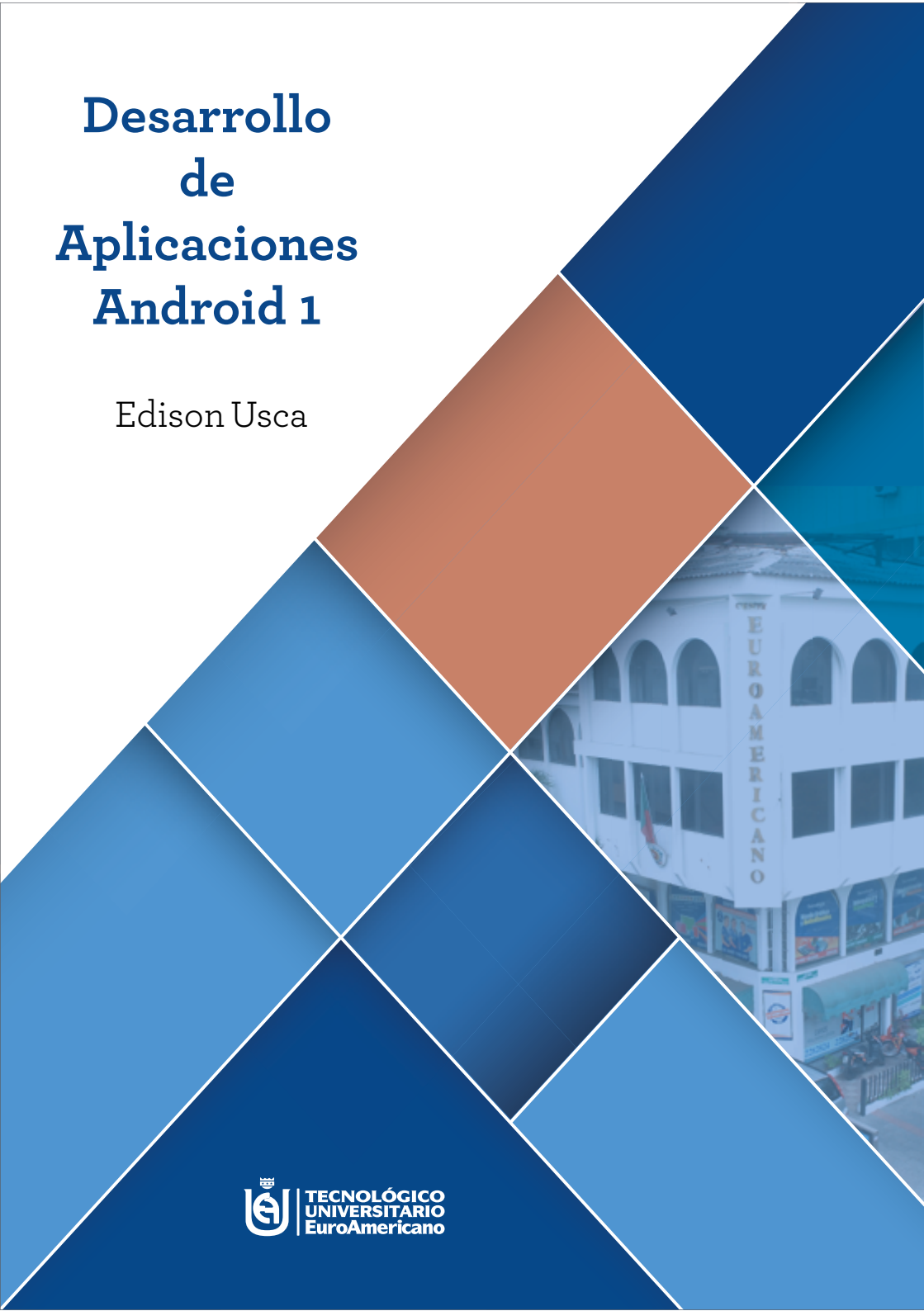
Desarrollo de Aplicaciones Android 1

Edison Usca



TECNOLÓGICO UNIVERSITARIO EuroAmericano

Desarrollo de Aplicaciones Android 1

Ing. Edison Javier Usca Paguay

Tecnológico Universitario EuroAmericano

DIRECCIÓN:
Quisquís 1317 y Los Ríos
Guayaquil – Guayas - Ecuador
(+593) 04-2288-440
www.euroamericano.edu.ec

RECTOR:
Mgtr. Antonio Manuel Marques Guitiérrez

AUTOR:
Ing. Edison Javier Usca Paguay

CORREO:
eusca@euroamericano.edu.ec

Primera Edición – mayo 2023
Editorial "R2ICS" | Pichincha | Quito | Ecuador

Datos de catalogación bibliográfica

USCA-PAGUAY, E.

Desarrollo de Aplicaciones Android 1
Primera Edición
Quito, Ecuador, 2023
Editorial: Red Internacional de Investigación en Ciencias Sociales y Humanidades "R2ICS"

ISBN: 978-9942-8940-8-3
Área: Aplicaciones Tecnológicas
Formato A5: 148 x 210 mm
Páginas: 102

Diseño y maquetación R2ICS

Reservados todos los derechos. Ni la totalidad ni parte de esta publicación puede reproducirse, registrarse o transmitirse, por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea electrónico, mecánico, fotoquímico, magnético o electroóptico, por fotocopia, grabación o cualquier otro, sin permiso previo por escrito del editor.
El préstamo, alquiler o cualquier otra forma de cesión de uso de este ejemplar requerirá también la autorización del autor o de sus representantes.
Conforme lo establece el Art. 71 y 72 del Reglamento de Carrera y Escalafón del Profesor e Investigador del Sistema de Educación Superior (Codificación), este texto ha sido sometido a un proceso de revisión de pares disciplinares así como la revisión metodológica. El detalle en anexo evaluación de pares.

Índice

Índice

Prólogo 11
Introducción 15
Capítulo 1 17
Introducción Sistema Operativo Android 19
Fundamentos de una Aplicación 19
Ventajas de Usar Kotlin 22
Componentes 24
Capítulo 2 25
Instalación del IDE Android Studio 27
Preparación del Equipo 27
Configuración del Emulador 30
Capítulo 3 33
Principales Librerías de Android Studio 35
Librerías Básicas 35
Librerías Avanzadas 37
Capítulo 4 39
Estructura de un Proyecto Android 41
La Carpeta Manifests 42
La Carpeta Res 42
Creación de Una App Básica en Android Studio con Lenguaje Kotlin 43
Capítulo 5 55
Ciclo de Vida de una Actividad 57
Prioridades y Estados de una Activity 61
Guardando el Estado de una Actividad 62
Capítulo 6 65
Componentes de Vistas Personalizadas 67
¿Qué son los Layouts y Cuáles Existen en Android Studio? 69
¿Qué son los Layouts en Android Studio? 69
Tipos de Layouts ConstraintLayout 69
ScrollView 72
Capítulo 7 75
Controles Básicos Utilizados en Android 77
TextView 77
EditText 78
CheckBox 78
RadioButton 79
Button 79
ImageButton 79
ImageView 80
Capítulo 8 81
Uso de Archivos XML 83
Conceptos Básicos de la Interfaz de Usuario 83
Diferentes Archivos XML Usados en Android 83
Uso de la Colección en Android 87

GridView 88
Referencias Bibliográficas 93

Prólogo

Prólogo

Android Studio es un entorno de desarrollo ideal para la creación de soluciones móviles que pueden realizarse desde dispositivos diminutos como relojes inteligentes hasta la plataforma de interacción y control para televisores de última tecnología.

El lenguaje de programación Kotlin ha sido el empleado oficialmente para la codificación de Android Studio y se ha oficializado por la gigante empresa tecnológica Google, la cual a diario publica actualizaciones en librerías y guías de acción para código abierto; cada vez va acrecentando más la comunidad de usuarios y desarrolladores para este tipo de aplicativos, permitiendo y contribuyendo con la generación de aplicaciones móviles recursivas y responsivas para brindar comodidad a millones y millones de usuarios a nivel mundial.

Introducción

Introducción

El desarrollo de aplicaciones y soluciones informáticas para diferentes entornos se ha convertido en la meta principal de ingenieros, arquitectos de software, desarrolladores, diseñadores y demás personajes relacionados con estos hechos de forma general. El estudiante del Tecnológico Universitario Euroamericano ha sido capacitado en varios entornos para la elaboración y desarrollo de soluciones de esta índole, sirviendo de plataforma previa para desenvolverse en cualquier área tecnológica y presentar soluciones software de acuerdo con sus conocimientos y destrezas.

El presente documento introduce las directrices y material didáctico que se emplea en el Tecnológico Universitario Euroamericano para capacitar sobre el desarrollo de aplicaciones para dispositivos móviles, televisores Inteligentes, así como de un sinnúmero de artefactos que incluyen el Sistema Operativo Android como nativo para su desempeño.

Se incorporan criterios y paradigmas como la programación orientada a objetos, así como la programación estructurada en un entorno para aplicaciones móviles que ofrece el lenguaje de programación Java, creando una estructura similar a su desempeño como lo es el lenguaje Kotlin, el mismo que se ha convertido en el lenguaje principal y reconocido como nativo por Google como parte oficial para entornos en los que se puede crear nuevas soluciones para dispositivos móviles.

El propósito de este estudio es plasmar en el conocimiento de los estudiantes e instruir sobre el manejo del código nativo en la actualidad para el desarrollo de aplicaciones en móviles, utilizando el entorno de desarrollo que Android Estudio ofrece para este cometido.

Capítulo 1

Introducción Sistema Operativo Android

Fundamentos de una Aplicación

En un entorno de desarrollo de aplicaciones móviles y para cierto tipo de dispositivos en el que se fundamenta los principios del código abierto, así como los de plataformas embebidas con control inteligente, se destaca el entorno de desarrollo de Android, donde se prioriza el criterio tomado desde los repositorios de Google que en uno de sus sitios oficiales dice textualmente: ***"En las aplicaciones Android trabajamos con programación orientada a objetos Java el SDK tiene herramientas que permiten trabajar en ambientes APK o bien con el paquete Android dicho ser**á **nuestro instalador. Est**á **catalogado con diferentes características tiene su propio sistema operativo dará permisos para todos los ficheros de una aplicación para el usuario app"*** (Google, n.d.)

Es necesario también identificar el manejo de licencias y derechos de uso de las herramientas de desarrollo, así como el uso en general de la plataforma Android Studio para la implementación de nuevas soluciones para todo tipo de dispositivos que ya tienen como Sistema Embebido general para electrodomésticos, dispositivos móviles o Sistemas Operativos de Televisores y demás artefactos modernos.

Android identifica el Lenguaje Kotlin como lenguaje oficial para el desarrollo, determinando que es un Lenguaje compatible de primer nivel, según lo destaca el sitio oficial de Android Studio que textualmente indica: ***"Kotlin es un len-***

guaje compatible con Android que es conciso y expresivo, y está diseñado para tener seguridad de tipos y contra nulos. Funciona sin inconvenientes con Java, de modo que los desarrolladores que aman este lenguaje puedan seguir usándolo y, a su vez, agregar de manera incremental el código Kotlin y aprovechar las bibliotecas. Además, como muchos desarrolladores de Android ya descubrieron que Kotlin permite un desarrollo más rápido y divertido, queremos brindar una mejor asistencia a los usuarios de este lenguaje. Obtén más información sobre el *enfoque de prioridad de Kotlin en Android.**”*** (Android, n.d.)

