” le proporcionaremos una palabra y obtenemos el documento XML. Reescribimos los métodos
   a. La primer aproximación es:

```
public class GetXmlDefWord extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... strings) {
        return null;
    }
}
```

   b. Incorporamos la constante base del URL y obtenemos el texto XML de la página de servicio.

```
public class GetXmlDefWord extends AsyncTask<String, Void, String> {
    private final String BASE_URL =

"http://services.aonaware.com/DictService/DictService.asmx/Define?word=";

    @Override
    protected String doInBackground(String... strings) {
        URL urlWord;
        String word;
        String defXML = "";
        word = strings[0] == null ? "Hello" : strings[0];
        try {
            urlWord = new URL(BASE_URL + word);
            defXML = getXMLString(urlWord);
        } catch (MalformedURLException e) {
            Log.v("GetXml",word);
        } catch (IOException e) {
            Log.v("GetXml","Error en la Red");
        } return defXML;
    }

    private static String getXMLString(URL url) throws IOException {
        HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();
        try {
            InputStream in = urlConnection.getInputStream();

            Scanner scanner = new Scanner(in);
            scanner.useDelimiter("\\A");

            boolean hasInput = scanner.hasNext();
            String response = null;
            if (hasInput) {
                response = scanner.next();
            }
            scanner.close();
            return response;
        } finally {
            urlConnection.disconnect();
        }
    }
}
```

6. Los elementos de nuestro recurso en “activity_princ” se muestra a continuación. Es un EditText lladado etBusca para almacenar la palabra para buscar sus definiciones. Hay una ProgressBar para indicar cuando esté haciendo la búsqueda notifique al usuario. Y finalmente los resultados en EditText donde se mostrarán las definiciones.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".PrincActivity">

    <LinearLayout
        android:id="@+id/linearHor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/etBusca"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_weight="1"
            android:inputType="text"
            android:text=""
            android:hint="@string/palabra_a_buscar_enter"/>

        <ProgressBar
            android:id="@+id/progressBar"
            style="?android:attr/progressBarStyle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginStart="8dp"
            android:layout_weight="1"
            android:visibility="invisible"
            app:layout_constraintEnd_toEndOf="parent" />

    </LinearLayout>

    <View
        android:id="@+id/divider"
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_weight="1"
        android:background="?android:attr/listDivider" />

    <LinearLayout
        android:id="@+id/linearVer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="8dp">
```

```
        <EditText
            android:id="@+id/editText"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:ems="10"
            android:inputType="none|textMultiLine" />

    </LinearLayout>

</LinearLayout>
```

7. Si buscamos una palabra como “Hello” en un navegador con el siguiente URL.

   http://services.aonaware.com/DictService/DictService.asmx/Define?word=Hello

   Obtendríamos el siguiente resultado:

```
<?xml version="1.0" encoding="utf-8"?>
<WordDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://services.aonaware.com/webservices/">
  <Word>hello</Word>
  <Definitions>
    <Definition>
      <Word>hello</Word>
      <Dictionary>
        <Id>gcide</Id>
        <Name>The Collaborative International Dictionary of English
v.0.44</Name>
      </Dictionary>
      <WordDefinition>Hello \Hel*lo"\, interj. &amp;amp; n.
   An exclamation used as a greeting, to call attention, as an
   exclamation of surprise, or to encourage one. This variant of
   {Halloo} and {Holloo} has become the dominant form. In the
   United States, it is the most common greeting used in
   answering a telephone.
   [1913 Webster +PJC]
</WordDefinition>
    </Definition>
    <Definition>
      <Word>hello</Word>
      <Dictionary>
        <Id>wn</Id>
        <Name>WordNet (r) 2.0</Name>
      </Dictionary>
      <WordDefinition>hello
     n : an expression of greeting; "every morning they exchanged
         polite hellos" [syn: {hullo}, {hi}, {howdy}, {how-do-you-do}]
</WordDefinition>
    </Definition>
  </Definitions>
</WordDefinition>
```

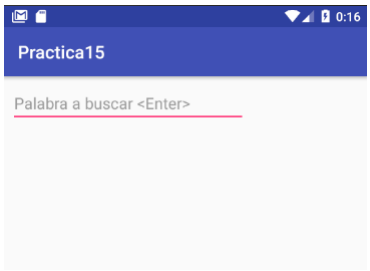
8. Si deseamos obtener las definiciones de la palabra hacemos una búsqueda de los elementos XML. Es a través de la clase XmlPullParser que nos permite leer las definiciones contenidas en el texto del tag “<WordDefinition>”. La clase PrincActivity.java nos queda como sigue:

