



UTPL
La Universidad Católica de Loja

Modalidad Abierta y a Distancia



Itinerario 2: Desarrollo Basado en Plataformas Móviles

Guía didáctica

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Departamento de Ciencias de la Computación y Electrónica

Sección departamental Tecnologías Avanzadas de la Web y SBC

Itinerario 2: Desarrollo Basado en Plataformas Móviles

Guía didáctica

Autor:

Ramírez Coronel Ramiro Leonardo



Asesoría virtual
www.utpl.edu.ec

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Itinerario 2: Desarrollo Basado en Plataformas Móviles

Guía didáctica

Ramírez Coronel Ramiro Leonardo

Universidad Técnica Particular de Loja



Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojainfo@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-39-162-9



La versión digital ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite: copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

29 de Marzo, 2021

Índice

1. Datos de información.....	8
1.1. Presentación de la asignatura	8
1.2. Competencias genéricas de la UTPL.....	8
1.3. Competencias específicas de la carrera.....	8
1.4. Problemática que aborda la asignatura.....	9
2. Metodología de aprendizaje.....	9
3. Orientaciones didácticas por resultados de aprendizaje.....	10
Primer bimestre	10
Resultado de aprendizaje	10
Contenidos, recursos y actividades de aprendizaje	10
Semana 1	11
Unidad 1. Plataformas y metodologías de desarrollo de aplicaciones móviles.....	11
1.1. Conceptos	12
1.2. Plataformas móviles	15
1.3. Entornos de desarrollo	16
1.4. Metodologías de desarrollo para aplicaciones móviles ..	16
Autoevaluación 1	22
Semana 2	25
Unidad 2. Tipos de aplicaciones y almacenamiento de información en dispositivos móviles.....	25
2.1. Aplicaciones nativas	25
2.2. Aplicaciones móviles híbridas	26
2.3. Tecnologías o framework para las aplicaciones móviles híbridas.....	27
Semana 3	31

Índice	
Primer bimestre	
Segundo bimestre	
Solucionario	
Referencias bibliográficas	
2.4. Bases de datos en dispositivos móviles.....	31
2.5. Gestión de archivos en dispositivos móviles.....	33
Actividades de aprendizaje recomendadas	34
Autoevaluación 2	35
Resultado de aprendizaje	38
Contenidos, recursos y actividades de aprendizaje	38
Semana 4	38
Unidad 3. Aplicaciones móviles en Android Studio. Parte 1.....	38
3.1. Conceptos	38
3.2. Estructura del proyecto	39
3.3. Instalación.....	42
Semana 5	43
3.4. Entorno de desarrollo	44
3.5. Flujo de trabajo de desarrollo en Android Studio	44
3.6. Desarrollo de un proyecto	47
Semana 6	52
3.7. Cómo ejecutar su app	53
Actividades de aprendizaje recomendadas	56
Autoevaluación 3	58
Actividades finales del bimestre.....	61
Semana 7	61
Semana 8	62
Segundo bimestre	62
Resultado de aprendizaje	62

Contenidos, recursos y actividades de aprendizaje	62
Semana 9	62
Unidad 4. Aplicaciones móviles con Android Studio. Parte 2.....	63
4.1. Construcción del FrontEnd.....	63
4.2. Editor de diseño	65
4.3. Cuadro de texto	68
4.4. Botones	69
4.5. Tamaño flexible del cuadro.....	70
4.6. Interacción de actividades	73
4.7. Intent	75
4.8. Actividad secundaria.....	77
4.9. Vistas de texto en la app.....	77
4.10. Mensajes.....	79
4.11. Navegación ascendente.....	80
Semana 10	81
4.12. Aspectos fundamentales de la aplicación.....	81
4.13. Componentes de la aplicación	83
4.14. Actividades	84
4.15. Servicios.....	85
4.16. Activación de componentes	86
4.17. El archivo de manifiesto.....	88
4.18. Declaración de componentes.....	89
Autoevaluación 4	92
Semana 11	95
Unidad 5. Aplicaciones móviles con Flutter.....	95
5.1. ¿Qué es Flutter?	95
5.2. Características.....	96
5.3. Instalación.....	97
Semana 12	102

5.4. Primer proyecto en Flutter	102
5.5. Emulador	103
5.6. Estructura del proyecto	103
5.7. Aplicación en Flutter.....	104
5.8. Integración y consumo de Apis con Flutter	107
Autoevaluación 5	113
Semana 13	116
Unidad 6. Puesta en marcha de una aplicación móvil.....	116
6.1. Publicación de la aplicación móvil en tiendas oficiales ..	117
6.2. El proceso de aprobación	119
Semana 14	120
6.3. Después del lanzamiento.....	120
6.4. Comentarios de los usuarios.....	121
6.5. Las actualizaciones.....	121
Actividades de aprendizaje recomendadas	123
Autoevaluación 6	124
Actividades finales del bimestre.....	127
Semana 15	127
Semana 16	127
4. Solucionario	129
5. Referencias bibliográficas	135

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Índice

Primer bimestre

Segundo bimestre

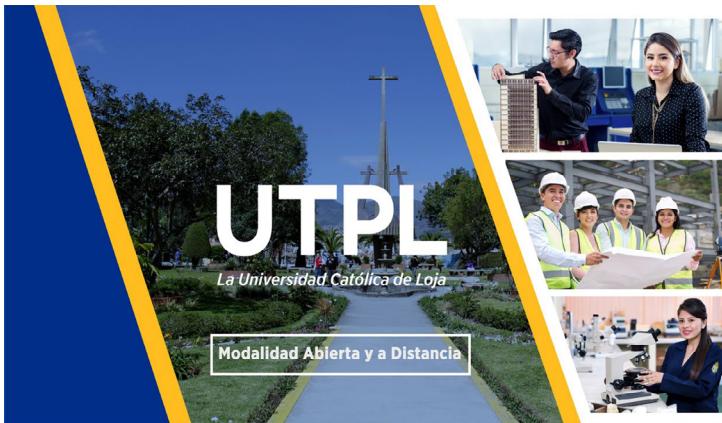
Solucionario

Referencias bibliográficas



1. Datos de información

1.1. Presentación de la asignatura



1.2. Competencias genéricas de la UTPL

- Comunicación en inglés.

1.3. Competencias específicas de la carrera

Implementar aplicaciones de baja, mediana y alta complejidad integrando diferentes herramientas y plataformas para dar solución a requerimientos de la organización.

1.4. Problemática que aborda la asignatura

Se identifica como *tensión* la escasez de conocimientos, propuestas o soluciones ante proyectos relacionados a tecnologías móviles; la falta de conocimiento en el proceso de análisis, construcción e implementación de apps móviles. La selección de la plataforma correcta ante una problemática tecnológica móvil es una debilidad muy común. Es por ello que se establecen estos temas de vital importancia en el transcurso de la materia.



2. Metodología de aprendizaje

La metodología de aprendizaje que se va a usar en el desarrollo de las actividades de la asignatura de desarrollo basado en plataformas móviles es una metodología de estudio de casos, el cual se caracteriza por precisar de un proceso de búsqueda e indagación, así como el análisis sistemático de uno o varios casos.

Para ser más exactos, por caso entendemos todas aquellas circunstancias, situaciones o fenómenos únicos de los que se requiere más información o merecen algún tipo de interés dentro del mundo de la investigación. (Psicología y Mente, 2021)

Para lograr este objetivo en nuestra materia, usted, como estudiante, tiene que desarrollar y poner en práctica los conocimientos adquiridos en un problema planteado, estos casos se irán detallando conforme va avanzando en los contenidos de la materia.



3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultado de aprendizaje

Compara y contrasta la programación móvil con la programación de propósito general.

En el transcurso de esta semana usted podrá revisar los contenidos de esta unidad, los recursos que se provee en la misma para comprender fácilmente sobre los conceptos del desarrollo móviles, los tipos de aplicaciones que existen y cómo proponer soluciones ante algún problema suscitado.

Para lograr el resultado de aprendizaje tiene que revisar los conceptos y contenidos de esta sección para poder determinar y comprender la diferencia que existe entre el desarrollo de aplicaciones generales y aplicaciones móviles.

Contenidos, recursos y actividades de aprendizaje

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Semana 1



Unidad 1. Plataformas y metodologías de desarrollo de aplicaciones móviles

Estimado estudiante, iniciamos el estudio de la asignatura mencionando temas fundamentales que se requiere conocer dentro del maravilloso mundo del desarrollo basado en plataformas móviles; comenzamos revisando algunos conceptos básicos de desarrollo de aplicaciones móviles, lenguajes de programación de esta área, entornos de desarrollo; esto le ayudará a comprender lo importante de este tema.

Se revisará conceptos para comprender la diferencia entre aplicaciones móviles web, aplicaciones nativas y aplicaciones híbridas, esto con el fin de poder entender las fortalezas de cada una y proponer soluciones móviles acorde a las necesidades de la problemática a resolver.

1.1. Conceptos

Vamos a revisar conceptos que le ayudaran a entender de mejor manera este mundo del desarrollo basado en plataformas móviles. ¡Mucho suerte!

¿Qué son las aplicaciones móviles?

Las aplicaciones también llamadas apps están presentes en los teléfonos desde hace tiempo; de hecho, ya estaban incluidas en los sistemas operativos de Nokia o Blackberry años atrás. Los móviles de esa época contaban con pantallas reducidas y muchas veces no táctiles, y son los que ahora llamamos feature phones, en contraposición a los smartphones, más actuales. En conclusión, una aplicación no deja de ser un software. Para entender un poco mejor el concepto, podemos decir que las aplicaciones son para los móviles lo que los programas son para los ordenadores de escritorio. Actualmente encontramos aplicaciones de todo tipo, forma y color, pero en los primeros teléfonos, estaban enfocadas en mejorar la productividad personal: se trataba de alarmas, calendarios, calculadoras y clientes de correo. Hubo un cambio grande con el ingreso de iPhone al mercado, ya que con él se generaron nuevos modelos de negocio que hicieron de las aplicaciones algo rentable, tanto para desarrolladores como para los mercados de aplicaciones, como App Store, Google Play y Windows Phone Store.

Al mismo tiempo, evolucionaron las herramientas tanto de diseño como las de desarrollo de aplicaciones móviles, facilitando la tarea de producir una aplicación y lanzarla al mercado, incluso por cuenta propia. (Cuello Javier, 2013)

Diferencias entre aplicaciones móviles y web móviles

Las aplicaciones comparten la pantalla del teléfono con las webs móviles, pero mientras las primeras tienen que ser descargadas e instaladas antes de usar, a una web puede accederse simplemente

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

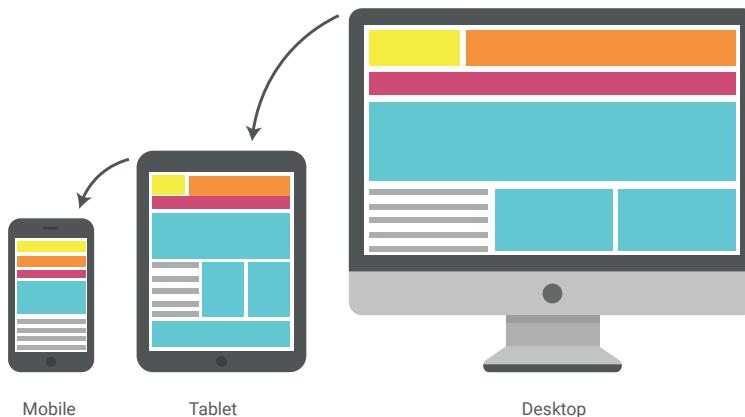
usando internet y un navegador; sin embargo, no todas pueden verse correctamente desde una pantalla generalmente más pequeña que la de un ordenador de escritorio. Las que se adaptan especialmente a un dispositivo móvil se llaman “web responsivas” y son ejemplo del diseño líquido, ya que se puede pensar en ellas como un contenido que toma la forma del contenedor, mostrando la información según sea necesario. Así, columnas enteras, bloques de texto y gráficos.

¿Qué es diseño web fluido o líquido?

Tal como se muestra en la figura 1.1 el diseño web fluido o líquido es aquel que tiene la propiedad de adaptarse al ancho de cualquier pantalla o dispositivo sin afectar la estética visual y funcional de la aplicación, para lograr este tipo de aplicaciones requiere de más tiempo de desarrollo y el uso de tecnologías como HTML3 y CSS3.

Una técnica para lograr un diseño fluido es utilizar la unidad de medida del ancho del elemento de la aplicación web en porcentajes tal como se muestra figura 1.2, este control se lo realiza en la hoja de estilos CSS.

Figura 1.1
Aplicaciones web responsivas



Nota: Adaptación de una aplicación web en diferentes dispositivos. Tomado de Aplicaciones Responsivas [Ilustración], 2020.

Índice

Primer bimestre

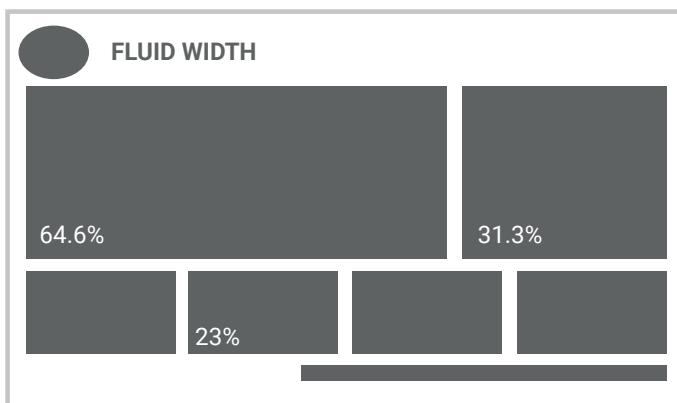
Segundo bimestre

Solucionario

Referencias bibliográficas

Una aplicación web puede acomodarse en el espacio de una manera diferente o incluso desaparecer de acuerdo a si se entra desde un teléfono, una tableta o un ordenador. Quienes cuentan ya con una web responsive pueden plantearse la necesidad de diseñar una aplicación, pero la respuesta a si esto es o no necesario, depende de entender tanto los objetivos de negocio, como las características que diferencian las aplicaciones de las webs. Por ejemplo, las aplicaciones pueden verse aun cuando se está sin conexión a internet, además, pueden acceder a ciertas características de hardware del teléfono como los sensores; capacidades que actualmente están fuera del alcance de las webs. Por lo anterior, puede decirse que una aplicación ofrece una mejor experiencia de uso, evitando tiempos de espera excesivos y logrando una navegación más fluida entre los contenidos. No siempre hay que elegir entre una u otra. Webs y aplicaciones no son competidoras, más bien, pueden complementarse entre ellas; por ejemplo, una web puede ser útil como canal de información para motivar la descarga de la aplicación.

Figura 1.2
Diseño líquido



Nota: Tamaño de los elementos de una aplicación web en porcentajes.
Tomado de idaBlog, 2020.

Primero el móvil

Es posible que cuando llegue la hora de diseñar una aplicación móvil ya exista una web como antecedente. En esos casos, la app móvil tiene que tomar las funciones y contenidos que se han pensado para la web y adaptarlos para que tengan sentido, de acuerdo al tamaño de pantalla y a la forma de interacción de un móvil. En otros casos, el diseño comienza desde cero, cuando todavía no hay ni web ni aplicación, y hay que decidirse por cuál de ellas empezar. Aquí es donde adquiere más trascendencia el concepto de mobile first que implica plantear el proceso de diseño teniendo en cuenta el móvil en primer lugar.

La ventaja de esta forma de trabajar es que, el pensar en el móvil como punto de partida, obliga a concentrarse en lo esencial de un producto y a hacer foco solo en lo que tiene sentido para este dispositivo. Una vez que la aplicación está diseñada, puede preguntarse cuál es la mejor forma de llevar lo hecho para el teléfono a una pantalla de ordenador o a otros dispositivos, extendiendo y escalando el contenido y repensando la diagramación. Todos los dispositivos tienen usos diferentes, y en el momento de adaptar el diseño, hay que tener en cuenta las características particulares de cada uno de ellos. Mobile first es una propuesta de trabajo que ha surgido recientemente; una tendencia emergente que aún está por consolidarse. (Cuello Javier, 2013)

1.2. Plataformas móviles

El mercado de las aplicaciones móviles es uno de los que mayor crecimiento está experimentando en estos momentos. Un número cada vez mayor de usuarios a nivel global utiliza a diario diferentes apps en sus teléfonos y tabletas, tanto en su vida personal como profesional.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Pues bien, para el desarrollo de dichas aplicaciones móviles es necesario hacer uso de las plataformas de desarrollo de apps, las cuales se pueden englobar en diferentes grupos en función de su naturaleza.

Plataformas móviles

1.3. Entornos de desarrollo

En el proceso de desarrollo móvil se ve la necesidad de la utilización de un entorno de desarrollo que facilite la velocidad de codificación; que la ayuda al programar muestre los resultados a través de emuladores.

Es por ello que se necesita un entorno de desarrollo accesible y usable para el desarrollador, en internet existe un sinnúmero de alternativas como:

- Android studio.
- Sublime text.
- Visual studio code.
- Entre otros.

1.4. Metodologías de desarrollo para aplicaciones móviles

Estimado estudiante, es hora de ver la metodología para el desarrollo de aplicaciones móviles, es importante mencionar que existen muchas metodologías que nos pueden ayudar al desarrollo de estas apps, una de ellas se la detalla a continuación:

El proceso de diseño y desarrollo de una aplicación móvil abarca desde la concepción de la idea hasta el análisis posterior a su publicación en las tiendas. Durante las diferentes etapas, diseñadores

y desarrolladores trabajan la mayor parte del tiempo de manera simultánea y coordinada. Se resume las fases de este proceso solo desde la perspectiva del diseño y desarrollo, es decir, sin tener en cuenta los roles de coordinación ni la participación del cliente. (Cuello Javier, 2013)

Metodologías de desarrollo.

1.4.1. Conceptualización

El resultado de esta etapa es una idea de aplicación, que tiene en cuenta las necesidades y problemas de los usuarios. La idea responde a una investigación preliminar y a la posterior comprobación de la viabilidad del concepto.

- Ideación.
- Investigación.
- Formalización de la idea.

1.4.2. Definición

En este paso del proceso se describe con detalle a los usuarios para quienes se diseñará la aplicación. También aquí se sientan las bases de la funcionalidad, lo cual determinará el alcance del proyecto y la complejidad de diseño y programación de la app.

- Definición de usuarios.
- Definición funcional.

1.4.3. Diseño

En la etapa de diseño se llevan a un plano tangible los conceptos y definiciones anteriores, primero en forma de wireframes que permiten crear los primeros prototipos para ser probados con usuarios, y posteriormente, en un diseño visual acabado que será provisto al desarrollador en forma de archivos separados y pantallas modelo para la programación del código.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

- Mockups.
- Prototipos.
- Test con usuarios.
- Diseño visual.

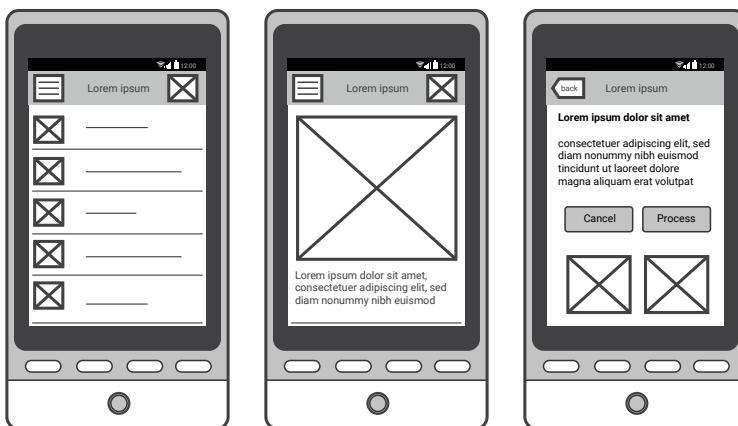
Wireframes.- Estructuración gráfica de los elementos o componentes de la aplicación.

En la fase de diseño se debe tener definida la funcionalidad y el aspecto general de nuestra aplicación, ya que será el primer paso para evaluar la viabilidad de nuestra aplicación móvil. Se puede utilizar múltiples aplicaciones para realizar mockups. En alguna ocasión he utilizado NinjaMock, Balsamiq, figma o InvisionApp, pero realmente existen múltiples opciones en internet, si no, siempre nos quedará el lápiz y el papel.



Actividades de aprendizaje recomendadas

Ingrese al foro y mencione las herramientas online para la creación de wireframes para una app móvil de registro de contactos.



Para esta actividad puede realizar una investigación en internet sobre las herramientas que ayudan a la creación de wireframes.

Índice

Primer bimestre

Segundo bimestre

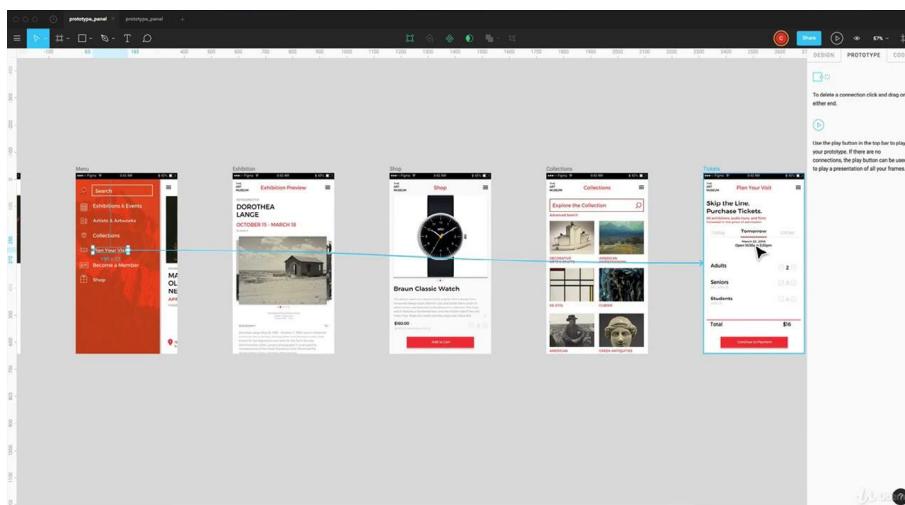
Solucionario

Referencias bibliográficas

Como herramienta de sugerencia para la realización de los wireframes de una app móvil y de esta actividad, se recomienda **FIGMA** tal como se muestra en la figura 1.3; esta herramienta nos ayuda al diseño y desarrollo de prototipos interactivos que ayudan de manera clara a visualizar la proyección de la app móvil.

