

Escuela de Código para PILARES

Descripción de actividades

Parte 4: Desarrollo de Aplicaciones Móviles (MM)



Escuela de Código para PILARES Descripción de actividades Parte 4: Desarrollo de aplicaciones móviles (MM) por Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México se distribuye bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).

Créditos¹

Redacción de actividades

Andrea García Ruiz, Elisa Mariana Valdés Armada, Héctor Alfonso Islas García y Luz Elena Rueda Rojas

Coordinación de módulo

Nora Isabel Pérez Quesadas

Coordinación de la Transversalización de la Perspectiva de Género

Yuliana Ivette López Rodríguez

Revisiones

Karen Itzel Bruno Sainos, Citlalli Sánchez Mendoza, Carmen Daniela Garrido Juvencio

Supervisión PILARES

Jesús Alanis Manriquez, René Alejandro Rivas Robles y María del Rocío Estrada Monroy

Supervisión IIMAS

Adrián Durán Chavesti, Alejandra Sarahí Monroy Velázquez, Carla Irena Blenda Palacios, Karen Alexa Alva Aguirre, Karina Flores García, Héctor Benítez Pérez, Helena Gómez Adorno, Ivan Vladimir Meza Ruiz, María del Pilar Ángeles, Víctor Manuel Lomas Barrie, Zian Fanti Gutierrez

Financiamiento:

Diseño de un programa de estudios para la capacitación en programación y habilidades en tecnologías de información y comunicación para la escuela de código dentro de PILARES de la Ciudad de México (SECTEI/284/2019)

¹ En orden alfabético.

Agradecimientos

Agradecemos el tiempo y la retroalimentación hecha a los materiales a:

- Ante Salcedo González, ITAM - Instituto Tecnológico Autónomo de México
- Blanca Esther Carvajal-Gámez, ESCOM - Escuela Superior de Cómputo - IPN
- Dagoberto Pulido Arias, IPN - Instituto Politécnico Nacional
- Eréndira Itzel García Islas, UNAM - Facultad de Ciencias
- Marco Antonio Moreno Ibarra, CIC - Centro de Investigación en Computación del IPN
- Ricardo Marcelín Jiménez, UAM-I Universidad Autónoma Metropolitana – Iztapalapa
- Salvador Elías Venegas Andraca, ITESM - Instituto Tecnológico y de Estudios Superiores de Monterrey

También agradecemos el apoyo y seguimiento al personal de SECTEI, en particular de:

- José Bernardo Rosas Fernandez
- Federico Antonio Hernández Loranca
- Rogelio Artemio Morales Martínez
- Adrián Eleazar Contreras Martínez
- Benigno Antonio González Núñez

Índice

Créditos	2
Agradecimientos	3
Agradecemos el tiempo y la retroalimentación hecha a los materiales a:	3
Ante Salcedo González, ITAM - Instituto Tecnológico Autónomo de México	3
Blanca Esther Carvajal-Gámez, ESCOM - Escuela Superior de Cómputo - IPN	3
Dagoberto Pulido Arias, IPN - Instituto Politécnico Nacional	3
Eréndira Itzel García Islas, UNAM - Facultad de Ciencias	3
Marco Antonio Moreno Ibarra, CIC - Centro de Investigación en Computación del IPN	3
Ricardo Marcelín Jiménez, UAM-I Universidad Autónoma Metropolitana – Iztapalapa	3
Salvador Elias Venegas Andraca, ITESM - Instituto Tecnológico y de Estudios Superiores de Monterrey	3
También agradecemos el apoyo y seguimiento al personal de SECTEI, en particular de:	3
José Bernardo Rosas Fernandez	3
Federico Antonio Hernández Loranca	3
Rogelio Artemio Morales Martínez	3
Adrián Eleazar Contreras Martínez	3
Benigno Antonio González Núñez	3
Índice	4
Taller 1: Mi primera aplicación Móvil	7

Actividad 1: Conociendo mi celular	8
Actividad 2: Estableciendo el ambiente de trabajo	11
Actividad 3: Diseñando mi primera aplicación	14
Actividad 4: Mi catálogo con sonidos	17
Actividad 5: Tomando fotos desde mi aplicación	21
Actividad 6: Haciendo un video con mi aplicación	26
Actividad 7: Mis coordenadas	30
Actividad 8: Obteniendo mi ubicación a partir de una fotografía desde mi teléfono	34
Actividad 9: Ubicación con un video desde mi app	38
Actividad 10: Enviando un mensaje SMS (Whatsapp, Red social) desde mi app	45
Actividad 11: Compartiendo mi ubicación a partir de una foto o video con un mensaje de texto en tiempo real con mi aplicación	52
Actividad 12: Mejorando mi aplicación	56
Actividad 13: Seguridad	63
Actividad 14: Probando el funcionamiento de mi aplicación con mi red de amigas e Instalando mi aplicación en un teléfono móvil	70

Taller 2: Mis primeros pasos en Kotlin	74
Actividad 1: Mis primeros pasos con Kotlin	76
Actividad 2: Un vistazo a Android Studio	81
Actividad 3: Elementos que permanecen y que van cambiando en mi programa	86
Actividad 4: Esos elementos que permanecen y cambian son de varios tipos	89
Actividad 5: ¿Cómo le digo a mi programa lo que debe de hacer con los datos?	103
Actividad 6: Tomando decisiones en mi programa	105
Actividad 7: Tomando una decisión entre varias opciones	110
Actividad 8: Repetir acciones hasta alcanzar el objetivo	113
Actividad 9: Repetir acciones mientras el objetivo esté vigente.	122
Actividad 10: Almacenando y recuperando datos del mismo tipo de elemento	125

Actividad 11: Una llave para recuperar un elemento de valor	130
Actividad 12: No llevar a la escuela una mochila sin útiles	134
Actividad 13: La reutilización es una buena práctica, ahorra tiempo y esfuerzo	141
Actividad 14: Elementos robustos de mi programa	145
Actividad 15: Uniendo cada parte para la creación de mi aplicación personalizada.	149

Taller 3: Mis primeras aplicaciones con Kotlin	155
---	------------

Actividad 1: Un paseo por el código de mi aplicación	156
Actividad 2: Diseñando mi primera App	161
Actividad 3: Uso de listas	167
Actividad 4: Usando mi cámara	172
Actividad 5: Obtener mi ubicación actual	176
Actividad 6: Mostrar ubicación en mapa	180
Actividad 7: Más pantallas	184
Actividad 8: Guardar información en una base de datos local	189
Actividad 9: Agregar, editar, actualizar y borrar datos en la aplicación	196
Actividad 10: Agregar barra de calificación y comentarios	202
Actividad 11: Compartiendo en redes sociales (Facebook, WhatsApp, Instagram)	207
Actividad 12: Privacidad y seguridad de los datos de mi aplicación	213
Actividad 13: Mejorando mi aplicación	218

Taller 1: Mi primera aplicación Móvil

En honor a **Sofia Vasílievna Kovalévskaya (1850- 1891)**. Doctora en matemática y escritora rusa. Fue la primera mujer en trabajar como profesora en la Universidad de Estocolmo en 1881. Tanto el acceso al empleo como a la educación no le fueron fáciles debido a las prohibiciones hacia las mujeres de la época, sin embargo, lo logró tras clases particulares y con ayuda de su profesor, quien presentó su trabajo de doctorado.

Sus investigaciones en matemáticas contribuyeron a las ecuaciones diferenciales parciales y la mecánica. Es conocida por el Teorema de Cauchy-Kovalevskaya.

Debido a su labor científica y social, recibió el Premio Bordin de la Academia de Ciencias de París y Suecia. Asimismo, fue invitada a colaborar en la revista *Acta Mathematica* hasta formar parte del consejo editorial. Al morir, se le reconoció su aporte por la simplificación de un Teorema de Bruns.

Como escritora, realizó dos colaboraciones de teatro para representar la problemática de desigualdad de derechos de las mujeres y escribió su autobiografía *Recuerdos de la infancia*.

Competencia del taller: Identificar los recursos del sistema operativo de un teléfono móvil, para acceder a ellos a partir de aplicaciones móviles desarrolladas con *MIT App Inventor* e integrar dichas funcionalidades en la elaboración de una aplicación móvil.

Actitudes: Curiosidad, disposición, constancia, persistencia, apertura a la incorporación de nuevos aprendizajes, capacidad de análisis, apertura al diálogo, escucha, trabajo en equipo, intercambio de opiniones, participación activa, interés y apertura a incorporar en las actividades la Perspectiva de Género para el logro de un bien común.

Actividad 1: Conociendo mi celular

Aprendizaje esperado: Identificar algunas aplicaciones de un dispositivo móvil, su tipo y funcionalidad.		Duración de la actividad: 4 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Teléfono móvil. • Diversas aplicaciones móviles instaladas. • Sistema operativo Android. • Anexo <i>MMT1A1_Anexos.pdf</i> 	Evidencia: <ol style="list-style-type: none"> 1. Organizador gráfico que contenga información referente a los sistemas operativos y las aplicaciones móviles. 2. Lista de las aplicaciones que posee el teléfono móvil de la participante. 3. Crucigrama globalizador de la tercera parte de la actividad. 	Retroalimentación: <ul style="list-style-type: none"> • Revisar que el organizador gráfico contenga la información solicitada en la segunda parte de la actividad. • Corroborar que la plantilla A del anexo <i>MMT1A1_Anexos.pdf</i> presente la información sobre las aplicaciones del dispositivo móvil de la participante. • Revisar que las respuestas del crucigrama de la plantilla B sean acordes a las plasmadas en la plantilla C (ambas disponibles en el anexo <i>MMT1A1_Anexos.pdf</i>).
Desarrollo de la actividad:		
<p>Primera parte: ¿Cómo me siento?</p> <p>Sugerencia de tiempo invertido: 10 minutos.</p> <p>La respiración es una actividad que si se hace desde la conciencia ayuda a conectar con nuestro cuerpo. Nos permite hacer un escaneo de cómo está nuestro cuerpo, atender sus necesidades y prevenir enfermedades.</p> <ol style="list-style-type: none"> 1. Se indicará a la participante cerrar los ojos y realizar tres respiraciones profundas. 2. Con la respiración deberá ir recorriendo su cuerpo, concentrándose en él y en sus sensaciones. Deberá realizarlo despacio, de manera tranquila y consciente. 3. Luego de ésto, en un lapso no mayor a cinco minutos, la participante elegirá una parte de su cuerpo, que quiera trabajar, y realizará las siguientes preguntas: <ol style="list-style-type: none"> a. En estos momentos, ¿qué sientes en esta parte del cuerpo que elegiste? 		

- b. La parte del cuerpo elegida, ¿se siente tensa o relajada?, ¿liviana o cansada?
 - c. Respira y describe brevemente tus sensaciones.
4. Al finalizar, hará tres respiraciones profundas y abrirá los ojos nuevamente.

Segunda parte: ¿Qué controla mi teléfono y sus apps?

Sugerencia de tiempo invertido: 1 hora y 50 minutos.

1. Se proporcionará a la participante varios sitios web para la búsqueda de información que ayude a responder las siguientes preguntas:
 - ¿Qué es un sistema operativo y cuáles son los más usados en los dispositivos?
 - ¿Qué es el sistema operativo Android y cuáles son sus principales características?
 - ¿Qué es una aplicación móvil y cuál es su utilidad?
 - ¿Cuáles son los tipos de aplicaciones móviles que existen? (Nativas, web, híbridas).
2. Se indicará realizar un organizador gráfico con la información obtenida, este puede ser elaborado en papel o de manera electrónica. (Se recomienda utilizar los enlaces proporcionados).
3. Posteriormente, la participante explorará su dispositivo móvil con el fin de identificar las diversas aplicaciones que posee, su tipo y utilidad.
4. Le proporcionará la plantilla A (disponible en el anexo *MMT1A1_Anexos.pdf*), en ella deberá anotar el nombre de la aplicación, tipo y utilidad.

Tercera parte :La memoria y los archivos de mi teléfono.

Sugerencia de tiempo invertido: 2 horas.

1. Se proporcionará a la participante varios sitios web para la búsqueda de información que ayude a responder las siguientes preguntas:
 - ¿Qué es la memoria interna física?
 - ¿Qué es la memoria root?
 - ¿Qué son las carpetas *SDCARD*, *data*, *app*?
 - ¿Qué es un archivo *APK*?
 - ¿Qué es un explorador de archivos para Android?
 - ¿En qué ayuda un explorador de archivos para Android?
 - ¿Qué tipos de explorador de archivos existen para Android?

- ¿Qué es el almacenamiento interno y externo en android?
2. Se le proporcionará a la participante la plantilla B (disponible en el anexo *MMT1A1_Anexos.pdf*) donde resolverá un crucigrama, el cual contiene preguntas referentes al tema, cuyas respuestas lo completan.

Links de apoyo (parte 2):

- Wikipedia La enciclopedia libre. (2008). *Android*.
<https://es.wikipedia.org/wiki/Android> (Noviembre, 2020).
- android.com (Sin fecha). *Qué es Android. La plataforma que lleva a los dispositivos móviles más allá*.
https://www.android.com/intl/es-419_mx/what-is-android/ (Noviembre, 2020).

Links de apoyo (parte 3):

- ChicaGeek. (2019). *MÁS MEMORIA para tu Android con tarjetas SD | ChicaGeek*. [video]
https://www.youtube.com/watch?v=DMCvih-0j7s&ab_channel=ChicaGeek (Noviembre, 2020).
- Ayuda de Android.(Sin fecha). *Cómo buscar y borrar archivos en Android*.
<https://support.google.com/android/answer/9110661?hl=es-419> (Noviembre, 2020).
- Teso, N. (2018). *Cómo utilizar el Explorador de Archivos integrado de Android*.
<https://androidayuda.com/2018/01/03/explorador-de-archivos-integrado-android/> (Noviembre, 2020).