En cuanto al manejo y uso de su licencia se puede definir que emplea el mismo concepto de código abierto, esto se verifica en el mismo sitio web oficial: ***“La licencia preferida para Kotlin es la Licencia del software Apache versión 2.0 (“Apache 2.0”), y la mayor parte de Kotlin cuenta con ella. Si bien nos esforzaremos para que el proyecto se adhiera a la licencia preferida, es posible que haya excepciones que se resolverán según cada caso. Por ejemplo, ciertas dependencias de terceros que usan Kotlin podrían tener licencias con códigos abiertos diferentes, pero que de igual manera son compatibles con Apache 2.”*** (Android, n.d.)

Historia y Fundamentos Esenciales del Lenguaje Nativo Kotlin

Desde hace más de una década, el lenguaje oficial para el desarrollo de aplicaciones móviles fue Kotlin, el cual se considera como un Java mejorado, ya que mantiene la estructura esencial de la programación orientada a objetos, pero realizando cierto tipo de mejoras que son resaltadas más aún con

el entorno de desarrollo que ofrece Android Studio, ya que, con ciertos atajos y código abreviado, entre otras ayudas, ha mejorado y agilizado el proceso de revisión en cuanto a sintaxis.

Además, podemos citar algo de historia referente a su aparición reemplazando al tradicional Java de Oracle, el cual aún mantiene su estructura como lenguaje, pudiendo hallar compatibilidad con versiones actuales de Kotlin, pero con carencia de nuevas librerías necesarias para su posible compilación.

"Kotlin fue creado en 2010 por JetBrains, la empresa detrás del famoso IDE para Java Intellij IDEA (aunque se puede usar en cualquier otro IDE, como ***Eclipse)."*** (Cabot, 2017)

"Todos los IDEs de JetBrains están escritos en Java y otros lenguajes basados en la JVM (Java Virtual Machine) y por lo tanto fácilmente interoperables con Java ya que todos acaban compilando a bytecode que la JVM es capaz de ejecutar." (Cabot, 2017)

"El mayor problema que tenían es que su codebase era muy grande (Java no es precisamente un lenguaje conciso) y pensaron que algunas de las características de lenguajes más recientes (como el uso de lambdas) podría ayudarlos. El problema es que en ese momento (2010) no había muchos lenguajes compatibles con JVM que tuvieran esas funcionalidades." (Cabot, 2017)

"Inicialmente Kotlin fue creado para aplicaciones Desktop (que es el mercado de JetBrains) pero ahora mismo

Kotlin es multiplataforma con soporte para Android, compilación a JavaScript e incluso ***compilación nativa vía LLVM*** para entornos donde JVM no sea una buena opción.**”** (Cabot, 2017)

Ventajas de Usar Kotlin

En el siguiente apartado, se incorporan algunas de las ventajas que ofrece el Lenguaje de Programación Kotlin, sobre el lenguaje tradicional Java:

“Ahorro de getters, setters y cia. *Para cada atributo normalmente quieres tener el correspondiente get y set, incluir la propiedad en el método equals,… Muchos IDEs te los generan automáticamente, pero al final, aunque sea código generado lo vas a tener que mantener (ej. cuando añadas un nuevo atributo). Kotlin te los proporciona automáticamente. Es decir, no los ves, el código no está, pero los puedes utilizar (Kotlin los va a añadir durante la compilación a bytecode).”* (Cabot, 2017)

“Casting automático*. Una vez compruebas si un objeto es de un cierto tipo, no necesitas después hacer un casting explícito para usar las propiedades de ese tipo. Kotlin asume que si la comprobación tiene éxito el objeto pasa a ser del tipo concreto inmediatamente y lo trata como tal”* (Cabot, 2017)

“Organización de paquetes*, ya no hace falta que la organización de tu código en paquetes siga la estructura de carpetas en tu directorio”* (Cabot, 2017)

"Valores nulos. Kotlin no tiene el concepto de *null. Lo que implica que nunca tendrás una excepción del tipo Null-PointerException* (¿Cuántas veces te has encontrado una de esas?). Está demostrado que evitar nulos ahorra mucho tiempo (y dinero) en el desarrollo y que hay maneras más seguras de indicar la ausencia de valor.*"* (Cabot, 2017)

"Funciones. *En Kotlin, las funciones están al mismo nivel que las clases. Puedes definir funciones sin tener que crear primero una clase para ellas. Además, las funciones pueden recibir como argumento otra función (o devolver una como resultado) permitiendo un alto nivel de programación funcional. También hay soporte para lambdas / funciones anónimas."*

"Extensiones a clases existentes. *Kotlin permite añadir nuevos métodos a una clase existente SIN tener que crear la subclase correspondiente. Esto permite que el nuevo método se pueda ejecutar sobre todas las instancias de la clase original y no sólo sobre las de la subclase que, en Java, estaría obligado a crear. De la misma forma, Kotlin también te permite añadir nuevas propiedades."* (Cabot, 2017)

En cuanto a la estructura de un proyecto de Android, se identifica la siguiente información, definida en el sitio de Google:

- Cada aplicación será un usuario diferente
- Cada proceso tiene su propia maquina
- Cada aplicación corre en su propio proceso Linux
- Android trabaja con normas principios de menor

privilegio es decir permisos para no dañar el sistema operativo es decir adquirir permisos básicos y lo más seguros posible. (Google, n.d.)

Componentes

Cuenta con 4 componentes principales:

Android Manifest.- Es un archivo de extensión .XML, en el cual se realizan la configuración a gran escala de la App., como, por ejemplo: permisos de acceso a hardware del dispositivo, permisos de acceso a internet para la aplicación, así como la clase principal desde donde arranca la misma.

MainActivity.kt.- Es donde se aloja la clase principal del proyecto, una vez se crea, en él se genera y establece la lógica o conocido también como el modelo del proyecto, el código establecido es kotlin y su extensión es kt.

activity_main.- Es el archivo donde se aloja la actividad del proyecto, las pantallas, objetos, en definitiva, el diseño de la App. El lenguaje es .XML, las etiquetas establecidas en la estructura del proyecto, definen el diseño y la integración de valores, estilos, colores, temas y demás detalles para dicha aplicación.

build.gradle.- En este archivo se establece las librerías y dependencias necesarias para la integración del proyecto hacia agentes externos, por ejemplo, la dependencia de un archivo json.

Capítulo 2

Instalación del IDE Android Studio

Preparación del Equipo

Para la elaboración de soluciones móviles en la plataforma de Android, se requiere la instalación del entorno de desarrollo Android Studio, para esto se tendrán las siguientes consideraciones referente a los requisitos básicos, tal como lo muestra la figura 1 del presente documento, además también se referencia el sitio de descarga oficial y el detalle paso a paso de como instalar la plataforma en un entorno Windows.

Cabe mencionar que, en el proceso de Instalación, se puede especificar el destino de los archivos, librerías y demás componentes que serán almacenados de forma permanente en la unidad de disco duro del equipo destino.

Es prioritario también recalcar que los archivos de instalación y librerías que se almacenan una vez se concluye con el proceso de instalación, son sujetos a cambios constantes, ya que cada vez que se genere una nueva actualización, sea en el compilador o sea en el entorno de desarrollo propiamente dicho (plugins, jdk), el tamaño de los ficheros en mención serán incrementados, el mismo escenario se tiene con la configuración del emulador para pruebas de cada aplicación creada.

La siguiente gráfica muestra los requerimientos básicos en cuanto a recursos hardware para poder lograr una instalación estable en un ordenador, dependiendo del sistema Operativo.

Figura 1

Requisitos básicos de equipo hardware para instalación de Android Studio

Fuente: Tomado de (developer.android, 2022)

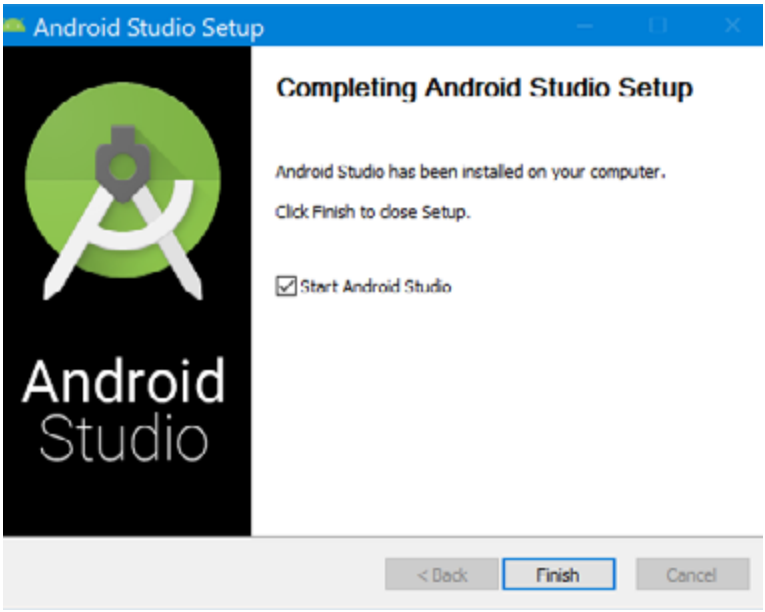
Nota: Esta imagen corresponde a los requerimientos mínimos de Hardware para la Instalación de Android en los Sistemas Operativos conocidos.

Una vez se conoce los requerimientos funcionales básicos de hardware, previo a la instalación, se determina si adicional al Entorno de desarrollo, se incluirá o no el emulador

y sus archivos de instalación, así como el aviso del espacio en disco físico que ocupará este proceso, tal como lo muestra la figura 2.

Figura 2

Proceso de Instalación de Android Studio en un equipo Windows 10



Fuente: Tomado de (developer.android.com/codelabs, 2017)
Nota: Esta gráfica contiene un asistente del instalador del Framework de Android Studio, con inclusión de Emulador de dispositivo Android.

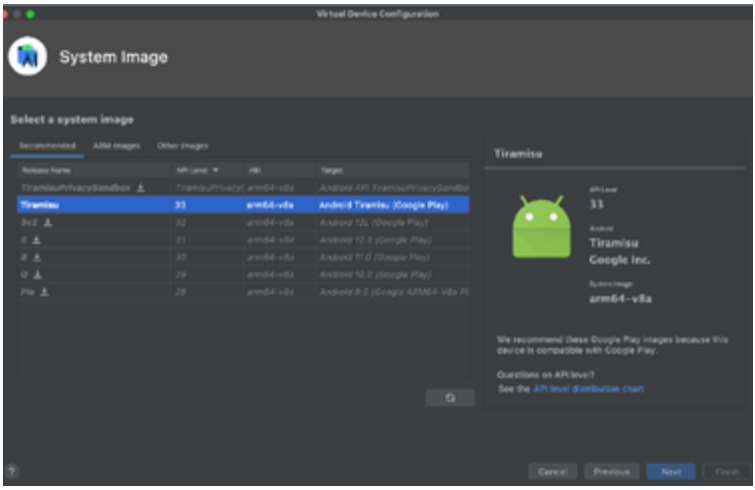
Una vez se procede con la instalación normal del Entorno de Desarrollo, se verifica el mensaje de Bienvenida, se trata de un asistente para la configuración por primera vez de mencionado entorno.