FIGMA.- Herramienta de diseño gráfico.

Figura 1.3
Herramienta de diseño FIGMA



Nota: Software libre para edición de imágenes. Tomado de Herramienta de FIGMA [Entorno de la herramienta], 2021.

Terminada esta tarea podemos pasar al siguiente tema que es:

1.4.4. Desarrollo

El programador se encarga de dar vida a los diseños y crear la estructura sobre la cual se apoyará el funcionamiento de la aplicación. Una vez que existe la versión inicial, dedica gran parte del tiempo a corregir errores funcionales para asegurar el correcto desempeño de la app y la prepara para su aprobación en las tiendas.

- Programación del código.
- Corrección de bugs.

1.4.5. Publicación

La aplicación es finalmente puesta a disposición de los usuarios en las tiendas. Luego de este paso trascendental se realiza un seguimiento a través de analíticas, estadísticas y comentarios de usuarios, para evaluar el comportamiento y desempeño de la app, corregir errores, realizar mejoras y actualizarla en futuras versiones.

- Lanzamiento.
- Seguimiento.
- Actualización.

Muy bien estimado estudiante, lo invito a continuar con las mismas energías con las cuales inició la materia, seguidamente tenemos una autoevaluación para comprobar sus conocimientos adquiridos.

CASO DE ESTUDIO

Para poner en práctica los conocimientos adquiridos durante el desarrollo de los contenidos de nuestra materia, en especial aplicar la metodología de desarrollo para aplicaciones móviles; se propone desarrollar el siguiente caso de estudio, para lo cual debe realizar la conceptualización, definición y diseño.

Objetivo del caso: Se requiere la creación de una aplicación móvil nativa que registre contactos de personas (nombres, apellidos, correo, teléfonos, dirección, parentesco, etc.) y tenga opciones para ver la fecha actual, activar cronómetro y mostrar la latitud y longitud del GPS.

Elaboración de preguntas: Cumpliendo la metodología de casos, estimado estudiante, determine el desarrollo del caso con ayuda de estas preguntas:

¿La aplicación garantiza una facilidad de uso?

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

¿La aplicación garantiza la facilidad de navegación?

¿La aplicación cumple con el acceso a todos los requerimientos planteados?

¿La app muestra un diseño original?

Análisis y desarrollo del caso: Conceptualizar y definir la idea del caso de estudio mediante la utilización de alguna herramienta de diseño propuesta en esta guía. Desarrollar los wireframes de la aplicación completa para su posterior desarrollo.

Elaboración de informe: Desarrolle un documento donde se argumente la herramienta utilizada, la estructura de la aplicación y el motivo de su propuesta de solución.

Comparta y comente su trabajo desarrollado en el entorno de aprendizaje EVA.

Estimado estudiante, lo invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuál es su nivel de conocimientos obtenidos durante el transcurso de este periodo transcurrido; en caso de no obtener resultados satisfactorios, le sugiero revisar nuevamente los contenidos en el cual tiene problemas.



Autoevaluación 1

Conteste correctamente las preguntas según sea el caso:

1. Seleccione de la siguiente lista cuál es la tecnología de una aplicación móvil híbrida.
 - a. Aplicación responsive desarrollada en Java.
 - b. Aplicación desarrollada en HTML, CSS y JavaScript.
 - c. Desarrollada en C++.
2. Seleccione el concepto de una aplicación móvil:
 - a. Son apps que están en smartphones.
 - b. Son apps que corren en un ordenador.
 - c. Son apps que están en un servidor en la nube.
3. La diferencia entre una aplicación web móvil y una app móvil es:
 - a. La aplicación móvil se tiene que descargar para ser utilizada y las aplicaciones web móviles se adaptan al dispositivo.
 - b. La aplicación móvil es de pago y las aplicaciones web móviles son gratuitas.
 - c. Ambas son aplicaciones de pago.
4. Una aplicación responsive o adaptativa es:
 - a. Una app que se instala en el dispositivo móvil.
 - b. Una app que es desarrollada en Java.
 - c. Una app que se adapta al dispositivo móvil.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. Seleccione la tecnología que no es una plataforma móvil:

- a. iOS.
- b. Android.
- c. Play store.

6. Seleccione el proceso que no pertenece a una metodología de desarrollo de una aplicación móvil:

- a. Definición.
- b. Maquetación y diagramación.
- c. Desarrollo.

7. Para evaluar la historia de las aplicaciones móviles, los primeros teléfonos contaban con aplicaciones de:

- a. Productividad personal.
- b. Economía mundial.
- c. Deportes.

8. Para acceder a una aplicación web móvil se necesita:

- a. Instalar el aplicativo en el móvil.
- b. Conectarse a una tienda.
- c. Conectarse a internet.

9. El diseño líquido es:

- a. Diseño conceptual de desarrollo móvil.
- b. Diseño de las apps de los primeros teléfonos.
- c. El contenido que toma la forma del contenedor.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

10. Para el desarrollo de aplicaciones móviles es necesario el uso de:
- Licencias de las plataformas móviles.
 - Plataformas de desarrollo de apps.
 - Contar por lo menos de un dispositivo móvil para las pruebas correspondientes.

[Ir al solucionario](#)

Para ver los resultados correctos de esta autoevaluación, acuda a la parte del solucionario de la unidad que se encuentra al final de esta guía.

¡Hemos terminado la primera unidad!

¡Felicitaciones!



Semana 2

Para lograr el resultado de aprendizaje tiene que revisar el contenido de esta sección, en especial los tipos de aplicaciones móviles que existen. Esto le ayudará a dar una visión general y diferenciar con el desarrollo de aplicaciones generales.

Estimado estudiante, lo invito a seguir revisando los contenidos de nuestra materia, ahora empezamos su segunda semana con temas importantes como:



Unidad 2. Tipos de aplicaciones y almacenamiento de información en dispositivos móviles

Estimado estudiante, le comento que en el mundo del desarrollo de las aplicaciones móviles en la actualidad existen dos alternativas para desarrollarlas; lo invito a revisar el contenido de la materia y conocer cuáles son.

2.1. Aplicaciones nativas

Las aplicaciones nativas son aquellas que han sido desarrolladas con el software que ofrece cada sistema operativo a los programadores, llamado genéricamente Software Development Kit (SDK). Así, Android, iOS y Windows Phone tienen uno diferente y

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

las aplicaciones nativas se diseñan y programan específicamente para cada plataforma en el lenguaje utilizado por el SDK. Este tipo de apps se descarga e instala desde las tiendas de aplicaciones, sacando buen partido de las diferentes herramientas de promoción y marketing de cada una de ellas. Las aplicaciones nativas se actualizan frecuentemente, y en esos casos, el usuario debe volver a descargarlas para obtener la última versión que a veces corrige errores o añade mejoras. Una característica generalmente menospreciada de las apps nativas es que pueden hacer uso de las notificaciones del sistema operativo para mostrar avisos importantes al usuario, aun cuando no se esté usando la aplicación, como los mensajes de Whatsapp, por ejemplo. Además, no requieren internet para funcionar, por lo que ofrecen una experiencia de uso más fluida y están realmente integradas al teléfono, lo cual les permite utilizar todas las características de hardware del terminal, como la cámara y los sensores (GPS, acelerómetro, giróscopo, entre otros). A nivel de diseño, esta clase de aplicaciones tiene una interfaz basada en las guías de cada sistema operativo, logrando mayor coherencia y consistencia con el resto de aplicaciones y con el propio SO. Esto favorece la usabilidad y beneficia directamente al usuario que encuentra interfaces familiares. (Cuello Javier, 2013)

GPS.- Sistema de geo posicionamiento.

SO.- Sistema Operativo móvil.

2.2. Aplicaciones móviles híbridas

Este tipo de aplicaciones son una especie de combinación de tecnologías. La forma de desarrollarlas es parecida a la de una aplicación web, usando HTML, CSS y JavaScript, y una vez que la aplicación está terminada, se compila o empaqueta de forma tal, que el resultado final es como si se tratara de una aplicación nativa. Esto permite casi con un mismo código obtener diferentes aplicaciones, por ejemplo, para Android y iOS, y distribuirlas en cada una de sus tiendas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

A diferencia de las aplicaciones web, estas permiten acceder, usando librerías, a las capacidades del teléfono, tal como lo haría una app nativa. Las aplicaciones híbridas, también tienen un diseño visual que no se identifica en gran medida con el del sistema operativo. Sin embargo, hay formas de usar controles y botones nativos de cada plataforma para apegarse más a la estética propia de cada una.

¿Cuál se debería usar?

Dadas las características de cada una de las aplicaciones, decidirse por una u otra estará determinado por unos pocos factores fundamentales y por la forma en que afectan finalmente la experiencia de uso. Cuando la disponibilidad de la aplicación móvil para utilizarla sin tener acceso a internet, la posibilidad de usar notificaciones y el acceso a los recursos de hardware del teléfono sean importantes, una aplicación nativa será la opción más indicada. Si ninguna de estas cosas es realmente importante para la aplicación, quizás sea más fácil diseñar una aplicación web; si es el caso de que ya se dispone del conocimiento para ello heredado del desarrollo de sitios web. En este caso, el costo de desarrollo es más bajo y la forma de trabajar un poco más ágil. Independientemente de esto, las aplicaciones nativas son las que ofrecen una mejor experiencia de uso y, sobre todo, rendimiento. Algunas apps como Facebook o LinkedIn, que antes eran híbridas, han pasado a ser nativas por este motivo. Adicionalmente, ellas responden más a las guías de diseño de cada sistema operativo.

2.3. Tecnologías o framework para las aplicaciones móviles híbridas

Un framework es un entorno de trabajo que utilizan los programadores y que cuenta con una estructura previa que se establece para organizar y desarrollar un software. Con estos espacios se pueden automatizar muchos procesos y por ello se

facilita la programación, bien sea de un desarrollo web o de una aplicación móvil. En la figura 2.1 se muestran algunos de estos frameworks de desarrollo.

Existe una infinidad de aplicaciones híbridas que han sido creadas con un framework de desarrollo. Apps de la talla de Uber, Airbnb o Instagram son algunos ejemplos. Tanto las funcionalidades, el acceso a los elementos del dispositivo y la experiencia de usuario son tan espectaculares como en las apps nativas. Y no somos los únicos que lo decimos, usted puede encontrar diversas aplicaciones muy conocidas que nunca hubiese pensado que eran híbridas.

Figura 2.1

Framework de desarrollo móvil



Nota: Tipos de frameworks para desarrollo de aplicaciones móviles.

Ionic Framework

La primera versión de Ionic salió al mercado en 2013, hace ya 7 años, basado en AngularJS y en Apache Cordova. Con este framework, podemos desarrollar nuestro código con AngularJS, ReactJS y VueJS.

Es uno de los frameworks más potentes y versátiles para el desarrollo de aplicaciones híbridas. Tiene una comunidad de desarrolladores muy grande y la curva de aprendizaje para los desarrolladores que vienen de tecnologías FrontEnd es mínima.

React Native

La primera versión de React Native salió al mercado en 2015. Es la apuesta de Facebook para el desarrollo de aplicaciones híbridas. Es decir, React Native va a seguir creciendo y evolucionando ya que tiene a un gigante detrás. Este framework se apoya en ReactJS, también desarrollado por Facebook.

Xamarin

Al igual que React Native es la apuesta de Facebook, Xamarin es la apuesta de Microsoft. El lenguaje que se utiliza para desarrollar en Xamarin es C#.

Flutter

Figura 2.2

Framework de desarrollo móvil Flutter



Nota: Framework de desarrollo móvil Flutter. Tomado de Flutter, 2021.

Flutter es un framework (ver la figura 2.2) de código abierto desarrollado por Google para crear aplicaciones nativas de forma fácil, rápida y sencilla. Su principal ventaja radica en que genera código 100% nativo para cada plataforma, con lo que el rendimiento y la UX es totalmente idéntico a las aplicaciones nativas tradicionales.

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

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

Funcionalidades de Flutter

- Calidad nativa: Las aplicaciones nativas se desarrollan específicamente para un sistema operativo, Flutter utiliza todas las ventajas de las aplicaciones nativas para conseguir calidad en el resultado final.
- Experiencia de usuario: Flutter incluye Material Design de Google y Cupertino de Apple, con lo que la experiencia de usuario es óptima y los interfaces de usuario idénticos a los de las aplicaciones desarrolladas por las propias compañías.
- Tiempo de carga: Una de las principales causas de abandono de una aplicación es el tiempo que tarda en cargar, con Flutter se experimentan tiempos de carga por debajo de un segundo en cualquiera de los soportes iOS o Android.
- Desarrollo ágil y rápido: Gracias a la característica hot-reload, puedes programar y ver los cambios en tiempo real en tu dispositivo o en los simuladores.

Otra de las ventajas de utilizar este framework es que da igual el sistema operativo que utilice, ya que puede descargarlo para Windows, Mac o Linux desde este [enlace](#).

El framework tiene una comunidad en la que podrá compartir sus conocimientos, experiencias y/o buscar ayuda para sus dudas o problemas puntuales.



Actividad de aprendizaje recomendada

Vídeo: Con ayuda de internet escuche detenidamente el siguiente [vídeo](#) donde se explica el inicio de Flutter.

¿Cómo le pareció el contenido del video? Bastante explicativo sobre la historia del framework Flutter, ¿verdad?

Hemos terminado la semana 2, lo invito a seguir trabajando y revisando el contenido de la materia.

¡Siga adelante!



Semana 3

Llegamos a la semana 3 y para lograr este resultado de aprendizaje en esta semana tiene que revisar el contenido de esta sección para poder determinar los tipos de almacenamiento y la gestión de archivos que es muy diferente en el ámbito del desarrollo de software general.

2.4. Bases de datos en dispositivos móviles

Con el uso de internet, ya no es necesario que las personas se encuentren físicamente dentro de las empresas para poderse conectar a las aplicaciones o las bases de datos. Con la portabilidad de la tecnología inalámbrica, nos podemos conectar a internet o la intranet de la empresa para tomar datos almacenados. Estos datos están estructurados y organizados en entidades y objetos que se

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

encuentran disponibles para los usuarios como información. La mayor ventaja se encuentra en la disponibilidad de la información, esta puede ser utilizada el momento que lo solicite el usuario.

Cada vez más aplicaciones móviles necesitan datos para funcionar, y las bases de datos han sido durante bastante tiempo la forma más común de almacenar y administrar datos.

Por lo tanto, en un escenario típico, una aplicación móvil usa una base de datos alojada en la nube y se conecta remotamente a ella para acceder a sus datos. Esto, por supuesto, implica que la aplicación móvil necesita una conexión de red activa y bastante rápida.

La base de datos es la forma más común de almacenar y administrar datos. Las bases de datos se manejan en el lado del servidor o en la nube y los dispositivos móviles solo se comunican con ellos a través de la red.

Sin embargo, para hacer que las aplicaciones sean más receptivas y menos dependientes de la conectividad de la red, la tendencia del uso fuera de línea o la menor dependencia de la red están ganando popularidad.

Hoy en día, las aplicaciones mantienen la base de datos localmente o hacen una copia de DB en la nube en el dispositivo local y se sincronizan con ella una vez al día o cada vez que hay una conectividad de red. Esto ayudará en aplicaciones más rápidas y receptivas que son funcionales, incluso cuando no hay conectividad a internet o es limitada. (Modelos de DB, 2012)

La validación de datos también se conoce como validación de entrada.

Las bases de datos deben ser:

- Ligeras, ya que el almacenamiento es limitado en dispositivos móviles.

- Sin requisito de servidor.
- En una forma de biblioteca con ninguna o muy limitada dependencia (incrutable), para que se pueda usar cuando sea necesario.
- Rápido y seguro.
- Fácil de manejar mediante código y opción para hacerlo privado o compartido con otras aplicaciones.
- Poca memoria y consumo de energía.

Las bases de datos incrustadas como se muestran en la tabla 1 son bibliotecas livianas y autónomas sin componentes de servidor, sin necesidad de administración, una pequeña huella de código y requisitos de recursos limitados. Las aplicaciones móviles pueden vincularse (estática o dinámicamente) con ellas y luego usarlas para crear y gestionar sus propias bases de datos privadas o compartidas localmente en el dispositivo.

Tabla 1.

Comparativa de bases de datos para dispositivos móviles

Base de datos	Tipo de datos	Licencia	Soporte
BerkeleyDB	Relacional, objetos, pares clave-valor, documentos	AGPL 3.0	Android, iOS
Couchbase Lite	Documentos	Apache 2.0	Android, iOS
LevelDB	pares clave-valor	New BSD	Android, iOS
SQLite	Relacional	Public Domain	Android, iOS, Windows Phone, Blackberry
BerkeleyDB	Relacional, objetos, pares clave-valor, documentos	AGPL 3.0	Android, iOS

Fuente: [Enlace](#)

2.5. Gestión de archivos en dispositivos móviles

El software de gestión de dispositivos móviles permite la distribución de aplicaciones, configuraciones y parches para dichos dispositivos. Idealmente, el software permite a los administradores supervisar los dispositivos móviles tan fácilmente como se hace con los equipos

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

de escritorio y proporciona un rendimiento óptimo para los usuarios. Las herramientas de gestión de archivos deben incluir administración de aplicaciones, sincronización y uso compartido de archivos, herramientas de seguridad de datos y soporte para dispositivos, ya sea que pertenezcan a la empresa o sean de propiedad personal. (TechTarget, 2017)

La herramienta ideal de gestión de dispositivos móviles:

- Es compatible con todas las plataformas y aplicaciones de operación de dispositivos portátiles comunes.
- Puede funcionar a través de múltiples proveedores de servicios.
- Puede desplegar rápidamente la próxima generación de hardware, plataformas operativas y aplicaciones.
- Puede agregar o quitar dispositivos del sistema según sea necesario para asegurar una eficiencia y seguridad de red óptimas.



Actividades de aprendizaje recomendadas

- Ahora con ayuda del contenido de la guía, argumente las características y diferencias entre las aplicaciones híbridas y nativas de los dispositivos móviles. Puede utilizar su cuaderno de apuntes para ir registrando las características y diferencias de cada una de ellas.

¡Suerte! ¡Siga adelante!

Estimado estudiante, lo invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuál es su nivel de conocimientos obtenidos durante el tránscurso de este periodo transcurrido, en caso de no obtener resultados satisfactorios le sugiero revisar nuevamente los contenidos en los que tiene problemas.



Autoevaluación 2

Conteste correctamente las siguientes preguntas:

1. Un framework de desarrollo es:
 - a. Una librería de trabajo que utilizan los programadores.
 - b. Un entorno de trabajo que utilizan los programadores.
 - c. Un conjunto de plantillas móviles que reutilizan los programadores.
2. Seleccione las aplicaciones que han sido desarrolladas con frameworks:
 - a. SIRI.
 - b. Uber.
 - c. Ionic.
 - d. Google Play.
3. Seleccione cuál de las alternativas es un framework de desarrollo móvil:
 - a. Ionic.
 - b. Xamarin.
 - c. Kotlin.
4. El framework React Native es propiedad de:
 - a. Google.
 - b. Facebook.
 - c. Twitter.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. El framework Flutter es propiedad de:
 - a. Google.
 - b. Facebook.
 - c. Twitter.
6. Las aplicaciones móviles nativas son desarrolladas con la tecnología:
 - a. C++.
 - b. Software Development Kit.
 - c. Windows phone.
7. Para implementar una actualización de las aplicaciones nativas es necesario:
 - a. Descargar e instalar la actualización en tiendas oficiales.
 - b. No se necesitan actualizar las aplicaciones nativas.
 - c. Las aplicaciones nativas se actualizan automáticamente.
8. Seleccione una característica fundamental de las apps nativas es:
 - a. No tienen costo.
 - b. Las apps pueden hacer uso de notificaciones del sistema operativo para mostrar a los usuarios.
 - c. Las notificaciones se las realiza vía correo electrónico.
9. El diseño FrontEnd de aplicaciones móviles nativas se basa en:
 - a. Los estándares de desarrollo web.
 - b. La guía de diseño de Google UI.
 - c. La guía de cada sistema operativo, para lograr coherencia.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

10. El diseño estético o FrontEnd de las aplicaciones híbridas tiene una característica que es:

- a. Tiene un diseño visual que se identifica con el sistema operativo.
- b. Tiene un diseño visual que no se identifica con el sistema operativo.
- c. Tiene un diseño visual con estándares web.

[Ir al solucionario](#)

Las respuestas a esta autoevaluación se encuentran al final de la presente guía didáctica, vaya y compare las respuestas, si no logró un buen resultado en la autoevaluación, no se preocupe, le recomiendo leer nuevamente el/los capítulos confusos y reforzar sus conocimientos. Y si aún tiene inquietudes no dude en preguntar al profesor.

¡Hemos terminado la segunda unidad!

¡Felicitaciones! Sigamos avanzando con la siguiente sección.

Resultado de aprendizaje Implementa aplicaciones para plataformas móviles.