Links de apoyo:

Para el organizador gráfico:

- Cacao by nulab. (Sin fecha). *Los equipos que hacen diagramas juntos crecen juntos*.
<https://cacao.com/es/> (Mayo, 2020).
- Popplet. (Sin fecha). *The simplest Way to Visualize Ideas. Capture, organize, collaborate*.
<http://popplet.com/> (Mayo 2020).
- Spicynodes.
Popplet. (Sin fecha). *The simplest Way to Visualize Ideas. Capture, organize, collaborate*.
<http://popplet.com/> (Mayo 2020).

Actividad 2: Estableciendo el ambiente de trabajo

Aprendizaje esperado: Establecer los requerimientos necesarios del entorno de trabajo para comenzar a desarrollar aplicaciones móviles mediante <i>App Inventor</i> .		Duración de la actividad: 3 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Cuenta de correo electrónico. • Acceso a la página web de <i>MIT App Inventor</i>. • Anexo <i>MMT1A2_Anexos.pdf</i>. 	<p>Evidencia:</p> <ol style="list-style-type: none"> 1. Captura de pantalla del sitio <i>MIT App Inventor</i>. <p>Producto:</p> <ol style="list-style-type: none"> 1. Generar un proyecto nuevo, listo para iniciar su primera aplicación. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> • Revisar la identificación de los componentes de AppInventor en la plantilla A del Anexo <i>MMT1A2_Anexos.pdf</i> • Mostrar a través de la captura de pantalla el proceso que siguió la participante en la generación de un proyecto nuevo y la incorporación de 6 herramientas de App Inventor. • Probar el funcionamiento de la aplicación creada en el emulador de Android.
Desarrollo de la actividad:		
<p>Primera parte: Agradezco a mi cuerpo.</p> <p>Sugerencia de tiempo invertido: 10 minutos.</p> <ol style="list-style-type: none"> 1. Se indicará a la participante cerrar los ojos y realizar tres respiraciones profundas. 2. A la par, deberá concentrarse en su cuerpo, lo recorrerá de forma pausada, tranquila y consciente. 3. Al tiempo que piensa en cada parte de su cuerpo, agradecerá lo que cada una (extremidades, órganos, cabeza, etc.) hace por ella. Por ejemplo, al recorrer su cabeza agradecerá por las ideas que ésta le permite crear. 4. Luego de ello, tomará algunos minutos para dialogar con su cuerpo y si así lo decide podrá crear un compromiso de cuidado con el mismo. 5. Para finalizar, realizará tres respiraciones profundas y podrá abrir nuevamente los ojos. <p>Segunda parte: Mi cuenta de correo.</p>		

Sugerencia de tiempo invertido: 30 minutos.

1. Se solicitará ingresar al sitio de Gmail, para crear una cuenta de correo electrónico.
2. Se recomienda crear un nombre de usuario para uso profesional (evitar pseudónimos y palabras aleatorias u ofensivas). Si la participante tiene una empresa o emprendimiento o piensa iniciar uno, se recomienda utilizar el nombre de la organización que posee.

Tercera parte: Conociendo MIT App Inventor.

Sugerencia de tiempo invertido: 50 minutos.

1. La participante deberá registrarse en el sitio web de *MIT App Inventor*, para esto deberá utilizar la cuenta Google que creó previamente.
2. Deberá explorar todo el sitio para identificar las herramientas que posee.
3. Se le proporcionará a la participante la plantilla A (disponible en el anexo *MMT1A2_Anexos.pdf*) la cual contiene unas capturas de pantalla del sitio *MIT App Inventor*. En ella, deberá anotar las herramientas que identifique y una breve descripción de su utilidad.

Cuarta parte: Mi primer proyecto.

Sugerencia de tiempo invertido: 30 minutos.

1. Una vez que la participante se haya relacionado con el sitio web *MIT App Inventor*, podrá crear su primer proyecto. Para esto, deberá situarse en el menú superior color gris, dar clic en *Proyecto> Iniciar nuevo proyecto* y asignar un nombre al proyecto.
2. Deberá implementar al menos seis de las herramientas identificadas en la segunda parte de la actividad.
3. Al finalizar, tomará una captura de pantalla y explicará de qué forma incorporó las herramientas en su proyecto.

Quinta parte: El emulador de Android.

Sugerencia de tiempo invertido: 1 hora.

App Inventor proporciona un emulador de Android que funciona igual que un dispositivo móvil con Android, sin embargo este aparece en la pantalla de la computadora. Su utilidad radica en que la participante pueda probar sus aplicaciones en un emulador y así distribuir la aplicación a otros.

1. Para usar el emulador, primero deberá instalar el software en la computadora (en los *Links de apoyo* se encuentra el enlace para la descarga).
2. Inicialá manualmente *aiStarter* el cual se encuentra en la carpeta `/usr/google/appinventor/command-for-Appinventor;` o

bien, desde la línea de comandos con `/usr/google/appinventor/command-for-appinventor/aiStarter &`.

3. Ejecutará de nuevo *App Inventor* y abrirá un proyecto (o crear uno nuevo).
4. Posteriormente, desde el menú de App Inventor, deberá ir al menú *Conectar* y hacer clic en la opción *Emulador*.
5. Realizado lo anterior, recibirá un aviso en el que se enuncia que el emulador se está conectando.
6. Se deberá iniciar el emulador (puede tomar un par de minutos). Esto se indicará con un fondo de pantalla a color.
7. Se desplegará el mensaje ¡Instalación completa! ¡Ahora estás listo para probar tu primera aplicación!

Nota para la tallerista:

- Existen otras dos formas de probar las apps desarrolladas en *MIT App Inventor*: por conexión WIFI o descargándolas en su dispositivo Android (celular o tableta).

Links de apoyo:

Emulador.

- MIT APP INVENTOR (Sin fecha). *Installing App Inventor 2 Setup on GNU/Linux*.
<http://appinventor.mit.edu/explore/ai2/linux> (Octubre, 2020).

Actividad 3: Diseñando mi primera aplicación

Aprendizajes esperados: Identificar los componentes básicos en el diseño de una aplicación para conocer los componentes disponibles en el menú <i>Interfaz de usuario</i> y el menú <i>Disposición</i> que se encuentran en la <i>Paleta</i> . Conocer los componentes disponibles en el menú <i>Interfaz de usuario</i> y el menú <i>Disposición</i> que se encuentran en la <i>Paleta de MIT App Inventor</i> .		Duración de la actividad: 4 horas y 30 minutos.
Recursos <ul style="list-style-type: none"> • Descarga de imágenes en formato jpg. • <i>MIT App Inventor</i>. • <i>MMT1A3_Anexo.pdf</i> 	Evidencia/producto Evidencia <ol style="list-style-type: none"> 1. Captura de pantalla del proyecto finalizado del tutorial <i>Hello Codi!</i> 2. Captura del área de trabajo (Visor, Componentes) del proyecto “MiBiografía”. Producto <ol style="list-style-type: none"> 1. Proyecto “MiBiografía”. 	Retroalimentación/Evaluación Retroalimentación <ul style="list-style-type: none"> • Verificar el término del proyecto Hello Codi. • Comprobar que reconoce el uso de los componentes para el diseño de una App en <i>MIT App Inventor</i>. Evaluación <ul style="list-style-type: none"> • El proyecto “MiBiografía” debe contener lo estipulado en la tercera parte de la actividad.
Desarrollo de la actividad: Primera parte: Desmontando la App. Sugerencia de tiempo invertido: 2 horas. La participante experimentará con los diversos componentes disponibles en la sección de <i>Interfaz de Usuario</i> en <i>MIT App Inventor</i> , para lo cual se recomienda hacer uso de las instrucciones contenidas en el anexo <i>MMT1A3_Anexo.pdf</i> . Segunda parte: Galería de Apps. Sugerencia de tiempo invertido: 30 minutos.		

La participante identificará los componentes básicos del diseño (mencionados en la primera parte de la actividad) en al menos dos aplicaciones de la *Play Store* o en alguno de los proyectos ya realizados en *MIT App Inventor* que llamen su atención (la liga a los proyectos puede encontrarse en los *Links de apoyo*).

Tercera parte: Mi Biografía.

Sugerencia de tiempo invertido: 1 hora.

La participante retomará y comenzará agregar contenido a su proyecto “MiBiografía”, haciendo uso de los elementos vistos a lo largo de la actividad. La biografía deberá contener una imagen y las etiquetas que considere necesarias para desplegar la información. Para verificar que se han aplicado los conceptos aprendidos hasta ahora, deberá ser identificable:

- El uso de por lo menos un elemento de *Disposición*.
- El ajuste del ancho y el alto de por lo menos dos formas distintas.
- Al menos un cambio en la posición del texto (centrado, derecha o izquierda), tamaño de letra y formato del texto (negritas o cursiva).

Se sugiere tomar en cuenta los siguientes puntos para la realización de “MiBiografía”:

- Nombrar al menos tres acontecimientos importantes en mi vida y preguntarse:
 - ¿Quién fui antes de ello?
 - ¿Qué personas estuvieron presentes en ese momento y qué función desempeñaron?
 - ¿Qué aprendiste de ese evento con respecto al cuidarse o descuidarse a sí misma?

La participante tomará una captura de pantalla de su proyecto.

Cuarta parte: Hello Codi!

Sugerencia de tiempo invertido: 1 hora.

La participante completará el tutorial del tutorial *Hello Codi* para principiantes que se encuentran en el enlace *Beginners Tutorials* (disponible en los *Links de apoyo*).

Notas:

- El sitio *MIT App Inventor* por defecto se encuentra en el idioma inglés, pero es posible cambiar la configuración al español.

- Al final del documento se anexa la liga para la herramienta *Color Tool de Material Design* que ayuda a realizar combinaciones de colores y mostrar cómo se visualizarán en la aplicación.

Notas para aprender más:

- La participante podrá completar el resto de los tutoriales para principiantes que se encuentran en el enlace (<http://appinventor.mit.edu/explore/ai2/beginner-videos>).

Links de apoyo:

- MIT APP INVENTOR (Sin fecha). *Beginners Tutorials*.
<http://appinventor.mit.edu/explore/ai2/beginner-videos> (Octubre, 2020).

Para la segunda parte de la actividad:

- Google Play (Sin fecha). *Botón de pánico Ixtapaluca*.
https://play.google.com/store/apps/details?id=mx.gob.ixtapaluca.alerta&hl=es_MX (Julio, 2020).
 - FGJ EDOMEX.
https://play.google.com/store/apps/details?id=com.oomovil.procurapp&hl=es_MX (Julio, 2020).
 - *Mujer Siempre Alerta*.
https://play.google.com/store/apps/details?id=com.igmm.musa&hl=es_MX (Julio, 2020).
 - *Vive Segura CDMX*.
https://play.google.com/store/apps/details?id=mx.gob.df.inmujeres.app.vivesegura&hl=es_MX (Julio, 2020).
- COLOR TOOL.(Sin fecha). USER INTERFACES ACCESSIBILITY.
<https://material.io/resources/color/> (Julio, 2020).
- MIT APP INVENTOR. (Sin fecha). *MIT APP INVENTOR OF THE MONTH*.
<http://appinventor.mit.edu/explore/app-month-gallery> (Julio, 2020).
- Tutorial Hello Codi!
MIT APP INVENTOR (Sin fecha). *Beginner Tutorials*.
<http://appinventor.mit.edu/explore/ai2/beginner-videos> (Julio, 2020).

Actividad 4: Mi catálogo con sonidos

Aprendizaje esperado: Elaborar una aplicación básica que relacione componentes visuales (imágenes) y auditivos. Generar y/o descargar archivos (imágenes y audio) para elaborar una aplicación básica que relacione estos componentes. Generar el acceso a la aplicación para su prueba a partir de un emulador o mediante conexión wifi con el dispositivo.		Duración de la actividad: 3 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Archivos de imágenes de un tema (por ejemplo: animales, coches, etc) en formato jpg. Archivos de sonidos relacionados con las imágenes en formato MP3. Emulador o dispositivo móvil. Anexo <i>MMT1A4_Anexos.pdf</i>. 	Evidencia: <ol style="list-style-type: none"> Documento nombrado “Evidencias de NOMBREDELPROYECTO”. El documento “Evidencias de NOMBREDEMIPROYECTO”. Producto: <ol style="list-style-type: none"> Aplicación que relacione imágenes con sonidos al hacer click en un botón mediante el uso de eventos. 	Retroalimentación: El documento “Evidencias de NOMBREDELPROYECTO”, tendrá que contener una imagen, resaltando las secciones descritas en el punto 2 Instrucciones de la “Tercera parte: Definiendo el funcionamiento”. El documento “Evidencias de NOMBREDEMIPROYECTO” deberá incluir una captura en su documento del <i>Visor</i> en modo <i>Bloques</i> mostrando su código sin errores. Evaluación: <ul style="list-style-type: none"> Probar la aplicación y su correcto funcionamiento a través de un emulador o dispositivo móvil.
Desarrollo de la actividad: Primera parte: Mis metas con la app. Sugerencia de tiempo invertido: 30 minutos. La participante:		

1. Identificará el objetivo de su aplicación. Para ello, se recomienda el uso de las siguientes preguntas guía, utilizando el formato tweet (140 caracteres por respuesta):
 - a. ¿Qué pretendo lograr con mi app? (Atender problemáticas o necesidades).
 - b. ¿Por qué? (Motivaciones).
 - c. ¿Para qué? (Problematizar).
2. Descargará o generará las imágenes y sonidos para su app, al menos tres de cada uno. Se recomienda que si la participante decide producir sus propios audios e imágenes, lo podrá hacer con una temática libre, para ello, puede utilizar de inspiración el proyecto *Sonidos de mi barrio* (el enlace se encuentra al final de esta sección).