Configuración del Emulador

El asistente de Android Studio especifica la posibilidad de configurar un emulador para pruebas de aplicaciones desarrolladas en el IDE (Entorno de Desarrollo), donde se destaca el espacio utilizado de disco duro, los recursos en cuanto a procesador y memoria física separada para este proceso y las librerías y ficheros necesarios para esta instalación, la cual debe ser especificada por el usuario, tal como lo indica la figura 3.

Figura 3

Configuración del Emulador Android



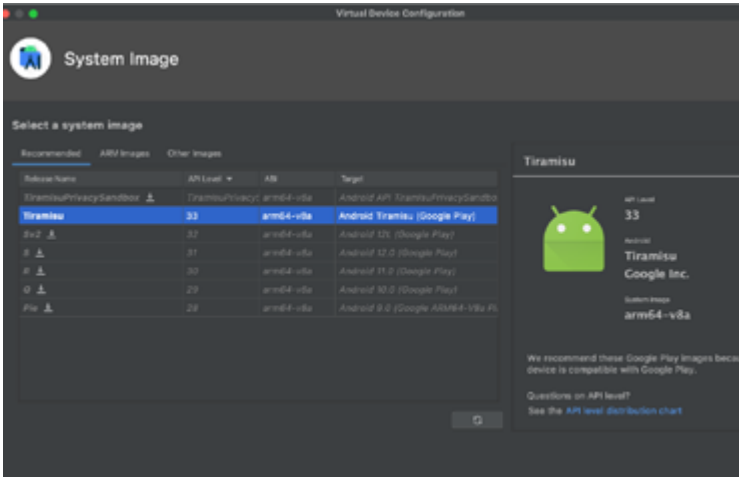
Fuente: Tomado de (filehorse.com, 2022)

Nota: La presente gráfica muestra el asistente de configuración del Emulador del Android Studio, selecciona el Hardware a asignar desde la computadora física para su funcionamiento.

Finalmente, en este proceso previo a la elaboración aun primer proyecto, se determina el tipo de emulador a emplear, sea este para dispositivos móviles telefónicos, tablets, radios, televisores o relojes con sistema Operativo compatible o nativo de Android, según la ilustración de la figura 4.

Figura 4

Diseño de pantalla de emulador a configurar para IDE ANDROID STUDIO



Fuente: Tomado de: (developer.android.com/studio, 2022)
Nota: Esta imagen representa el asistente de diseño del Emulador a definir de acuerdo con el modelo de equipo, tipo de dispositivo, tamaño de pantalla, así como las cámaras requeridas.

Capítulo 3

Principales Librerías de Android Studio

Android Studio es un entorno de desarrollo muy completo que determina una estructura con múltiples herramientas, empleando como otros lenguajes definidos con mucha trayectoria ya, varias utilidades, entre ellas el paradigma de la Programación Orientado a Objetos en las que intervienen conceptos como herencia, polimorfismo, ocultamiento, entre otras, además se refiere una serie de librerías que se destaca a continuación:

"Android ofrece una serie de APIs para el desarrollo de sus aplicaciones. La siguiente lista proporciona una visión de lo que está disponible. Todos los dispositivos Android ofrecen al menos estas APIs:" (Google, n.d.)

Librerías Básicas

android.util El paquete básico de servicios públicos contiene las clases de bajo nivel, como contenedores especializados, formateadores de cadenas, y de análisis XML de servicios públicos. (Google, n.d.)

android.os El paquete de sistema operativo permite el acceso a los servicios básicos como el paso de mensajes, la comunicación entre procesos y funciones de reloj. (Google, n.d.)

android.graphics La API de gráficos, es el suministro de las clases de bajo nivel como lienzos de apoyo, colores y las primitivas de dibujo. También le permite dibujar sobre lienzos.

(Google, n.d.)

android.text Las herramientas de procesamiento de texto para mostrarlo y analizarlo. (Google, n.d.)

android.database Proporciona las clases de bajo nivel necesario para la manipulación de cursores cuando se trabaja con bases de datos. (Google, n.d.)

android.content El contenido de la API se utiliza para admirar el acceso a los datos y a la publicación, proporcionando los servicios para hacer frente a los recursos, los proveedores de contenido y los paquetes. (Google, n.d.)

android.view Las vistas son un núcleo de la interfaz de usuario. Todos los elementos de la interfaz se construyen utilizando una serie de vistas que proporcionan los componentes de interacción con el usuario. (Google, n.d.)

android.widget Construido sobre el paquete de Vista, están las clases widget "aquí está uno que creamos antes", elementos de la interfaz de usuario para su uso en las aplicaciones. Se incluyen listas, botones y diseños. (Google, n.d.)

com.google.android.maps API de alto nivel que proporciona acceso a los controles de mapas que usted puede utilizar en su aplicación. Incluye el control MapView, así como la superposición y la clase MapController utilizados para anotar y controlar dichos mapas. (Google, n.d.)

android.app Paquete de alto nivel que proporciona el acceso al modelo de solicitud. Este, incluye la actividad de servicios y las API que forman la base de todas sus aplicaciones. (Google, n.d.)

android.provider Para facilitar el acceso a los desarrolladores a determinados proveedores de contenidos estándar, el paquete proveedor ofrece clases para todas sus distribuciones. (Google, n.d.)

android.telephony Las API´s de telefonía le dan la posibilidad de interactuar directamente con el dispositivo de Teléfono, permitiéndole realizar, recibir y controlar las llamadas de teléfono, su estado y mensajes SMS. (Google, n.d.)

android.webkit Ofrece funciones para trabajar con contenido basado en web, incluyendo un control WebView para incrustar los navegadores en sus actividades y un administrador de cookies. (Google, n.d.)

Librerías Avanzadas

"Las bibliotecas ofrecen toda la funcionalidad que necesita para comenzar a crear aplicaciones en Android, pero no pasará mucho tiempo antes de que esté dispuesto a profundizar en API´s avanzadas que ofrecen una funcionalidad más emocionante." (Google, n.d.)

"Android pretende abarcar una gran variedad de equipos móviles, de modo que tenga en cuenta que la adecuación y aplicación de las siguientes API´s variará en función del dispositivo a que se apliquen:" (Google, n.d.)

android.location Da las aplicaciones de acceso a la ubicación física del dispositivo actual. Los servicios de localización ofrecen acceso genérico a información de localización

utilizando cualquier posición de hardware xing-Fi o tecnología disponible en el dispositivo. (Google, n.d.)

android.media Las API de los medios de comunicación proporcionan soporte para reproducción y grabación de audio. (Google, n.d.)

android.opengl Android ofrece un potente motor 3D que utiliza la API de OpenGL ES que usted puede utilizar para crear interfaces de usuario en 3D dinámico para sus aplicaciones. (Google, n.d.)

android.hardware Cuando sea posible, el hardware de la API expone un sensor incluyendo la cámara, acelerómetro, sensores y una brújula. (Google, n.d.)

android.bluetooth, android.net.wifi, android and telephony Proporciona el acceso a las plataformas hardware, incluyendo Bluetooth, Wi-Fi y el hardware de telefonía. (Google, n.d.).

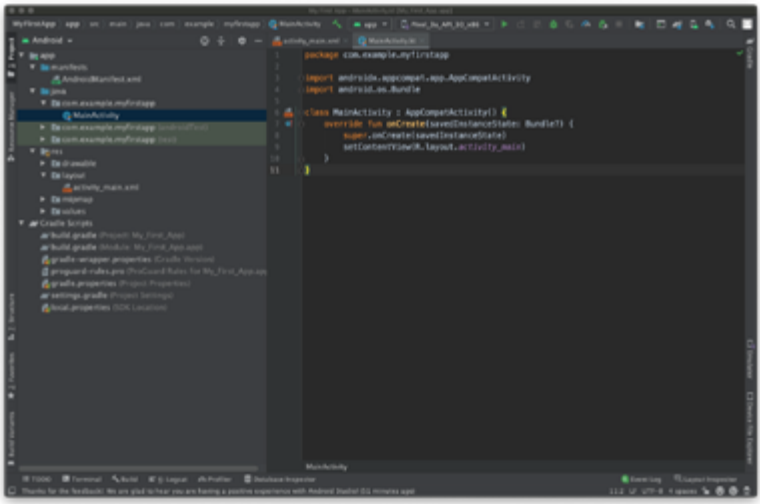
Capítulo 4

Estructura de un Proyecto Android

"Cada vez que comienzas un proyecto nuevo, Android Studio crea la estructura de carpetas y archivos necesarios para esa aplicación. Echemos un vistazo a los diferentes archivos involucrados en un proyecto de aplicación de Android." (Vaati, 2020)

Figura 5

Estructura de un proyecto Android



Fuente: Tomado de (developer.android.com/training, 2022)
Nota: La imagen representa la estructura de un proyecto creado en Android Studio desde la vista de Android con código Kotlin.

La Carpeta Manifests

"La carpeta *manifests* ***contiene el archivo*** *AndroidManifest.XML.* ***El archivo manifiesto describe información esencial sobre tu aplicación."***

La Carpeta Java

"Esta carpeta contiene los archivos del código fuente de Java. Como puedes ver arriba en la ventana del editor, el archivo *MainActivity.java* ***contiene el código fuente de Java para la actividad principal de la aplicación."***

La Carpeta Res

Esta carpeta incluye todos los recursos que no son de código, por ejemplo:

Diseños (layouts): los diseños son archivos XML que definen la arquitectura de la interfaz de usuario en una actividad o en un componente de una interfaz de usuario. Por ejemplo, en nuestra aplicación, el archivo activity_main.XML corresponde a la actividad principal.

Valores (values): contiene el color, estilo y los archivos XML de cadenas para la aplicación.

Dibujables (drawable): este es un repositorio general para todos los gráficos que pueden ser dibujados en la pantalla, por ejemplo, las imágenes.

build.gradle: este es un archivo generado automáticamente que contiene información sobre los detalles

de la aplicación, por ejemplo la versión del SDK, la versión de las herramientas de compilación, el ID de la aplicación y más.

Creación de Una App Básica en Android Studio con Lenguaje Kotlin

Para este cometido, se cumple con el mismo esquema, donde se detallará paso a paso este proceso empleando el IDE de desarrollo actualizado de Android Studio Dolphin 2021.3.01, versión con compilador y plugin actualizado a diciembre del 2022, tal como detalla la siguiente figura.