Contenidos, recursos y actividades de aprendizaje



Semana 4

Vamos avanzando, y para lograr el resultado de aprendizaje en esta semana tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual para poder así desarrollar aplicaciones móviles en las plataformas.



Unidad 3. Aplicaciones móviles en Android Studio. Parte 1

3.1. Conceptos

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android, basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan su productividad cuando desarrolla apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puede desarrollar para todos los dispositivos Android.
- Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación.
- Integración con GitHub y plantillas de código para ayudarle a compilar funciones de apps comunes y también importar código de muestra.
- Variedad de marcos de trabajo y herramientas de prueba.
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros.
- Compatibilidad con C++ y NDK.
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine.

3.2. Estructura del proyecto

Estimado estudiante, para poder revisar la estructura de un proyecto en Android pude hacer uso de este recurso publicado en el repositorio [GitHub](#) para poder hacer referencia a los contenidos de esta sección, esto le ayudará a comprender de mejor manera las partes importantes de un proyecto.

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

- Módulos de apps para Android.
- Módulos de biblioteca.
- Módulos de Google App Engine.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyecto de Android, como se ve en la figura 3.1. Esta vista está organizada en módulos para que pueda acceder rápidamente a los archivos fuente clave de su proyecto.

Puede ver todos los archivos de compilación en el nivel superior de secuencias de comando de Gradle y cada módulo de app contiene las siguientes carpetas:

- **Manifests:** Contiene el archivo `AndroidManifest.xml`.
- **Java:** Contiene los archivos de código fuente Java, incluido el código de prueba de JUnit.
- **Res:** Contiene todos los recursos sin código, como diseños XML, strings de IU e imágenes de mapa de bits.

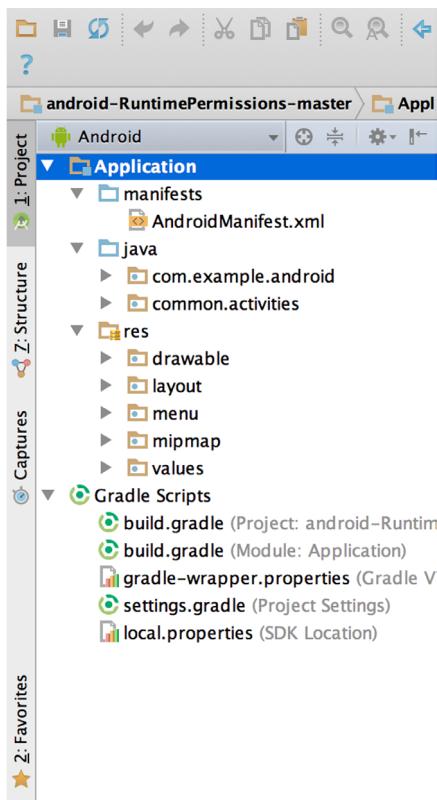
La estructura del proyecto de Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, se selecciona Project en el menú desplegable Project como se muestra en la figura 3.1; se muestra como Android.

También se puede personalizar la vista de los archivos del proyecto para concentrarse en aspectos específicos del desarrollo de su app.

Editor es de gran ayuda en la detección de errores, ya que los archivos con problemas reflejan alertas tal como se muestra en la figura 3.2, esto ayuda a localizar posibles inconvenientes en el desarrollo del proyecto. (Android Studio, 2021)

Figura 3.1

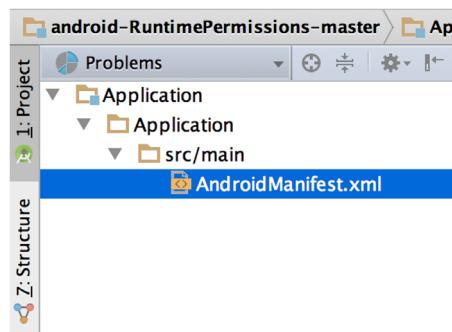
Los archivos de proyecto en la vista de Android



Nota: Estructura de directorios y archivos de un proyecto en Android Studio. Tomado de Android Studio, 2021.

Figura 3.2

Archivos defectuosos de Android



Nota: Archivos del proyecto en la vista "Problemas", en la que se muestra un archivo de diseño con un problema. Tomado de Android Studio, 2021.

3.3. Instalación

Estimado estudiante, para el proceso de instalación y poner en práctica los conocimientos adquiridos de la herramienta, realice la instalación de Android Studio; hay que tomar en cuenta que el proceso es diferente para cada sistema operativo, pondremos como ejemplo la instalación en el sistema operativo Windows.

Instalación de Android Studio en Windows

Para instalar Android Studio en Windows, haga el siguiente proceso:

1. Primero debemos instalar el compilador de Java y la máquina virtual. Estas herramientas las podemos descargar de: [Java SE Development Kit \(JDK\)](#).
2. El segundo paso es la descarga del Android Studio (que contiene todo lo necesario para comenzar el desarrollo de aplicaciones en Android), lo hacemos del sitio: [Android Studio](#).

Siga los pasos del asistente de configuración en Android Studio y asegúrese de instalar los paquetes de SDK que recomienda.

Para instalar el SDK puede revisar el siguiente [recurso](#) el cual le ayudará al proceso de instalación.

Es importante mencionar que cuando haya nuevas herramientas y otras API disponibles, Android Studio se lo informará por medio de una ventana emergente. También puede buscar actualizaciones si hace click en **Help > Check for Update**.

Figura 3.3

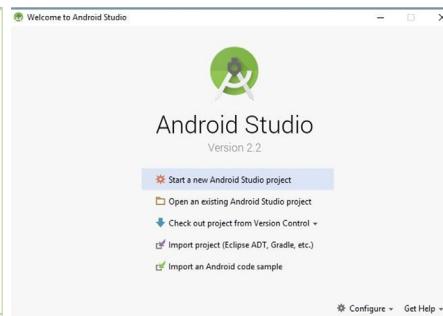
Interfaz de instalación de Android Studio



Nota: Proceso de instalación de Android Studio. Tomado de Android Studio, 2021.

Figura 3.4

Interfaz para la creación de un nuevo proyecto



Nota: Interfaz para la gestión de proyectos en Android Studio. Tomado de Android Studio, 2021.



Semana 5

Para lograr el resultado de aprendizaje de la semana 4 tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual, en especial, desarrollar las prácticas de instalación y configuración de la herramienta de estudio.

Estimado estudiante, es hora de revisar detalladamente las partes de nuestro entorno de desarrollo, la siguiente sección le ayudará a entender los elementos que componen el editor.

3.4. Entorno de desarrollo

Android Studio cuenta con una interfaz debidamente estructurada e intuitiva haciendo que los componentes utilizados para el desarrollo móvil sean accesibles, a continuación, en la figura 3.5 se muestra los elementos que contiene el editor.

Ventana principal de Android Studio

Se puede organizar la ventana principal para tener más espacio en pantalla si oculta o desplaza las barras y ventanas de herramientas.

En cualquier momento se puede realizar búsquedas en el código fuente, las bases de datos, las acciones y los elementos de la interfaz de usuario, entre otros. Para ello, presione dos veces la tecla Mayús o haga click en la lupa en la esquina superior derecha de la ventana de Android Studio.

3.5. Flujo de trabajo de desarrollo en Android Studio

El flujo de trabajo de desarrollo de una app para Android es conceptualmente el mismo que el de otras plataformas de apps. Sin embargo, compilar de manera eficiente una app para Android bien diseñada requiere algunas herramientas especializadas. En la siguiente lista, se proporciona una descripción general del proceso de compilación de una app para Android y se incluyen vínculos a algunas herramientas de Android Studio que debería usar en cada fase del desarrollo.

Flujo de trabajo de desarrollo en Android Studio

3.5.1. Cómo configurar el espacio de trabajo

Esta es la fase que probablemente ya terminó: instalar Android Studio y crear un proyecto.

3.5.2. Cómo escribir su app

Android Studio incluye inteligencia y una variedad de herramientas que ayudarán a agilizar su trabajo, escribir códigos de calidad, diseñar una IU y crear recursos para diferentes tipos de dispositivos.

3.5.3. Cómo compilar y ejecutar

Durante esta fase, compilará su proyecto en un paquete APK depurado que puede instalar y ejecutar en el emulador o en un dispositivo que ejecute Android.

APK.- Extensión de un archivo de instalación de un dispositivo Android.

3.5.4. Depura, genera perfiles y realiza pruebas

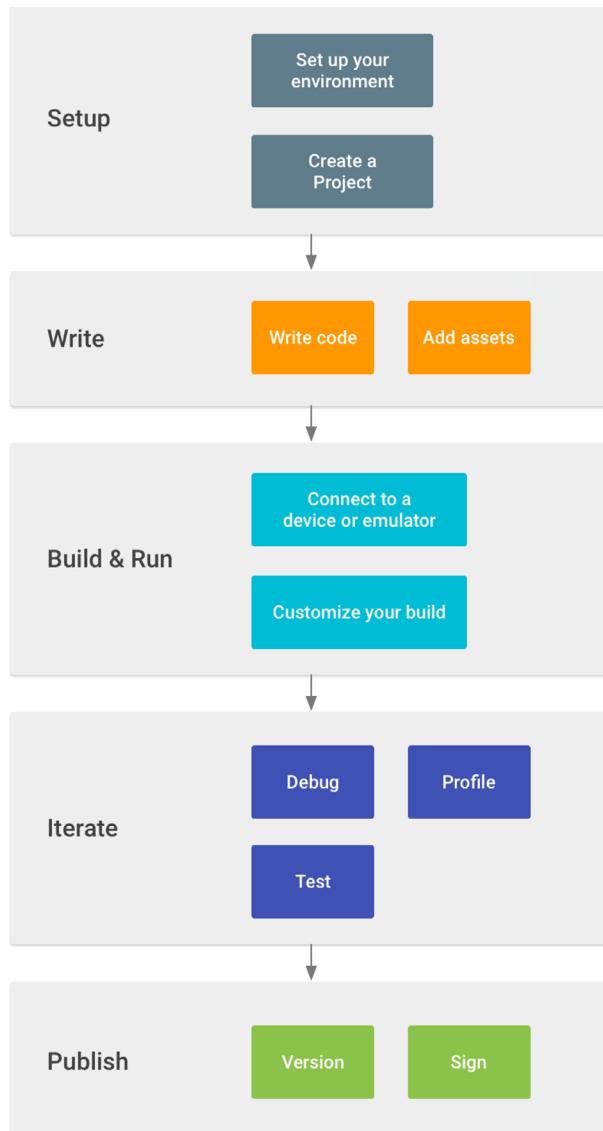
Esta es la fase iterativa en la que continúa escribiendo su app, pero con mayor énfasis en la solución de errores y optimización del rendimiento. Por supuesto, crear pruebas le ayudará con esas tareas.

Para ver y analizar las diferentes métricas de rendimiento, como el uso de memoria, el tráfico de red y el impacto de la CPU, entre otras, consulte “Herramientas de generación de perfiles de rendimiento”.

3.5.5. Cómo publicar

Cuando esté listo para lanzar su app a los usuarios, solo debe considerar algunos aspectos más, como el control de versiones y la firma con clave.

Figura 3.5
Proceso de publicación de una app



Nota: Proceso de creación de una aplicación en Android Studio. Tomado de Android Studio, 2021.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

3.6. Desarrollo de un proyecto

Android Studio facilita la creación de apps de Android para varios factores de forma, como teléfonos celulares, tablets, TVs y dispositivos Wear. En este apartado, se muestra cómo iniciar un nuevo proyecto de app para Android o importar un proyecto existente.

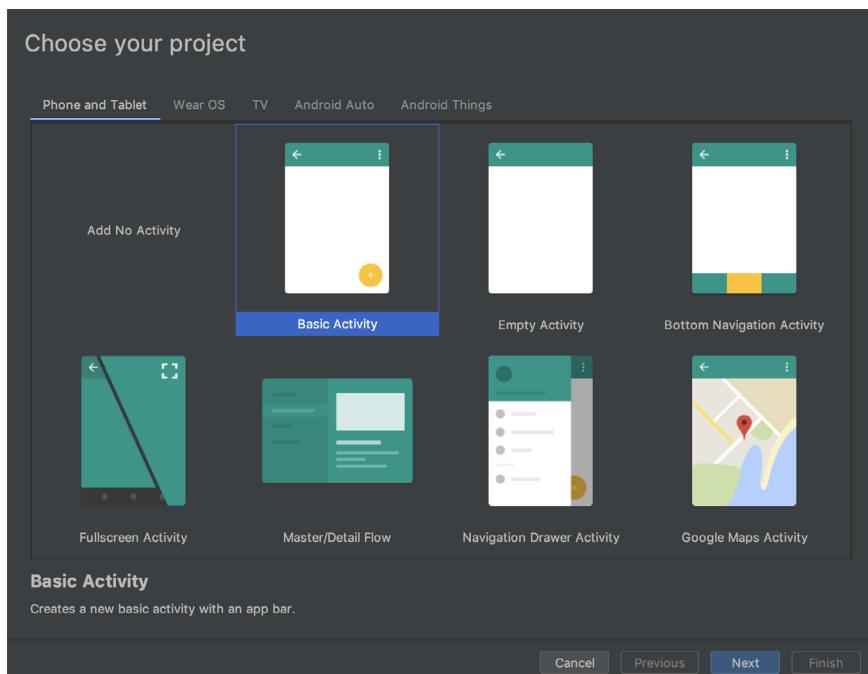
Si no tiene un proyecto abierto, Android Studio muestra la pantalla de bienvenida, donde puede hacer click en “Start a new Android Studio project” para crear un nuevo proyecto.

Si tiene un proyecto abierto, para comenzar a crear un nuevo proyecto selecciona File > New> New Project en el menú principal.

Debería ver el asistente “Create New Project”, que le permite elegir el tipo de proyecto que desea crear y se completa con código y recursos para comenzar. Esta página sirve como guía para crear un nuevo proyecto desde el asistente Create New Project.

En la figura 3.6 se muestra los tipos de proyectos que se pueden seleccionar, incluso el tipo de dispositivo con el cual se va a trabajar como TV, Wear OS, etc.

Figura 3.6
Tipos de proyectos en Android



Nota: En la primera pantalla del asistente, seleccione el tipo de proyecto que desea crear. Tomado de Android Studio, 2021.



Actividades de aprendizaje recomendadas

Realice la instalación de Android Studio en su máquina personal con todo el proceso mencionado anteriormente, documente y comente en el foro sobre esta actividad. Puede ayudarse realizando capturas de pantalla de cada proceso de instalación para ayudarse con la documentación.

¿Cómo le fue con la actividad? Espero que muy bien, y si tiene preguntas puede asistir en el horario de tutorías para poderle ayudar con sus preguntas.

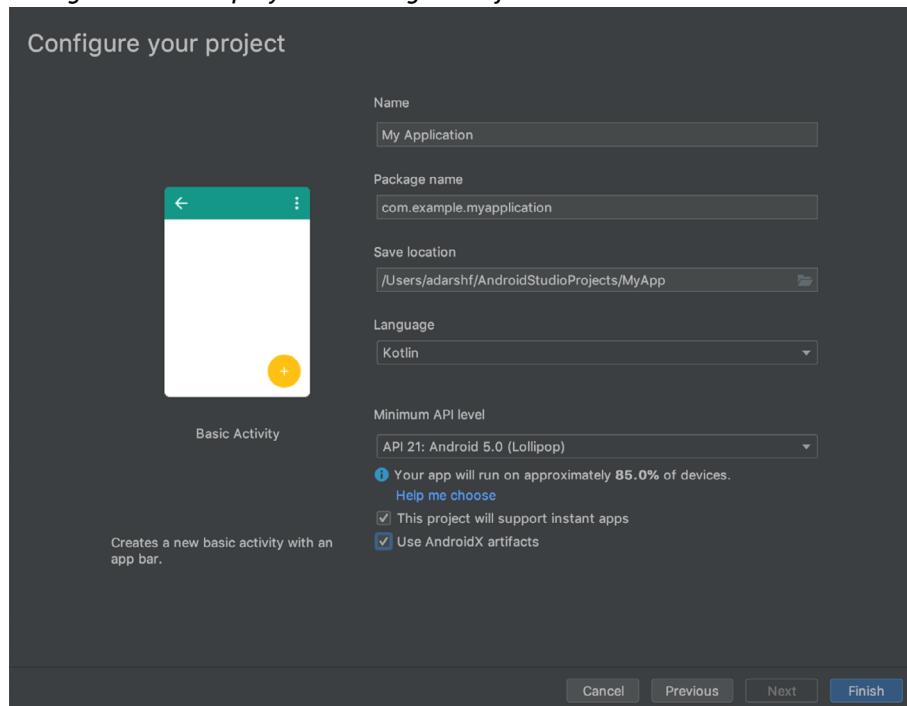
Terminada esta tarea podemos pasar al siguiente tema que es:

Configuración del proyecto

El siguiente paso consiste en configurar algunos ajustes y crear un nuevo proyecto, tal como se describe a continuación y se muestra en la figura 3.7.

Figura 3.7

Configure su nuevo proyecto con algunos ajustes



Nota: Configure su nuevo proyecto con algunos ajustes. Tomado de Android Studio, 2021.

1. Especifique el nombre del proyecto en "Name".
2. Especifique el nombre del paquete en "Package name". De forma predeterminada, el nombre del paquete también se usa como ID de su aplicación, aunque puede cambiar este valor más adelante.

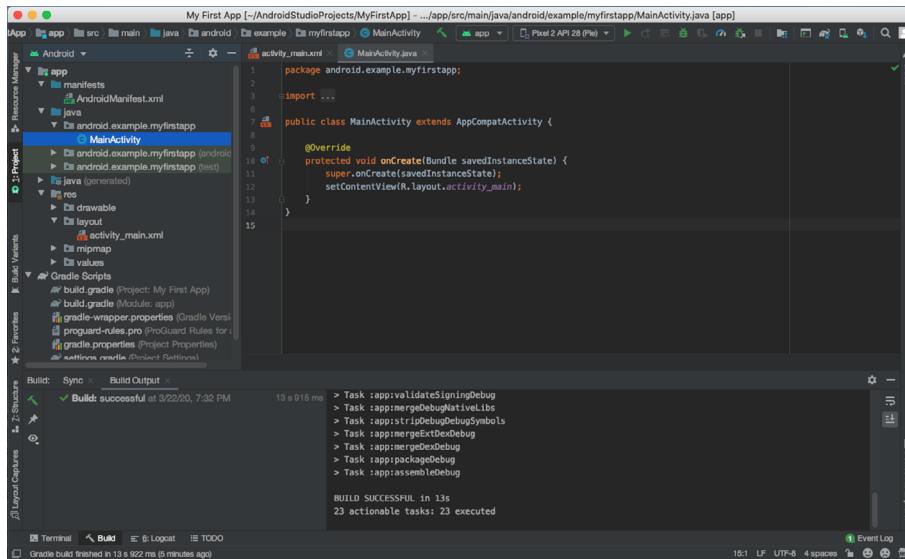
3. Especifique en “Save location” la ruta local donde va a guardar el proyecto.
4. Seleccione en “Language” el lenguaje que desea que use Android Studio al crear código de muestra para el nuevo proyecto. Tenga en cuenta que no está limitado a usar solo ese lenguaje para crear el proyecto.
5. Seleccione el nivel mínimo de API que admitirá la app en “Minimum API level”. Si selecciona un nivel de API más bajo, es posible que la app confíe en menos API de Android modernas. Sin embargo, también habrá un porcentaje mayor de dispositivos Android que podrán ejecutar su app. Cuando elige un nivel de API más alto, ocurre lo contrario.
6. Si desea que su proyecto use las bibliotecas de AndroidX de forma predeterminada, que son reemplazos mejorados de las bibliotecas de compatibilidad de Android, marque la casilla junto a “Use AndroidX artifacts”. Para obtener más información, consulte la sección de la descripción general de AndroidX.
7. Cuando esté todo listo para crear su proyecto, haga click en “Finish”.

Android Studio crea el nuevo proyecto con código y recursos básicos para comenzar. Si luego decide agregar compatibilidad con un factor de forma de dispositivo diferente, puede agregar un módulo al proyecto. Y si desea compartir código recursos entre módulos, puede crear una biblioteca de Android.

Después de un tiempo de procesamiento, aparecerá la ventana principal de Android Studio.

Figura 3.8

Entorno de desarrollo de Android Studio



Nota: Ventana principal del entorno de Android Studio. Tomado de Android Studio, 2021.

1. Primero, asegúrese de que la ventana Project esté abierta (selecione View > Tool Windows > Project) y la vista Android esté seleccionada en la lista desplegable de la parte superior de la ventana. Podrá ver los siguientes archivos:

app > java > com.example.myfirstapp > MainActivity

2. Esta es la actividad principal. Es el punto de entrada de su app. Cuando compila y ejecuta la app, el sistema inicia una instancia de esta Activity y carga su diseño.

app > res > diseño > activity_main.xml

3. Este archivo XML define el diseño de la interfaz de usuario (IU) de la actividad. Contiene un elemento TextView con el texto "Hello, World!".

app > manifiestos > AndroidManifest.xml

4. En el archivo de manifiesto, se describen las características fundamentales de la app y se define cada uno de sus componentes.

Secuencias de comandos de Gradle > build.gradle

5. Hay dos archivos con este nombre: uno para el proyecto, “Proyecto: mi primera app”, y otro para el módulo de la aplicación, “Módulo: app”. Cada módulo tiene su propio archivo build.gradle, pero este proyecto por el momento tiene un solo módulo. Use el build.file de cada módulo para controlar cómo el complemento Gradle compila su app.
6. Para ejecutar la app, continúe con el siguiente contenido, “Cómo ejecutar su app”, también [descargue](#) el siguiente recurso donde encontrará un ejemplo básico de una aplicación en Android para ejecutarlo correctamente.