```
public class PrincActivity extends AppCompatActivity {

    ProgressBar progressBar;
    EditText editText;
```

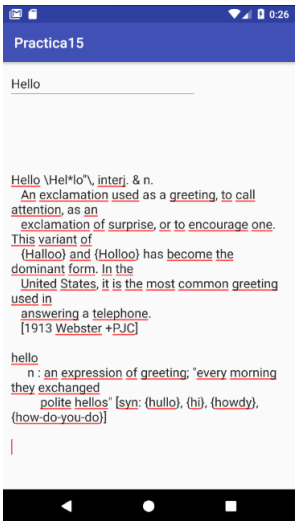
```
    EditText wordText;
    boolean buscando;

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

        progressBar = findViewById(R.id.progressBar);
        editText = findViewById(R.id.editText);
        wordText = findViewById(R.id.etBusca);

        wordText.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                if (hasFocus) {
                    wordText.setText("", TextView.BufferType.EDITABLE);
                }
            }
        });

        wordText.setOnKeyListener(new View.OnKeyListener() {
            @Override
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                if ((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode
== KeyEvent.KEYCODE_ENTER)) {
                    BuscaWord();
                    return true;
                }
                return false;
            }
        });
    }

    public void BuscaWord() {
        String words[], word;
        Log.v("Busca", "Hello");
        if (!buscando) {
            buscando = true;
            progressBar.setVisibility(View.VISIBLE);
            words = TextUtils.split(wordText.getText().toString(), " ");
            word = TextUtils.isEmpty(words[0]) ? "Hello" : words[0];
            new GetJavaFromXML().execute(word);
        } else {
            Toast.makeText(this, "Buscando...", Toast.LENGTH_SHORT).show();
        }
    }

    class GetJavaFromXML extends GetXmlDefWord {
        @Override
        protected String doInBackground(String... strings) {
            return super.doInBackground(strings);
        }

        @Override
        protected void onPostExecute(String s) {
            progressBar.setVisibility(View.INVISIBLE);
            buscando = false;
            ProcesaXML(s);
            editText.setText("");
            for (String si : list) {
                editText.append(si+"\n");
            }
        }
    }

    private static final String TAG_DEFINICION = "WordDefinition";
    private static final String ns = null;
    private static XmlPullParser xpp;
```

```
    private static List<String> list = new ArrayList<>();

    public void ProcesaXML(String s) {
        xpp = Xml.newPullParser();
        list.clear();
        try {
            xpp.setInput(new StringReader(s));
            xpp.nextTag();
            leerDefiniciones();
        } catch (XmlPullParserException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void leerDefiniciones() throws XmlPullParserException, IOException
{
        String s, et;
        xpp.require(XmlPullParser.START_TAG, ns, TAG_DEFINICION);
        while (xpp.next() != XmlPullParser.END_DOCUMENT) {
            if (xpp.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }
            et = xpp.getName();
            Log.v("Etiqueta", et);
            if (et.equals(TAG_DEFINICION)) {
                s = obtenerTexto();
                list.add(s);
            }
        }
    }

    // Obtiene el texto de la etiqueta
    private String obtenerTexto() throws IOException, XmlPullParserException
{
        String resultado = "";
        if (xpp.next() == XmlPullParser.TEXT) {
            resultado = xpp.getText();
            xpp.nextTag();
        }
        return resultado;
    }

}
```

9. Al correr nuestro programa nos presenta una interface como la siguiente. Donde nos pide las definiciones de una palabra.



10. Si buscamos la palabra Hello. Se actualiza el editText con las definiciones de la palabra buscada.



11. Ejercicio finalizado ☺.

Bibliografía Preliminar

1. **World Wide Web Consortium.** W3C Working Group Note. *Web Services Glossary.* [En línea] https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice.
2. **Lee, Wei Meng.** *Beginning Android™ 4 Application Development.* Indianapolis, IN 46256 : John Wiley & Sons, Inc., 2012.
3. **Refsnes Data.** XML Tutorial. *w3schools.com.* [En línea] 2018 de Jun. https://www.w3schools.com/xml/default.asp.