Figura 6

IDE Android Studio Dolphin 2021.3.1

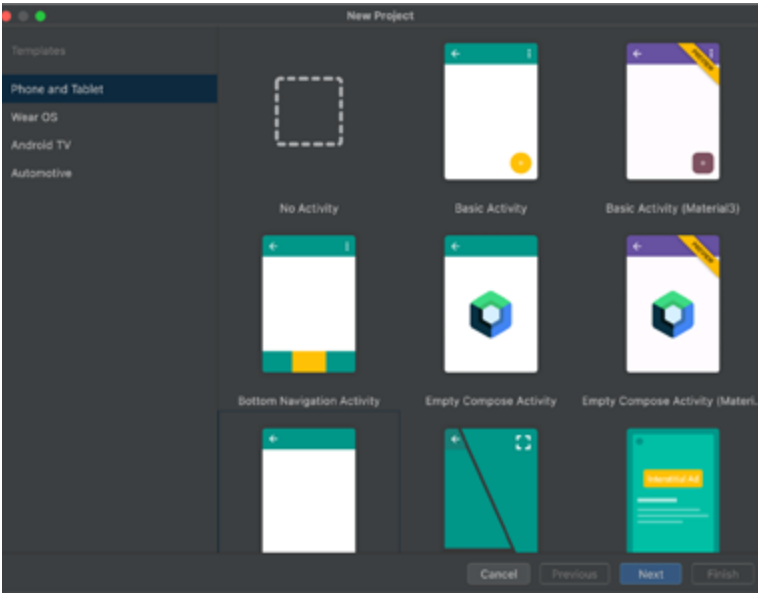
Fuente: Tomado de: (Yuri Blaise, 2022)

Nota: La imagen representa el inicio del IDE de desarrollo de Android con el plugin Dolphin del 2021.

Una vez inicia el asistente, se podrá ir a la opción New Project, el cual especifica la siguiente pantalla, donde se podrá escoger un tipo de plantilla a emplear, siempre que se tenga un modelo predefinido, al ser el primer proyecto en Kotlin, se establecerá una actividad en blanco.

Figura 7

Plantillas específicas para aplicaciones Android para teléfonos y tablets



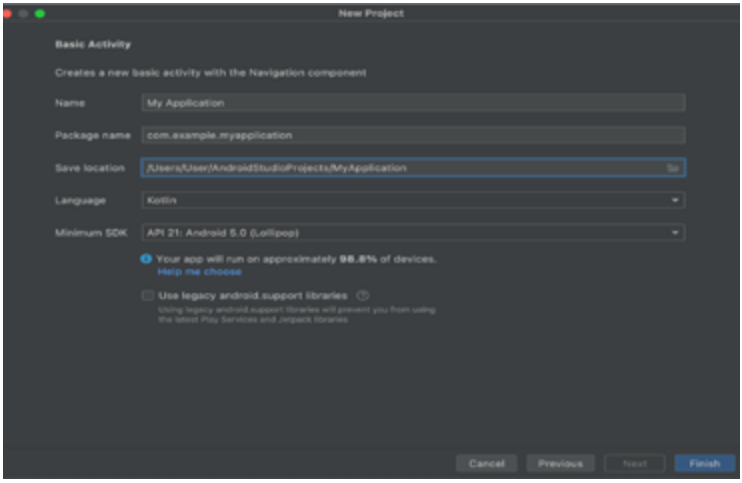
Fuente: Tomado de: (/developer.android.com/studio/past-releases, 2022)

Nota: La imagen representa el menú de plantillas de Activity que representa el modelo de pantalla para la creación de una aplicación.

Una vez especificada la plantilla, se debe determinar el nombre del proyecto, que a su vez se define como el nombre de la aplicación de forma predeterminada, pudiendo en lo posterior cambiar el mismo en el Android Manifest (Archivo de configuración a gran escala), también se muestra en este escenario el nombre del paquete donde se almacenará recursivamente los ficheros, librerías y demás archivos relacionados al proyecto, la ubicación de almacenamiento físico de los recursos del proyecto en el computador.

Figura 8

Detalles de aplicación móvil a desarrollar en Android Studio con Kotlin



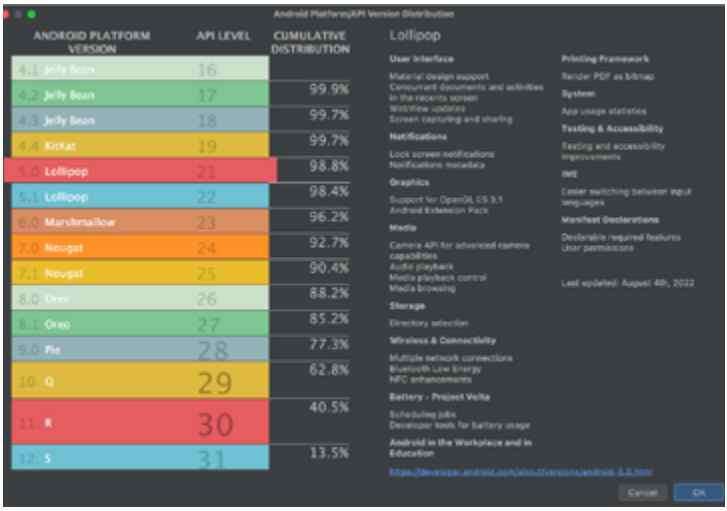
Fuente: Tomado de (developer.android.com/studio, 2022)
Nota: La imagen representa la configuración del nombre del proyecto y paquete, almacenamiento físico del proyecto, la elección del lenguaje y App mínima compatible.

Otro aspecto importante es el lenguaje a incorporar en la sintaxis nativa del proyecto, a diferencia del anterior ejemplo, se escoge lenguaje Kotlin, finalmente el mínimo SDK (compilador compatible) o versión compatible de Android donde se podrá ejecutar la aplicación a desarrollar, ante alguna duda de cuál podría ser la versión mínima óptima, este asistente otorga una opción Help me choose, donde se abre otra pantalla en la que se visualiza un listado de versiones estables de Android Studio con su correspondiente porcentaje de aceptación y de permanencia en dispositivos móviles a nivel mundial, aunque en los sitios oficiales se recomienda se escoja la más cercana al 95%.

Aunque es necesario aclarar que, a mayor número de versiones compatibles, más librerías y recursos se requieren para poder incorporar al proyecto, lo que lo convierte en una aplicación que requiere más recursos del dispositivo donde se ejecuta.

Figura 9

Pantalla de un asistente que muestra la distribución acumulativa de diferentes niveles de API

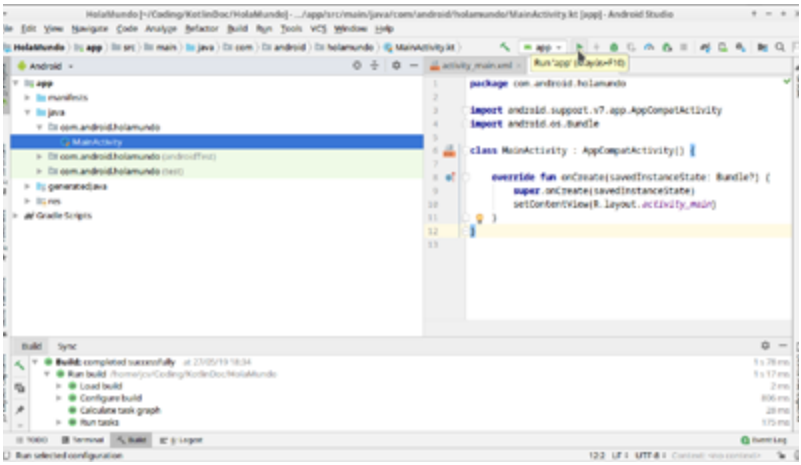


Fuente: Tomado de (developer.android.com/studio, 2022)
Nota: Recomendación del IDE Android para elección de la versión de Plataforma Android Mínima compatible a escoger como base de la aplicación creada.

Ya en la estructura del proyecto, se reconoce el mismo escenario que para una aplicación para JAVA, a diferencia de la sintaxis, la cual es un tanto diferente, requiere menos código, aunque guarda cierta similitud, se puede decir que su escritura es más intuitiva, dejando la apertura para que el desarrollador se preocupe más en la lógica que en la sintaxis.

Figura 10

Visión de Archivo MainActivity de un proyecto en Lenguaje Java.



Fuente: Tomado de (kotlindoc.blogspot.com, 2019)
Nota: La imagen representa el contenido de código del archivo MainActivity para lenguaje de programación Java.

En cuanto a la estructura del proyecto es la misma, a diferencia de la extensión de ciertos archivos que al tener la sintaxis Kotlin, cambia a la extensión kt en reemplazo de JAVA, así cada vez que se cree una clase, interfaz o archivo de lógica Kotlin, su extensión será kt.

Quedando los siguientes archivos como principales en el proyecto:

AndroidMAnifest.XML, el cual es el que mantiene las configuraciones de es el elemento que posee la configuración

a gran escala del Proyecto, por ejemplo, la configuración de: Etiqueta o nombre de la App.(Hello Kotlin), tema principal(Theme.HelloKotlin), La Api principal(31), Clase de inicio(MainActivity), la actividad a lanzarse, permiso de acceso a medios, desde y hacia el dispositivo, etc.

```
“<?XML version=”1.0” encoding=”utf-8”?>
<manifest XMLns:android=”http://schemas.android.com/apk/
res/android”
    XMLns:tools=”http://schemas.android.com/tools”>

    <application
        android:allowBackup=”true”
        android:dataExtractionRules=”@XML/data_extraction_
rules”
        android:fullBackupContent=”@XML/backup_rules”
        android:icon=”@mipmap/ic_launcher”
        android:label=”@string/app_name”
        android:roundIcon=”@mipmap/ic_launcher_round”
        android:supportsRtl=”true”
        android:theme=”@style/Theme.HolaEuro2”
        tools:targetApi=”31”>
        <activity
            android:name=”.MainActivity”
            android:exported=”true”>
            <intent-filter>
                <action android:name=”android.intent.action.
MAIN” />

                <category android:name=”android.intent.category.
LAUNCHER” />
            </intent-filter>
```

```
        <meta-data
            android:name=”android.app.lib_name”
            android:value=”” />
    </activity>
  </application>
</manifest>”
```
(Android Studio, 2022)

MainActivity.kt, la clase principal, la cual se ejecutará desde el inicio, configurado por default desde el Manifest, en este archivo y en la carpeta java, se alojarán las clases, interfaces y objetos de la programación nativa en Kotlin, prácticamente mantiene la misma estructura que JAVA, aunque la sintaxis difiere en varios aspectos, reduciendo al máximo la escritura del código.

```
“package ec.edu.euro.holaeuro2
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
class MainActivity: AppCompatActivity () {
  override fun onCreate (savedInstanceState: Bundle?) {
    super.onCreate (savedInstanceState)
    setContentView (R.layout.activity_main)
  }
}”
```
(Android Studio, 2022)

Activity_main.XML, es el archivo de diseño de las vistas e interfaces gráficas del proyecto, considerando una serie de técnicas necesarias empleando el lenguaje XML con el uso de etiquetas y referencias para lograr un diseño adecuado y conforme las necesidades del proyecto a elaborar.

```
“<?XML version=”1.0” encoding=”utf-8”?>
<androidx.constraintlayout.widget.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=”Hello World!”
      App: layout_constraintBottom_toBottomOf=”parent”
      app: layout_constraintEnd_toEndOf=”parent”
      app: layout_constraintStart_toStartOf=”parent”
      app: layout_constraintTop_toTopOf=”parent” />
</androidx.constraintlayout.widget.ConstraintLayout>”
```

(ANDROID STUDIO, 2022)