Hemos terminado la semana 5 y para ver los resultados de nuestra aplicación móvil le invito a seguir la siguiente sección donde se muestra como ejecutar la app.



Semana 6

Para lograr este resultado de aprendizaje de la semana 4 tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual, con mayor énfasis en la ejecución de la aplicación móvil en el emulador de prueba.

3.7. Cómo ejecutar su app

En la sección anterior, aprendió a crear una app para Android que muestra el mensaje “Hola mundo”. Ahora podrá ejecutar la app en un dispositivo real o en un emulador.

Cómo ejecutar la app en un dispositivo real:

Configure su dispositivo de la siguiente manera:

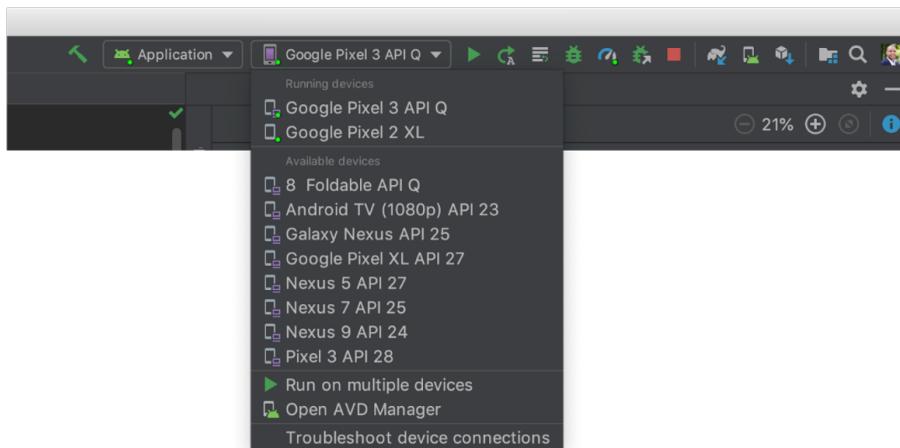
1. Conecte el dispositivo a la máquina de desarrollo con un cable USB. Si desarrolló una app en Windows, es posible que tenga que instalar el controlador USB apropiado para su dispositivo.
2. Realice los siguientes pasos a fin de habilitar la depuración de USB en la ventana “Opciones para desarrolladores”:
 - Abra la app de “Configuración”.
 - Si su dispositivo usa Android 8.0 o una versión posterior, seleccione “Sistema”. De lo contrario, continúe con el paso siguiente.
 - Desplácese hasta la parte inferior y seleccione “Acerca del teléfono”.
 - Desplácese hasta la parte inferior y presione “Número de compilación” siete veces.
 - Regrese a la pantalla anterior, desplácese hasta la parte inferior y presione “Opciones para desarrolladores”.
 - En la ventana “Opciones para desarrolladores”, desplácese hacia abajo para buscar y habilitar la depuración de USB.

Ejecute la app en su dispositivo de la siguiente manera:

1. En Android Studio, seleccione su app en el menú desplegable de configuraciones de ejecución/depuración de la barra de herramientas.
2. En la barra de herramientas, seleccione el dispositivo en el que desea ejecutar la app desde el menú desplegable del dispositivo de destino como lo muestra la figura 3.9.

Figura 3.9

Ejecución de una app



Nota: Menú desplegable del dispositivo de destino. Tomado de Android Studio, 2021.

3. Haga click en **Run**

Android Studio instala su app en el dispositivo conectado y lo inicia. Ahora verá el mensaje “Hello, World!” que se muestra en su app en el dispositivo.

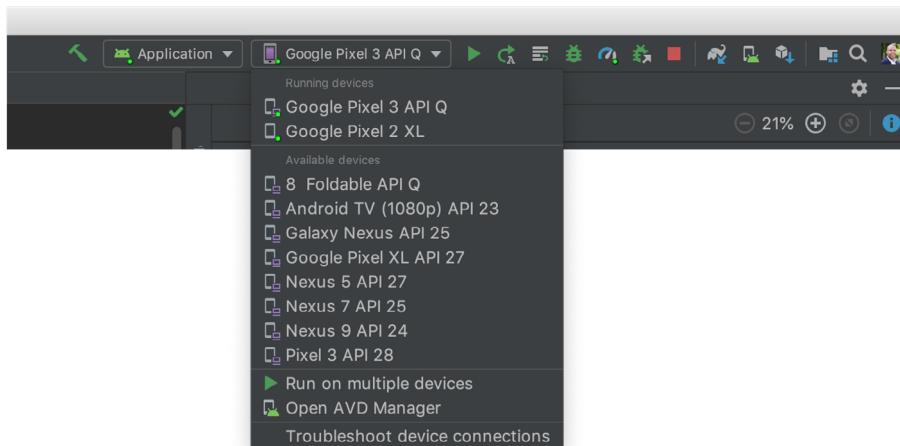
Cómo ejecutar la app en un emulador

Ejecute la app en un emulador de la siguiente manera; también revise la figura 3.10 donde se muestra gráficamente la forma de ejecutar la app en un dispositivo:

- En Android Studio, cree un dispositivo virtual de Android (AVD) que el emulador pueda usar para instalar y ejecutar su app.
- En la barra de herramientas, seleccione su app en el menú desplegable de configuraciones de ejecución y depuración.
- En el menú desplegable del dispositivo de destino, seleccione el AVD en el que desea ejecutar su app.

Figura 3.10

Despliegue de una app móvil



Nota: Menú desplegable del dispositivo de destino en Android Studio.
Tomado de Android Studio, 2021.

Ejecutar o compilar la aplicación móvil haga click en Run.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Android Studio instala la app en el AVD y luego inicia el emulador. Ahora verás el mensaje “Hello, World!” en la app.

Con los conocimientos revisados hasta este momento usted ha adquirido la competencia de diseñar aplicaciones móviles en base a estándares y metodologías de desarrollo móvil creando así soluciones móviles efectivas para empresas.

Para comenzar a desarrollar su app, continúe con la siguiente sección.



Actividades de aprendizaje recomendadas

Investigue cuál es el proceso para instalar GenyMotion como emulador de Android. Para esta actividad, puede ingresar en siguiente [enlace](#) donde se encuentra la documentación del emulador, encuentre el proceso de instalación y coméntelo en el foro.

¿Cómo le fue con la actividad? Muy interesante, ¿verdad?

Continuemos con el siguiente caso de estudio para poner en práctica lo aprendido.

¡Suerte! ¡Siga adelante!

CASO DE ESTUDIO

Para poner en práctica los conocimientos adquiridos durante el avance de los contenidos de nuestra materia, en especial el desarrollo de aplicaciones móviles en Android Studio; se propone desarrollar el siguiente caso de estudio, para lo cual debe realizar la fase de desarrollo de la metodología para aplicaciones móviles.

Objetivo del caso: Se requiere la creación de una aplicación móvil que registre contactos de personas (nombres, apellidos, correo, teléfonos, dirección, parentesco, etc.) y tenga opciones para ver la fecha actual, activar cronómetro, entre otros.

Elaboración de preguntas: Cumpliendo la metodología de casos, estimado estudiante, determine el desarrollo del caso con ayuda de estas preguntas:

- ¿La aplicación permite el menú de opciones solicitadas?
- ¿La aplicación tiene un sistema de versionamiento?

Análisis y desarrollo del caso: Cree un repositorio en GitHub para el versionamiento de código y conforme siga desarrollando, suba el estado de la aplicación móvil hasta completar los requerimientos de la misma.

Elaboración de informe: Desarrolle un documento donde se argumente el proceso que utilizó para cumplir el objetivo, utilice capturas de pantallas y código para explicar su desarrollo.

Comparta y comente su trabajo desarrollado en el entorno de aprendizaje EVA.

Estimado estudiante, lo invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuál es su nivel de conocimientos obtenidos durante el tránscurso de este periodo transcurrido, en caso de no obtener resultados satisfactorios le sugiero revisar nuevamente los contenidos en los cuales tiene problemas.



Autoevaluación 3

Conteste correctamente las siguientes preguntas de la unidad 3:

1. En el proceso de instalación de Android Studio lo primero que tiene que comprobar es que:
 - a. Esté instalado Java Development Kit.
 - b. La memoria RAM del ordenador sea superior a 4Gb.
 - c. El path de instalación de Java esté en las variables globales del sistema operativo.
2. Seleccione las funciones que aumentan la productividad en el editor Android Studio:
 - a. Emulador más rápido y cargado de funciones.
 - b. Una base de datos interna para mayor productividad del dispositivo.
 - c. Variedad de marcos de trabajo y herramientas de prueba.
3. Seleccione los módulos que se incluyen en un proyecto de Android Studio
 - a. Módulo de SDK.
 - b. Módulo de versionamiento.
 - c. Módulo de apps para Android.
4. La opción de compilar el proyecto en el editor de Android Studio se encuentra en:
 - a. La barra de navegación.
 - b. La ventana de herramientas.
 - c. La barra de herramientas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. Antes de instalar Android se debe instalar:
 - a. El emulador.
 - b. El compilador de Java.
 - c. La base de datos que se va a trabajar.
6. El editor Android Studio tiene funciones como:
 - a. Un sistema de compilación basado en Gradle.
 - b. Un sistema de compilación basado en iOS.
 - c. Un sistema de compilación de C++.
7. Una función del editor Android Studio para el versionamiento de código es:
 - a. Una consola DOS para ejecutar comando de Git.
 - b. La integración con GitHub.
 - c. Una cuenta automática para subir código a Git.
8. La compatibilidad de Android Studio es con:
 - a. Java.
 - b. PHP.
 - c. C++.
9. El editor de Android Studio ofrece módulo de:
 - a. Módulo de Google Engine.
 - b. Módulo de plantillas.
 - c. Módulo de emuladores.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

10. Para la localización de archivos en el editor de Android Studio:

- a. El editor solo muestra las carpetas del proyecto.
- b. El editor muestra la estructura en la vista de proyecto de Android.
- c. El editor muestra las herramientas de desarrollo del proyecto.

[Ir al solucionario](#)

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Actividades finales del bimestre



Semana 7

En esta semana nos vamos a dedicar a revisar las temáticas estudiadas en las semanas anteriores con el fin de ir afianzando los temas previos a la evaluación presencial.

Además de eso tenemos algunas actividades calificadas que es importante que usted realice como parte del proceso de preparación, como es el chat académico que se ha programado para esta semana, además de la actividad suplementaria para el caso de los alumnos que por algún motivo no puedan realizar la actividad síncrona.



Semana 8

En la última semana del primer bimestre nos vamos a enfocar en la revisión del contenido previa la evaluación presencial, se recomienda aprovechar esta semana revisando nuevamente cada uno de los temas y si surgieran dudas por favor contactarse con el docente-tutor para resolverlas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Segundo bimestre

Resultado de aprendizaje Implementa aplicaciones para plataformas móviles.

Para lograr este resultado de aprendizaje en esta semana tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual, enfocándose en el desarrollo e implementación de elementos como ventanas, botones, etc. También se debe poner mayor esfuerzo en el desarrollo de caso de estudio que se planteó durante el desarrollo de esta materia.

Contenidos, recursos y actividades de aprendizaje



Semana 9

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Unidad 4. Aplicaciones móviles con Android Studio. Parte 2

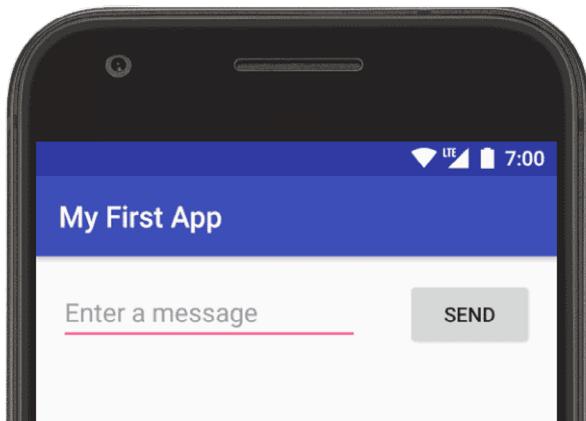
Para iniciar el estudio del tema del desarrollo de aplicaciones móviles en Android le recomendamos leer el siguiente recurso digital en el siguiente [enlace](#).

¿Qué le pareció la lectura? ¿Tiene algunas dudas?
No se preocupe, las vamos a ir resolviendo.

4.1. Construcción del FrontEnd

El desarrollo visual y estético de una aplicación móvil esta englobado en el tema del FrontEnd de una aplicación móvil, es por ello que, en esta unidad, aprenderá a usar el editor de diseño de Android Studio para crear un diseño que incluya ventanas, un cuadro de texto y un botón. De esta manera, se configura la sección siguiente, en la que puede obtener información sobre cómo hacer que la app envíe el contenido del cuadro de texto a otra actividad cuando se presione el botón tal como se ve en la figura 4.1.

Figura 4.1
Pantalla del diseño final

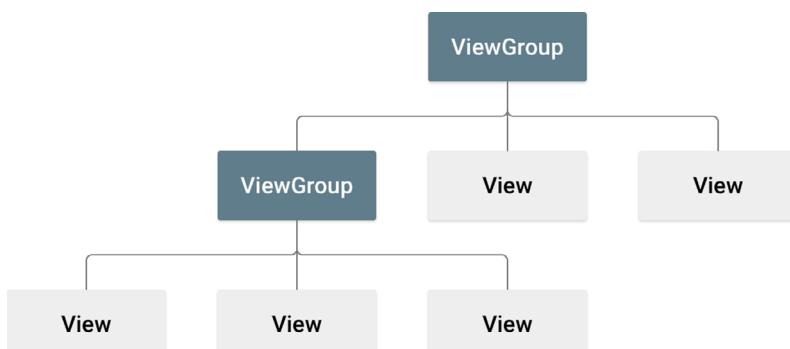


Nota: Captura de pantalla del diseño final de la app móvil. Tomado de Android Studio, 2021.

La interfaz de usuario (IU) de una app de Android se compila como una jerarquía de diseños y widgets. Los diseños son objetos ViewGroup, contenedores que controlan cómo se posicionan sus vistas secundarias en la pantalla. Los widgets son objetos View, componentes de la IU, como botones y cuadros de texto.

Figura 4.2

Objetos ViewGroup de Android Studio



Nota: Ilustración de cómo los objetos ViewGroup forman ramas en el diseño y contienen objetos View. Tomado de Android Studio, 2021.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Android proporciona un vocabulario XML para las clases ViewGroup y View, por lo que, la mayor parte de su IU se define en archivos XML. Sin embargo, en lugar de enseñarle a escribir en formato XML, en esta sección se le mostrará cómo crear un diseño con el editor de diseño de Android Studio. El editor de diseño escribe el archivo XML a medida que arrastra y suelta las vistas para compilar su diseño.

4.2. Editor de diseño

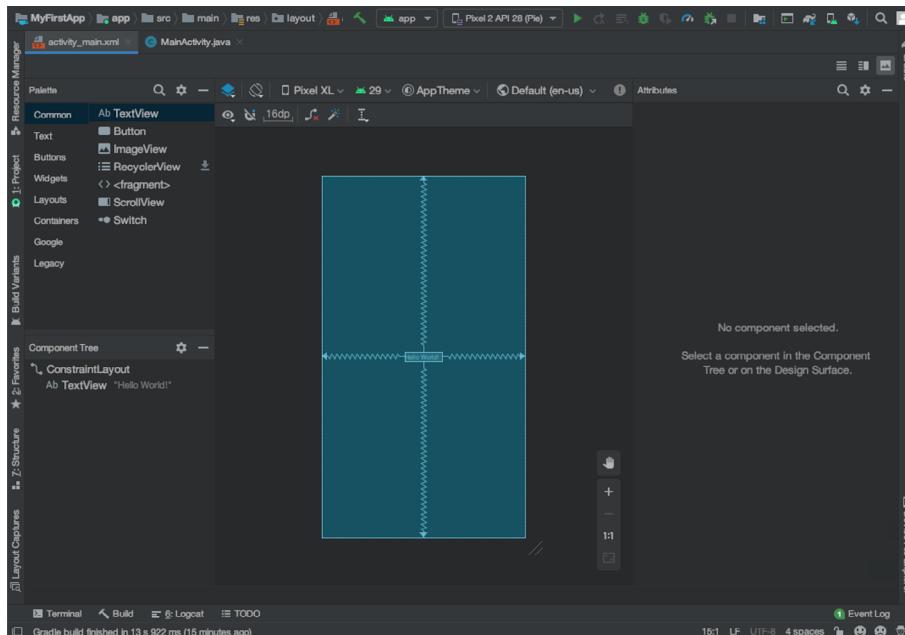
Para comenzar abriendo el editor de diseño, configure su lugar de trabajo de la siguiente manera:

- En la ventana Project, abra app > res > layout > activity_main.xml
- A fin de hacer más espacio para el editor de diseño, oculte la ventana Project. Para ello, seleccione View > Tool Windows > Project o haga click en Project a la izquierda de la pantalla de Android Studio.
- Si en el editor se muestra el código fuente XML, haga click en la pestaña Design, en la parte inferior de la ventana.
- Haga click en Select Design Surface y seleccione Blueprint.
- Haga click en Show en la barra de herramientas del editor de diseño y asegúrese de que la opción Show All Constraints esté marcada.
- Compruebe que la opción de conexión automática esté desactivada. La información sobre la barra de herramientas muestra Enable Autoconnection to Parent cuando la conexión automática está desactivada.

- Haga click en Default Margins en la barra de herramientas y seleccione 16. Si es necesario, puede ajustar los márgenes para cada vista más adelante.
- En la barra de herramientas, haga click en Device for Preview y seleccione 5.5, 1440 × 2560, 560 dpi (Pixel XL).

El editor de diseño ahora se ve como se muestra en la figura 4.3.

Figura 4.3
Activity_main de Android Studio



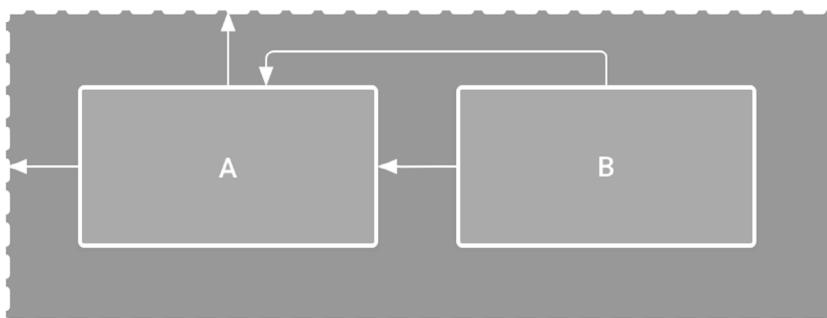
Nota: Visualización de activity_main.xml en el editor de diseño.
Tomado de Android Studio, 2021.

En el panel Component Tree en la parte inferior izquierda, se muestra la jerarquía de vistas del diseño. En este caso, la vista de raíz es un ConstraintLayout, que contiene solo un objeto TextView.

ConstraintLayout es un diseño que define la posición de cada vista a partir de restricciones sobre el diseño principal y las vistas secundarias. De esta manera, puede crear diseños simples y complejos con una jerarquía de vista plana. Este tipo de diseño evita la necesidad de diseños anidados. Un diseño anidado, que es un diseño dentro de otro, como se muestra en la figura 4.4, puede aumentar el tiempo necesario para dibujar la IU.

Figura 4.4

ConstraintLayout



Nota: Ilustración de dos vistas ubicadas dentro de un ConstraintLayout.
Tomado de Android Studio, 2021.

Por ejemplo, puede declarar el siguiente diseño, que se muestra en la figura 4.4:

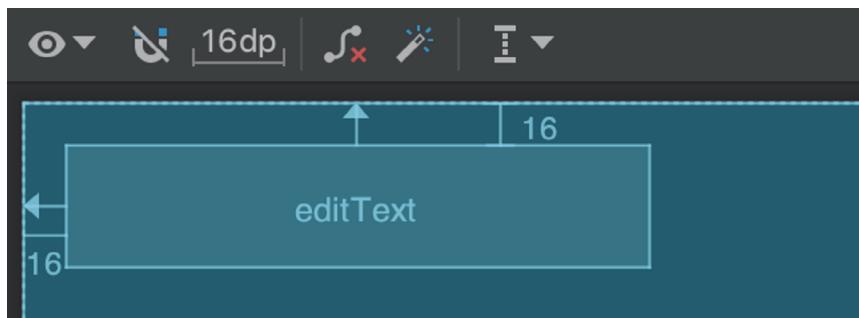
- La vista A se muestra a 16 dp de la parte superior del diseño de nivel superior.
- La vista A se muestra a 16 dp de la parte izquierda del diseño de nivel superior.
- La vista B se muestra a 16 dp a la derecha de la vista A.
- La vista B se encuentra alineada con la parte superior de la vista A.

4.3. Cuadro de texto

Para insertar un cuadro de texto siga estos pasos para agregar un cuadro de texto:

Figura 4.5

Insertar cuadro de texto en Android Studio



Nota: El cuadro de texto tiene restricciones respecto de las partes izquierda y superior del diseño de nivel superior. Tomado de Android Studio, 2021.

- Primero debe quitar todo lo que hay en el diseño. Haga click en TextView en el panel Component Tree y luego, presione la clave Delete.
- En el panel Palette, haga click en Text para mostrar los controles de texto disponibles.
- Arrastre la opción de texto sin formato al editor de diseño y suéltela cerca de la parte superior del diseño. Este es un widget EditText que acepta entradas de texto sin formato.
- Haga click en la vista del editor de diseño. Ahora puede ver los controladores cuadrados para cambiar el tamaño de la vista en cada esquina y los anclajes de restricciones circulares en cada lado. Si desea obtener un mejor control, quizás le convenga acercar el editor. Para hacerlo, use los botones Zoom en la barra de herramientas del editor de diseño.