Segunda parte: Creando la interfaz.

Sugerencia de tiempo invertido: 1 hora.

La participante creará una nueva app utilizando los archivos recabados en la primera parte de esta actividad. Se recomienda utilizar las instrucciones proporcionadas en el anexo *MMT1A4_Anexos.pdf*.

Tercera parte: Definiendo el funcionamiento.

Sugerencia de tiempo invertido: 1 hora.

La participante dotará de funcionamiento a su aplicación, para lo cual, trabajará en la pestaña de *Bloques*.

1. El primer paso será identificar la pestaña *Bloques* (el botón para cambiar de pestaña se encuentra en la parte superior derecha).
2. Habrá dos secciones: *Visor* y *Bloques*. En *Visor*, se arrastrarán los elementos que se seleccionen del menú *Bloques*, contendrá los elementos que se agreguen a la aplicación desde *Diseñador*, además de la sección *Integrados*, misma que cuenta con instrucciones similares a las de los lenguajes de programación.
3. Explorará cada una de las opciones disponibles en *Integrados* dentro de la sección *Bloques* y escribirá en el documento “Evidencias de NOMBREDELPROYECTO”, con qué secciones se encuentra familiarizada y con cuáles no. En esta parte es indispensable orientar a la participante.
4. Posteriormente, seleccionará uno de los componentes *Sonido*, que al ser presionado desplegará una lista con elementos de distinto color. Analizando las acciones disponibles para cada color, relacionará cada color con una de las siguientes opciones (deberá incluir su respuesta en el documento “Evidencias de NOMBREDELPROYECTO”):
 - a. Evento (realizar una acción cuando algo ocurra).

- b. Función (realizar una acción).
 - c. Asignar un determinado valor a las propiedades del componente.
 - d. Referencia a las propiedades del componente.
5. Dotará a cada botón con la acción de reproducir el sonido correspondiente cuando sea presionado.
 6. Tomará una captura del *Visor* en modo *Bloques* mostrando su código, dicha captura se incorporará al documento “Evidencias de NOMBREDELPROYECTO”.

Cuarta parte: Probando mi aplicación.

Sugerencia de tiempo invertido: 30 minutos.

En esta parte de la actividad, la participante deberá probar su aplicación, apoyándose de las tres formas que ofrece *MIT App Inventor*.

1. La participante identificará la sección *Generar* que se encuentra en la barra superior del sitio, basándose en sus observaciones e investigando los conceptos desconocidos, responderá lo siguiente:
 - a. ¿Cuáles son las dos opciones que se muestran al presionar *Generar*?
 - b. ¿Alguna de las dos te suena familiar?
 - c. De ser así, ¿Cuál y cómo crees que funcione?
 - d. ¿Qué es un código QR?
 - e. ¿Qué es un apk?
2. La tallerista deberá guiar a la participante en la implementación de las dos formas para probar su aplicación. Con ambas propuestas se debe tener cuidado, ya que en ocasiones es necesario indicar en las configuraciones del celular en el que se va a probar la aplicación, el permiso para instalar aplicaciones de terceros.
3. Una vez que la aplicación se haya instalado en el celular, se deberá destacar la creación del icono en el menú. La participante entrará a la aplicación y verificará el correcto funcionamiento de su catálogo de sonidos.
4. La última forma de probar la aplicación es a través del emulador que se instaló en la actividad 2 del taller. Será necesario identificar la sección *Conectar* ubicada en el mismo menú que *Generar*.
5. De entre las opciones se deberá seleccionar *Emulador*, al presionar esta opción, el *Emulador* se ejecutará automáticamente permitiendo probar la aplicación.
6. Probadas las tres formas de correr la aplicación, se plantearán las siguientes cuestiones:
 - a. ¿Cuál es la principal diferencia entre probar la aplicación utilizando el emulador e instalarla por el apk?
 - b. ¿Qué es un emulador?

- c. ¿Cuál consideras que es la mejor forma y por qué?
- d. ¿Cuál es más sencilla?
- e. ¿Por qué crees que existen tantas opciones distintas?

Nota:

- Puede haber problemas si el emulador no se encuentra instalado, además una vez que se ejecute se debe esperar hasta que se encuentre listo (esto nos lo indicará nuestro sistema).

Links de apoyo:

- SONIDOS DE MI BARRIO. (Sin fecha). *Pasajes sonoros de La Merced CDMX*.
<https://sonidosdemibarrío.org/> (Julio, 2020).

Actividad 5: Tomando fotos desde mi aplicación

Aprendizaje esperado: Implementar el acceso a la cámara del teléfono móvil para tomar fotos mediante una aplicación desarrollada con <i>App Inventor</i> .		Duración de la actividad: 4 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Dispositivo móvil con cámara. Medio para probar la aplicación. 	<p>Evidencia:</p> <ol style="list-style-type: none"> Captura de pantalla del teléfono móvil con los botones identificados. Fotografías capturadas y guardadas en la galería. <p>Producto:</p> <ol style="list-style-type: none"> Una aplicación que muestre en la pantalla un botón para iniciar la cámara, otro para tomar foto, y la opción para guardar o descartar la foto tomada. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> Presentar la captura de pantalla con los botones identificados. Visualizar las imágenes guardadas y comprobar la utilización de una imagen como fondo de la App. <p>Evaluación:</p> <ul style="list-style-type: none"> Mostrar el funcionamiento de la aplicación con la revisión de cada uno de los componentes utilizados (botones).
Desarrollo de la actividad:		
<p>Primera parte: Los botones en una app. Sugerencia de tiempo invertido: 1 hora.</p> <ol style="list-style-type: none"> Se indicará a la participante que investigue el funcionamiento de la cámara del dispositivo móvil que posee. Posteriormente, deberá indagar sobre el componente <i>Botón</i> para responder las siguientes preguntas: <ul style="list-style-type: none"> ¿Qué es el componente <i>Botón</i>? ¿Para qué se usa? ¿Qué propiedades tiene? Tomará una captura de pantalla de cualquier app de su dispositivo móvil donde identifique diferentes botones. Por último, investigará el funcionamiento de la sección de bloques de <i>MIT App Inventor</i> (se recomienda ver el video ubicado en los <i>Links de apoyo</i>). 		

Segunda parte: Creando botones.

Sugerencia de tiempo invertido: 1 hora.

1. Se iniciará un nuevo proyecto en *MIT App Inventor*, el cual se nombrará “App de Fotografía”.
2. Deberá hacer clic en *Conectar* y configurará su dispositivo (o emulador) para realizar pruebas en vivo.
3. A continuación, deberá posicionarse a la derecha del *Diseñador*, dirigirse al panel de *Propiedades* y cambiar el nombre de la pantalla; lo sustituirá por el nombre que se le vaya a dar a la app. Para comprobar el cambio en el dispositivo móvil deberá ver el nuevo título en la pantalla del emulador.
4. Realizado el cambio, hará clic en el componente *Botón* (se encuentra en la parte superior de la *Paleta* de interfaz de usuario en el *Diseñador*) y lo arrastrará al *Visor*. El *Botón* mostrará automáticamente "Texto para Botón1", éste se podrá cambiar fácilmente en el cuadro de texto del panel de propiedades (ubicado cerca de la parte inferior derecha de la pantalla).
5. El *Botón* podrá mostrar instrucciones como: "Enviar", "Siguiente", "Reproducir", etc., dependiendo de para qué se utilizará.
6. Deberá crear tres botones, uno para iniciar la cámara, otro para tomar foto, y otro para la opción de guardar o descartar la foto tomada. Cada botón deberá ser de un color diferente.
7. Para terminar, deberá probar en el emulador y comprobar que los botones aparezcan en la pantalla.

Tercera parte: Creando conexiones.

Sugerencia de tiempo invertido: 1 hora.

1. Una vez que la participante haya creado los tres botones, lo siguiente será hacerlos funcionar. Los botones estarán apilados uno encima de otro, sin embargo, se necesita que todos estén alineados de forma contigua en la parte superior de la pantalla. Para realizar esto, utilizará el componente *Arreglo Horizontal*.
2. En la lista de componentes del proyecto, se verán los tres botones indentados bajo el componente *Arreglo Horizontal*. Esto indicará que los botones son ahora subcomponentes del componente *Arreglo Horizontal*. (Todos los componentes estarán indentados en Screen1).
3. Desde la *Paleta*, deberá arrastrar un segundo arreglo horizontal y colocarlo debajo del lienzo. Luego, arrastrará los componentes de *Botón* a la pantalla y los colocará en el arreglo horizontal inferior.
4. Realizado lo anterior, cambiará el nombre del primer botón a “TomaUnaFoto” y su propiedad de Texto a “Captura”.

5. Repetirá la acción con el segundo botón y cambiará el nombre a “BotonDeGuardado” y su propiedad de texto a “Guardar”.
6. Se realizará lo mismo con el tercer botón y cambiará el nombre a “BotonBoteDeBasura” y su propiedad de texto a “Eliminar”.
7. Posteriormente desde el cajón de Medios, arrastrará un componente de la *Cámara* al *Visor*. Este deberá aparecer en el área del componente no visible.

Cuarta parte: Mi autorretrato.

Sugerencia de tiempo invertido: 1 hora.

Cuidar el cuerpo es parte del autoconocimiento. El cuerpo para las mujeres debe ser el primer territorio a explorar. En los lugares donde se ha instaurado el patriarcado como un régimen político, el cuerpo de las mujeres se expropia y se convierte en un vínculo solo para la procreación y el agrado de los “otros”, es por ello, que existen discursos culturales y mediáticos que hacen objeto el cuerpo de las mujeres, procurando que se desentiendan de su cuerpo. Un ejemplo de esta alienación es pensar que el cuerpo está separado de la persona, estar preocupada por parecerse a los modelos de mujeres que salen en las revistas y pensar que al cuerpo le falta o le sobra “algo” y estar a disgusto o sentir vergüenza por él, lo que a su vez ocasiona que las mujeres hagan dietas o quieran tener más curvas para cumplir con el estándar de lo que el cuerpo de una mujer debe ser, por ejemplo.

Ahora bien, en esta actividad la participante tomará tres fotos, la primera de la parte del cuerpo que más le guste, la segunda de la parte del cuerpo que menos le guste y la tercera en el lugar donde más segura y respetada se sienta.

Describirá cada una de las fotos y responderá a las siguientes preguntas:

- a. Foto 1. ¿Por qué te gusta esa parte de tu cuerpo?
- b. Foto 2. ¿Por qué te disgusta esa parte de tu cuerpo?
- c. ¿Qué tanto tu elección en la foto 1 o 2 está atravesada por los discursos que estereotipan el cuerpo de las mujeres?
- d. Foto 3. Enumera al menos tres elementos o personas que se encuentran en la foto para que te sientas segura.

Para darle vida a la aplicación, la participante tomará una foto con la cámara del dispositivo y la establecerá como fondo de la app.

1. Se creará dos bloques “Camara.TomaUnaFoto” y “Camara.DespuesDeFoto”.
2. El bloque de “Camara.TomaUnaFoto” deberá llamar a la cámara a tomar una foto cuando se haga click en el botón de

capturar.

3. Se solicitará tomar una foto.
4. Cuando el bloque anterior termine, comenzará el bloque “Camara.DespuesDeFoto” el cual deberá tener un argumento llamado *Imagen* el cual será la foto capturada, para que se pueda colocar como fondo en la aplicación y la participante la visualice.
5. Posteriormente, cuando la imagen aparezca en pantalla deberán mostrarse los botones creados en la tercera parte de la actividad, llamados “Guardar” y “Eliminar”.
6. La participante deberá indicar con los botones la acción que ejecutará el bloque de “Camara.DespuesDeFoto”.
7. Se probará la app, e indicará a la participante tomar distintas fotos de las partes de su cuerpo que más aprecia, deberá guardar las fotos que más le agraden y eliminar las que no le sean útiles.
8. Para finalizar, deberá seleccionar una foto para establecerla como fondo de la app.

Notas:

Para la primera parte de la actividad:

- Cambiar el texto de visualización del botón no debe confundirse con cambiar el nombre del botón. Cambiar el nombre del botón le ayuda al diseñador/codificador, a mantener sus botones organizados. Por lo tanto, "SubmitButton" es un nombre apropiado para un botón que muestra el texto "Enviar".
- Es una buena práctica cambiar el nombre de los botones para describir claramente su función, de modo que cuando comience a codificar en el Editor de bloques, el propósito de cada botón sea obvio.

Para la cuarta parte de la actividad

- Las aplicaciones *App Inventor* pueden interactuar con las potentes funciones de un dispositivo Android, incluyendo la cámara. Se recomienda que la cámara tenga dos bloques clave. El bloque “Camara.TomaUnaFoto” inicia la aplicación de la cámara en el dispositivo. El evento “Camara.DespuesDeFoto” se activa cuando el usuario ha terminado de tomar la foto y esta se establece como fondo de imagen para el usuario.

Links de apoyo:

- Villalpando, J. (Sin fecha). *App Inventor 2 en español*. Cómo programar los teléfonos móviles con Android “ 2W.- Botones en Paneles.”
http://kio4.com/appinventor/2W_Botones_Paneles.htm (Junio, 2020).
- CÓDIGO21 TECNOLOGÍAS CREATIVAS (Sin fecha). *Descripción de los bloques integrados de App Inventor 2*.
<https://codigo21.educacion.navarra.es/autoaprendizaje/descripcion-de-los-bloques-integrados-de-app-inventor-2/#control> (Junio, 2020).