Build.gradle, este archivo contiene entre otros aspectos, la compatibilidad de versiones mínimas y target preferencial, así como la implementación de las librerías externas o dependencias que requiera la App. para ejecutarse.

```
“plugins {
   id ‘com.android.application’
   id ‘org.jetbrains.kotlin.android’
}
android {
   namespace ‘ec.edu.euro.holaeuro2’
```

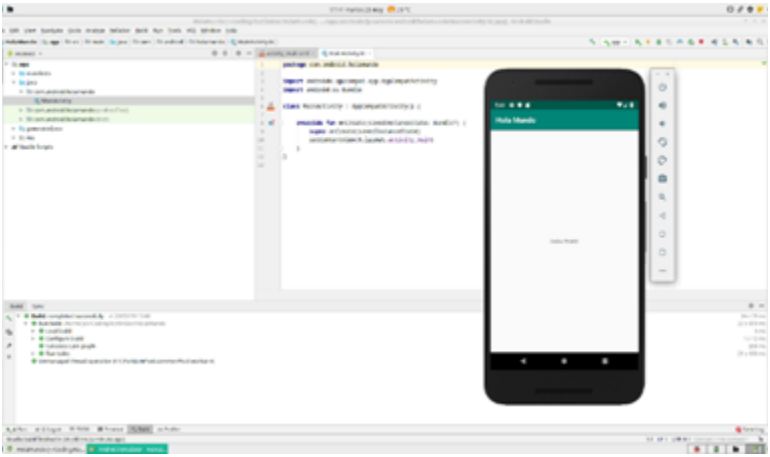
```
    compileSdk 32

    defaultConfig {
        applicationId “ec.edu.euro.holaeuro2”
        minSdk 26
        targetSdk 32
        versionCode 1
        versionName “1.0”
        testInstrumentationRunner “androidx.test.runner.
AndroidJUnitRunner”
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile(‘proguard-
android-optimize.txt’), ‘proguard-rules.pro’
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = ‘1.8’
    }
}
dependencies {
    implementation ‘androidx.core:core-ktx:1.7.0’
    implementation ‘androidx.appcompat:appcompat:1.5.1’
    implementation ‘com.google.android.
material:material:1.7.0’
```

implementation ‘androidx.constraintlayout:constraintlayout:2.1.4’
testImplementation ‘junit:junit:4.13.2’
androidTestImplementation ‘androidx.test.ext:junit:1.1.4’
androidTestImplementation ‘androidx.test.espresso:espresso-core:3.5.0’”
(Android Studio, 2022)

La simple creación de un proyecto determina una aplicación que de forma predeterminada posee un mensaje de saludo, el tradicional Hola Mundo, el cual se ejecuta a través de un emulador para el presente ejemplo en un emulador configurado para esta clase.

Figura 11

Ejecución de App. Hola Mundo en Android Studio Lenguaje Kotlin



Fuente: Tomado de (kotlindoc.blogspot.com, 2019)
Nota: La imagen representa la App. Hola mundo desde un proyecto en IDE Android Studio en un emulador.

Capítulo 5

Ciclo de Vida de una Actividad

"El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros S.O., como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario." (Universidad Politécnica de Valencia, 2017)

Una aplicación en Android está formada por un conjunto de elementos básicos de interacción con el usuario, conocidos como actividades. Además de varias actividades, una aplicación también puede contener servicios. El ciclo de vida de los servicios se estudiará en el Unidad 8. Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad.

El sistema mantiene una pila con las actividades previamente visualizadas, de forma que el usuario puede regresar a la actividad anterior pulsando la tecla "retorno". (Universidad Politécnica de Valencia, 2017)

Una aplicación Android corre dentro de su propio proceso Linux. Este proceso se crea con la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignársela a otra aplicación. (Universidad Politécnica de Valencia, 2017)

Una característica importante, y poco usual, de Android es que la destrucción de un proceso no es controlada directamente por la aplicación, sino que es el sistema el que determina cuándo destruir el proceso. Lo hace basándose en el conocimiento que tiene de las partes de la aplicación que están corriendo (actividades y servicios), en la importancia de dichas partes para el usuario y en cuánta memoria disponible hay en un determinado momento. (Universidad Politécnica de Valencia, 2017)

Si tras eliminar el proceso de una aplicación, el usuario vuelve a ella, se crea de nuevo el proceso, pero se habrá perdido el estado que tenía esa aplicación. En estos casos, será responsabilidad del programador almacenar el estado de las actividades, si queremos que cuando sean reiniciadas conserven su estado. (Universidad Politécnica de Valencia, 2017)

Como vemos, Android es sensible al ciclo de vida de una actividad; por lo tanto, necesitas comprender y manejar los eventos relacionados con el ciclo de vida si quieres crear aplicaciones estables. (Universidad Politécnica de Valencia, 2017)

Una actividad en Android puede estar en uno de estos cuatro estados:

Activa (*Running*): La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco. (Universidad Politécnica de Valencia, 2017)

Visible (*Paused*): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada. (Universidad Politécnica de Valencia, 2017)

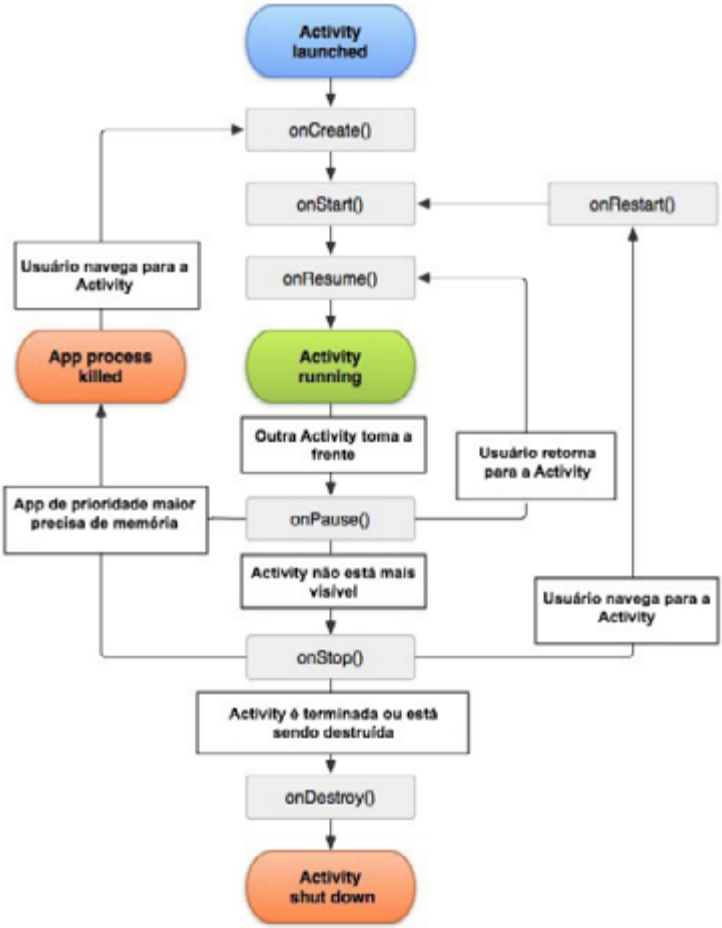
Parada (*Stopped*): Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc. (Universidad Politécnica de Valencia, 2017)

Destruida (*Destroyed*): Cuando la actividad termina al invocarse el método *finish()*, o es matada por el sistema. (Universidad Politécnica de Valencia, 2017)

Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación, se muestra un esquema que ilustra los métodos que capturan estos eventos. (Universidad Politécnica de Valencia, 2017)

Figura 12

Ciclo de vida de una Activity



Fuente: Tomado de (Bittencourt, 2022)
Nota: La imagen representa los estados a los que accede la aplicación en la transición de los ciclos de vida de una aplicación en el IDE de Android Studio.

Prioridades y Estados de una Activity

onCreate(Bundle): Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado de la actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear. (Universidad Politécnica de Valencia, 2017)

onStart(): Nos indica que la actividad está a punto de ser mostrada al usuario.

onResume(): Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música. (Universidad Politécnica de Valencia, 2017)

onPause(): Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición. (Universidad Politécnica de Valencia, 2017)

onStop(): La actividad ya no va a ser visible para el usuario. !Ojo si hay muy poca memoria! es posible que la actividad se destruya sin llamar a este método. (Universidad Politécnica de Valencia, 2017)

onRestart(): Indica que la actividad va a volver a ser representada después de haber pasado por *onStop()*. (Universidad Politécnica de Valencia, 2017)

onDestroy(): Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón de volver o cuando se llama al método *finish()*. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método. (Universidad Politécnica de Valencia, 2017)

Guardando el Estado de una Actividad

"Cuando el usuario ha estado utilizando una actividad, y tras cambiar a otras, regresa a la primera, lo habitual es que esta permanezca en memoria y continúe su ejecución sin alteraciones. Como hemos explicado, en situaciones de escasez de memoria, es posible que el sistema haya eliminado el proceso que ejecutaba la actividad. En este caso, el proceso será creado de nuevo, pero se habrá perdido su estado, es decir, se habrá perdido el valor de sus variables y el puntero de programa. Como consecuencia, si el usuario estaba a mitad de un proceso de edición o estaba reproduciendo un audio en un punto determinado perderá esta información. En este apartado estudiaremos un mecanismo sencillo que nos proporciona Android para resolver este problema." (Universidad Politécnica de Valencia, 2017)

Para guardar el estado de una actividad has de utilizar los siguientes dos métodos:

onSaveInstanceState(Bundle): Invoca el sistema cuando ha de destruir una actividad, que más adelante ha de restaurar (por cambiar su inclinación o por falta de memoria), para permitir a la actividad guardar su estado. (Universidad

Politécnica de Valencia, 2017)

onRestoreInstanceState(Bundle): Se invoca cuando se restaura la actividad para recuperar el estado guardado por onSaveInstanceState().

```
var v: String? = null
var pos: Int = 0
override fun onSaveInstanceState(guardarEstado: Bundle) {
  super.onSaveInstanceState(guardarEstado)
  guardarEstado.putString("variable", v)
  guardarEstado.putInt("posicion", pos)
}
override fun onRestoreInstanceState(recEstado: Bundle) {
  super.onRestoreInstanceState(recEstado)
  v = recEstado.getString ("variable")
  pos = recEstado.getInt("posicion")
}
```

Capítulo 6

Componentes de Vistas Personalizadas

Android ofrece un modelo componen tizado para compilar la IU que es sofisticado y potente, y que se basa en las clases de diseño fundamentales: View y ViewGroup. Para comenzar, la plataforma incluye una variedad de subclases de View y ViewGroup precompiladas, llamadas widgets y diseños, respectivamente, que puedes usar para compilar tu IU. (developer.android, n.d.)

Una lista parcial de widgets disponibles incluye Button; TextView, EditText; ListView; CheckBox; RadioButton; Gallery; Spinner; y, para objetivos más especiales, AutoCompleteTextView, ImageSwitcher y TextSwitcher. (developer.android, n.d.)

Entre los diseños disponibles, se encuentran LinearLayout, FrameLayout y RelativeLayout. Para ver más ejemplos, consulta Objetos de diseño comunes. (developer. android, n.d.)