- Mantenga presionado el anclaje en la parte superior, arrástrelo hasta que se ajuste a la parte superior del diseño y luego suéltelo. De esta manera, se limita la vista dentro del margen predeterminado que se definió. En este caso, establece el valor en 16 dp desde la parte superior del diseño.
- Use el mismo proceso para crear una restricción desde el lado izquierdo de la vista hasta el lado izquierdo del diseño.

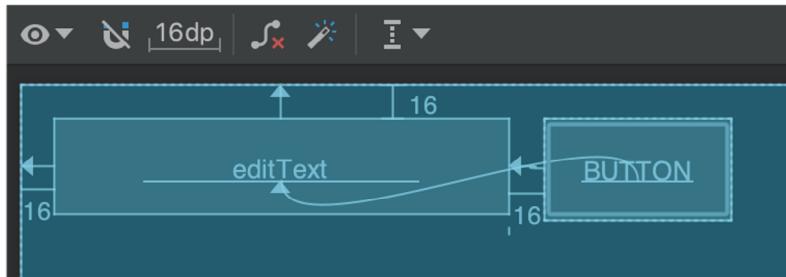
Estimado estudiante, este resultado debería verse como en la figura 4.5.

4.4. Botones

- En el panel Palette, haga click en Buttons.
- Arrastre el widget Button al editor de diseño y suéltelo cerca del lado derecho.
- Cree una restricción desde el lado izquierdo del botón hasta el lado derecho del cuadro de texto.
- Para restringir las vistas en una alineación horizontal como se indica en la figura 4.6, cree una restricción entre las líneas de base del texto. Para ello, haga click con el botón derecho y, luego, seleccione Show Baseline, como mostrar la acción del modelo de referencia en el editor de diseño. El anclaje de línea de base aparece dentro del botón. Mantenga presionado este anclaje y arrástrelo hacia el anclaje de modelo de referencia que se muestra en el cuadro de texto adyacente.

Figura 4.6

Alineación de elementos en la app móvil



Nota: El botón tiene restricciones respecto del lado derecho del cuadro de texto y de su línea de base. Tomado de Android Studio, 2021.

El resultado debería verse como en la figura 4.6.

4.5. Tamaño flexible del cuadro

Para crear un diseño que reaccione a distintos tamaños de pantalla, se debe hacer que el cuadro de texto se extienda para completar todo el espacio horizontal del dispositivo.

Antes de continuar, haga click en Select Design Surface en la barra de herramientas y seleccione Blueprint.

Para que el cuadro de texto sea flexible, siga estos pasos:

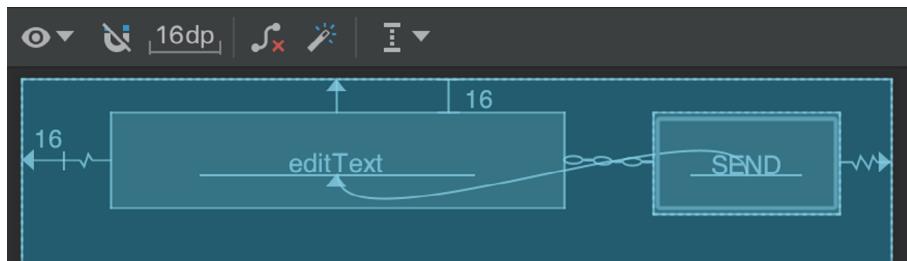
- Seleccione ambas vistas. Para ello, haga click en una, mantenga presionada la tecla Mayúsculas y luego haga click en la otra. A continuación, haga click con el botón derecho en cualquiera de ellas y seleccione Chains > Create Horizontal Chain.
- Una cadena es una restricción bidireccional entre dos o varias vistas que le permite organizar las vistas encadenadas de forma simultánea.

- Seleccione el botón y abra la ventana Attributes. Luego, use el inspector de vistas en la parte superior de la ventana Attributes para establecer el margen derecho en 16 dp tal como se muestra en la figura 4.7.
- Haga click en el cuadro de texto para ver sus atributos. Luego, haga click en el indicador de ancho dos veces para establecerlo en Match Constraints.

Match Constraints significa que el ancho se expande para cumplir con la definición de las limitaciones horizontales y los márgenes. Por lo tanto, el cuadro de texto se extiende hasta llenar el espacio horizontal que queda después del botón y se registran todos los márgenes, esto se refleja en la figura 4.8.

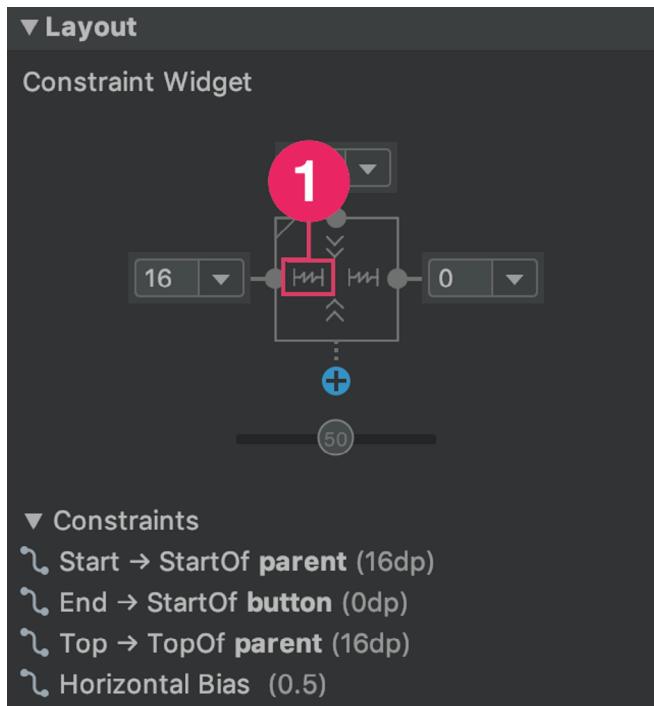
Figura 4.7

Create Horizontal Chain



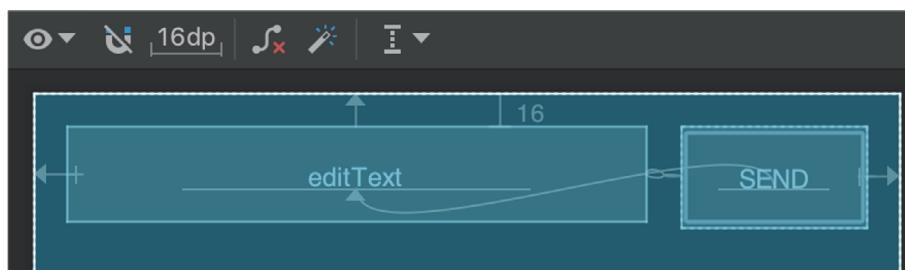
Nota: El resultado que se obtiene cuando selecciona Create Horizontal Chain. Tomado de Android Studio, 2021.

Figura 4.8
Ajuste de restricciones en Android



Nota: Haga click para cambiar el ancho a fin de ajustar a las restricciones.
Tomado de Android Studio, 2021.

Figura 4.9
Adaptación de elementos en Android



Nota: El cuadro de texto se estirará hasta llenar el espacio sobrante. Tomado de Android Studio, 2021.

El diseño quedará listo y tendrá el aspecto de la figura 4.9.

Si le parece que su diseño no quedó como esperaba, haga click en el vínculo para ver el XML final del diseño a continuación y accederá a una vista de cómo debería verse el XML. Compárela con lo que ve en la pestaña Text. (No se preocupe si sus atributos aparecen en un orden diferente; no es un error).

4.6. Interacción de actividades

Ahora, terminado el proceso anterior, tendrá una app que mostrará una actividad que consiste en una sola pantalla con un campo de texto y un botón Enviar. En este apartado podrá agregar código a MainActivity con el objetivo de iniciar una nueva actividad para mostrar un mensaje cuando el usuario presione el botón Enviar.

Cómo responder al botón Enviar

Siga estos pasos para agregar un método a la clase MainActivity a la que se llama cuando presiona el botón Enviar:

- En el archivo app > java > com.example.myfirstapp > MainActivity, agregue el siguiente método auxiliar sendMessage():Kotlin

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
  
    /** Called when the user taps the Send button */  
    fun sendMessage(view: View) {  
        // Do something in response to button  
    }  
}
```

Es posible que vea un error porque Android Studio no puede resolver la clase View que se usa como argumento del método. Para borrar el error, haga click en la declaración View, coloque el cursor sobre ella y presione Alt + Intro (o, en Mac, Opción + Intro) a fin de realizar una corrección rápida. Si aparece un menú, seleccione Import class.

- Regrese al archivo activity_main.xml para llamar al método desde el botón, de la siguiente manera:
- Seleccione el botón en el editor de diseño.
- En la ventana Attributes, ubique la propiedad onClick y seleccione sendMessage [MainActivity] en la lista desplegable.

De esta manera, cuando se presione el botón, el sistema llamará al método sendMessage().

Tome nota de los detalles de este método. Son necesarios para que el sistema reconozca el método como compatible con el atributo android:onClick. Específicamente, el método tiene las siguientes características:

- Acceso público.
 - Un valor de retorno unit vacío o, en Kotlin, implícito.
 - Un objeto View como único parámetro.
-
- A continuación, deberá completar este método para leer el contenido del campo de texto y proporcionar dicho texto a otra actividad.

4.7. Intent

Un Intent es un objeto que proporciona vinculación en tiempo de ejecución entre componentes separados, como dos actividades. El Intent representa la intención que tiene una app de realizar una tarea. Puede usar los Intents para varias tareas; pero, en esta sección, su Intent inicia otra actividad.

En MainActivity, agrega la constante EXTRA_MESSAGE y el código sendMessage(), como se muestra a continuación:

Código en Kotlin

```
const val EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE"

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    /** Called when the user taps the Send button */
    fun sendMessage(view: View) {
        val editText = findViewById<EditText>(R.id.editText)
        val message = editText.text.toString()
        val intent = Intent(this,
DisplayMessageActivity::class.java).apply {
            putExtra(EXTRA_MESSAGE, message)
        }
        startActivity(intent)
    }
}
```

Puede esperar que Android Studio vuelva a mostrar errores Cannot resolve symbol. Para borrar los errores, presione Alt + Intro o, en Mac, Opción + Regresar. Debería terminar con las siguientes importaciones:

Código en Kotlin

```
import androidx.appcompat.app.AppCompatActivity
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.EditText
```

Todavía queda un error para `DisplayMessageActivity`, pero no se preocupe porque lo solucionará en el siguiente paso.

Esto es lo que sucede en `sendMessage()`:

- El constructor `Intent` toma dos parámetros, un `Context` y un `Class`.
- El parámetro `Context` se usa primero porque la clase `Activity` es una subclase de `Context`.
- El parámetro `Class` del componente de la app, al que el sistema entrega el `Intent`, es, en este caso, la actividad que va a comenzar.
- El método `putExtra()` agrega el valor de `EditText` al `intent`. Un `Intent` puede transportar tipos de datos como pares clave-valor llamados extras.
- Su clave es una constante pública `EXTRA_MESSAGE` porque la actividad siguiente usa la clave para obtener el valor de texto. Le recomendamos definir las claves para extras de `Intents` con el nombre del paquete de la app como prefijo. De esta manera, se garantiza que las claves sean únicas en el caso de que su app interactúe con otras apps.
- El método `startActivity()` inicia una instancia de `DisplayMessageActivity` que especifica el `Intent`. Luego, deberá crear esa clase.

4.8. Actividad secundaria

Para crear la segunda actividad, y poder visualizar otra pantalla en nuestra aplicación móvil se procede de la siguiente manera:

- En la ventana Project, haga click con el botón derecho en la carpeta de la app y seleccione New > Activity > Empty Activity.
- En la ventana Configure Activity, ingrese "DisplayMessageActivity" en Activity Name. Deje todas las demás propiedades con sus valores predeterminados y haga click en Finish.

Android Studio realiza tres acciones automáticamente:

- Crea el archivo DisplayMessageActivity.
- Crea el archivo de diseño activity_display_message.xml, que se corresponde con el archivo DisplayMessageActivity.
- Agrega el elemento <activity> obligatorio en AndroidManifest.xml.

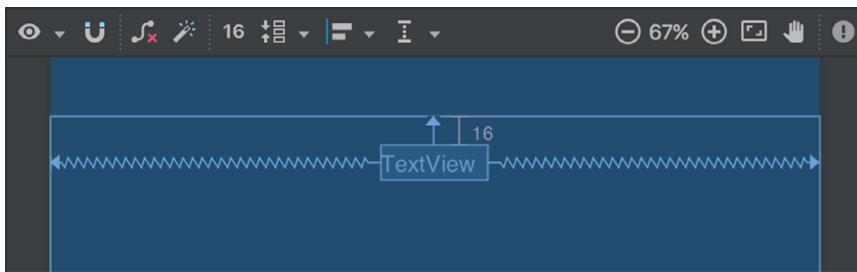
Si ejecuta la app y presiona el botón en la primera actividad, se iniciará la segunda, pero estará vacía porque usa el diseño vacío proporcionado por la plantilla.

4.9. Vistas de texto en la app

La vista de texto centrada en la parte superior del diseño

Figura 4.10

Texto centrado en la parte superior del diseño



Nota: La vista de texto centrada en la parte superior del diseño. Tomado de Android Studio, 2021.

La nueva actividad incluye un archivo de diseño en blanco. Siga estos pasos para agregar una vista de texto al lugar donde aparece el mensaje:

- Abra el archivo app > res > diseño > activity_display_message.xml.
- Haga click en Enable Autoconnection to Parent en la barra de herramientas. De esta manera, se habilita la conexión automática.
- En el panel Palette, haga click en Text, arrastre un elemento TextView al diseño y suéltelo cerca de la parte central superior del diseño de manera que se acople a la línea vertical que aparecerá. La opción de conexión automática agrega restricciones a la derecha y a la izquierda para ubicar la vista en el centro horizontal.
- Cree una restricción más desde la parte superior de la vista de texto hasta la parte superior del diseño para que se vea como en la figura 4.10.

De manera opcional, puede realizar algunos ajustes en el estilo de texto si expande textAppearance en el panel Common Attributes de la ventana Attributes y cambia los atributos como textSize y textColor.

4.10. Mensajes

En este paso, puede modificar la segunda actividad para mostrar el mensaje que pasó la primera actividad.

En DisplayMessageActivity, agregue el siguiente código al método onCreate():

Código en Kotlin

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_display_message)

    // Get the Intent that started this activity and extract the string
    val message = intent.getStringExtra(EXTRA_MESSAGE)

    // Capture the layout's TextView and set the string as its text
    val textView = findViewById<TextView>(R.id.textView).apply {
        text = message
    }
}
```

Presione Alt + Intro o, en Mac, Opción + Regresar para importar estas otras clases necesarias:

Código en Kotlin

```
import androidx.appcompat.app.AppCompatActivity
import android.content.Intent
import android.os.Bundle
import android.widget.TextView
```

4.11. Navegación ascendente

En cada pantalla de su app que no sea el punto de entrada principal, es decir, todas las pantallas que no sean la “pantalla principal”, se debe proporcionar navegación para que el usuario pueda regresar a la pantalla superior lógica en la jerarquía de la app. Para ello, agregue un botón Arriba en la barra de la app.

Para agregar un botón Arriba, debe declarar qué actividad es la lógica principal en el archivo AndroidManifest.xml. Abra el archivo que se encuentra en app > manifests > AndroidManifest.xml, busque la etiqueta <activity> para DisplayMessageActivity y reemplácela por lo siguiente:

```
<activity android:name=".DisplayMessageActivity"
    android:parentActivityName=".MainActivity">
    <!-- The meta-data tag is required if you support API level 15 and
lower -->
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

El sistema Android agregará automáticamente el botón Arriba en la barra de app.

Resultado al ejecutar la app

Al realizar click en el apartado Apply Changes en la barra de herramientas para ejecutar la app. Cuando se abra, escriba un mensaje en el campo de texto y presione Send para ver el mensaje en la segunda actividad.

La figura 4.11 de la app abierta, con texto que se ingresa a la izquierda de la pantalla y se muestra a la derecha

Figura 4.11
Resultados de la app móvil con los elementos



Nota: Imagen de la app abierta, con texto que se ingresa a la izquierda de la pantalla y se muestra a la derecha. Tomado de Android Studio, 2021.



Semana 10

4.12. Aspectos fundamentales de la aplicación

Para escribir aplicaciones de Android, es posible usar los lenguajes Kotlin, Java y C++. Las herramientas de Android SDK compilan su código, junto con los archivos de recursos y datos, en un APK: un paquete de Android, que es un archivo de almacenamiento con el sufijo .apk. Un archivo APK incluye todos los contenidos de una aplicación de Android y es el archivo que usan los dispositivos con tecnología Android para instalar la aplicación.

Cada aplicación de Android reside en su propia zona de pruebas de seguridad y está protegida por las siguientes características de seguridad de Android:

- El sistema operativo Android es un sistema Linux multiusuario en el que cada aplicación es un usuario diferente.
- De forma predeterminada, el sistema le asigna a cada aplicación un ID de usuario de Linux único (solo el sistema utiliza el ID y la aplicación lo desconoce). El sistema establece permisos para todos los archivos en una aplicación de modo que solo el ID de usuario asignado a esa aplicación pueda acceder a ellos.
- Cada proceso tiene su propia máquina virtual (VM), por lo que el código de una aplicación se ejecuta de forma independiente de otras aplicaciones.
- De forma predeterminada, cada aplicación ejecuta su propio proceso de Linux. El sistema Android inicia el proceso cuando se requiere la ejecución de alguno de los componentes de la aplicación y, luego, lo cierra cuando el proceso ya no es necesario o cuando el sistema debe recuperar memoria para otras aplicaciones.

De esta manera, el sistema Android implementa el principio de mínimo privilegio. Es decir, de forma predeterminada, cada aplicación tiene acceso solo a los componentes que necesita para llevar a cabo su trabajo y nada más. Esto crea un entorno muy seguro, en el que una aplicación no puede acceder a partes del sistema para las que no tiene permiso. Sin embargo, hay maneras en las que una aplicación puede compartir datos con otras aplicaciones y en las que una aplicación puede acceder a servicios del sistema:

- Es posible coordinar que dos aplicaciones compartan el mismo ID de usuario de Linux para que puedan acceder a los archivos de la otra. Para conservar recursos del sistema, las aplicaciones con el mismo ID de usuario también pueden coordinar la ejecución en el mismo proceso de Linux y compartir la misma VM. Las aplicaciones también deben estar firmadas con el mismo certificado.

- Una aplicación puede solicitar permiso para acceder a datos del dispositivo, como los contactos de un usuario, los mensajes de texto, el dispositivo de almacenamiento (tarjeta SD), la cámara y la conexión Bluetooth. El usuario debe conceder de manera explícita estos permisos.

A continuación, se describen los siguientes conceptos:

- Los componentes del marco de trabajo central que definen su aplicación.
- El archivo de manifiesto en el que declara los componentes y las funciones necesarias del dispositivo para su aplicación.
- Los recursos que son independientes del código de la aplicación y que permiten que su aplicación optimice correctamente su comportamiento en diversas configuraciones de dispositivos.

4.13. Componentes de la aplicación

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

Las aplicaciones tienen cuatro tipos de componentes diferentes:

- Actividades.
- Servicios.
- Receptores de emisiones.
- Proveedores de contenido.

Cada tipo tiene un fin específico y un ciclo de vida característico que define cómo se crea y se destruye el componente. En las siguientes secciones, se detallan los cuatro tipos de componentes de la aplicación.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4.14. Actividades

Una actividad es el punto de entrada de interacción con el usuario. Representa una pantalla individual con una interfaz de usuario. Por ejemplo, una aplicación de correo electrónico tiene una actividad que muestra una lista de los correos electrónicos nuevos, otra actividad para redactar el correo electrónico y otra actividad para leerlo. Si bien las actividades funcionan juntas para ofrecer una experiencia de usuario uniforme en la aplicación de correo electrónico, cada una es independiente de las demás. De esta manera, una aplicación diferente puede iniciar cualquiera de estas actividades (si la aplicación de correo electrónico lo permite). Por ejemplo, para que el usuario comparta una imagen, una aplicación de cámara puede iniciar la actividad correspondiente en la aplicación de correo electrónico que redacta el nuevo mensaje. Una actividad posibilita las siguientes interacciones clave entre el sistema y la aplicación:

- Realizar un seguimiento de lo que realmente le interesa al usuario (lo que está en pantalla) para garantizar que el sistema siga ejecutando el proceso que aloja la actividad.
- Saber que los procesos usados con anterioridad contienen elementos a los que el usuario puede regresar (actividades detenidas) y, en consecuencia, priorizar más esos procesos que otros.
- Ayudar a la aplicación a controlar la finalización de su proceso para que el usuario pueda regresar a las actividades con el estado anterior restaurado.
- Permitir que las aplicaciones implementen flujos de usuarios entre sí y que el sistema los coordine (el ejemplo más común es compartir).

Las actividades se implementan como subclases de la clase Activity.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4.15. Servicios

Un servicio es un punto de entrada general que permite mantener la ejecución de una aplicación en segundo plano por diversos motivos. Es un componente que se ejecuta en segundo plano para realizar operaciones de ejecución prolongada o para realizar tareas de procesos remotos. Un servicio no proporciona una interfaz de usuario. Por ejemplo, un servicio podría reproducir música en segundo plano mientras el usuario se encuentra en otra aplicación, o podría capturar datos en la red sin bloquear la interacción del usuario con una actividad. Otro componente, como una actividad, puede iniciar el servicio y permitir que se ejecute o enlazarse a él para interactuar. De hecho, existen dos semánticas muy diferentes que los servicios usan para indicarle al sistema cómo administrar una aplicación: Los servicios iniciados le indican al sistema que los siga ejecutando hasta que finalicen su trabajo. Dos ejemplos serían la sincronización de datos en segundo plano o la reproducción de música después de que el usuario sale de la aplicación. La sincronización de datos en segundo plano o la reproducción de música también son dos tipos de servicios iniciados muy diferentes que modifican la manera en que el sistema los administra:

- La reproducción de música es algo de lo que el usuario es consciente, por lo que la aplicación se lo comunica al sistema indicándole que quiere ejecutarse en segundo plano y le envía una notificación al respecto al usuario. En este caso, el sistema sabe que debe hacer todo lo posible para mantener el proceso de ese servicio en funcionamiento porque el usuario no estará satisfecho si se interrumpe.
- Un servicio habitual en segundo plano no es algo de lo que el usuario sea consciente, por lo que el sistema tiene más libertad para administrarlo. Puede permitir que se interrumpe (y reiniciarlo posteriormente) si necesita memoria RAM en procesos que son más urgentes para el usuario.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Los servicios enlazados se ejecutan porque otra aplicación (o el sistema) indicó que quiere usarlos. Básicamente, lo que sucede es que un servicio le brinda una API a otro proceso. Por lo tanto, el sistema sabe que hay una dependencia entre estos procesos. En consecuencia, si el proceso A está enlazado a un servicio en el proceso B, sabe que debe mantener funcionando el proceso B (y el servicio correspondiente) para el proceso A. Además, si el proceso A es de interés para el usuario, también sabe que debe tratar el proceso B teniendo esto en cuenta. Gracias a su flexibilidad (o a pesar de ella), los servicios se han convertido en un componente muy útil para todos los tipos de conceptos generales del sistema. Los fondos de pantalla animados, los receptores de notificaciones, los protectores de pantalla, los métodos de entrada, los servicios de accesibilidad y muchas otras funciones básicas del sistema se compilan como servicios que las aplicaciones implementan y que el sistema vincula cuando deben ejecutarse.

Un servicio se implementa como una subclase de Service.

4.16. Activación de componentes

De los cuatro tipos de componentes, tres (actividades, servicios y receptores de emisión) se activan mediante un mensaje asíncrono denominado Intent. Las Intents vinculan componentes entre sí durante el tiempo de ejecución. Imagine que son los mensajeros que solicitan acciones de otros componentes, ya sea que estos pertenezcan a su aplicación o a alguna otra.

Una Intent se crea con un objeto Intent, que define un mensaje para activar un componente específico o un tipo específico de componente.

Activación de componentes

Para actividades y servicios, una Intent define la acción que se realizará (por ejemplo, ver o enviar algo) y puede especificar el URI de los datos en los que debe actuar, entre otras cosas, que el componente que se está iniciando podría necesitar saber. Por ejemplo, una intent podría transmitir una solicitud para que una actividad muestre una imagen o abra una página web. En algunos casos, puede iniciar una actividad para recibir un resultado. En ese caso, dicha actividad también devuelve el resultado en una Intent. Por ejemplo, puede emitir una Intent para permitir que el usuario elija un contacto personal y se lo devuelva. Esa Intent devuelta incluye un URI dirigido al contacto elegido.

En el caso de los receptores de emisión, la Intent tan solo define el anuncio que se emite. Por ejemplo, una emisión para indicar que el nivel de batería del dispositivo es bajo incluye solo una cadena de acción conocida que indica batería baja.

A diferencia de las actividades, los servicios y los receptores de emisión, los proveedores de contenido no se activan con Intents. En cambio, se activan mediante solicitudes de un ContentResolver. El solucionador de contenido aborda todas las transacciones directas con el proveedor de contenido, de modo que el componente que realiza las transacciones con el proveedor no deba hacerlo y, en su lugar, llame a los métodos del objeto ContentResolver. Esto deja una capa de abstracción entre el proveedor de contenido y el componente que solicita información (por motivos de seguridad).

Existen métodos independientes para activar cada tipo de componente:

- Puedes iniciar una actividad o asignarle una tarea nueva al pasar un Intent a startActivity() o startActivityForResult() (cuando quiere que la actividad devuelva un resultado).

- Con Android 5.0 (nivel de API 21) y las versiones posteriores, puede usar la clase JobScheduler para programar acciones. En versiones anteriores de Android, puede iniciar un servicio (o darle instrucciones nuevas a un servicio en curso) al pasar un Intent a startService(). Puede establecer un enlace con el servicio al pasar un Intent a bindService().
- Puede iniciar una emisión al pasar un Intent a métodos como sendBroadcast(), sendOrderedBroadcast() o sendStickyBroadcast().
- Puede realizar una consulta a un proveedor de contenido si llama a query() en un ContentResolver.

4.17. El archivo de manifiesto

Para que el sistema Android pueda iniciar un componente de la aplicación, debe reconocer la existencia de ese componente leyendo el archivo de manifiesto de la aplicación (AndroidManifest.xml). Su aplicación debe declarar todos sus componentes en este archivo, que debe encontrarse en la raíz del directorio de proyectos de la aplicación.

El manifiesto puede hacer ciertas cosas además de declarar los componentes de la aplicación, por ejemplo:

- Identificar los permisos de usuario que requiere la aplicación, como acceso a Internet o acceso de lectura para los contactos del usuario.
- Declarar el nivel de API mínimo que requiere la aplicación en función de las API que usa.
- Declarar características de hardware y software que la aplicación usa o exige, como una cámara, servicios de Bluetooth o una pantalla multitáctil.

- Declarar bibliotecas de la API a las que la aplicación necesita estar vinculada (además de las API del marco de trabajo de Android), como la biblioteca de Google Maps.

4.18. Declaración de componentes

La tarea principal del manifiesto es informarle al sistema acerca de los componentes de la aplicación. Por ejemplo, un archivo de manifiesto puede declarar una actividad de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
            ...
        </activity>
    ...
</application>
</manifest>
```