Actividad 6: Haciendo un video con mi aplicación

Aprendizaje esperado: Establecer el acceso a la cámara del teléfono móvil mediante una aplicación desarrollada por la participante para obtener video y localizar el archivo.		Duración de la actividad: 4 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Dispositivo móvil con cámara. Medio para probar la aplicación. 	Evidencia: <ol style="list-style-type: none"> Videos grabados y guardados en la galería. Producto: <ol style="list-style-type: none"> Una aplicación que muestre en la pantalla un botón para iniciar la cámara, otro para grabar video, y la opción para guardar o descartar el video. 	Retroalimentación: <ul style="list-style-type: none"> Visualizar los vídeos guardados y comprobar que la app puede guardarlos en los formatos Windows Media Video (. Wmv), 3GPP (.3gp), o MPEG-4 (. Mp4). Evaluación: <ul style="list-style-type: none"> Mostrar el funcionamiento de la aplicación con la revisión de cada uno de los componentes utilizados (grabar, borrar, guardar video, pausar y controles multimedia reproducción / pausa, saltar hacia delante y saltar hacia atrás).
Desarrollo de la actividad:		
Primera parte: Mis videos en Android. Sugerencia de tiempo invertido: 1 hora. <ol style="list-style-type: none"> Se indicará a la participante realizar una búsqueda en la web para responder las siguientes preguntas: <ul style="list-style-type: none"> ¿Qué es un archivo de video? ¿En dónde se guardan los archivos de video en Android? ¿Qué formatos de video son compatibles con Android? 		

Segunda parte: Mis botones de video.

Sugerencia de tiempo invertido: 1 hora.

Para esta parte de la actividad se recomienda retomar el proyecto de la *MMT1A5* en donde se instruye a la participante en la creación de botones y acceso a la cámara del dispositivo móvil.

1. Se deberá contar con cuatro botones en la pantalla, cada uno de diferente color e identificarlos como “Grabar” “Captura” “Borrar” y “Guardar”.
2. Los botones indentados deberán estar alineados uno al lado del otro. Para realizar esto, deberá utilizar el componente *Arreglo Horizontal* nuevamente.
3. Desde la *Paleta*, deberá arrastrar un segundo *Arreglo Horizontal* y colocarlo debajo del lienzo. Luego, arrastrará los componentes de Botón a la pantalla y los colocará en el arreglo horizontal inferior.
4. Desde *Media* deberá arrastrar los componentes de *Video* y *Cámara*.

Tercera parte: 3, 2, 1, ¡Acción!

Sugerencia de tiempo invertido: 1 hora.

1. La participante deberá dirigirse a *Bloques* y realizará lo siguiente:
 - Al hacer clic en el botón “Captura” se deberá abrir la cámara.
 - Al hacer clic en el botón “Grabar” se deberá accionar el video.
2. Agregará las conexiones a los bloques de “Borrar” y “Guardar” usando las sentencias *si entonces*.
3. Para finalizar, probará la app grabando un vídeo. Se sugiere que la participante se grabe así misma respondiendo brevemente la pregunta ¿qué entiendes por autocuidado?.
4. Una vez realizado el vídeo, se solicitará realizar una investigación rápida acerca del concepto desde el feminismo. Para ello, se propone revisar el vídeo: Caleidoscopio 11. Autocuidado en https://www.youtube.com/watch?v=ypNiY_i1cmM.
5. La participante deberá realizar un segundo vídeo donde comparta, con base en lo investigado, uno o varios consejos de autocuidado que realiza en su vida cotidiana.

Cuarta parte: ¿Y dónde quedó todo?

Sugerencia de tiempo invertido: 1 hora.

La participante deberá poder revisar y usar los archivos multimedia que creó en la actividad *MMT1A5* y *MMT1A6*.

1. Se proporcionará un vínculo a la participante (ver sección *Links de apoyo*) que deberá consultar para responder de forma precisa a la siguiente pregunta:
 - ¿En dónde y cómo se guardan los archivos creados en *MIT App Inventor*?
 - a. Deberá utilizar un componente *Player* para reproducir archivos de sonido largos, archivos de video y para hacer vibrar el teléfono.
 - b. Para reproducir archivos de sonido cortos, tales como efectos de sonido, utilizará un componente *Sound*.
2. La aplicación deberá poder controlar el comportamiento de la reproducción mediante una llamada a los métodos *Start*, *Pause* y *SeekTo*.
3. Los archivos de vídeo deberá guardarse en los formatos Windows Media Video (. Wmv), 3GPP (.3 gp), o MPEG-4 (. Mp4).

Notas:

- *AfterPicture* (ruta al archivo de imagen) se dispara después de tomar la fotografía. El argumento es la ruta que se puede utilizar para localizar la imagen en el teléfono.
- El *Reproductor de video* es un componente multimedia que reproduce vídeos. Un reproductor de vídeo aparece en su aplicación como un rectángulo. Si el usuario pulsa el rectángulo, los controles multimedia aparecen: reproducción/pausa, saltar hacia delante y saltar hacia atrás.

Links de apoyo:

- Villalpando, J. (Sin fecha). *App Inventor 2 en español*. Cómo programar los teléfonos móviles con Android. “Almacenamiento.”
http://kio4.com/appinventor/8archivo_basedatos.htm (Julio, 2020).
- MIT App inventor Beta. Aprende App Inventor. (Sin fecha). *Componentes multimedia*.
<https://sites.google.com/site/aprendeappinventor/documentacion/componentes-multimedia> (Julio, 2020).
- Tal Y MIT App Inventor tutorials (2012). *Camcorder, Camera, Player, Sound, VideoPlayer*. [video]

https://www.youtube.com/watch?v=Rrf_PP_upv4 (Julio, 2020).

- La Quimera Feminista (2019). *Caleidoscopio* 11-Autocuidado. [video]
https://www.youtube.com/watch?v=ypNiY_i1cmM (Julio, 2020).

Actividad 7: Mis coordenadas

Aprendizaje esperado: Implementar el acceso sensor de ubicación GPS de mi teléfono celular mediante una aplicación desarrollada por la participante para obtener las coordenadas de ubicación en tiempo real.		Duración de la actividad: 4 horas.
Recursos <ul style="list-style-type: none"> Dispositivo móvil con sensor GPS. Medio para probar la aplicación. <i>MMT1A7_Anexos.pdf</i> 	Evidencia/producto <p>Evidencia:</p> <ol style="list-style-type: none"> Organizador gráfico sobre el funcionamiento del GPS. Mapeo el GPS de mi cuerpo. <p>Producto:</p> <ol style="list-style-type: none"> Una aplicación que muestre en la pantalla un botón y/o icono para pedir las coordenadas de localización actuales y las guarde en una lista. Con botón que pregunte “¿guardar coordenadas?” y otros dos para “sí” o “no”. Lista de tareas. 	Retroalimentación/Evaluación <p>Retroalimentación:</p> <ul style="list-style-type: none"> Presentar un organizador gráfico que exhiba el funcionamiento más simple (ubicación y traslado de punto A a B) del GPS. Presentar el Mapeo del GPS de mi cuerpo, el cual contiene una silueta, momentos y emociones. Verificar que los Componentes hayan sido renombrados. <p>Evaluación:</p> <ul style="list-style-type: none"> Lista de tareas que contengan el funcionamiento de la aplicación por pasos. Documento que contenga 3 capturas de pantalla en las que se muestren las coordenadas de las 3 ubicaciones elegidas, obtenidas mediante la aplicación.
Desarrollo de la actividad:		
Primera parte: GPS cotidiano. Sugerencia de tiempo invertido: 30 minutos. La participante:		

1. Identificará el funcionamiento del GPS en dispositivos móviles, se sugiere realice un diagrama de flujo haciendo uso del material recopilado en los “links de apoyo”.
2. Explorará la herramienta Google Maps y distinguirá sus componentes.
3. Posteriormente, escribirá un uso de la herramienta así como alguna ventaja y desventaja.

Segunda parte: El GPS de mi cuerpo.

Sugerencia de tiempo invertido: 30 minutos.

La participante:

1. Realizará una reflexión sobre tres situaciones del pasado que haya disfrutado.
2. Reconocerá las partes de su cuerpo que se vieron involucradas en esos momentos importantes.
3. Hará un dibujo del lugar donde se encontraba y dentro de él ubicará su silueta.
4. Posteriormente, coloreará las partes del cuerpo en las que sintió alguna emoción y las nombrará.
Ejemplo. Lugar: Fiesta de cumpleaños en casa de mis padres. Dibujo: Una casa que contiene mi silueta, en ella, está coloreado el estómago y hay una línea que indica que sentí “felicidad”.
5. Este mapeo servirá como evidencia para la actividad.

El propósito de la reflexión es realizar un reconocimiento de las partes que constituyen el cuerpo, de manera que permita apreciar cuán importante es cada parte, aunque a veces no le demos tanta importancia.

Tercera parte: Mis pasos en la ciudad.

Sugerencia de tiempo invertido: 2 horas.

La participante elaborará una aplicación que le permitirá llevar un registro de su ubicación a lo largo de su día.

1. Se recomienda realizar una pequeña reflexión respecto a ¿cómo esta aplicación podría ser útil en su día a día? Por ejemplo, medir tiempos de traslado entre dos ubicaciones.
2. Posteriormente, creará un nuevo proyecto en la página de *MIT App Inventor* titulado “Localizapp”.

Interfaz

3. Deberá crear la interfaz para su proyecto, la cual contendrá los siguientes elementos:
 - Para desplegar última ubicación:

- Dos etiquetas que contengan latitud, longitud.
 - Un visor web.
 - Para tomar una ubicación:
 - Un botón.
 - Sensor de localización.
 - Para tomar la hora:
 - Reloj.
 - Para mostrar el historial de ubicaciones:
 - Visor de lista.
4. Se recomienda utilizar el diseño que se muestra en el anexo *MMT1A7_Anexos.pdf*, aunque la participante podrá generar cualquier otro diseño personalizado, siempre y cuando cuente con todos los elementos descritos.
 5. Por último, renombrará sus componentes (principalmente las etiquetas) por unos más significativos, con la finalidad de que se puedan reconocer fácilmente desde la sección *Bloques*. Esto sugiere una buena práctica para el futuro, por lo que será considerado como parte de la evaluación.

Funcionamiento

6. La participante realizará una pequeña lista de tareas con las acciones que se deberán implementar para lograr el objetivo de la aplicación, esta lista será revisada en conjunto con la tallerista, a fin de establecer una guía de los pasos a seguir.

El funcionamiento de la aplicación será el siguiente:

- Al presionar el botón, comenzará la lectura de la ubicación. Leída la información, la ubicación será desplegada en las etiquetas correspondientes, en el *visor web* se cargará la página de google maps con las coordenadas leídas y por último, se añadirá la información de la ubicación al registro de ubicaciones (se añadirá un elemento al *Visor de lista*).
7. Es posible brindar este enunciado a la participante, para que descomponga el problema en partes y de esta forma arme su lista de tareas. La lista de tareas será parte de la evidencia de la actividad.
 8. Establecidas las acciones, la participante implementará las actividades de su lista, una por una, procurando hacer uso de procedimientos. Si la participante ya ha tenido acercamiento con el mundo de la programación, no le será tan difícil

comprender el concepto, de cualquier manera, en la sección de “links de apoyo” se incluyen materiales que pueden ser útiles para abordar esta parte.

Cuarta parte: De ruta conmigo.

Sugerencia de tiempo invertido: 1 hora.

La participante probará su aplicación, para ello, podrá implementar alguna de las formas mencionadas en *MMT1A4*.

Para comprobar la aplicación, la participante identificará tres ubicaciones de su día a día, tomará una captura de pantalla de cada una de esas ubicaciones y las incorporará en un documento entregable.

Para reflexionar: Dadas las circunstancias actuales de violencia, es recomendable variar las rutas para trasladarse de un punto A a un punto B, responder las preguntas presentadas a continuación.

- ¿Sueles variar de ruta cuando acudes a algún destino con frecuencia?
- ¿Compartes tu ubicación con familiares o amigas? ¿Por qué?
- ¿Consideras que las aplicaciones móviles pueden ser herramientas útiles para atender la violencia que se vive actualmente?

Links de apoyo:

- Date un Voltio. (2015). *¿Cómo funciona el GPS?* [video]
<https://www.youtube.com/watch?v=8tL-UBNsCv8> (Junio, 2020).
- GENIAL. (2019). *Explicación sencilla de cómo funciona el GPS.* [video]
<https://www.youtube.com/watch?v=IT7RzYcZnSc> (Junio, 2020).
- fcaenlinea1.unam.mx (Sin fecha). *Organizadores Gráficos.*
http://fcaenlinea1.unam.mx/anexos/organizadores_graficos.pdf (Julio, 2020).
- Herrera, F. (2018). *Funciones en programación.* [video]
<https://www.youtube.com/watch?v=Mih3pbP4EFc> (Julio, 2020).
- Palomares, K. (2019). *¿QUÉ es una FUNCIÓN en programación? [Diccionario del PROGRAMADOR].* [video]
<https://www.youtube.com/watch?v=HWeN8mCP7i0> (Julio, 2020).