Si ninguno de los widgets o diseños precompilados satisface tus necesidades, puedes crear tu propia subclase de View. Si solo necesitas hacer pequeños ajustes en un widget o diseño existente, puedes subclasificar el widget o el diseño, y anular sus métodos. (developer.android, n.d.)

La creación de tus propias subclases de View te brinda un control preciso sobre el aspecto y la función de un elemento de pantalla. Para tener una idea del control que obtienes con vistas personalizadas, a continuación, encontrarás algunos ejemplos de lo que podrías hacer con ellas: (developer.android, n.d.)

Podrías crear un tipo de vista con renderización completamente personalizada, como un “control de volumen” renderizado con gráficos 2D que se asemeje a un control electrónico analógico. (developer.android, n.d.)

Podrías combinar un grupo de componentes de View en un nuevo componente único, por ejemplo, para hacer algo como un ComboBox (una combinación de lista emergente y campo de texto de entrada libre), un control de selector de doble panel (paneles izquierdo y derecho con una lista en cada uno, donde puedes reasignar qué elemento hay en cada lista), y así sucesivamente. (developer.android, n.d.)

Podrías anular la forma en que se renderiza en la pantalla un componente EditText (en el instructivo del Bloc de notas, se usa esta función para crear una página de bloc de notas alineada). (developer.android, n.d.)

Podrías capturar otros eventos, como presionar teclas, y controlarlos de alguna manera personalizada (como para un juego). (developer.android, n.d.)

En las siguientes secciones, se explica cómo crear vistas personalizadas y usarlas en tu app. Para obtener información de referencia detallada, consulta la clase View. (developer.android, n.d.)

¿Qué son los Layouts y Cuáles Existen en Android Studio?

"Cuando trabajas en Android Studio necesitas definir los Layouts sobre los cuales reposarán los elementos de tu aplicación, como los botones, Recycler View, Image View, Elementos de Formularios, etc. Debes saber que Layout es el adecuado para que tu aplicación este bien organizada y no presente errores cuando tu aplicación esté funcionando, en este artículo te vamos a explicar que son los Layouts en Android Studio y para qué sirve cada uno" (nubecolectiva, 2018)

¿Qué son los Layouts en Android Studio?

"Son un conjunto de contenedores en donde podemos colocar muchos elementos según el diseño de nuestra aplicación, por ejemplo, dentro de un Layout puedes colocar, botones, imágenes, formularios, textos, etc. todos estos deben tener un orden y armonía entre si dentro del diseño, para esto debemos haber elegido un Layout primero para luego colocar dentro de este los elementos que necesitemos." (nubecolectiva, 2018)

Tipos de Layouts ConstraintLayout

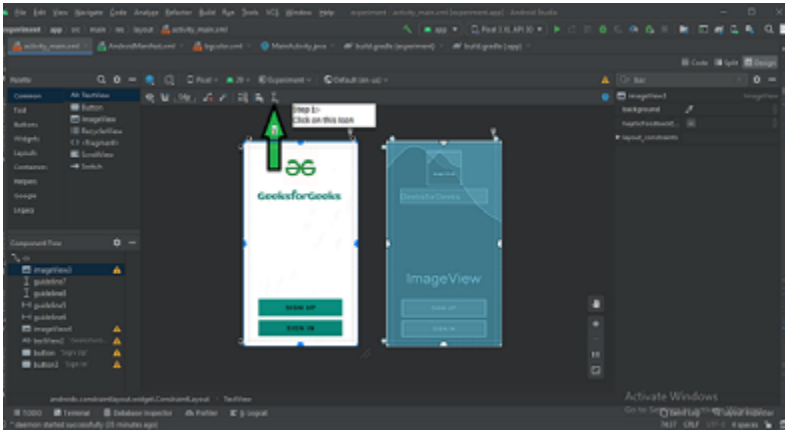
Fue lanzado en Android Studio 2 junto con el Layout Editor y nos permite trabajar sin grupos de vistas anidadas similar a Relative Layout, este Layout nos ayuda mucho a

diseñar interfaces de nuestra aplicación sin tener que depender de muchos Layouts como Linear, Frame, Table o Grid Layout. (nubecolectiva, 2018)

Este Layout es útil para trabajar con Grandes grupos de elementos porque nos crea un orden jerárquico dinámico, podemos agregar elementos adicionales y darles restricciones para que cuando muevas la posición de la pantalla del móvil, este mantenga en su lugar los elementos de una vista, a continuación, giramos la pantalla de Portrait (Vista Normal) a Landscape (Vista de Costado) (nubecolectiva, 2018)

Figura 13

Ejemplo de ConstraintLayout



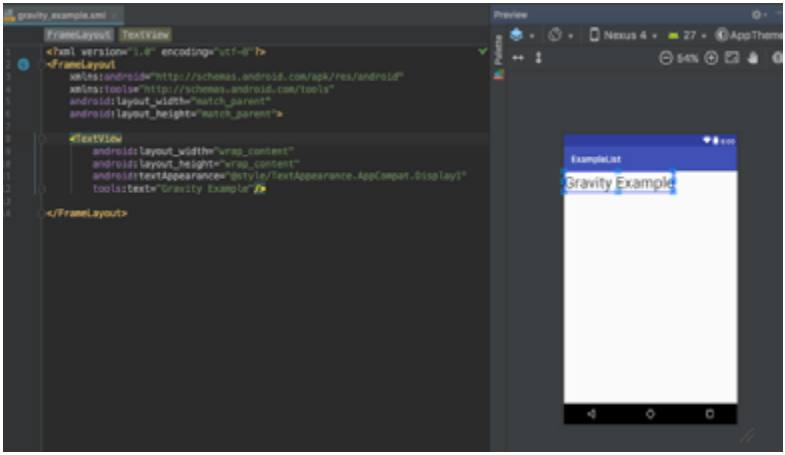
Fuente: Tomado de (https://www.geeksforgeeks.org/, 2021)
Nota: La imagen representa el diseño de una Actividad en una App. A través del código XML de un proyecto con Constraint-Layout.

FrameLayout

Este Layout alinea todos los elementos de tu diseño (botones, Image View, etc.) al lado izquierdo, podemos aplicar márgenes para mostrar un elemento en un punto específico de la pantalla, puedes agregar varios elementos a tu diseño, pero mantenido el orden de los elementos par que tu diseño se vea agradable. (nubecolectiva, 2018)

Figura 14

Estructura de un FrameLayout



Fuente: Tomado de (Budzar, 2017)

Nota: La imagen representa el diseño de una Actividad en una Aplicación móvil empleando código XML de un proyecto Android usando FrameLayout.

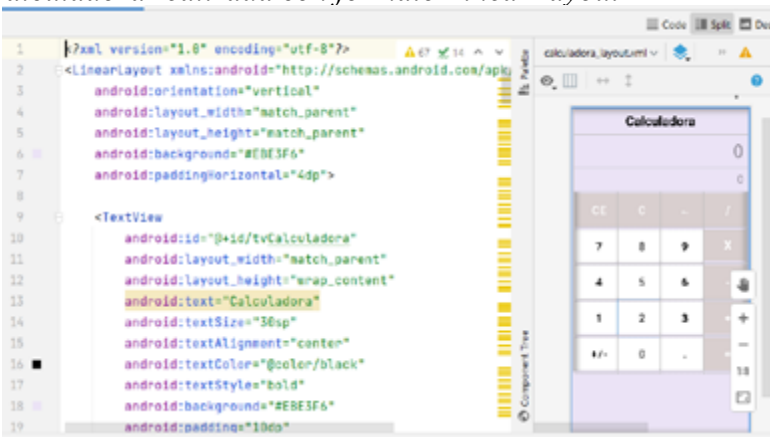
LinearLayout

Se utiliza para alinear uno o varios elementos de tu diseño de forma vertical u horizontal, la orientación lo definimos con el atributo **android:orientation**=”horizontal”

este tipo de Layout los podemos aplicar a los elementos que deben de centrarse en su totalidad, por ejemplo, un formulario de contacto o una imagen con un botón, una orientación general a todos los elementos de tu vista. (nubecolectiva, 2018)

Figura 15

Calculadora realizada con formato LinearLayout



Fuente: Tomado de Imagen propiedad del Autor Edison Usca, 2023 a través de IDE Android Studio.

Nota: La imagen representa el diseño de una Actividad en una App. A través del código XML de un proyecto con LinearLayout.

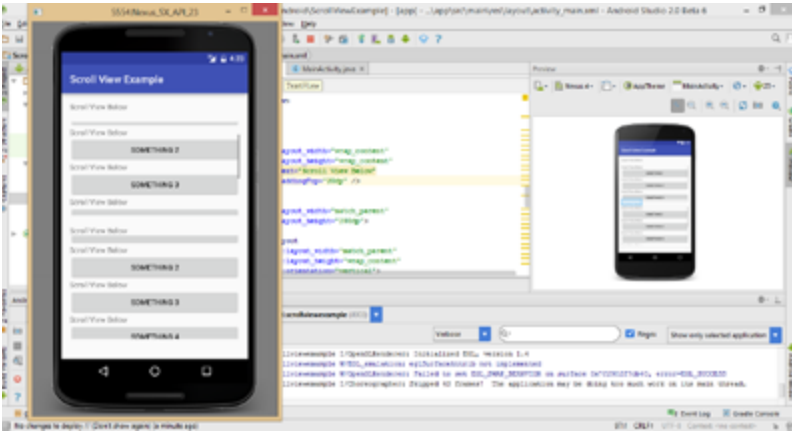
ScrollView

Aunque no se considera un tipo de Layout, se puede disponer de este tipo de organizador de Vistas, ya que permite incluir elementos en una vista principal agregando al final una barra de progreso Scroll, la cual permite al usuario avanzar en

la inclusión de elementos que no se pueden visualizar en una sola pantalla. Cabe recalcar que este tipo de contenedores de elementos solo permite la inclusión de un solo elemento por lo que se recomienda incluir un layout o contenedor propiamente dicho, de preferencia un LinearLayout tal como lo muestra la figura 21.

Figura 16

Implementación de un ScrollView



Fuente: Tomado de (blogspot.com, 2017)

Nota: La imagen representa el diseño de una Actividad en una App. A través del código XML de un proyecto con Scroll-View.

Capítulo 7

Controles Básicos Utilizados en Android

Los controles básicos de Android nos permitirán diseñar aplicaciones simples de manera rápida. Este grupo de controles básicos está compuesto por los TextView, EditText, CheckBox, RadioButton, Button, ToggleButton, ImageButton e ImageView. A continuación, los describiremos brevemente y comentaremos los parámetros principales más importantes de cada uno. (Canovas, 2014)

TextView

Este control es uno de los más usados y sencillos de utilizar, ya que su única función es la mostrar texto en pantalla. Sus parámetros principales son:

• android:text; Permite establecer un texto determinado, el cual se mostrará en pantalla.

• android:background; Permite establecer el color de fondo del textview.

• android:textColor; Permite establecer el color del texto. Deben especificarse en hexadecimal.

• android:textSize; Permite establecer el tamaño del texto. Se recomienda utilizar la unidad de medida sp.

• android:textface; Permite establecer el estilo del texto; negrita (bold) o cursiva (italic).

La manera de añadir texto a un textview mediante código es con el método *setText()*.