En el elemento `<application>`, el atributo `android:icon` señala los recursos de un ícono que identifica la aplicación.

En el elemento `<activity>`, el atributo `android:name` especifica el nombre de clase plenamente calificado de la subclase Activity, y el atributo `android:label` especifica una cadena para usar como etiqueta de la actividad visible para el usuario.

Debes declarar todos los componentes de la aplicación mediante estos elementos:

- Elementos `<activity>` para las actividades.
- Elementos `<service>` para los servicios.
- Elementos `<receiver>` para los receptores de emisión.
- Elementos `<provider>` para los proveedores de contenido.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Si incluye actividades, servicios y proveedores de contenido en su archivo de origen, pero no los declara en el manifiesto, no estarán visibles para el sistema y, por consiguiente, no se podrán ejecutar. No obstante, los receptores de emisión pueden declararse en el manifiesto o crearse dinámicamente en forma de código como objetos BroadcastReceiver y registrarse en el sistema llamando a registerReceiver().



Actividades de aprendizaje recomendadas

Una vez compilado el proyecto inicial, localice el archivo .apk que es el instalador y ejecútelo en su dispositivo móvil; comente los resultados en el foro de trabajo.

Para esta actividad tiene que investigar y examinar dentro de los directorios del proyecto el archivo en mención.

Puede [descargar](#) el siguiente recurso donde encontrará un ejemplo completo de una aplicación en Android donde se muestra la interacción de ventanas, botones y ejecuta funciones de almacenamiento que le ayudarán a entender mucho mejor el desarrollo de una aplicación en Android Studio.

CASO DE ESTUDIO

Para poner en práctica los conocimientos adquiridos durante el avance de los contenidos de nuestra materia, en especial el desarrollo de aplicaciones móviles en Android Studio; se propone desarrollar el siguiente caso de estudio, para lo cual debe finalizar la fase de desarrollo de la metodología para aplicaciones móviles.

Objetivo del caso: Se requiere la creación de una aplicación móvil que registre contactos de personas (nombres, apellidos, correo, teléfonos, dirección, parentesco, etc.) y tenga opciones para ver la fecha actual, activar cronómetro, entre otros.

Elaboración de preguntas: Cumpliendo la metodología de casos, estimado estudiante, determine el desarrollo del caso con ayuda de estas preguntas:

- ¿La aplicación permite el menú de opciones solicitadas?
- ¿Las opciones de cada menú cumplen correctamente su funcionamiento?

Análisis y desarrollo del caso: Documente en base a su cuenta del repositorio en GitHub para el versionamiento de código, el número de actualizaciones del proyecto que realizó para completar la aplicación, argumente el estado final de la aplicación mediante capturas de pantallas de la aplicación y códigos, ¿cuál fue su experiencia al terminar el proyecto en Android Studio?

Elaboración de informe: Desarrolle un documento donde se argumente el proceso que utilizó para cumplir el objetivo, utilice capturas de pantallas y código para explicar su desarrollo.

Comparta y comente su trabajo desarrollado en el entorno de aprendizaje EVA.

Hemos terminado la semana 10 y vamos avanzando con los contenidos de nuestra materia.

Estimado estudiante, le invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuál es su nivel de conocimientos obtenidos durante el tránscurso de este periodo transcurrido, en caso de no obtener resultados satisfactorios le sugiero revisar nuevamente los contenidos en los cuales tiene problemas

¡Mucha suerte!



Autoevaluación 4

Conteste las siguientes preguntas de la unidad 4

1. La interfaz de usuario (IU) de una app en Android se compila como:
 - a. Una jerarquía de diseños y widgets.
 - b. Una secuencia aleatoria de las actividades y widgets.
 - c. Una jerarquía de actividades.
2. Un ViewGroup es:
 - a. Contenedores que controlan los resultados y comportamientos en la pantalla.
 - b. Contenedores que controlan las acciones en la pantalla.
 - c. Contenedores que controlan las posiciones en la pantalla.
3. Para las clases ViewGroup utiliza el vocabulario:
 - a. JSON.
 - b. XML.
 - c. Kotlin.
4. El panel component tree muestra y se encuentra en:
 - a. Parte inferior izquierda y muestra la jerarquía de vista de diseño.
 - b. Parte inferior derecha y muestra la jerarquía de vista de diseño.
 - c. Parte inferior izquierda y muestra el xml de las actividades.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. Para colocar un botón flexible se utiliza el formato de medida:
 - a. cm.
 - b. px.
 - c. dp.
6. Para editar el archivo de diseño del proyecto es:
 - a. Project, abre app > res > layout > activity_main.xml
 - b. Project, abre app > res > layout > activity_main.java
 - c. Project, abre app > res > desing > activity_main.xml
7. El editor puede gestionar el diseño de la app mediante:
 - a. Solo código.
 - b. Solo gráficamente.
 - c. Con el código en xml o gráficamente.
8. El panel component tree muestra:
 - a. La jerarquía de vistas de diseño.
 - b. Los widgets que se puede usar.
 - c. Los botones o ventanas utilizados.
9. Para colocar un cuadro de texto se utiliza:
 - a. String.
 - b. TextComponent.
 - c. TextView.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

10. Para restringir la vista horizontalmente se configura:

- a. Utilizar la propiedad Show Baseline.
- b. Utilizar la propiedad Show bellow.
- c. Utilizar la propiedad Show inline.

[Ir al solucionario](#)

Las respuestas a esta autoevaluación se encuentran al final de la presente guía didáctica, acuda y compare las respuestas, si no logró un buen resultado en la autoevaluación, no se preocupe le recomiendo leer nuevamente el/los capítulos confusos y reforzar sus conocimientos. Y si aún tiene inquietudes no dude en preguntar al profesor.

¿Cómo le fue en su evaluación? ¡Espero que muy bien!

Lo invito a seguir participando y revisando los contenidos de esta guía, a continuación, tenemos el proceso de instalación de la herramienta de desarrollo. Mucha suerte.!

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



Semana 11

Para lograr este resultado de aprendizaje del segundo bimestre tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual enfocándose especialmente en la instalación y configuración del framework de desarrollo Flutter el cual nos ayudará a implementar la solución móvil en diferentes plataformas.

Continuamos avanzando con los contenidos de nuestra materia, ahora con un nuevo tema:



Unidad 5. Aplicaciones móviles con Flutter

5.1. ¿Qué es Flutter?

Flutter es un framework propiedad de Google que permite crear aplicaciones multiplataforma de forma nativa en el lenguaje Dart.

Este framework es un SDK (Software Development Kit) que incluye librerías, elementos de UI y herramientas de testeo para que nuestra aplicación sea fácil y eficiente.

5.2. Características

- Es un framework que nos permite crear aplicaciones multiplataforma de forma nativa.
- El framework, a pesar de estar creado en C++, utiliza un único lenguaje llamado Dart para crear aplicaciones para todas las plataformas. Al estar su núcleo creado con C++ permite que nuestras aplicaciones sean rápidas.
- Tiene elementos ya predefinidos de interfaz de usuario que nos facilitan el desarrollo como botones, barras de navegación, etc.
- Más rápido y eficiente que Ionic o React Native.
- Es un framework que tiene la cualidad de Hot reload lo que permite que mientras ejecuta la aplicación en el emulador, los cambios se ven al instante en ella.
- Tiene elementos ya predefinidos de interfaz de usuario que nos facilitan el desarrollo como botones, barras de navegación, etc.
- Posee un motor propio de renderizado de aplicaciones basado en Skia, lo que implica que no utiliza las Web Views ni los OEM Widgets de los dispositivos ya que posee sus propias librerías de widgets.
- Además, es el framework oficial para Fucshia el primer sistema operativo desarrollado por Google que no está basado en Linux/Unix.

5.3. Instalación

Vamos avanzando en este maravilloso mundo del desarrollo de aplicaciones móviles; a continuación, vamos al proceso de instalación de la herramienta de desarrollo. Para el proceso de instalación, inicialmente se debe trabajar con un entorno de versionamiento.

- Descarga [Git For Windows¹](#).
- Instálelo, la configuración por defecto es más que suficiente.

Configurar el SDK de Flutter

- Descargue el archivo .zip del SDK de Flutter.
- Extraiga el archivo .zip en el disco C. Quedará así **C:\flutter**.
- Presione la tecla Windows y en la barra de búsqueda digite **variable**.
- Seleccione Editar las variables de entorno del sistema.
- Dé click en Variables de entorno.
- En Variables del sistema, seleccione la variable Path y luego click en Editar.
- Dé click en Nueva y pegue la ruta de la carpeta bin de Flutter: **C:\flutter\bin**
- Ahora dé click en Aceptar en las 3 ventanas para guardar cambios.

Instalar Android Studio

- Descargue el [Instalador de Android Studio](#).
- Ejecute el instalador.
- Dé click en Next.
- Marque todas las casillas y dé click en Next 2 veces más.
- Dé click en Install.
- Cuando finalice, dé click en Next y luego en Finish.

1. Software de versionamiento de código.

Índice

Primer bimestre

Segundo bimestre

Solucionario

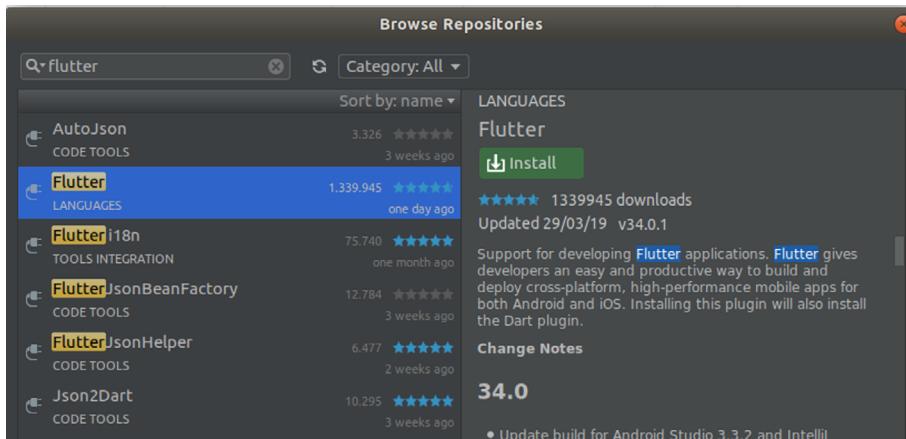
Referencias bibliográficas

- Ahora seleccione Do not import settings y dé click en **OK**.
- Se abrirá Android Studio Setup Wizard.
- Dé click en Next.
- Seleccione Custom y nuevamente click en Next.
- Seleccione el tema del editor (Le recomiendo Darcula) y dé click en Next.
- Ahora descargue el SDK de Android, para ello marque todas las casillas.
- Dé click en el botón que tiene 3 puntos “...”.
- Se abrirá el explorador de archivos.
- Seleccione la carpeta de su usuario de Windows; dentro, cree una carpeta llamada Android, y dentro de esta, una carpeta llamada SDK, y luego seleccione la carpeta SDK y dé click en OK. **C:\Users\<TU_USUARIO>\Android\SDK**
- Ahora dé click en Next dos veces y luego click en Finish.
- Cuando finalice la descarga, haga click en Finish y se abrirá Android Studio.

Instalar el Plugin de Flutter en Android Studio

- En la ventana Welcome to Android Studio dé click en Configure y luego en Plugins.
- En la barra de búsqueda digite flutter y dé click en Search in repositories.
- Dé click en el que dice Flutter, luego en Install, Accept y Yes como lo muestra la figura 5.1.

Figura 5.1
Flutter en Android Studio



Nota: Instalación de Flutter en Android Studio. Tomado de Android Studio, 2021.

- Ahora dé click en Restart Android Studio, luego en OK, y en Restart.
- Se abrirá nuevamente Android Studio.

Agregar el SDK de Android a Flutter

- Abrimos nuevamente las variables de entorno.
- Y en variables del sistema, dé click en Nueva.
- En nombre de la variable, digite ANDROID_HOME.
- Dé click en Examinar directorio y seleccione la ruta en donde descargarse el SDK de Android:
C:\Users\<TU_USUARIO>\Android\SDK
- Ahora dé click en Aceptar en las 3 ventanas para guardar cambios.
- Presione la tecla Windows y en la barra de búsqueda digite cmd y seleccione la Command Prompt o Símbolo del sistema.
- Digite el comando flutter doctor y presione Enter como lo muestra la figura 5.2.

Figura 5.2

Flutter en la consola de instalación

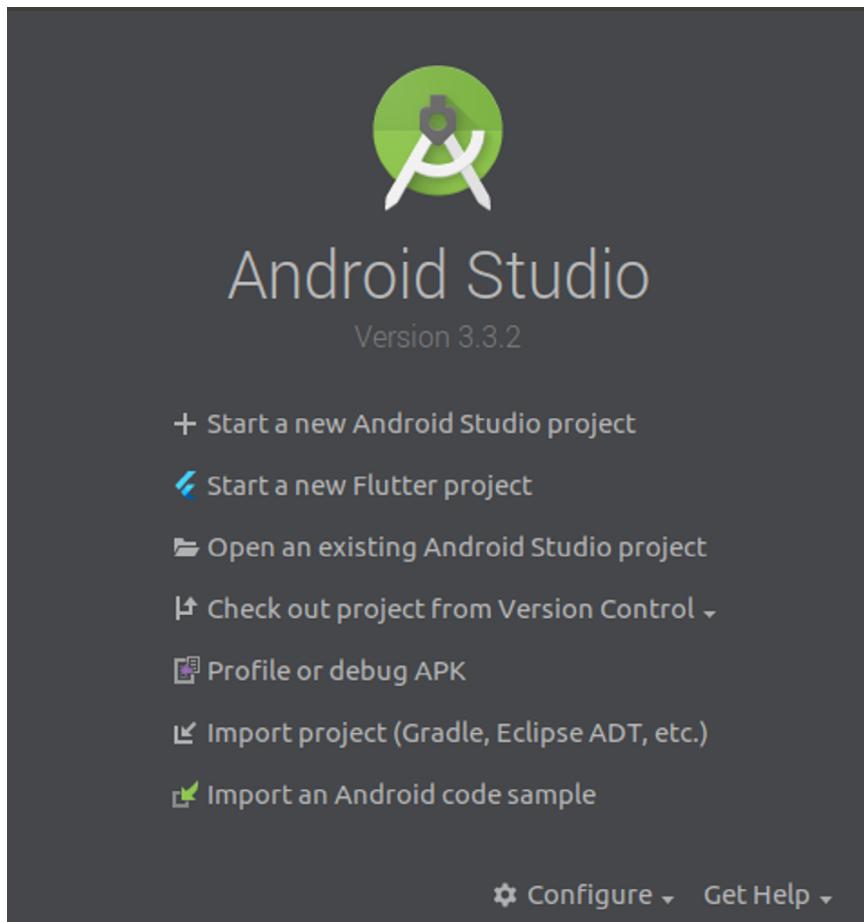
```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Linux, locale es_ES.UTF-8)
[✓] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[✓] Android Studio (version 3.3)
[✓] Connected device (1 available)

• No issues found!
```

Nota: Ejecución de flutter en la consola de instalación. Tomado de Android Studio, 2021.

- Notará que le falta aceptar las licencias de Android, así que digite el siguiente comando y presione Enter:
flutter doctor --android-licenses
- Ahora le preguntará si acepta cada licencia, en la cual debe digitar Y y presionar Enter para cada licencia.
- Cuando haya aceptado todas las licencias le saldrá All SDK package licenses accepted y cierra la ventana. Finalmente se mostrará el resultado como se visualiza en la figura 5.3.

Figura 5.3
Proyectos en Flutter



Nota: Creación de nuevos proyectos en Flutter. Tomado de Android Studio, 2021.



Actividades de aprendizaje recomendadas

Realice el proceso de instalación y configuración en su máquina personal, luego comente los resultados en el foro de trabajo.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Para esta actividad puede revisar el siguiente [enlace](#) donde encontrará la documentación detallada de Flutter, puede ayudarse con capturas de pantalla para ir documentando el proceso de instalación en su máquina local.