Actividad 8: Obteniendo mi ubicación a partir de una fotografía desde mi teléfono

Aprendizaje esperado: Desarrollar la integración de la cámara y el GPS del teléfono móvil en una aplicación para que al tomar una foto se obtenga la ubicación en tiempo real.		Duración de la actividad: 6 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Dispositivo móvil con cámara y sensor GPS. Medio para probar la aplicación. 	<p>Evidencia: Una galería de tres fotos con ubicaciones accesibles mediante la app diseñada por la participante.</p> <p>Producto: 1. Una aplicación que muestre en la pantalla botones para tomar foto y una vez tomada activar otro botón para obtener las coordenadas de localización actuales. Guardar información relacionando el archivo de la imagen con la ubicación.</p>	<p>Retroalimentación: Verificar que las tres fotos contengan la información de la ubicación.</p> <p>Evaluación:</p> <ul style="list-style-type: none"> Mostrar una lista de las tres últimas imágenes generadas en la aplicación con sus correspondientes coordenadas de ubicación. Presentar un organizador gráfico que muestre la información de una fotografía obtenida desde sus metadatos y descripción de al menos dos funciones que pueden ser ejecutadas con esos datos (Ejemplo: Ubicación vía GPS y fecha de creación para realizar respaldos en la nube).
Desarrollo de la actividad:		
<p>Primera parte: ¿Cuánta información hay en una foto?</p> <p>Sugerencia de tiempo invertido: 30 minutos.</p> <p>La participante:</p> <ol style="list-style-type: none"> Identificará el concepto básico de <i>metadatos</i> y elaborará un organizador gráfico que tenga como centro una fotografía de celular, explicará los datos que contiene una fotografía y sus posibles aplicaciones. Se sugiere utilizar los materiales proporcionados en los “links de apoyo”. Visualizará los metadatos de tres diferentes tipos de archivos, a elección libre. 		

- Windows -> Para poder visualizar los metadatos de un archivo se debe dar clic derecho sobre el archivo y seleccionar la opción *Propiedades*, una vez ahí se debe seleccionar *Detalles* lo cual desplegará los metadatos de nuestros archivos.
- Linux -> Es necesario instalar la herramienta *exiftool* mediante el siguiente comando `sudo apt install libimage-exiftool-perl`, ya que está instalada es posible ver los metadatos de los archivos haciendo uso del comando `<<exiftool nombre_archivo.extension>>`.

Segunda parte: Desarrollando la app.

Sugerencia de tiempo invertido: 3 horas.

La participante realizará una aplicación que le permita conocer su ubicación a partir de una fotografía tomada con la cámara de su teléfono. Es posible retomar el proyecto de la actividad *MMT1A6*, en el que se tomaba una fotografía y se mostraba la imagen recién tomada o, de creerlo necesario, puede realizar un nuevo proyecto con el nombre sugerido “LocalizappMe”.

Para esta aplicación es necesario incluir una extensión al proyecto, que como su nombre lo indica, nos ayudará a extender funcionalidades. Para añadir la extensión *Image Metadata Extension* es necesario seguir los siguientes pasos:

1. Descargar el archivo de la extensión, cuyo link se adjunta al final del documento.
2. Abrir el proyecto con el que se va a trabajar.
3. Al fondo del panel *Paleta*, localizar la opción con la leyenda *Extensión*.
4. Presionar *Extensión* y seleccionar *Import extension*.
5. Elegir la opción *subir archivos de mi computadora* y seleccionar el archivo de la extensión cuya terminación debe ser *aix*.
6. Si todo funcionó de manera correcta, debajo de *Import extension* aparecerá la extensión con nombre *Taifun Metadata*.

Interfaz

- Añadir el componente *Taifun Metadata* arrastrándolo al screen como cualquier componente.
- Añadir dos etiquetas para almacenar latitud, longitud.
- Añadir un componente *VisorWeb* para desplegar la ubicación en el mapa.
- Es preciso recordar que todos los componentes deben poseer nombres descriptivos como requisito indispensable.

Funcionamiento

- Implementar el procedimiento *get* que recibe como parámetros: *listMetadata*, *directory* y *tag*. El cual puede encontrarse

en la página *Image Metadata Extension* incluida en los “links de apoyo”.

- Para obtener la ubicación a partir de la foto tomada deberá realizar lo siguiente: en el bloque de la cámara después de tomar foto, mandar a llamar al procedimiento que se acaba de crear, con los parámetros;
 - *listMetadata* -> será igual a lo que devuelva el bloque <<llamar TaifunMetadata1.Obtener>> pasándole como parámetro la imagen que se acaba de tomar;
 - *directory* -> será igual a la cadena “GPS”;
 - *tag* -> será la cadena “GPS Latitude” o “GPS Longitude” según lo que se requiera conocer.
- Obtenidos los datos, se deberán mostrar en las etiquetas correspondientes.
- Finalmente, para desplegar la información en un mapa deberá realizar el mismo proceso que en la actividad *MMT1A7*.

Tercera parte: Probando la app.

Sugerencia de tiempo invertido: 2 horas.

Para probar la aplicación es necesario que el celular de la participante tenga configurada la localización en sus fotografías, de otra forma siempre se obtendrá el mensaje “GPS not found”.

En el caso de que no sea posible modificar estas configuraciones, es factible editar los metadatos de ubicación a través de Google Fotos. Los pasos son los siguientes:

1. Subir la fotografía a Google Fotos.
2. Seleccionar la fotografía.
3. Abrir la información de la fotografía. Al fondo estará la opción ¿Dónde se tomó esta foto?
4. Al dar click permitirá añadirle una ubicación.

Subiendo una foto desde mi celular:

Ahora se debe subir la fotografía con los metadatos editados a la aplicación para posteriormente mostrar el mapa a partir de sus coordenadas. Para realizar esto, la participante agregará los siguientes componentes a su interfaz:

- Un componente de tipo *SelectorDeImagen*.
- Un componente *ActivityStarter*.
- Un botón para indicar que se quiere *subir una imagen*.

El funcionamiento que debemos agregar a partir de la sección *Bloques* se dividirá de la siguiente forma:

- Colocar la foto seleccionada en *Imagen*; presionar el *SelectorDeImagen* y elegir el bloque *DespuésDeSelección*, dentro de éste se seleccionará el *bloque de Imagen* <<poner Imagen.Foto como...>> y como parámetro se le pasará *SelectorDeImagen.Selección*.
- Llamar al selector de imágenes cuando se presione el botón: Seleccionar el bloque <<cuando Boton. Clic...>> de *Botón* y dentro colocar el bloque de *ActivityStarter* para poner acción, la cual será la cadena “*android.intent.action.PICK*”. Colocar dentro de Boton.Clic el bloque poner *TipoDeDato* de *ActivityStarter* como “*image/**” y por último, llamar a *ActivityStarter.iniciarActividad*.
- Desplegar el mapa y las coordenadas: Dentro del bloque <<cuando ActivityStarter.DespuésDeActividad>> colocar los bloques necesarios para mostrar la información en las etiquetas y mostrar el mapa.

Con lo anterior, se ha añadido otra nueva funcionalidad a la aplicación, por lo que se debe probar el proyecto nuevamente.

Es importante recordar que como el código para mostrar la información en las etiquetas y mostrar el mapa es igual, se recomienda utilizar un procedimiento.

Cuarta parte: Mi árbol genealógico en un mapa.

Sugerencia de tiempo invertido: 30 minutos.

La participante utilizará fotografías (analógicas o digitales) de tres mujeres pertenecientes a su familia (ejemplo: abuela materna, abuela paterna y madre) agregando los datos en cumplimiento con los metadatos, utilizando la ubicación del lugar de nacimiento de cada una de las mujeres.

Links para apoyar la actividad:

- Maestro TV SNTE. (2016). *Log In-¿Sabes qué son los Metadatos? T5E4-SNTE Nacional*. [video]
<https://www.youtube.com/watch?v=WMQoXaM2oqw> (Julio, 2020).
- OpenWebinars. (2018). *QUÉ SON LOS METADATOS*. [video]
<https://www.youtube.com/watch?v=7Gu0zGwDi9Y> (Julio, 2020).

Actividad 9: Ubicación con un video desde mi app

Aprendizaje esperado: Desarrollar la integración del video y el GPS del teléfono móvil en una aplicación para que al grabar un video se obtenga la ubicación en tiempo real.		Duración de la actividad: 6 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">● Dispositivo móvil con cámara y sensor GPS.● Medio para probar la aplicación.	<p>Evidencia:</p> <ol style="list-style-type: none">1. Guardar información relacionando los datos del video con la ubicación.2. Cuestionario de la Primera parte “¿Y los datos?” y su complemento de la tercera parte “Utilizando la API de Dropbox”, en el documento “<i>MMT1A9_respuestas.txt</i>”. <p>Producto:</p> <ol style="list-style-type: none">1. Una aplicación que muestre en la pantalla botones para grabar un video, una vez obtenido el vídeo activar otro botón para obtener las coordenadas actuales de localización.	<p>Retroalimentación:</p> <ul style="list-style-type: none">● Reforzar los conocimientos vistos en los cuestionarios realizados, priorizando el conocimiento práctico sobre los usos e identificación de elementos en situaciones cotidianas. <p>Evaluación:</p> <ul style="list-style-type: none">● Mostrar tres videos cortos generados en la aplicación y sus correspondientes coordenadas de ubicación.
Desarrollo de la actividad:		
<p>Primera parte: ¿Y los datos?</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <p>La participante identificará lo que sucede con la información contenida en aplicaciones, en situaciones como el pausado y cierre, para ello verificará el funcionamiento de alguna de las aplicaciones que haya desarrollado, seleccionará uno o varios de los producto de las actividades partiendo de la <i>MMT1A5</i> hasta la <i>MMT1A9</i>.</p> <p>Una vez seleccionada la aplicación, seguirá los siguientes pasos:</p> <ol style="list-style-type: none">1. Abrir la aplicación y navegar en ella, es decir, tomar alguna ubicación, fotografía o video.2. Cambiar a cualquier otra aplicación o volver al menú principal.		

3. Volver a su aplicación y contestar la siguiente pregunta: ¿Aún es posible visualizar la imagen, video o ubicación?
4. Eliminar la aplicación de sus pestañas (en el menú de multifunciones).
5. Volver a acceder a la aplicación y contestar nuevamente la pregunta del punto 3.

A partir de lo observado en los puntos anteriores la participante contestará las siguientes preguntas, que fungirán como evidencia de esta primera parte:

- ¿Los datos siempre se conservaron en tu aplicación?
- ¿Consideras que este comportamiento es óptimo para tus aplicaciones?
- ¿Tienes alguna idea de cómo sería posible conservar los datos de tu aplicación?
- ¿Qué es una base de datos?
- ¿Qué opciones de bases de datos existen en *MIT App Inventor*?
- ¿Por qué crees que existan tantas opciones?
- Investiga las características de cada una de ellas y realiza un pequeño cuadro comparativo.
- ¿Hay alguna otra forma de guardar información?

Guardar las respuestas en el documento “*MMT1A9_respuestas.txt*”.

Segunda parte: Tejiendo la red.

Sugerencia de tiempo invertido: 1 hora.

1. La participante identificará tres negocios dirigidos por mujeres de su comunidad: tiendas de abarrotes, estéticas, consultorías, despachos, clínicas, artesanas, etc.
2. Grabará un video corto de dicho negocio, el material servirá para el desarrollo de una app sencilla de tipo directorio, que tenga la capacidad de agrupar negocios locales y su dirección (localización GPS) mediante el video, para ello, deberá contar con el consentimiento de los negocios.
3. En caso de ser necesario, puede agregar algún tipo de información adicional, por ejemplo: reseñas cortas donde se detalle que los negocios son atendidos o emprendidos por mujeres, contacto, servicios que se ofrecen etc.

Tercera parte: Utilizando la API de Dropbox.

Sugerencia de tiempo invertido: 1 hora.

Para familiarizarse con los conceptos que se utilizarán en la actividad, se deberán agregar las siguientes preguntas al documento de la primera parte “*MMT1A9_respuestas.txt*”:

Nota: Utilizar el apartado “links de apoyo”, como referencia para contestar las preguntas.

- ¿Qué es el almacenamiento en la nube?
- Menciona al menos dos ventajas y dos desventajas del almacenamiento en la nube.
- ¿Qué es Dropbox?
- ¿Existen servicios similares? Menciónalos.
- Realiza una pequeña comparativa entre las distintas formas de almacenamiento en la nube.
- ¿Qué es una API?
- ¿Cuál es el concepto de token en las APIs?
- ¿Existe una API de Dropbox?
- Menciona al menos tres acciones que se puedan realizar con la API de Dropbox.
- ¿Cuentan con API las opciones que mencionaste?

Cuarta parte: Conectando con la API de Dropbox.

Sugerencia de tiempo invertido: 3 horas.

La participante conectará su aplicación con una cuenta de Dropbox, a fin de poder compartir información entre ambas partes. Los pasos a seguir serán los siguientes:

Obteniendo el Access Token

1. Visitará la página para crear una nueva aplicación en el API de Dropbox.
2. Accederá al sitio con una cuenta (si la participante ya cuenta con una puede hacer uso de ésta o puede crear una nueva).
3. Dará click en el botón *Create App*.
4. En la ventana que se abrirá, deberá seleccionar Dropbox API, acceso a un solo folder, nombrará la aplicación con el nombre “MiDirectorio” y marcará la casilla de aceptación de términos y condiciones.
5. Con las opciones seleccionadas, dará click en *Create App*.

6. La acción anterior redirigirá a una nueva página, en ésta deberá obtener el *Access Token* de la aplicación. Para obtenerlo, deberá buscar la opción *Generate Access Token* y dar click en el botón *Generate*.

Creando un nuevo proyecto

La participante creará un nuevo proyecto en la consola de *MIT App Inventor*, al que nombrará “MiDirectorio”.

Diseñador

La interfaz de la aplicación contendrá los siguientes elementos:

- Botón para grabar video.
- Reproductor de video.
- Botón para guardar el video.
- Botón para tomar otro video.
- Botón para reproducir.
- Botón para obtener la ubicación.
- Etiqueta para mostrar el estado.
- Etiqueta para mostrar la ubicación.
- Componente *Grabador*.
- Componente *Notificador*.
- Componente *Web*.

Es posible personalizar la interfaz a gusto de la participante, siempre y cuando se incluyan todos los elementos anteriormente descritos.

Bloques

1. Para conectar la aplicación con Dropbox: en la sección *Bloques* creará una variable global llamada *accessToken* de tipo *Obfuscated Text* (este tipo de dato indica que se codificará y no podrá ser accedido desde el apk) y pegará el *Access Token* generado.
2. Codificar la variable es importante, ya que si se comparte este código, cualquiera tendría acceso al folder que definimos en nuestra cuenta.