EditText

Este control es la típica caja de texto que se emplea para la introducción de datos por teclado. Para la captura de dichos datos se utiliza el método *getText()*, mientras que para establecer texto puede utilizarse el método *setText()*. Podemos destacar dos parámetros de EditText; *android:text*, que es el parámetro de diseño que permite establecer texto (aunque no suele utilizarse mucho) y *android:hint*, que sirve para establecer un texto semitransparente dentro de la caja de texto que desaparece cuando se hace clic con el ratón dentro de ella. (Canovas, 2014)

CheckBox

Los checkboxes son casillas que se pueden marcar o desmarcar y que se utilizan para indicar que se ha seleccionado o deseleccionado una opción. Por lo tanto, un checkbox tiene dos estados; true cuando está marcado y false cuando no lo está. Para saber el estado en que se encuentra un checkbox se utiliza el método booleano *isChecked()*. Con el método *setChecked(boolean)* se puede establecer cualquiera de los dos estados mediante código. (Canovas, 2014)

El evento más importante de un checkbox es *onCheckedChanged(Compound checkview, boolean isChecked)*, y permite capturar el momento en que éste cambia de estado. Compound checkview hace referencia al checkbox en sí. (Canovas, 2014)

RadioButton

Los RadioButton tienen un funcionamiento similar a los checkboxes, sin embargo, su principal diferencia es que se agrupan en los llamados RadioGroups y que en un mismo grupo sólo puede haber un RadioButton seleccionado. El resto se desmarcan automáticamente. Con el parámetro android:text podemos asignar texto a cada uno de los RadioButtons. (Canovas, 2014)

El evento más importante de un RadioButton es *onCheckedChanged (RadioGroup group, int checkedId)*, y permite capturar el momento en que se produce un cambio de selección en un RadioGroup. Este tipo de control básico tiene varios métodos que se pueden usar en la parte del código, como por ejemplo *clearCheck(),* que establece que no haya ningún RadioButton seleccionado por defecto. (Canovas, 2014)

Button

Este control es un simple botón. Su evento más importante es *onClick(View v)* el cual se ejecuta cuando se pulsa sobre el botón. (Canovas, 2014)

ImageButton

Tiene la misma funcionalidad que Button. La diferencia respecto a éste es que se le puede asignar una imagen cubriendo la superficie del ImageButton. Para ello es necesario utilizar

el parámetro *android:src* en el área de diseño. Para iniciar un recurso de imagen se suele utilizar la ruta /res/drawable, donde hay almacenadas imágenes predefinidas de Android. (Canovas, 2014)

ImageView

Este control no es más que una imagen establecida en una view. No tiene funcionalidad ninguna, salvo que dicho control se relacione con otro y en cuanto se capture un evento concreto se quiera cambiar la imagen del ImageView. Para establecer la imagen se utiliza el parámetro android:src en el área de diseño. (Canovas, 2014)

Capítulo 8

Uso de Archivos XML

XML significa Lenguaje de etiquetas extensible. XML es un lenguaje de etiquetas muy parecido al **HTML** utilizado para describir datos. Las etiquetas XML no están predefinidas en XML. Debemos definir nuestras propias etiquetas. XML ya que es bien legible tanto por humanos como por máquinas. Además, es escalable y simple de desarrollar. En Android usamos XML para diseñar nuestros diseños porque XML es un lenguaje liviano por lo que no hace que nuestro diseño sea pesado. (ANDROFAST, 2017)

Conceptos Básicos de la Interfaz de Usuario

El concepto completo de la interfaz de usuario de **Android** se define utilizando la jerarquía de los objetos **View** y **ViewGroup**. Un **ViewGroup** es un contenedor invisible que organiza vistas secundarias. Estas vistas secundarias son otros **widgets** que se utilizan para crear las diferentes partes de la **IU**. Un ViewGroup puede tener otro ViewGroup como elemento secundario como se muestra en la imagen que se muestra a continuación: (ANDROFAST, 2017)

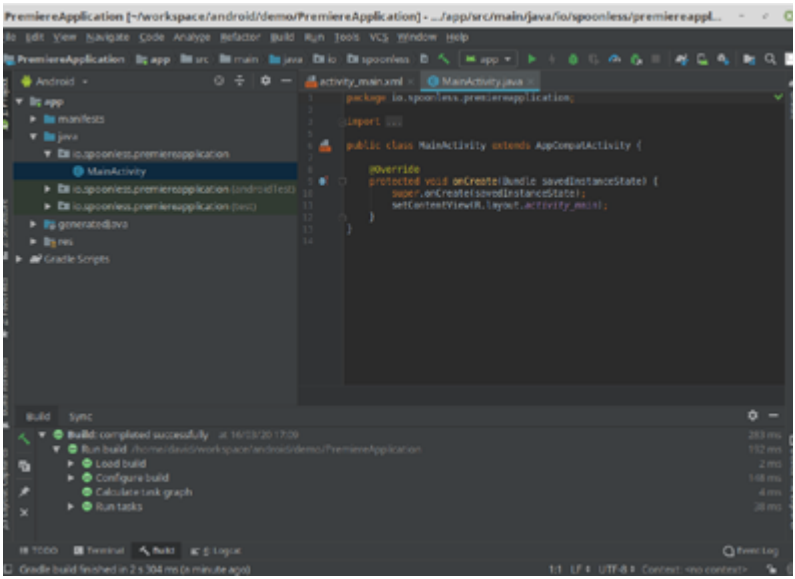
Diferentes Archivos XML Usados en Android

En **Android** hay diferentes tipos de archivos **XML** utilizados para varios propósitos diferentes. A continuación, definiremos todos y cada uno de ellos (ANDROFAST, 2017)

Layout XML: Los archivos **XML** de diseño se utilizan para definir la interfaz de usuario real de nuestra aplicación. Contiene todos los elementos (vistas) o las herramientas que queremos usar en nuestra aplicación. Al igual que los Text-View, los botones y otros elementos de la interfaz de usuario. (ANDROFAST, 2017)

Figura 17

Estructura del activity_main



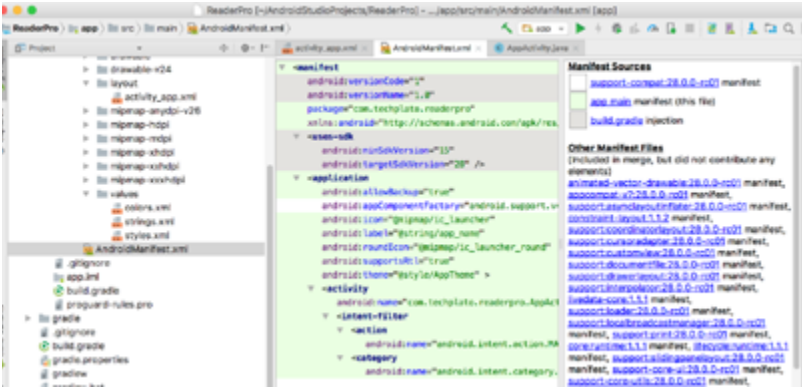
Fuente: Tomado de (https://gayerie.dev/, s.f.)
Nota: La imagen representa la estructura de un proyecto y la ubicación del archivo activity_main, que contiene el diseño de la pantalla de una APP en Android Studio.

Manifest XML: Este archivo XML se usa para definir todos los componentes de nuestra aplicación. Incluye los nom-

bres de nuestros paquetes de la app, nuestros Activitys, receptores, servicios y los permisos que nuestra aplicación necesita. Por ejemplo: supongamos que necesitamos usar la cámara de nuestro dispositivo, entonces necesitamos definir el permiso del uso de la cámara en nuestro **Manifest**. (ANDROFAST, 2017)

Figura 18

Estructura del AndroidManifest.xml



Fuente: Tomado de (imgur.com, 2019)

Nota: La imagen representa la estructura de un proyecto y la ubicación del archivo AndroidManifest, que contiene la configuración a gran escala de una APP en Android Studio.

• Strings XML: Este archivo XML se usa para reemplazar las cadenas codificadas con una sola cadena. Definimos todas las cadenas en este archivo XML y luego accedemos a ellas desde nuestra aplicación desde este archivo. Este archivo nos permite mejorar la reutilización del código. (ANDROFAST, 2017)

• Styles XML: Este XML se usa para definir diferentes estilos de la aplicación. Definimos nuestros temas y estilos personalizados en este archivo. (ANDROFAST, 2017)

Figura 19

Estructura del archivo Styles.XML

Fuente: Tomado de (https://stackoverflow.com/, 2014)
Nota: La imagen representa la estructura archivo styles.XML, contiene uno de los recursos que es el diseño del estilo de una APP en Android Studio.

• Color XML: Este archivo se usa para definir los códigos de color que usamos en nuestra aplicación. Simplemente definimos los colores en este archivo y los usamos en nuestra aplicación, por ejemplo, cambiar el fondo de nuestra app o darle el color que elijamos algún texto de nuestra aplicación. (ANDROFAST, 2017)

Uso de la Colección en Android

"Cuando trabajaba en un proyecto de Android recientemente, encontré algunos problemas de clasificación de colecciones. También he usado algunos métodos de Colecciones antes." (programmerclick, n.d.)

Por ejemplo, Collections.sort (), Collections.swap () ..., por lo que algunas funciones comunes de las colecciones están unificadas. (programmerclick, n.d.)

1).java.util.Collection es una interfaz de colección (una interfaz de nivel superior de la clase de colección). Proporciona métodos de interfaz comunes para operaciones básicas en objetos de colección. La interfaz Collection tiene muchas implementaciones concretas en la biblioteca de clases de Java. El significado de la interfaz Colección es proporcionar un método de operación unificado maximizado para varias colecciones específicas, y sus interfaces de herencia directa incluyen Lista y Conjunto. (programmerclick, n.d.)

2) collections es una clase de herramienta de la clase de colección, que proporciona una serie de métodos estáticos para ordenar, buscar y operaciones seguras para subprocesos de los elementos de la colección. (programmerclick, n.d.)

Ordenar (ordenar)

Aunque ya existen muchos algoritmos de clasificación (clasificación de burbujas, clasificación de selección, clasificación de inserción, etc.), aquí hay un método proporcionado por el marco de Colecciones en Java. (programmerclick, n.d.)

Con este método podemos implementar fácilmente muchas funciones, solo una línea de código, (programmerclick, n.d.)

Como utilizar: Collections.sort(lista) (programmerclick, n.d.)

GridView

Un GridView es un tipo de *layout* (diseño) que nos muestra una serie de elementos en forma de tabla y que podemos deslizar verticalmente. El GridView mostrará una cuadrícula de vistas, cada una de ellas puede representar diversos elementos. Estas vistas se mostrarán en forma de cuadrícula, automáticamente *scrolleable* arriba y abajo. (Universidad Politécnica de Valencia, n.d.)

Para definir un GridView se requieren cuatro elementos. Para empezar, deberemos introducir el GridView en un *layout* para mostrarlo al usuario. En segundo lugar, necesitaremos una actividad que lo muestre y lo controle. Esta actividad también será la que recibirá los eventos que el GridView genere. En tercer lugar, precisaremos un adaptador, definido en otra clase, que rellene cada elemento del GridView, introduciendo la información necesaria en cada uno. Por último, necesitaremos un *layout* para los ítems del GridView. El adaptador utilizará este diseño y lo modificará con la información de cada elemento. (Universidad Politécnica de Valencia, n.d.)