Terminada esta tarea, podemos pasar al siguiente tema que es la Creación de proyectos en Flutter:



Semana 12

Para lograr este resultado de aprendizaje, en esta semana tiene que revisar los contenidos y realizar la práctica en su computador personal o laboratorio virtual y poniendo en práctica los ejemplos en el desarrollo de proyectos en Flutter.

5.4. Primer proyecto en Flutter

Para empezar a trabajar con Visual Studio debemos añadir la extensión Flutter a este. Esta extensión también añade la extensión para el lenguaje Dart, que como hemos visto es el que utiliza Flutter.

Para comprobar que todo ha ido bien podemos utilizar Flutter Doctor a través de Command Palette de Visual Studio; para esto utilizamos Ctrl+shift+p y tecleamos doctor para seleccionar Flutter: Run Flutter Doctor. Ahora en la consola de Visual Studio nos devolverá el retorno de Flutter Doctor.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Ahora que todo funciona volvemos a la Command Palette, en la cual ahora escribimos flutter para seleccionar Flutter: New Project, el cual nos pedirá el nombre de nuestra aplicación y el directorio donde crearla. Este comando nos creará una aplicación básica de Flutter a partir de la cual realizaremos unas modificaciones para observar cómo funciona el HotReload de Flutter.

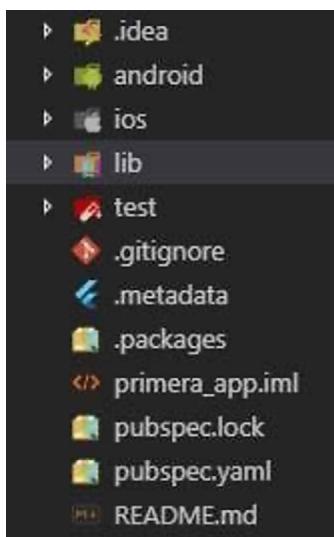
5.5. Emulador

Ahora intentaremos iniciar nuestra aplicación sobre un dispositivo móvil, en nuestro caso vamos a crear un emulador en Android Studio para utilizarlo para esto. Lo primero que tenemos que hacer es iniciar Android Studio y seleccionar Configure > AVD Manager como se muestra en el recurso.

[Configuración de Android Studio para el emulador](#)

5.6. Estructura del proyecto

Ahora que vemos que el proyecto funciona, vamos a hablar de la estructura de este. Como se muestra en la figura 5.12, puede hacer uso de este recurso, [descargando](#) o revisando en [internet](#) el proyecto básico para hacer referencia al contenido estudiado:

Figura 5.12*Estructura del proyecto en Flutter*

Nota: Estructura del proyecto en Flutter. Tomado de Android Studio, 2021.

Las carpetas más importantes son:

- Android: aquí se despliega la aplicación para Android.
- iOS: en la cual se añade el proyecto para iOS.
- Test: para testear nuestro proyecto.
- lib: la carpeta principal del proyecto de Flutter, en la cual nos encontramos.

Dentro de lib está el archivo main, que es el archivo principal de la aplicación de Flutter a partir del que realizaremos toda la aplicación.

5.7. Aplicación en Flutter

Después de instalar Flutter y el entorno de desarrollo (o editor) deseado, puedes crear tu primera aplicación en Flutter. Como ya hemos mencionado, en este tutorial de Flutter utilizamos Android

Índice

Primer bimestre

Segundo bimestre

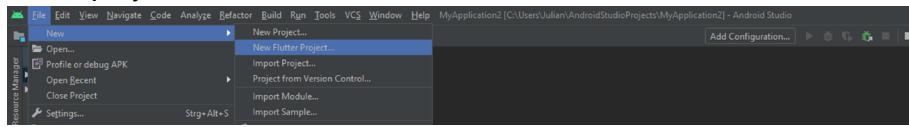
Solucionario

Referencias bibliográficas

Studio, por lo que ahora lanzaremos el IDE para este propósito. Abre la pestaña “File”, selecciona “New” y, luego, “New FlutterProject”, tal como se muestra en la figura 5.13 para iniciar un nuevo proyecto basado en el marco.

Figura 5.13

Nuevo proyecto en Flutter

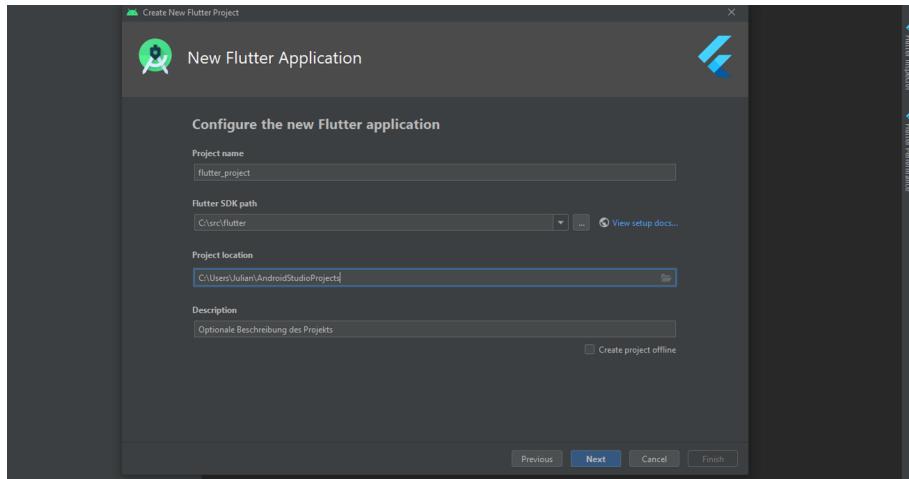


Nota: Crear un nuevo proyecto en Flutter. Tomado de Android Studio, 2021.

Seleccione “Flutter Application” como tipo de proyecto deseado y pulse “Next”. En el menú de configuración, defina ahora un título de trabajo y la ubicación local donde se guardará la aplicación. También puede añadir una descripción del proyecto. En la figura 5.14 se muestra la configuración de la línea “Flutter SDK path”, el cual indica la ruta al marco de Flutter instalado.

Figura 5.14

La ruta al SDK de Flutter



Nota: La ruta al SDK Flutter es personalizable y depende de dónde haya guardado el kit en su sistema (en este tutorial de Flutter, C:\src\Flutter). Tomado de Android Studio, 2021.

Índice

Primer bimestre

Segundo bimestre

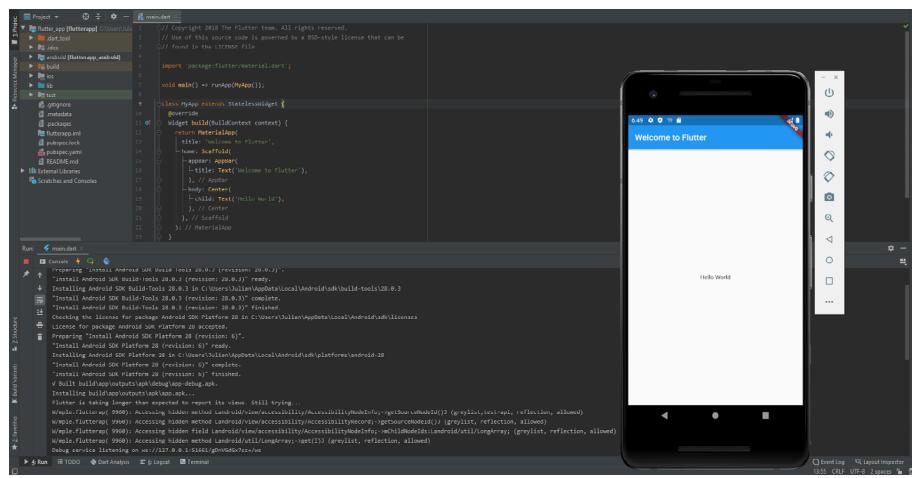
Solucionario

Referencias bibliográficas

Cuando termine, haga clic en “Finalizar” para crear la nueva aplicación en Flutter. En el archivo main.dart, el archivo de trabajo básico de proyectos, que también usamos en este tutorial de Flutter, escriba el siguiente código para que la aplicación presente un simple mensaje de “Hello World” (puede borrar el código existente en el archivo main.dart):

```
import 'package:flutter/material.dart';
void main() => runApp(MyApp());
class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'Welcome to Flutter',
            home: Scaffold(
                appBar: AppBar(
                    title: Text('Welcome to Flutter'),
                ),
                body: Center(
                    child: Text('Hello World'),
                ),
            ),
        );
    }
}
```

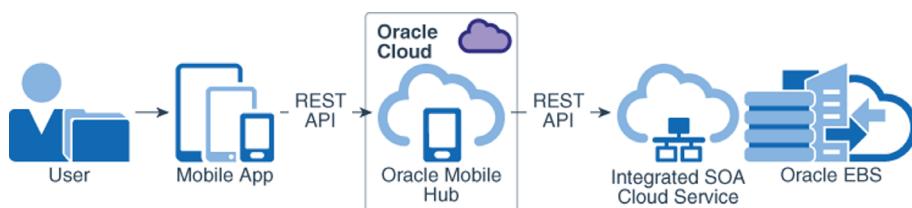
Una vez haya insertado el fragmento, puede ejecutar esta primera versión de su propia aplicación. En Android Studio, seleccione la máquina virtual correspondiente (en “Flutter Device Selection”) y, luego, pulse el botón de ejecución (“Run main.dart”) tal como se muestra en la figura 5.15:

Figura 5.15*Ejecución de la aplicación en Flutter*

Nota: Ejecución de la aplicación en Flutter. Tomado de Android Studio, 2021.

5.8. Integración y consumo de APIs con Flutter

Estimado estudiante, para explicar de mejor manera la integración de las aplicaciones móviles con Cloud, se pone como referencia el consumo de APIs como medio de conexión entre una aplicación móvil y el internet, esto nos ayuda a centralizar toda la información que es recolectada y consumida por los usuarios que utilizan los dispositivos móviles.

Figura 5.16*Arquitectura del consumo de una aplicación móvil con APIs*

Nota: Arquitectura de una app móvil que consume datos vía Api rest.

Tomado de Oracle, 2021.

¿Concepto de API?

API es un servicio web propio o de terceros, el cual tiene como medio de conexión una URL única que al hacer referencia esta nos devuelve la información solicitada. Los APIs pueden tener varias propiedades según las necesidades para las que fueron creadas y que son ejecutadas en el lado del servidor.

Para entender de mejor manera este tema lo invito a revisar este [recurso](#) donde encontrará mayor detalle de lo que son las APIs y el consumo de las mismas mediante una aplicación desarrollada en Flutter.

Consumo de Apis en Flutter

Para implementar el servicio http en Flutter, necesitamos importar el paquete que nos sirve para eso. Para importar el paquete a nuestro proyecto debemos ir al archivo pucspecl.yaml y agregar el nombre del paquete en la sección de dependencies: http.

Recuerde que para agregar el nombre del paquete, debe guardar el archivo para que se ejecute automáticamente el **flutter pub get**. Debe agregar el número de versión después de los dos puntos “: flutter”. Entienda que tiene que buscar y usar siempre la versión más reciente.

Cuando termine de instalar el paquete ya podemos usarlo en cualquiera de nuestros widgets. Solo tendremos que importarlo al archivo y asignarlo a una variable. (Código Correcto, 2021)

Implementación del paquete HTTP en Flutter

Con un ejemplo o proyecto básico en Flutter debemos importarlo a nuestro archivo .dart mediante un import de toda la vida. Así:

```
import 'package:http/http.dart' as http;
```

Ahora tenemos que hacer una función que es la encargada de hacer el llamado a la API, la función tiene que ser asíncrona ya que estamos intentando acceder a información que no se encuentra en nuestro dispositivo cargada, así que puede tardar en recogerse. El código se lo representa de la siguiente manera:

```
String url = "https://api.thecatapi.com/v1/categories";

Future<dynamic> _getListado() async {
    final respuesta = await http.get(url);
    if (respuesta.statusCode == 200) {
        return jsonDecode(respuesta.body);
    } else {
        print("Error con la respuesta");
    }
}
```

Mostrando el resultado del consumo del API en Flutter

Con el código tenemos una función que nos devuelve información, ahora tenemos que usarla en la App para poder ver esa información.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Para eso usaremos un FutureBuilder en el cuerpo de nuestro Scaffold. La aplicación quedaría así:

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text("Listado API"),
        ),
        body: FutureBuilder<dynamic>(
            future: _getListado(),
            builder: (context, snapshot) {
                if (snapshot.hasData) {
                    print(snapshot);
                    return ListView(
                        children: listado(snapshot.data)
                    );
                } else {
                    print("No hay información");
                    return Text("Sin data");
                }
            },
            initialData: Center(child: CircularProgressIndicator()),
        )
    );
}