3. Creará además otras 2 variables globales llamadas *rutaArchivo* y *nombreArchivo* inicializadas con cadena vacía.
4. El primer paso consistirá en poder tomar un video y mostrarlo en el reproductor, para realizar lo anterior será posible tomar como referencia los pasos descritos en la actividad *MMT1A6*.
5. Logrado lo anterior, dentro del bloque “cuando Grabacion DespuesDeGrabacion ejecutar” guardar el valor de clip en la variable global *rutaArchivo*.
6. Obtener la ubicación a partir del video de la misma forma en que se realizó en *MMT1A8* y mostrarla en la etiqueta que se reservó para la ubicación.
7. Crear un procedimiento llamado “obtenerNombreArchivo”, que obtendrá el nombre del archivo a partir de una ruta (las rutas están construidas de la siguiente forma: content://media/external/video/media/nombreArchivo).
8. Los pasos para obtener el nombre del archivo son los siguientes:
Definir una variable local llamada “lista” en el procedimiento, esta variable guardará el resultado de dividir la cadena “rutaArchivo” cada que se encuentre una “/” a través del bloque “recorta texto ... en ...”. Donde en “texto” se colocará el contenido de rutaArchivo y en “en” el separador.
Definir el valor de la variable global “nombreArchivo” como el último elemento de la variable “lista” que recién se definió. Esto puede lograrse a partir del bloque “seleccionar elemento de la lista ... índice ...”.
9. Crear un nuevo procedimiento llamado “guardarVideoDropbox”, las acciones que se realizarán dentro de este procedimiento serán:
Llamar al procedimiento “obtenerNombreArchivo”
Colocar la cadena “Tratando de subir archivo..” en la etiqueta que se reservó para mostrar el estado.
Seleccionar el bloque “poner Web Url ...” y colocar la url: <https://content.dropboxapi.com/2/files/upload>
Dentro del bloque “poner Web.PedirCabeceras como ...”, construir una lista para tres elementos (cada uno para un header).
10. Cada elemento a su vez contará con una lista de dos elementos, a continuación se mencionan los contenidos de cada uno de ellos:
Primer header. El primer elemento contendrá la cadena “Authorization” y el segundo elemento hará uso del bloque “unir ...” para combinar las cadenas “Bearer ” (es muy importante dejar el espacio) y la variable global “accessToken”.
Segundo header. El primer elemento contendrá la cadena “Content-Type” y el segundo la cadena “application/octet-stream”.
Tercer header. El primer elemento contendrá la cadena “Dropbox-API-Arg” y el segundo será la unión de tres cadenas:

```
{"path": "/Aplicaciones/MiDirectorio/  
La unión de nombreArchivo y la cadena ".mp4"  
", "mode": "add"}
```

Por último se colocará el bloque “llamar Web publicar Archivo camino...” y se le pasará la variable global “rutaArchivo”

11. Agregar el bloque “cuando Web.obtuvoTexto ... ejecutar ...” para detectar el resultado de la petición y saber si existieron errores. Dentro de este bloque, agregar una validación para saber si el códigoDeRespuesta fue igual a 200 (lo que indica que todo ha salido bien).
12. Si el código fue igual a 200, colocar en la etiqueta de estado el resultado del bloque “llamar Web.DecodificarTextoJson...” pasándole como parámetro “contenidoDeRespuesta”. En caso de que el código no haya sido de éxito, utilizar el bloque “llamar Notificador.MostrarDialogoMensaje...” pasándole como mensaje el “contenidoDeRespuesta”, como título la unión de la cadena “Error” y “códigoDeRespuesta” y como “textoEnBotón” la cadena “Ok”.

Notas:

Para la segunda parte de la actividad:


- Se sugiere que los negocios procuren el autocuidado entre mujeres o buscar un negocio emprendido o atendido por mujeres, rescatar el valor que aporta al mercado por ello.

Para la cuarta parte de la actividad:

- Es posible ver ejemplos de la construcción de la url tanto en la documentación como en los dos links a blogs provistos al final del documento.

Links de apoyo:

- EDteam. (2019). *¿Qué son las APIs y para qué sirven?*
<https://www.youtube.com/watch?v=u2Ms34GE14U> (Junio, 2020).
ChicaGeek. (2019). *Los 5 mejores servicios de ficheros EN LA NUBE* ☁️ / ChicaGeek. [video]

- <https://www.youtube.com/watch?v=BiTQGcXhB6w> (Junio, 2020).
- Unocero. (2018). *Almacenamiento en la Nube - Lo bueno, lo malo y lo feo*. [video] <https://www.youtube.com/watch?v=3nFZj3bB9g8> (Junio, 2020).
- Crear aplicaciones Dropbox API.
Dropbox. (Sin fecha). Prueba de *Dropbox Business*.
<https://www.dropbox.com/developers/apps> (Junio, 2020).
- Documentación de Dropbox API para la acción upload.
Dropbox. (Sin fecha). *Dropbox for HTTP Developers*.
<https://www.dropbox.com/developers/documentation/http/documentation#files-upload> (Junio, 2020).
- Villalpando, J. (Sin fecha). *75 L.-subir y bajar archivos Dropbox. Upload and Download files Dropbox. Android. App inventor*.
http://kio4.com/appinventor/75L_dropbox.htm (Junio, 2020).
- Tutorial subir archivos Dropbox.
Pura Vida Apps. (Sin fecha). *How to upload a file into your  Dropbox*.
<http://puravidaapps.com/dropbox.php> (Junio, 2020).
- MIT APP INVENTOR. (Sin fecha). *Storage*.
<http://ai2.appinventor.mit.edu/reference/components/storage.html> (Junio, 2020).

Actividad 10: Enviando un mensaje SMS (Whatsapp, Red social) desde mi app

Aprendizaje esperado: Desarrollar una aplicación para enviar mensajes de texto y foto para compartir a través de mensaje o red social.		Duración de la actividad: 9 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Dispositivo móvil con capacidad para enviar y recibir mensajes SMS, y que tenga instalada alguna red social como <i>Whatsapp</i>, <i>Facebook</i> u otra. 	<p>Evidencia:</p> <ol style="list-style-type: none"> Infografía del tema aplicaciones de mensajería instantánea. Organizador gráfico del componente <i>Activity Starter</i> y los componentes sociales. <p>Producto:</p> <ol style="list-style-type: none"> Una aplicación que contenga un cuadro de texto donde redactar un mensaje y botones correspondientes para adjuntar una imagen, enviar el mensaje o la imagen vía un SMS y otro botón por si se desea enviar por <i>Whatsapp</i>, <i>Facebook</i> u otra. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> Revisar que el contenido de la infografía sea el solicitado. Revisar que el organizador gráfico estructure la información solicitada. <p>Evaluación:</p> <ul style="list-style-type: none"> Se debe comprobar el funcionamiento de la app por medio de la redacción, el envío de un mensaje y la respuesta de los contactos.
Desarrollo de la actividad:		
<p>Primera parte: Las apps de mensajería. Sugerencia de tiempo invertido: 2 horas.</p> <ol style="list-style-type: none"> Se le proporcionará a la participante varios sitios web para la búsqueda de información que ayude a responder las siguientes preguntas: <ul style="list-style-type: none"> ¿Qué es una aplicación de mensajería instantánea? ¿Cómo funciona (técnicamente)? ¿Cuáles son sus características? 		

- ¿Qué elementos debe contener una aplicación de mensajería instantánea?
- 2. Deberá realizar una infografía con base en la información obtenida, se recomienda utilizar el sitio web *Canva*.
- 3. Posteriormente, anotará en una hoja de papel tres aplicaciones de mensajería que utilice frecuentemente y cómo le sirven para fomentar su autocuidado (se recomienda a la participante revisar el documento “Yo en el centro: el autocuidado feminista” que se encuentra en el documento de detalles).
- 4. Esta elaboración se podrá compartir con la tallerista y/o compañeras del taller para socializar y compartir consejos de autocuidado utilizando aplicaciones de mensajería móvil.

Segunda parte: Aprendiendo nuevos componentes.

Sugerencia de tiempo invertido: 2 horas.

1. La participante investigará el componente *Activity Starter* y los componentes sociales disponibles en *MIT App Inventor*.
2. Deberá responder las preguntas siguientes:
 - ¿Para qué se utiliza el componente *Activity Starter* y los componentes sociales?
 - ¿Cómo es su funcionamiento?
 - ¿Cómo se puede vincular a otras aplicaciones móviles?
 - ¿Cuáles son las propiedades, eventos y métodos disponibles de cada uno de los componentes sociales?
3. Por último, deberá realizar un organizador gráfico en donde se estructure la información obtenida, se puede realizar en papel o por medio de las páginas web sugeridas ya que esta información se utilizará más adelante.

Tercera parte: Diseño de mi interfaz.

Sugerencia de tiempo invertido: 2 horas.

1. La participante iniciará un nuevo proyecto en *MIT App Inventor* y lo nombrará *TextGroup*.
2. Deberá ajustar el *título* de la pantalla a *TextGroup*; abrir el editor de bloques y conectarlo al teléfono o emulador.
3. Se deberá usar el diseñador de componentes para crear esta interfaz.
4. La siguiente tabla describe los componentes que se necesitan. Deberá arrastrar cada componente desde la *Paleta* hasta dentro de *Viewer* (el visor) y nombrarlos como se especifica a continuación:

Tipo de componente	Grupo de la paleta	Como se llamaría	Propósito del componente
Caja de texto	Básico	Mensaje de texto	El usuario introducirá el mensaje aquí.
Botón	Básico	Botón de Texto	El usuario hará clic aquí después de introducir el mensaje.
Botón	Básico	Enviar por <i>Whatsapp</i>	El mensaje abrirá la aplicación <i>Whatsapp</i> para enviar un mensaje.
Botón	Básico	Imagen	El usuario podrá seleccionar desde su dispositivo una imagen o tomar una foto para enviar.
Etiqueta	Básico	Estado del mensaje	Informa cuando el mensaje ha sido enviado.
<i>Texting</i>	Social	<i>texting1</i>	El componente que envía el mensaje.
<i>Activity Starter</i>	Social	<i>ConexionWA</i>	Componente que permitirá redirigir a la aplicación <i>WhatsApp</i> para poder enviar un mensaje.

5. Se recomienda Ajustar las propiedades de los componentes de la siguiente manera:

- Cambiar la propiedad *Hint* (sugerencia) de mensaje de texto a "introduce un mensaje".
- Cambiar la propiedad de *Botón de texto* a "Enviar".
- Cambiar la propiedad de *Estado del mensaje* a "enviado".
- Cambiar la propiedad de *Imagen* a "Adjuntar archivo".
- Cambiar la propiedad de *Enviar por WhatsApp* a "WhatsApp" y agregar como imagen el icono correspondiente a la aplicación.
- Cada botón debe de ser de un color diferente, preferentemente todos indentados y distribuidos en la pantalla del

dispositivo.

Cuarta parte: Construyendo bloques.

Sugerencia de tiempo invertido: 2 horas.

1. Una vez que la interfaz de la aplicación haya sido creada y probada lo siguiente será añadir un comportamiento a los botones y etiquetas creadas.
2. Se abrirá el editor de bloques y se arrastrarán los componentes necesarios para lograr el siguiente funcionamiento:
 - El comportamiento de *Mensaje de texto* deberá ser el siguiente; cuando el usuario haga clic en el botón *enviar*, la aplicación deberá enviar el mensaje introducido por el usuario en *mensaje de texto* a todos los números de teléfono de una lista vía SMS.
 - Cuando todos los mensajes hayan sido enviados, el estado del mensaje deberá de mostrarse como *enviado*.
 - Si el usuario hace clic en el botón *WhatsApp* la aplicación deberá de redirigirse automáticamente a la app *WhatsApp* del teléfono móvil y ahí deberá aparecer el mensaje que el usuario escribió en mensaje de texto.
 - Si es una imagen lo que se enviará, la aplicación mostrará una lista desplegable al dar clic sobre el botón *imagen*, mostrará las opciones de tomar una foto o adjuntar desde la galería.

Quinta parte: Conectando todo.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Una vez que la app envía mensajes de texto lo que resta es hacer la conexión con la aplicación de *WhatsApp* o cualquier otra con la que se desee enviar mensajes además de agregar la opción de enviar fotos.

1. En la *paleta* se dirigirá a *conectividad* y seleccionará *Activity Starter* (este componente ya había sido agregado previamente en la cuarta parte de esta actividad) deberá asegurarse que aparezca en pantalla.
2. Se usará como contenedor el *botón de texto*, deberá seleccionar un bloque de la parte de control en el cual cuando el usuario haga clic en el botón, active este controlador de evento; dentro de éste se deberá anexar otro bloque que tome el texto introducido por el usuario y dentro de éstos se usarán los bloques de *Activity Starter* (todos de color verde) seguidos de un bloque de texto (color rosa).

“ActivityStarter1. acción a ” agregar en un bloque de texto “android.intent.action.SEND”

- “ActivityStarter1. paquete ” agregar en un bloque de texto “com.whatsapp”
- “ActivityStarter1. clase a ” agregar en un bloque de texto “com.whatsapp.ContactPicker”
- “ActivityStarter1.TipoDeDato a ” agregar en un bloque de texto “text/plain”
- “ActivityStarter1. ClaveAdicional a ” agregar en un bloque de texto “android.intent.extra.TEXT”
- “ActivityStarter1. ValorAdicional a ” agregar en un bloque verde que conecte con el texto.
3. Se utilizará el bloque lógico “si... entonces, si no entonces” y se anexará en la última parte pero aún dentro del bloque contenedor principal, cuya condición será que si no se encuentra la extensión “com.whatsapp” se enviará un mensaje al usuario indicando que la app no está instalada.
 4. Se anexará otro componente de *Activity Starter* para enlazar la app de fotos que se creó en la *MMT1A5* con el *botón imagen* que se creó en la tercera parte de la actividad.
 5. Para finalizar, se revisará el funcionamiento de la app, para esto se enviarán mensajes diferentes a sus contactos de confianza vía SMS o *WhatsApp* y se tomará captura de pantalla de que los mensajes o imágenes fueron enviadas y se mostrarán a la tallerista para confirmar su funcionamiento.