El uso de un GridView es muy parecido al de un ListView. Puedes encontrar información sobre el uso de

ListView en androidcurso.com en el tercer Unidad de El gran libro de Android. (Universidad Politécnica de Valencia, n.d.)

Uso de MAPVIEW

El SDK de Maps para Android proporciona varias clases que tu app puede usar para administrar el ciclo de vida, la funcionalidad y los datos de un mapa. Las clases admiten interacciones de los usuarios según el modelo de IU de Android, como configurar el estado inicial del mapa y responder a la entrada de gestos de los usuarios en el tiempo de ejecución. (developers.google.com, n.d.)

A continuación, se incluyen la interfaz y las clases principales para controlar los mapas:

GoogleMap: Es el punto de entrada para administrar las funciones y los datos del mapa subyacente. Tu app solo puede acceder a un objeto GoogleMap si este se recupera desde un objeto SupportMapFragment o MapView. (developers.google.com, n.d.)

SupportMapFragment: Es un fragmento que permite administrar el ciclo de vida de un objeto GoogleMap. (developers.google.com, s.f.)

MapView: Es una vista que permite administrar el ciclo de vida de un objeto GoogleMap. (developers.google.com, s.f.)

OnMapReadyCallback: Es una interfaz de devolución de llamada que controla los eventos y las interacciones de los usuarios del objeto GoogleMap. (developers.google.com, s.f.)

Un objeto GoogleMap realiza estas operaciones automáticamente:

- Conexión al servicio de Google Maps
- Descarga de mosaicos de mapa
- Visualización de mosaicos en la pantalla del dispositivo
- Visualización de varios controles, como el desplazamiento lateral y el zoom
- Respuesta a gestos de desplazamiento lateral y zoom a través del movimiento del mapa y su acercamiento o alejamiento

Para utilizar un objeto GoogleMap en tu app, debes usar un objeto SupportMapFragment o MapView como contenedor del mapa, y luego recuperar el objeto GoogleMap del contenedor. Dado que las clases de contenedor se derivan de un fragmento o una vista de Android, proporcionan al mapa las funciones de IU y administración del ciclo de vida de sus clases base de Android. La clase SupportMapFragment es el contenedor más común y moderno para un objeto GoogleMap. (developers.google.com, n.d.)

Referencias Bibliográficas

Referencias Bibliográficas

Androfast. (27 de noviembre de 2017). androfast.com. Obtenido de https://www.androfast.com/2017/11/xml-en-android-conceptos-basicos-y.html

Android. (s.f.). android.com. Obtenido de https://developer.android.com/kotlin/faq?hl=es

Android Studio. (13 de diciembre de 2022). ARCHIVO activity_main.xml diseño de una Vista para una APP en Android Studio.

Android Studio. (13 de diciembre de 2022). Código de Build. Gradle en proyecto de Android Studio.

Android Studio. (13 de Diciembre de 2022). Estructura de la clase MainActivity.kt en lenguaje Kotlin.

Android Studio. (13 de diciembre de 2022). Estructura del archivo AndroidManifest.xml.

Aprendeit. (12 de octubre de 2018). aprendeit.com. Obtenido de https://aprendeit.com/como-instalar-centos-y-entorno-grafico/

Bittencourt, J. (17 de febrero de 2022). Obtenido de https://www.alura.com.br/artigos/activity-lifecycle-por-que-conhecer-ciclo-de-vida-activity

blogspot.com. (2017). Obtenido de https://1.bp.blogspot.com/-Sd-55-RMEcQ/VvO1h_NpXcI/AAAAAAAABk4/jRZKXWIqneIBGMOPDUZQZQSaXK-tyX9Iw/s1600/android%2Bmultiple%2Bscroll%2Bview%2Bapp%2Bscreenshot.png

Budzar, M. (15 de noviembre de 2017). Obtenido de https://proandroiddev.com/android-developer-beginner-faq-2-gravity-vs-layout-gravity-3c461bdeae12

Cabot, J. (31 de mayo de 2017). https://ingenieriadesoftware.es/. Obtenido de https://ingenieriadesoftware.es/kotlin-java-mejorado/#:~:text=Kotlin%20fue%20creado%20en%202010,a%20crear%20un%20nuevo%20lenguaje.

Canovas, J. J. (19 de octubre de 2014). juanjosecanbus.com. Obtenido de https://juanjosecanbus.wordpress.com/2014/10/19/controles-basicos-utilizados-en-android/

delfstack.com. (s.f.). delfstack.com. Obtenido de https://www.delftstack.com/es/howto/linux/how-to-install-rpm-pacakges-on-centos/

developer.android. (2022). Obtenido de https://developer.android.com/studio?gclid=Cj0KCQiA54KfBhCKARIsAJzSrdq1mhEI6KjISVVbNMDEhlNejbPgKDDvbSkzsWR7coiMLlIrQOHOh8saApB_EALw_wcB&gclsrc=aw.ds

developer.android. (s.f.). developer.android.com. Obtenido de https://developer.android.com/guide/topics/ui/custom-components?hl=es

developer.android.com/codelabs. (2017). Obtenido de https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio?hl=es-419#2

developer.android.com/studio. (1 de agosto de 2022). Obtenido de https://developer.android.com/studio/run/managing-avds?hl=es-419

developer.android.com/training. (2022). Obtenido de https://developer.android.com/training/basics/firstapp/creating-project?hl=es-419

developers.google.com. (s.f.). developers.google.com. Obtenido de https://developers.google.com/maps/documenta-

tion/android-sdk/map?hl=es-419

DOCENCIA.UDEA. (2008). docencia.usea.edu.ec. Obtenido de http://docencia.udea.edu.co/cci/linux/dia1/introduccion.htm

filehorse.com. (25 de noviembre de 2022). Obtenido de https://www.filehorse.com/es/descargar-intel-haxm/

Google. (s.f.). google.com. Obtenido de https://sites.google.com/site/aplicacionesiraismoviles/sistema-operativo-android-fundamentos-cap-1

Google. (s.f.). sites.google.com. Obtenido de https://sites.google.com/site/swcuc3m/home/android/api

Hendrix. (10 de octubre de 2019). YO SOY HENDRIX. Obtenido de https://yosoyhendrix.com/2019/10/10/como-actualizar-el-kernel-de-linux-en-centos-7/

https://gayerie.dev/. (s.f.). Obtenido de https://gayerie.dev/docs/java-android/android/premiere_app.html

https://stackoverflow.com/. (14 de agosto de 2014). Obtenido de https://stackoverflow.com/questions/25265328/why-does-android-studio-highlight-theme-red-in-styles-xml

https://www.geeksforgeeks.org/. (13 de septiembre de 2021). Obtenido de https://www.geeksforgeeks.org/guidelines-in-android-constraintlayout/

imgur.com. (2019). Obtenido de https://i.stack.imgur.com/sqW54.png

kotlindoc.blogspot.com. (28 de mayo de 2019). Obtenido de https://kotlindoc.blogspot.com/2019/05/hola-mundo-con-android-studio.html

Medina, E. (20 de marzo de 2018). MUYSEGURIDAD. NET. Obtenido de https://www.muyseguridad.

net/2018/03/20/editores-texto-linux-unix-problemas-seguridad/

Microsoft.com. (s.f.). Obtenido de https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/configuring-iis-for-wcf

Morales, R. (8 de Mayo de 2013). TICARTE.COM. Obtenido de http://wmw.ticarte.com/contenido/comandos-para-el-manejo-de-directorios-y-archivos

nubecolectiva. (15 de agosto de 2018). nubecolectiva.com. Obtenido de https://blog.nubecolectiva.com/que-son-los-layouts-y-cuales-existen-en-android-studio/

programmerclick. (s.f.). programmerclick.com. Obtenido de https://programmerclick.com/article/91151871750/

TECNOLOGIAINFORMATICA. (s.f.). tecnologia-informatica.com/. Obtenido de https://tecnologia-informatica.com/el-sistema-operativo/

Universidad Politécnica de Valencia. (2017). androidcurso.com. Obtenido de http://www.androidcurso.com/index.php/tutoriales-android/37-unidad-6-multimedia-y-ciclo-de-vida/158-ciclo-de-vida-de-una-actividad

Universidad Politécnica de Valencia. (s.f.). androidcurso.com. Obtenido de http://www.androidcurso.com/index.php/51-tutoriales/arquitectura-y-diseno-avanzado-iu/unidad-1-diseno-avanzado-de-interfaces-de-usuario/358-gridview

Vaati, E. (30 de junio de 2020). tutsplus.com. Obtenido de https://code.tutsplus.com/es/tutorials/the-simplest-android-app-hello-world--cms-35175

Ventura, P. (12 de diciembre de 2010). pedroventura.com. Obtenido de https://www.pedroventura.com/linux/administracion-de-usuarios-en-linux-crear-borrar-modifi-

car-usuarios-y-grupos/

Yuri Blaise, P. M. (15 de septiembre de 2022). Obtenido de https://android-developers.googleblog.com/2022/09/android-studio-dolphin.html

Evaluación
de pares

Quito, 15 de mayo de 2023

Señor Magister
Renato Esteban Revelo Oña
Presidente "R2ICS"
Presente,

De mi consideración:

Una vez realizada la valoración metodológica para la **evaluación par** de la obra **Desarrollo de Aplicaciones Android 1,** del autor: Ing. Edison Javier Usca Paguay; se certifica que la obra cumple con los criterios de relevancia y pertinencia que especifica el respectivo reglamento de Educación Superior, por lo tanto, la misma es recomendable como una **OBRA RELEVANTE**.

Cabe indicar que los contenidos cumplen con estándares de calidad para los procesos de enseñanza y aprendizaje, es inédita y contribuyen al conocimiento y formación de los estudiantes universitarios, de tal manera que resultan fundamentales y sustanciales en la Educación Superior.

Informe que pongo a vuestra consideración para los fines pertinentes.

Atentamente,

Mgtr. Luis Alfonso Yela Lascano
CC: 1707782197
Especialista Sistemas Informáticos & Auditoría
CONTRALORIA GENERAL DEL ESTADO

r2ics
Red Internacional de Investigación en Ciencias Sociales y Humanidades

CERTIFICADO DE REVISIÓN METODOLÓGICA (PAR EVALUADOR)

Quito, 03 de mayo de 2023

Una vez realizada la valoración metodológica para la **evaluación par** de la obra **Desarrollo de Aplicaciones Android 1,** del autor: Ing. Edison Javier Usca Paguay; se certifica que la obra cumple con los criterios de relevancia y pertinencia que especifica el respectivo reglamento de Educación Superior, por lo tanto, la misma es recomendable como una **OBRA RELEVANTE**.

Cabe indicar que los contenidos cumplen con estándares de calidad para los procesos de enseñanza y aprendizaje, es inédita y contribuyen al conocimiento y formación de los estudiantes universitarios, de tal manera que resultan fundamentales y sustanciales en la Educación Superior.

Atentamente,

Mgtr. Anita Lucía Mata Velasteguí
CC: 1712685831
Revisor Metodológico

www.euroamericano.edu.ec