List<Widget> listado( List<dynamic> info ) {
List<Widget> lista = [];
info.forEach((elemento) {
    lista.add(Text(elemento["name"]));
});
return lista;
}
```

En la propiedad future del FutureBuilder hace el llamado a nuestra función, luego en la propiedad builder usamos el contexto y escogemos un nombre para la variable que va a contener la información. En este caso usaremos el nombre de snapshot que es un estándar, pero puede usar el nombre que quiera.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Los children del ListView son la función que está al final. Esta debe estar fuera de la clase principal del Widget. Lo que hace prácticamente es recorrer la información y agregar un Widget Text con la información que esté en el campo name de la información recibida en un sentido clave-valor. (Código Correcto, 2021)



Actividades de aprendizaje recomendadas

Una vez que ejecutó el proyecto, encuentre el archivo .apk para ejecutarlo en su dispositivo móvil, comente los resultados en el foro.

Para el desarrollo de esta actividad puede revisar este [video](#) para poder instalar la app en su dispositivo móvil, documente el proceso que realizó para cumplir este objetivo.

Muy bien, ¡sigamos avanzando!

Para mejorar sus conocimientos puede [descargar](#) el siguiente recurso donde encontrará una aplicación en Flutter, donde se muestra el código explicado, generado anteriormente y con algunas funciones de gestión de datos de la nube.

Es hora de poner en práctica los conocimientos adquiridos, lo invito a desarrollar el siguiente caso de estudio, ¡mucha suerte!

CASO DE ESTUDIO

Para poner en práctica los conocimientos adquiridos durante el avance de los contenidos de nuestra materia, en especial el desarrollo de aplicaciones móviles en Flutter; se propone desarrollar el siguiente caso de estudio, para lo cual debe desarrollar la siguiente app móvil.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Objetivo del caso: Se requiere la creación de una aplicación móvil que permita el consumo y publicación de productos, estos productos deben ser consumidos desde la nube, puede investigar APIs de su conveniencia o puede utilizar el repositorio del siguiente [enlace](#). Utilice técnicas amigables para la visualización de estos productos.

Elaboración de preguntas: Cumpliendo la metodología de casos, estimado estudiante, determine el desarrollo del caso con ayuda de estas preguntas:

- ¿La aplicación permite el consumo de datos desde la nube?

Análisis y desarrollo del caso: Cree un repositorio en GitHub para el versionamiento de código y constantemente vaya subiendo el estado de la aplicación móvil hasta completar los requerimientos de la misma.

Elaboración de informe: Desarrolle un documento donde se argumente el proceso que utilizó para cumplir el objetivo, utilice capturas de pantalla y código para explicar su desarrollo.

Documente, comparte y comente su trabajo desarrollado en el entorno de aprendizaje EVA.

Terminada esta tarea y caso de estudio podemos pasar al siguiente tema que es la autoevaluación:

Estimado estudiante lo invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuál es su nivel de conocimientos obtenidos durante el tránscurso de este periodo transcurrido, en caso de no obtener resultados satisfactorios le sugiero revisar nuevamente los contenidos en los cuales tiene problemas



Autoevaluación 5

Conteste la siguiente autoevaluación de la unidad 5

1. Flutter es:
 - a. Un framework de Google que permite crear aplicaciones móviles multiplataforma de forma nativa.
 - b. Un framework de Facebook que permite crear aplicaciones móviles multiplataforma de forma nativa.
 - c. Un framework de Google que permite crear aplicaciones móviles multiplataforma de forma híbrida.
2. Flutter utiliza el lenguaje para desarrollo:
 - a. C++.
 - b. Dart.
 - c. Java.
3. Flutter posee la cualidad de hot reload que significa:
 - a. Mientras se ejecuta el emulador los cambios se compilan en dispositivo.
 - b. Mientras se ejecuta el emulador los cambios no se ven al instante.
 - c. Mientras se ejecuta el emulador los cambios se ven al instante.
4. Flutter posee elementos ya predeterminados como:
 - a. Módulos de gestión de usuarios.
 - b. Elementos predefinidos de interfaz de usuario.
 - c. Librerías de conexión a bases de datos predefinidas.

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

5. Para la instalación de Flutter se debe descargar de:
 - a. Google.
 - b. Facebook.
 - c. Git.
6. Para la implementación del paquete de consumo de Apis HTTP en Flutter es:
 - a. import 'package:http/http.dart' as http;
 - b. import 'package:http/dart.http' as http;
 - c. import 'package:http/http' as dart;
7. El proceso de compilación multiplataforma de Flutter es:
 - a. Compilar aplicaciones multiplataforma con el lenguaje Dart.
 - b. Compilar las apps multiplataforma con HTML.
 - c. Compilar las apps multiplataforma con Java.
8. Una característica de Flutter es:
 - a. Su compatibilidad con Ionic.
 - b. Más rápido que sus competidores Ionic o React Native.
 - c. Más fácil de desarrollo por sus módulos desarrollados.
9. La ruta de la carpeta bin de Flutter está en:
 - a. C:\java\bin.
 - b. C:\java\flutter\bin.
 - c. C:\flutter\bin.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

10. Para aceptar las licencias de licencias de Android se necesita la siguiente línea de comando:
- flutter doctor -licenses.*
 - flutter doctor --android-licenses.*
 - flutter --android-licenses.*

[Ir al solucionario](#)

Las respuestas a esta autoevaluación se encuentran al final de la presente guía didáctica, acuda y compare las respuestas, si no logró un buen resultado en la autoevaluación, no se preocupe le recomiendo leer nuevamente el/los capítulos confusos y reforzar sus conocimientos. Y si aún tiene inquietudes no dude en preguntar al profesor.

¡Hemos terminado la quinta unidad!

¡Felicitaciones!



Semana 13

Para lograr el resultado de aprendizaje propuesto en el segundo bimestre, en esta semana tiene que revisar los contenidos de forma general la investigación de cuáles son los requerimientos y el presupuesto para poner en marcha la aplicación móvil.



Unidad 6. Puesta en marcha de una aplicación móvil

Estimado alumno, empezamos el sexto capítulo de la asignatura tomando temas fundamentales que se requiere entender dentro de la puesta en marcha de una app.

Para iniciar el estudio de este tema sobre la puesta en marcha de una app móvil, acuda al siguiente [recurso digital](#) y proceda a realizar una lectura comprensiva del tema.

¿Qué opina de lo aprendido? ¿Tiene inquietudes o dudas? ¡A continuación las iremos resolviendo!

La fase final de una aplicación llega cuando finalmente es cargada, revisada y publicada en la tienda, para lograrlo, hay que cumplir una serie de pasos y requisitos. Después del lanzamiento existe el proceso de mejora continua de la app, ello se logra escuchando a los usuarios y mejorando la app.

Índice

Primer bimestre

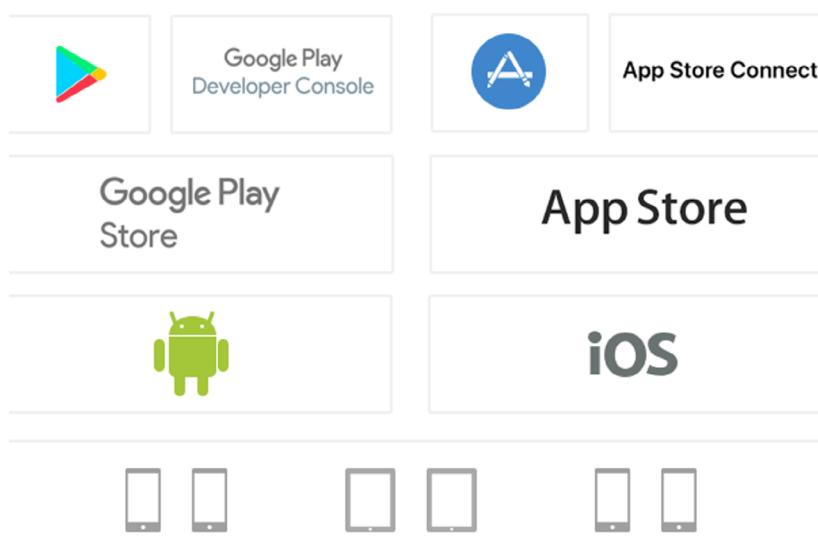
Segundo bimestre

Solucionario

Referencias bibliográficas

6.1. Publicación de la aplicación móvil en tiendas oficiales

Una aplicación se publica después de la etapa de pruebas, cuando ya se tiene seguridad acerca de su correcto funcionamiento, estabilidad, desempeño y se considera libre de errores, tanto de usabilidad como de diseño. El proceso de publicación en cada una de las tiendas es una tarea relativamente fácil y está bien documentada. Antes de comenzar, es importante prepararse y disponer de todos los recursos de diseño requeridos y hay que asegurarse de que la aplicación cumpla con las políticas de publicación de las tiendas para evitar que sea rechazada o bloqueada. Una aplicación puede tardar varios días en ser aprobada, sobre todo cuando es la primera vez que se envía. Hay que tener en cuenta este tiempo, especialmente si se quiere hacer coincidir su salida al mercado con alguna fecha especial. También cabe recordar que la publicación no es gratuita. Tanto en Google Play, como en App Store y en Windows Phone Store, hay que asumir un costo para iniciar el proceso que varía en cada uno de los casos. En Google Play es necesario pagar 25 dólares por única vez al momento de crear la cuenta para la publicación. En el caso de App Store el pago es mayor, asciende a 99 dólares que, además, deberán pagarse anualmente; esta situación es idéntica a la que se presenta para desarrolladores de Windows Phone. Crear la cuenta para publicación y realizar el pago correspondiente, permite acceder a una serie de herramientas de gestión y estadísticas sobre la cantidad de descargas de la app y su monetización. (Cuello Javier, 2013)

Figura 6.1*Tiendas oficiales para publicación de apps móviles*

Nota: Tiendas oficiales para publicación de apps móviles.

Imágenes y elementos promocionales

Cuando se publica una aplicación, no solo debe subirse el archivo de la app, sino también aquellos elementos que la acompañarán en la página promocional de la tienda y que, en la mayoría de los casos, se trata de capturas de pantalla, textos descriptivos, ícono de lanzamiento e imágenes. La importancia de preparar bien estos recursos radica en que serán visibles al público y servirán como herramientas de promoción y comunicación, lo cual puede ser determinante para la descarga de la aplicación. Textos e imágenes deben cuidar hasta el último detalle y cumplir con los requerimientos de cada una de las tiendas en cuanto a formato, dimensiones y resoluciones.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

En el caso particular de las capturas de pantalla, es conveniente elegir las más representativas y atractivas de la aplicación. Si una pantalla no dice algo significativo de la app, la recomendación es no incluirla. También se pueden aprovechar estas imágenes para incluir, conjuntamente, otros gráficos que expliquen el funcionamiento de la app o refuerzen el concepto que se quiere comunicar. Cuando el contenido de las pantallas incluya nombres de personas o lugares, fotos o títulos, es aconsejable reemplazarlos por datos que simulen lo mejor posible una situación real, para que estas no luzcan como pantallas creadas durante los test de la aplicación. Aunque pueda parecer un detalle menor, también es bueno tener en cuenta que la barra de estado superior del teléfono operativo esté lo más limpia posible de íconos que distraigan la atención del contenido principal —por ejemplo, correos sin leer— aunque para ello haga falta retocar ligeramente la imagen y hacerla más presentable. Se aconseja darle importancia a la elección de estos elementos ya que, junto al ícono de la app, son gráficos fundamentales para convencer al usuario: muchos de ellos tomarán la decisión de descargarla por lo que vean aquí.

6.2. El proceso de aprobación

Subir una aplicación no garantiza que esta sea publicada, ya que debe someterse a un proceso de aprobación que es diferente en cada tienda. El proceso en Google Play es más abierto y tolerante con una ventaja evidente: la mayoría de las aplicaciones son publicadas, a menos que incumplan claramente alguna de las políticas de la tienda, caso en el cual pueden ser retiradas y en última instancia, si la situación lo requiere, puede ser suspendida la cuenta del desarrollador. Por supuesto, esta facilidad para publicar significa que pueden encontrarse muchas aplicaciones de dudosa utilidad entre todas las alternativas que ofrece la tienda. Por el contrario, en App Store la aprobación es especialmente rigurosa, con una política

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

pensada para garantizar cierto estándar de calidad en la publicación de aplicaciones, pero que en algunos casos puede rozar la exageración. La guía completa de causas de rechazo está disponible al entrar a la cuenta de desarrollador. Solo por nombrar algunas, podemos mencionar aquellas que hacen referencia al contenido no original o similares a otras existentes, aplicaciones que no estén terminadas, por ejemplo, si están en forma de demostración o prueba o que representen una ofensa para personas, religiones o incluyan contenido inapropiado para niños. En el caso de Windows Phone, la aplicación también debe pasar un proceso de certificación en el cual se verifica que cumpla las normas de la tienda. Los aspectos más importantes a tener en cuenta para que una app sea aprobada son la apariencia general —que no deje lugar a dudas sobre a qué sistema operativo pertenece, no contenga adornos ni elementos innecesarios y haga buen uso de los botones de sistema— y la calidad del contenido.

Continuemos con algunos temas más de la puesta en marcha de una app.



Semana 14

6.3. Después del lanzamiento

El trabajo en una aplicación no termina una vez que esta ha sido lanzada y está publicada en las tiendas. De hecho, ahí comienza una etapa más emocionante porque el producto está en manos de usuarios reales. La app se ha puesto su traje y finalmente está en la calle. Esto significa que quienes la usen empezarán a compartir su experiencia e impresiones que, junto con las estadísticas de uso

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

y descargas, servirán como referencia para mejorar la aplicación y corregir aquellas fallas de diseño o funcionalidad que, a pesar de los esfuerzos anteriores, se pasaron por alto.

Este tipo de mejoras conducirá a aumentar la calidad de la aplicación, lo que, a su vez, se traducirá en un mayor número de descargas y más comentarios positivos. Los usuarios son quienes corren la voz de la aplicación y es mejor mantenerlos contentos. De la misma forma, las opiniones de la prensa y medios especializados pueden derivar en más descargas. (Cuello Javier, 2013)

6.4. Comentarios de los usuarios

Una buena forma de retroalimentación son las opiniones de los usuarios. Y esto no se refiere solamente a los comentarios casi siempre positivos del círculo de amigos, sino a aquellos que provienen de personas que no tienen nada que ver con nuestro entorno. Una vez subida a la tienda, cada aplicación recibe valoraciones y reseñas. Aunque no todas son de gran ayuda, es importante prestar atención a estos comentarios para identificar entre ellos los que sean de verdadera utilidad. Nadie mejor que los usuarios que ya han descargado la aplicación para queriendo o no hacer pruebas con ella. Reportes de fallos funcionales, opiniones sobre diseño o usabilidad, entre otras cosas, pueden encontrarse entre los comentarios de usuarios; seguirlos, es una buena herramienta para mejorar constantemente la app.

6.5. Las actualizaciones

Las observaciones que se obtienen a través de los comentarios y estadísticas de uso, se transforman en una serie de mejoras que pueden implementarse en la aplicación. Es entonces cuando deja de existir solo la versión inicial y se tiene una app mejorada, lista para

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

sustituir a la anterior. Esto es posible a través de las actualizaciones. Cuando lo que se ha corregido en la app son fallos realmente graves, que impiden el uso futuro de la aplicación, puede realizarse lo que conocemos como «actualización forzosa». Como el nombre lo indica, es aquella que obliga a los usuarios a descargar la última versión. El riesgo de este tipo de actualizaciones es que puede hacer que algunos usuarios dejen de usar la aplicación porque no quieren actualizarla y tampoco pueden usar la versión que tienen instalada. En otros casos, la descarga de las actualizaciones suele ser opcional y puede realizarse cuando el usuario lo prefiera. Las actualizaciones generalmente responden a un ciclo de desarrollo, que dura una determinada cantidad de días, al cabo del cual se liberan. De esta forma, se va trabajando en la prueba e implementación de cambios futuros que resultarán en una nueva versión de la aplicación. La ventaja de esta forma de trabajar es que se realiza una mejora constante de la app y los usuarios saben que pueden esperar una versión mejorada cada cierto tiempo. En Android y iOS las aplicaciones se actualizan automáticamente, evitando al usuario la preocupación de asegurarse tener las últimas versiones de las apps instaladas. Por el contrario, en Windows Phone sigue siendo necesaria su intervención, ya que se tiene que actualizar manualmente las apps, cuando el SO así se lo indica con una notificación visual. Es importante recalcar que subir una nueva versión de la aplicación a la tienda de Apple o de Microsoft obliga a someterla nuevamente a la revisión del cumplimiento de las políticas por parte de estas tiendas. En el caso de Google Play, como hemos comentado, el proceso es más simple y directo.





Actividades de aprendizaje recomendadas

Realice un presupuesto para la implementación de una app móvil en iOS.

Para el desarrollo de esta actividad puede revisar la sección 6.1 donde se detalla el presupuesto para una aplicación en Google Play o revisar el siguiente [enlace](#) donde se detalla el tema del presupuesto de una aplicación móvil, posteriormente coméntelo en el foro del Eva.

Como ayuda para reforzar y tomar en cuenta los aspectos que se necesita para elaborar un presupuesto y la puesta en marcha de una aplicación móvil le invito a revisar este [enlace](#) donde se explica los puntos importantes que se debe tomar en cuenta para la elaboración de una aplicación móvil.

Vamos finalizando nuestros contenidos, pero antes le invito a realizar la siguiente autoevaluación para medir sus conocimientos de esta sección.

¡Suerte! ¡Siga adelante!



Autoevaluación 6

Estimado estudiante, le invito a medir sus conocimientos de esta unidad con la siguiente autoevaluación, esta le ayudará a valorar cuáles su nivel de conocimientos obtenidos durante el trascurso de este periodo transcurrido, en caso de no obtener resultados satisfactorios le sugiero revisar nuevamente los contenidos en los cuales tiene problemas.

Seleccione la opción correcta:

1. Una aplicación móvil se publica después de:
 - a. La fase de desarrollo.
 - b. La fase de pruebas.
 - c. La fase de recolección de comentarios.

2. Para que una aplicación sea publicada en una tienda esta puede durar un tiempo de:
 - a. 1 hora.
 - b. Varios días.
 - c. 3 meses.

3. La publicación de una app en una tienda tiene:
 - a. No tiene costo.
 - b. Si tiene costo.
 - c. El costo es después de existir ganancias de la app.

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

4. Seleccione de la lista la que corresponde a una tienda:
 - a. Google.
 - b. Apple store.
 - c. Twitter.
5. La mejora continua es:
 - a. Aumentar la calidad de la aplicación.
 - b. Aumentar ventas de la app.
 - c. Aumentar requerimientos de la app.
6. Antes de publicar las apps en tiendas establecidas se necesita:
 - a. Debe cumplir estándares de codificación.
 - b. La app debe estar terminada en su totalidad.
 - c. La app debe cumplir con todas las políticas de publicación de las tiendas.
7. El costo de publicación de la app en Google Play es de:
 - a. \$25.
 - b. \$50.
 - c. \$99.
8. El costo de publicación de la app en App Store:
 - a. \$25.
 - b. \$50.
 - c. \$99.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

9. Para subir la app en las tiendas es necesario:
 - a. Contar con imágenes, textos que ayuden a la publicidad de la app.
 - b. Contar con un equipo de desarrollo listo para cambios impuestos por las tiendas.
 - c. Contar con un equipo de QA.
10. Cuando se sube la app en las tiendas, la app:
 - a. Siempre se publica.
 - b. Puede o no publicarse.
 - c. No se publica.

[Ir al solucionario](#)

Las respuestas a esta autoevaluación se encuentran al final de la presente guía didáctica, vaya y compare las respuestas, si no logró un buen resultado en la autoevaluación, no se preocupe, le recomiendo leer nuevamente el/los capítulos confusos y reforzar sus conocimientos. Y si aún tiene inquietudes no dude en preguntar al profesor.





Actividades finales del bimestre

- Resultados descritos en cada una de las asignaturas.



Semana 15

En esta semana nos vamos a dedicar a revisar las temáticas estudiadas en las semanas anteriores para que se puedan ir afianzando los temas previos a la evaluación presencial del segundo bimestre.

Además de eso se tiene algunas actividades calificadas que es importante que usted realice como parte del proceso de preparación, como es el chat académico que se ha programado para esta semana, además de la actividad suplementaria para el caso de los alumnos que por algún motivo no puedan realizar la actividad síncrona.



Semana 16

Actividad:

En la última semana del segundo bimestre nos vamos a enfocar en la revisión del contenido previa la evaluación presencial, se recomienda aprovechar esta semana revisando nuevamente cada uno de los temas y si surgieran dudas por favor contactarse con el docente-tutor para resolverlas.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas



4. Solucionario

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1	b	Las aplicaciones híbridas están desarrolladas en HTML, CSS y JavaScript, posteriormente son compiladas para crear el ejecutable de la app móvil.
2	a	Las aplicaciones móviles son instaladas en los dispositivos móviles.
3	a	Existe una serie de aplicaciones híbridas y responsivas que se adaptan los dispositivos móviles.
4	c	Las aplicaciones responsivas están desarrolladas con HTML y CSS las cuales se adaptan a cualquier dispositivo móvil que la utilice.
5	c	Play store no es una tecnología de desarrollo, es una tienda donde se sube las apps móviles.
6	b	El proceso de diagramación por lo regular se lo realiza en una aplicación web.
7	a	Los primeros teléfonos solo contaban con apps básicas enfocadas a la productividad personal, como tiempo, cronómetro, etc.
8	c	Una app web móvil solo se requiere conectarse al internet para su funcionamiento.
9	c	El diseño líquido es la forma que toma la app al desplegarse y adaptarse en la pantalla de los dispositivos móviles.
10	b	Se requiere la herramienta de desarrollo la cual compilará la app para pasar a producción.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 2

Pregunta	Respuesta	Retroalimentación
1	b	Es una aplicación que ayuda al desarrollo de aplicaciones móviles.
2	b	Uber es una app para solicitar movilidad y fue desarrollada como una app híbrida.
3	a	Ionic es un framework de desarrollo móvil.
4	b	La propiedad de React native es Facebook.
5	a	Google creó Flutter como alternativa para desarrollo de aplicaciones híbridas que al compilarlas se compilan como nativas.
6	b	SDK es la base para el desarrollo de aplicaciones nativas en Android.
7	a	Es necesario descargar e instalar las aplicaciones nativas de Android.
8	b	Con ayuda del sistema operativo ayuda a notificar de los requerimientos de las apps al usuario.
9	c	Es necesario cumplir con los requerimientos del sistema operativo para desarrollar apps nativas.
10	b	El diseño de apps híbridas es el más flexible e independiente de las apps nativas.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 3

Pregunta	Respuesta	Retroalimentación
1	a	El JDK es el requisito principal para la instalación de Android Studio y para que compilen las aplicaciones desarrolladas sobre el IDE.
2	a	El emulador para compilar las aplicaciones es más rápido al cargar las actualizaciones de la app.
3	a	Este módulo viene por defecto en el instalador de Android ya que es un requerimiento para compilar las apps.
4	c	La barra de herramientas posee varias opciones de ejecución sobre el proyecto de desarrollo.
5	b	El compilador de Java es la base fundamental para compilar las apps en Android.
6	a	Una función importante de Android Studio es la compilación basada en Gradle.
7	b	Android Studio trae integrado el versionamiento de código con GitHub.
8	c	C++ es la compatibilidad para aplicaciones móviles.
9	a	Android Studio trae con la function para la aceleración de las apps y tener acceso a las diferentes herramientas de Google.
10	b	El editor es accesible para el desarrollo ya que muestra la estructura completa de directorios y archivos para su rápida edición.

Ir a la
autoevaluación



Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 4

Pregunta	Respuesta	Retroalimentación
1	a	Estos componentes son las ventanas que se visualiza en la pantalla del dispositivo móvil.
2	c	Se encarga de englobar los elementos visuales de una app móvil.
3	b	El viregroup utiliza xml para configurar los elementos de posición.
4	a	A manera de grafos se muestra la jerarquía del diseño con los componentes de la ventana.
5	c	Es la medida oficial para realizar elementos flexibles a la pantalla del dispositivo.
6	a	La ruta correcta para la edición del archivo de diseño es la planteada anteriormente.
7	c	Android Studio permite la edición del archivo de forma de código y visualmente, esto ayuda al desarrollador a tener más opciones de desarrollo.
8	a	EL componente muestra de jerarquía de los elementos de diseño de los objetos.
9	c	El textview permite incrustar texto en la app móvil.
10	a	Esta propiedad permite desactivar que los elementos de la app se coloquen de forma horizontal.

Ir a la
autoevaluación

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 5

Pregunta	Respuesta	Retroalimentación
1	a	Este es el concepto de Flutter.
2	b	Dart es el lenguaje oficial para el desarrollo de apps en Flutter.
3	c	Los cambios que se van desarrollando en la app móvil se ven reflejadas automáticamente en el emulador sin tener que compilar.
4	b	Son elementos ya desarrollados que se pueden reutilizar para el desarrollo de apps.
5	c	Gil es un elemento requerido pro Flutter para el desarrollo y tiene como objetivo el versionamiento de código.
6	a	La forma correcta de implementar el paquete HTTP es invocando http.dart as http.
7	a	Crear aplicaciones multiplataforma a través de código de dart, transformando apps nativas.
8	b	Es más rápido de sus principales rivales.
9	c	La ruta correcta cuando se instala en el sistema operativo Windows.
10	b	El comando que se ejecuta en la consola permite aceptar las licencias que posee Flutter para su desarrollo.

Ir a la
autoevaluación



Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Autoevaluación 6

Pregunta	Respuesta	Retroalimentación
1	b	Después del proceso de desarrollo viene la fase de pruebas para su posterior implementación.
2	b	El proceso de aprobación dura algún tiempo porque entra a un tiempo de revisión.
3	b	Todas las tiendas tienen un costo para publicarlas en las tiendas.
4	b	Apple store es una tienda oficial para iOS.
5	a	Mejora continua es mejorar los problemas presentados e incrementar servicios para los usuarios.
6	c	Las tiendas donde se alojan las apps móviles solicitan que se cumpla todas las políticas de la tienda, caso contrario, no se las publica.
7	a	El costo es de \$25 dólares y solo se paga la primera vez.
8	c	El costo es de 99 dólares y brinda servicios estadísticos de la app.
9	a	Es necesario contar con recursos de publicidad e información que van a ser de utilidad de los usuarios.
10	b	Si no cumple con todas las políticas de las tiendas corre el riesgo de no publicar la app.

Ir a la
autoevaluación



5. Referencias bibliográficas

5.1 Básica

Amaro. (2019). *Android Programación. Dispositivos Móviles 2ed.*. España: ALFAOMEGA Editorial. ISBN: 9789587786118

Este libro posee un importante contenido ilustrativo e intuitivo que nos ayudará a cumplir nuestros objetivos planteados a lo largo de nuestra materia, como es el desarrollo de aplicaciones móviles.

Ramírez, R. (2021): *Guía didáctica de Desarrollo Basado en Plataformas Móviles*, Loja-Ecuador, Editorial Universidad Técnica Particular de Loja.

En la guía didáctica encontrará los lineamientos necesarios para que se pueda guiar a través del texto básico, además encontrará explicaciones adicionales sobre algunos temas que le permitirán entenderlos mejor.

5.2 Bibliografía

Android Studio . (Marzo de 2021). Android Studio . Obtenido de <https://developer.android.com/studio/>

Aplicaciones Responsivas [Ilustración]. (Enero de 2020). Obtenido de <https://www.fabricamos.software>

Axarnet. (Febrero de 2021). Axarnet. Obtenido de <https://axarnet.es/blog/plataformas-desarrollo-aplicaciones-moviles>

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Cuello Javier, V. J. (Marzo de 2013). Aprende a diseñar apps nativas.

Catalina Duque Giraldo. Obtenido de <https://appdesignbook.com/es/contenidos/las-aplicaciones/>

Flutter. (Marzo de 2021). Framework Flutter [Fotografía]. Obtenido de <https://medium.com/@fernandoquinterosgutierrez/de-web-a-flutter-839a34144295>

Herramienta de Figma [Entorno de la herramienta]. (Marzo de 2021).
Obtenido de figma.com

Invidgroup. (Enero de 2021). Invidgroup [Tipos de aplicaciones web].
Obtenido de <https://invidgroup.com/es/que-es-el-desarrollo-de-aplicaciones-moviles/>

Modelos de DB. (Marzo de 2012). Modelos de DB. Obtenido de <https://modelosbd2012t1.wordpress.com/2012/03/15/base-de-datos-moviles-3/>

TechTarget. (Marzo de 2017). Search DataCenter . Obtenido de <https://searchdatacenter.techtarget.com/es/definicion/Gestion-de-dispositivos-moviles-MDM>

5.3 Complementaria

Instalación de Android Studio [en línea] Disponible en: <https://developer.android.com/studio/install?hl=es-419> [consultado a: enero del 2021].

Recurso digital donde se encuentra el proceso de instalación de Android Studio.

Crear interfaz en Android Studio [en línea] Disponible en: <https://developer.android.com/training/basics/firstapp/building-ui?hl=es> [consultado a: enero del 2021].

Recurso digital donde muestra cómo crear interfaces para las aplicaciones móviles.

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Conceptos de Flutter [en línea] Disponible en: <https://www.qualitydevs.com/2019/07/05/que-es-flutter/> [consultado a: enero del 2021].

Recurso digital donde se argumenta los conceptos del framework Flutter

Documentación de Flutter [en línea] Disponible en: <https://flutter.dev/docs> [consultado a: enero del 2021].

Recurso digital donde se encuentra la documentación completa de Flutter

Desarrollo de aplicaciones híbridas [en línea] Disponible en: <https://www.armadilloamarillo.com/blog/desarrollo-de-aplicaciones-hibridas-cuando-son-buena-opcion/> [consultado a: enero del 2021].

Recurso digital donde se explica el desarrollo de aplicaciones híbridas.

Bases de datos para Android [en línea] Disponible en: <https://www.tecnologias-informacion.com/basedatosandroid.html> [consultado a: enero del 2021].

Recurso digital donde se explica cuáles son las bases de datos que se puede trabajar con Android.

Puesta en marcha de una aplicación móvil [en línea] Disponible en: <https://repository.usta.edu.co/bitstream/handle/11634/9915/PinedaJuan2016.pdf?sequence=1&isAllowed=y> [consultado a: enero del 2021].

Recurso digital donde están los elementos básicos para la puesta en marcha de una aplicación móvil.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Puesta en marcha de una aplicación móvil [en línea] Disponible en:

<https://www.cursor.cl/puesta-en-marcha-de-una-app-movil-de-marketing-a-cobranza/> [consultado a: enero del 2021].

Recurso digital donde se muestra elementos básicos para la puesta en marcha de una aplicación móvil.