Nota:

- Las aplicaciones que vienen con el teléfono pueden ser invocadas con los nombres de paquete y los nombres de clase.

Para la cuarta parte de la actividad

- Para enviar un mensaje, se necesitará ajustar dos propiedades del componente *Texting: PhoneNumber* (el número de teléfono) y *Message* (el mensaje). Una vez que estas propiedades estén ajustadas, se llamará a la función *SendMessage* para enviar el mensaje.

Links para apoyar la actividad:

Mensajería instantánea:

- Wikipedia La enciclopedia libre. (2017). *Mensajería instantánea*.
https://es.wikipedia.org/wiki/Mensajer%C3%ADa_instant%C3%A1nea (Noviembre, 2020).
- platea.pntic.mec.es. (Sin fecha). 4.2 *La mensajería instantánea*.
http://platea.pntic.mec.es/vgonzale/trabcolab_0910/archivos/_110/Tema_4.2.htm (Noviembre, 2020).

Información sobre Activity Starter

- MIT App Inventor BETA. Aprende App Inventor. (Sin fecha) *Activity Starter*.
<https://sites.google.com/site/aprendeappinventor/documentacion/activity-starter> (Julio, 2020).

Información sobre “Componentes sociales”

- MIT App Inventor BETA. Aprende App Inventor. (Sin fecha) *Componentes Sociales*.
<https://sites.google.com/site/aprendeappinventor/documentacion/componentes-sociales> (Julio, 2020).

Construcción de bloques

- App Inventor BETA. (Sin fecha) *Text Group*.
http://recopilation.com/3_DIRECTORIO_ES/informatica/programacion/appinventor/012aplicaciontextgroupsms/textgroup.html (Julio, 2020).
- Enviar mensaje vía Whatsap con App Inventor 2.
TuAppInventor Crea tu App sin programar. (2015). *Crea aplicaciones Android con MIT App Inventor en español.*
<https://www.tuappinventorandroid.com/2014/12/09/enviar-mensaje-v%C3%ADa-whatsapp-con-app-inventor-2/> (Julio, 2020).
- App Inventor Tutorials - SeblogApps. (2015) *Android Tutorial - How to send WhatsApp messages with MIT App Inventor 2*. [video]
<https://www.youtube.com/watch?v=DnGKgBeqGos> (Julio, 2020).

Para la infografía

- Canva.
<https://www.canva.com/> (Julio, 2020).

Para el organizador gráfico

- creately. (Sin fecha). *Organizadores gráficos para la escritura*.
<https://creately.com/es/usage/organizadores-gr%C3%A1ficos-de-escritura/> (Julio, 2020).
- Mind Meister.

Coronel, M. (Sin fecha). *Elaborando Organizadores visuales Online*.

<https://www.mindmeister.com/es/712924797/elaborando-organizadores-visuales-online> (Julio, 2020).

- Lucidchart. (Sin fecha). *Programa para hacer mapas conceptuales online*.

<https://www.lucidchart.com/pages/es/ejemplos/mapa-conceptual> (Julio, 2020).

Actividad 11: Compartiendo mi ubicación a partir de una foto o video con un mensaje de texto en tiempo real con mi aplicación

Aprendizaje esperado: Desarrollar la integración de foto o video y el GPS del teléfono móvil en una aplicación para que al tomar una foto o grabar un video se obtenga la ubicación en tiempo real y se comparta en mensajería o red social.		Duración de la actividad: 3 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Dispositivo móvil. Aplicaciones creadas durante el taller 1. Medio para probar la aplicación. 	<p>Evidencia:</p> <ol style="list-style-type: none"> Tabla de la primera parte “Reciclando mis app’s”. Documento donde se explique el comportamiento de la app de autor. Video/triller de su app de autor. <p>Producto:</p> <ol style="list-style-type: none"> App de autor. Una aplicación que al ingresar permita usar la cámara del dispositivo para tomar una foto o video corto, que obtenga las coordenadas del lugar actual y que permita enviar esta información con un mensaje de texto vía SMS o <i>WhatsApp</i> a uno o varios destinatarios. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> Verificar la viabilidad del reciclado de las app’s ya diseñadas, como buena práctica. Examinar que el documento sea conciso en su explicación y sea entendible para todo público. Asegurarse de que el video presentado sea conciso y atractivo para el público al que vaya dirigida la app. <p>Evaluación:</p> <ul style="list-style-type: none"> La aplicación debe presentar elementos reciclados y originales, además de contar con un objetivo contextualizado y ser funcional para el cumplimiento del mismo. Se debe verificar el correcto funcionamiento de la aplicación.
Desarrollo de la actividad:		
Previo a la actividad: Tu lista de contactos.		
Sugerencia de tiempo invertido: 30 minutos.		

1. La participante realizará una lista de dos contactos telefónicos, el criterio de selección será considerar a quién le pediría apoyo en caso de las siguientes situaciones:
 - Situación A. El transporte que utilizas se desvió del camino.
 - Situación B. Te sientes en riesgo dentro de tu casa (porque alguien enfermó, se metieron a tu casa o algún integrante tuvo conductas agresivas).
2. Las personas, podemos crear diferentes redes de apoyo dependiendo la situación en que nos encontremos, sin embargo, para crear la lista de contactos de seguridad en tu vida, se deben considerar diferentes aspectos. Para que la selección sea oportuna le pedirá a la participante que responda en una hoja de papel, las siguientes preguntas para cada situación:
 - ¿Cuándo estás con esa persona puedes expresar una opinión libremente?
 - Si tuvieras que decir “no” frente a alguna situación que no te agrada, ¿te sentirías cómoda al decírselo?
 - Cuando llegas a estar en desacuerdo con esta persona, ¿su manera de comunicar su desacuerdo es respetuosa?
 - ¿Esta persona cuenta con conocimiento básico sobre el uso del celular?
 - ¿Crees que la persona en la que estás pensando cuenta con información oportuna para poder apoyarte?
 - ¿En qué situación no podrías contarla? Propón a otra persona, ¿quién sería?

Primera parte: Reciclando mis aplicaciones.

Sugerencia de tiempo invertido: 30 minutos.

Reciclar (y reutilizar) material de autoría propia o de otras personas, siguiendo y respetando los derechos de autor, tiene como finalidad la reducción de esfuerzos, tiempo y costos. Es importante que la calidad se mantenga con respecto al material original o se mejore.

El reciclaje se refiere a la transformación de un producto (no necesariamente terminado) en otro diferente, otorgándole la capacidad de adaptarse a diferentes contextos, usuarios y propósitos.

1. El primer paso para generar un nuevo producto partiendo de otros, consiste en definir el nuevo objetivo. Para esta actividad se propone generar una nueva aplicación que permita compartir una ubicación obtenida a partir de una foto o video y enviar un mensaje de texto en tiempo real con esta información.

La participante podrá seleccionar este camino y si así lo desearse añadir nuevas funcionalidades, o podrá partir de una idea de su autoría.

Para ambos casos se deberá definir, el objetivo, el público al que va dirigido y la problemática/necesidad que atiende.

2. Identificará elementos reciclables de sus aplicaciones ya elaboradas para utilizarlas en el diseño de una app nueva. Se puede hacer uso de la siguiente tabla como apoyo para realizar la identificación de los elementos a reciclar.

Nombre de la App	Elemento Reciclable	Encuadre en la nueva App

Segunda parte: Uniendo el rompecabezas.

Sugerencia de tiempo invertido: 1 hora.

La participante iniciará la construcción de su aplicación partiendo de la línea o temática elegida basándose en la reutilización de componentes.

- Mochila

1. Partiendo de la tabla generada en la primera parte de la actividad “Reciclando mis app’s”, generará un nuevo proyecto en el que copiará y pegará los bloques necesarios.
2. Es posible comunicar bloques de componentes entre proyectos de *MIT App Inventor*, a través de la mochila que se encuentra en la sección *Bloques*.
3. Para verificar el comportamiento de esta utilidad la participante creará el nuevo proyecto para su aplicación. Después abrirá cualquier otro proyecto ya elaborado, del que desee copiar algún componente.
4. Una vez identificado el proyecto y lo que se desea copiar, entrará a la sección *Bloques* del proyecto elegido en donde deberá mantener presionado el bloque y arrastrarlo hacia la mochila para soltarlo una vez que se abra.
5. Deberá abrir el proyecto recién creado y dirigirse al lugar donde se encuentra la mochila, al dar click verá el componente añadido desde el otro proyecto, que podrá arrastrar para incluirlo a su proyecto actual.

6. Si la participante eligió componentes relacionados con la interfaz, procedimientos o variables y estos no han sido definidos, aparecerán marcados como error. Para resolverlo se deberán declarar los elementos pertinentes y la alerta desaparecerá.

- Múltiples pantallas.

1. Con el motivo de profundizar algunas secciones o distribuir mejor la información en la aplicación, la participante aprenderá a incluir nuevas pantallas y navegar entre ellas.
2. Para crear una nueva pantalla, en la ventana de *Diseñador* se deberá localizar el botón que se encuentra en la parte superior con la leyenda *Añadir Ventana*, una vez presionado el botón, se deberá asignar un nombre a la nueva pantalla para poder identificarla.
3. Cada una de las nuevas pantallas agregadas contendrá su propia sección de *Diseñador* y *Bloques*.
4. Para cambiar de pantalla en el área de desarrollo se deberá seleccionar la pantalla en la lista desplegable que se encuentra a lado de *Añadir Ventana*.
5. Dentro de la programación de la aplicación para cambiar de pantalla se utiliza el bloque “abrir otra pantalla Nombre de la pantalla ...” al que se deberá pasar como parámetro el nombre de la nueva ventana que se desea abrir. Es posible encontrar este bloque dentro el área de *Control* dentro de *Bloques*.
6. Como evidencia la participante deberá adjuntar un documento en donde explicará el comportamiento de su aplicación, además de mostrar los bloques y pantallas añadidos con una breve explicación del porqué los agregó. Este documento se centrará en la parte técnica de la aplicación.

Tercera parte: Presentando mi aplicación.

Sugerencia de tiempo invertido: 1 hora.

Cuando se desea publicar una nueva aplicación en la tienda de aplicaciones resulta conveniente incluir fotos o capturas de cómo luce la aplicación, así como un video en el que se explique el propósito o la necesidad que resuelve. Siguiendo esta línea, la participante elaborará un vídeo en el que expondrá porqué su aplicación es importante y qué problemática pretende resolver. El vídeo también deberá incluir una muestra del funcionamiento de la aplicación.

Actividad 12: Mejorando mi aplicación

Aprendizaje esperado: Modificar parámetros de los componentes de la aplicación elaborada en la <i>MMT1A11</i> para mejorar su diseño así como algunas de las funcionalidades de la aplicación. Hacer más inmediato su funcionamiento.		Duración de la actividad: 10 horas.
Recursos <ul style="list-style-type: none"> Dispositivo móvil o medio para probar la aplicación. Anexo <i>MMT1A12_Anexos.pdf</i> 	Evidencia/producto <p>Producto:</p> <ol style="list-style-type: none"> Mejoras a la aplicación realizada en la <i>MMT1A11</i>, la nueva aplicación podrá enviar un mensaje de texto predeterminado a una lista de seguridad con un solo botón además de sus coordenadas de ubicación. 	Retroalimentación/Evaluación <p>Evaluación:</p> <ul style="list-style-type: none"> La aplicación deberá contar con un nuevo icono y pantalla de inicio; así como también nuevos componentes (una pantalla redefinida con los botones colocados de manera más accesible y diseño personalizado) y mostrar su funcionamiento enviando a la lista de contactos un mensaje predeterminado y la ubicación del dispositivo con solo oprimir un botón.
Desarrollo de la actividad:		
<p>Primera parte: ¿El diseño importa o no?</p> <p>Sugerencia de tiempo invertido: 1 hora y 30 minutos.</p> <ol style="list-style-type: none"> Se le brindarán a la participante varios sitios web que abordan el tema del diseño en las apps móviles y el impacto que este tiene, para que responda las siguientes preguntas: <ul style="list-style-type: none"> ¿Consideras el diseño como una parte importante o trascendental para el desarrollo de aplicaciones móviles? ¿Qué elementos son los que se toman en cuenta? ¿La aplicación móvil que desarrollas es visualmente sencilla de usar? ¿Por qué? Con lo anterior, la participante deberá evaluar la app móvil en la que ha estado trabajando. Se recomienda utilizar la lista de cotejo disponible en el anexo <i>MMT1A12_Anexos.pdf</i> y con base en ella, anotar en una hoja las mejoras que pretende realizar y una justificación de porqué éstas ayudarían en su diseño. 		

Segunda parte: Una imagen para mi app.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante:

1. Deberá definir el nombre de la app móvil.
2. Tendrá que buscar y seleccionar una imagen para utilizarla como el icono de su app móvil. Se recomienda utilizar el link sugerido, para descargar imágenes con licencia *Creative Commons Zero* (CC0).
3. Para colocarla como fondo predeterminado deberá:
 - Abrir la aplicación móvil en *APP Inventor* y seleccionar la pantalla principal (Screen 1).
 - En la parte inferior izquierda, en *Media*, seleccionará subir un archivo y elegirá la imagen previamente descargada en la carpeta de “iconoAppInventor”.
 - Una vez que la imagen se haya subido, en la parte izquierda de la pantalla específicamente en *Icon* se deberá seleccionar la imagen y establecerla como fondo predeterminado.
 - Agregará el nombre que ha elegido para su app, mediante la casilla *Título* ubicada en las herramientas.
 - Para finalizar, comprobará que los cambios anteriores hayan sido aplicados, para ello será necesario descargar la app para que al abrirla muestre el icono seleccionado.
4. Diseñará la pantalla de *inicio* de la aplicación.
 - Desde la *paleta* deberá arrastrar la imagen seleccionada al fondo de *screen 1*, dicha imagen será el icono distintivo de su aplicación.
 - En *propiedades* deberá alinear la imagen a lo ancho y largo de la pantalla.
 - Posteriormente, definirá un color de fondo que cubra toda la pantalla.
 - En sensores, deberá agregar un reloj en el que se personaliza el tiempo que aparecerá en la pantalla de inicio, se recomienda de 3 a 5 segundos (*AppInventor* cuenta en mili segundos).
 - En *bloques* se recomienda usar lo siguiente:
 - Cuando la pantalla inicie, se debe de cambiar el color de fondo y mostrarse el icono de la aplicación.
 - Aparecerá un recuadro de texto con el nombre de la app.
 - Se habilitará el reloj para que comience a contar el tiempo desde que se abra la app.

- Probar en el emulador la calidad y resolución de la pantalla, ajustar las dimensiones de la imagen y color de fondo de ser necesario.

Tercera parte: Redefiniendo la pantalla.

Sugerencia de tiempo invertido: 1 hora.

Con base en lo que la participante escribió en la primera parte de la actividad, deberá aplicar los cambios anotados en sus botones de la app considerando lo siguiente:

- El cuadro texto deberá ser accesible y el recuadro más grande de la aplicación.
- Los botones no deberán interponerse unos con otros y de preferencia ser de colores distintos o con imágenes distintivas.

Cuarta parte: Red de seguridad (lista de contactos de seguridad).

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá crear una lista de contactos de su confianza, los cuales le puedan ayudar o mantenerse al tanto en caso de encontrarse en una situación de peligro, es importante que se considere a personas con un dispositivo móvil conectado a internet o que puedan recibir SMS.
2. Una vez se haya creado la lista se solicitará a la participante verificar en su agenda que se tengan agregados a los contactos seleccionados, de lo contrario deberá agregar de forma manual los números a *ApplInventor*.

Agregar contactos:

1. Deberá modificar los bloques correspondientes al recuadro de mensajes o *texting* considerando lo siguiente:
 - Definirá una *global variable* (variable global) para almacenar los datos para la aplicación (los números de teléfono).
 - Definir la variable *NumerosTelefonicos*, ésta se deberá renombrar.
 - Posteriormente se añadirá un bloque de *crear una lista* a la variable de *NumerosTelefonicos*.
 - A la nueva lista se añadirán todos los bloques de texto necesarios de acuerdo al tamaño de la lista de contactos.
 - Se utilizará un bloque “para cada elemento en la lista... ejecutar” que se encuentra dentro de los bloques de control, para especificar lo que se quiere enviar en el mensaje a cada número de la lista. Este se debe anidar a los bloques de la

variable principal, ya que esta va a actuar sobre estos.

- Finalmente se probará el funcionamiento, para esto se enviará un mensaje de saludo a la lista de contactos predeterminada y se asegurará de que todos los contactos lo reciban.

Quinta parte: Mensajes predeterminados.

Sugerencia de tiempo invertido: 2 horas.

Crear y almacenar:

1. La participante creará una lista de mensajes para situaciones de emergencia donde con solo presionar un botón, se enviarán de inmediato a su red de confianza junto con las coordenadas de su dispositivo móvil, en caso de no contar con conexión a internet estas se enviarán por vía SMS.
2. El criterio del contenido de los mensajes es de carácter personal de la participante, pero se recomiendan frases breves que enfaticen la situación de peligro en la que se encuentre.
3. Se trabajará de nuevo con el componente *Texting*, *TinyBD* y el *sensor de localización (GPS)*.
4. Lo primero que se realizará, será crear un botón llamado “ALERTA” el cual al ser presionado por el usuario enviará un mensaje de texto predeterminado a su lista de contactos de emergencia de forma automática.
5. Desde la *paleta* se arrastrará el componente *etiqueta* y se renombrará como “Respuesta”.
6. Dentro de las propiedades de la etiqueta “Respuesta” se escribirá el mensaje predeterminado que se creó.
7. De *almacenamiento* se arrastrará a la pantalla el componente *TinyDB*, el cual se ocupará de guardar los mensajes en la base de datos.

Respuesta automática:

1. Para programar la respuesta automática deberá utilizarse el componente *Texting*.
2. Para programar el texto de respuesta se colocará el bloque *Texting1.EnviarMensaje* dentro del bloque del botón de “ALERTA” que se creó.
3. El componente *Texting1.EnviarMensaje* se encargará de enviar el texto, así que se deberá indicar lo que se debe enviar y quién será el destinatario, para lograr ésto se deberán anexar los bloques de la lista de contactos que se crearon en la cuarta parte de la actividad, pues éstos serán los contactos que recibirán las alertas.
4. Se deberá definir *Respuesta.Texting*, para esto se asignará el texto que se haya escrito en *Respuesta*. Cuando se haya definido, la aplicación llamará a *Texting.EnviarMensaje* para que proceda con el envío de la respuesta. Se deberá repetir en un bloque

de texto diferente para cada mensaje personalizado.

Almacenamiento de las respuestas:

1. Si el usuario cierra el programa, cuando vuelva a abrirlo la contestación se habrá perdido y se empleará la opción predeterminada, para evitar que esto suceda, se deberá de guardar las respuestas en un elemento con memoria a largo plazo *la base de datos del teléfono*.
2. Se utilizará el componente *TinyDB* para recuperar el texto contenido en *Respuesta* y almacenarlo en la base de datos.
3. Cuando se almacena algo en la base de datos se tiene que asignar una etiqueta para identificarlo. Esta etiqueta se llamará *TinyDB1.GetValue* mediante la etiqueta de *MensajeRespuesta*.
4. El valor que se recupere de ahí se colocará en la variable *Responder*. Así, la aplicación podrá verificarla antes de colocarla en *Respuesta*. Esto se deberá concatenar en una variable global que se inicie cuando se abra la aplicación.

Ligar el sensor de ubicación:

1. Se deberá ligar el sensor de ubicación, para esto se indicará que al presionar el botón de alerta, éste deberá enviar la ubicación actual registrada a los contactos de la lista usando los bloques.
2. Finalmente se realizarán pruebas de la aplicación, se escribirán los mensajes predeterminados y se asegurará que los contactos reciban la ubicación del usuario.

Sexta parte: Mi aplicación favorita.

Sugerencia de tiempo invertido: 1 hora.

1. La participante vinculará su celular con la red social que más use o en la que se sienta más cómoda al comunicarse con sus red de confianza, ya que en esta se compartirán los mensajes de forma predeterminada.
2. Se usará *Activity Starter* para que la participante pueda vincular las aplicaciones de las redes sociales que seleccione y de esta manera se puedan compartir los mensajes predeterminados o su ubicación en caso de encontrarse en peligro.
3. Se recomienda que sea principalmente con *WhatsApp* o alguna otra app que permita enviar mensajes de texto instantáneos además de compartir la ubicación del usuario si éste así lo desea.
4. Se deberá crear un nuevo botón o vincular esta actividad al botón de alerta previamente creado si así lo desea.
5. Una vez realizado lo anterior, la participante deberá asegurarse que el botón funciona, se solicitará compartir un mensaje en sus redes sociales platicando sobre su aplicación y el impacto que esta podría tener para las mujeres, si lo desea puede etiquetar o mencionar a las redes de PILARES.

Séptima parte: Uniendo fuerzas.

Sugerencia de tiempo invertido: 1 hora.

1. Con el objetivo de visualizar el funcionamiento completo de la aplicación, la participante deberá ponerse en contacto con su red de confianza para crear una simulación donde pueda poner a prueba los distintos componentes que ha desarrollado en esta actividad.
2. Deberá comprobar que todos cada uno de los componentes funcionan y presentan el comportamiento esperado en su programación, se recomienda utilizar una lista de cotejo para identificar de manera ordenada cada componente y su función.
3. La participante podrá invitar a las mujeres que conozca (estas pueden ser sus amigas, familiares o colegas) a utilizar la aplicación que desarrolló con el fin de que puedan auxiliarse si se llegaran a encontrar en alguna situación de peligro.

Notas:

Para la cuarta parte de la actividad.

- Las variables globales no son vistas por el usuario; son memoria "oculta" de la aplicación. En este caso, se definirá la lista de números de teléfono a los cuales se envía el texto.
- Para los números de teléfono se usarán bloques de texto, no bloques de número. Esto permitirá usar guiones, los cuales no se pueden usar en los bloques de número. Los bloques número se utilizarán para datos o para realizar cálculos matemáticos.

Para la sexta parte de la actividad.

- Es importante que se tomen precauciones a la hora de compartir información en las redes sociales, ya que estas pueden dejar de ser seguras si la información se vuelve pública, se recomienda inspeccionar la configuración de privacidad de las aplicaciones que utilice.

Links de apoyo:

Importancia del diseño de aplicaciones móviles

- MovilApps®. (Sin fecha) *La importancia del diseño en tu APP*.
<https://movilapps.eu/la-importancia-del-diseno-en-tu-app/> (Julio, 2020).
- Tarragó, A. (2018) *buen diseño en las apps*, dribba.
<https://www.dribba.com/post/la-importancia-del-buen-dise%C3%B1o-en-las-apps-m%C3%B3viles> (Marzo, 2021)
- Cuella, J. y J. Vittone (2013) *Diseñando apps para móviles*.
<https://books.google.es/books?hl=es&lr=&id=ATiqsjH1rvwC&oi=fnd&pg=PA7&dq=diseño+de+apps+móviles&ots=a3fpYY3t5m&sig=uZ8D8Do9DKjDSAi1Kvs4MI5TzAg#v=onepage&q=diseño%20de%20apps%20móviles&f=false> (Julio, 2020).

Imágenes con licencia Creative Commons Zero (CC0).

- Pexels. (Sin fecha). *Imágenes con licencia Creative Commons*.
<https://www.pexels.com/es-es/creative-commons-images/> (Julio, 2020).

Íconos:

- icon-icons.com. (Sin fecha). *Iconos App inventor*.
<https://icon-icons.com/es/buscar/iconos/app+inventor> (Julio, 2020).

Ayuda en bloques para la creación de respuesta automática:

- Destruels, V. (Sin fecha) *Sms y Localización GPS*.
<http://www.aulainformatica.eu/datos/programacion/appinventor/capitulo4/appInventor-cap4-final.pdf> (Julio, 2020).

Actividad 13: Seguridad

Aprendizaje esperado: Revisar los conceptos básicos sobre seguridad en las aplicaciones móviles.		Duración de la actividad: 8 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Teléfono móvil. 	Evidencia 1. Identificar todos los elementos de seguridad estudiados para la aplicación desarrollada en la actividad A12. Reflexionar sobre los elementos vulnerables y cómo sobrepasar estos problemas de vulnerabilidad.	Retroalimentación <ul style="list-style-type: none"> • Se debe verificar que se identifiquen correctamente los elementos de seguridad y mostrarlos en una tabla de información.
Desarrollo de la actividad:		
<p>Primera parte: Mi información en la red. Sugerencia de tiempo invertido: 1 hora.</p> <p>La participante:</p> <ol style="list-style-type: none"> 1. Realizará una inspección de la relación que mantiene con diversos medios digitales, como las redes sociales (<i>Facebook, Twitter, Instagram</i>, etc.), o herramientas como cuentas google, con el fin de identificar las funciones que le permitan compartir su información como parte de su perfil o las condiciones y políticas a las que accede para el uso de su información personal, algunos ejemplos guía pueden ser la revisión de los términos y condiciones que son aceptados para utilizar una app de red social, la información que es agregada al realizar una publicación (fotografía, video o “historia”) en la red social <i>Facebook</i>, como ubicación, personas cercanas o sitios que se frecuentan. 2. Responderá las siguientes preguntas detonadoras y comentará con la tallerista (u otras participantes) las conclusiones, reflexiones e impresiones a las que llegó: <ol style="list-style-type: none"> a. ¿Qué ventajas otorga el compartir información personal con una app? b. ¿Qué desventajas otorga compartir información personal con una app? c. ¿Qué factores de seguridad son identificables en el uso diario de apps? (acceder a cuentas, navegar a través de buscadores, que una app registre la ruta utilizada para llegar a un sitio determinado, etc). 		

- d. ¿Cómo plantear interacciones con las apps que permitan el uso de información personal de manera beneficiosa?
- 3. Revisará los permisos de las redes sociales instaladas en su equipo móvil e inspeccionará si todo lo que piden es necesario.
 - a. ¿Cuántas aplicaciones tienen acceso a tu ubicación?
 - b. ¿Qué relación existe entre la información que se sube a una APP y la publicidad que suele aparecer en la misma?
 - c. ¿La información que cada persona decide subir, es un procedimiento de consentimiento consciente?

Segunda parte: ¿Qué es la seguridad?

Sugerencia de tiempo invertido: 1 hora.

La participante revisará los recursos provistos en el apartado de *Links de apoyo* o en su defecto realizará una búsqueda en internet para discutir las preguntas que se plantean a continuación:

- ¿Consideras que la seguridad en las aplicaciones móviles es importante? ¿Por qué?
- ¿Estabas consciente de la cantidad de información que compartimos a diario a través del uso de la tecnología?
- ¿Cómo te impacta la seguridad como consumidor de aplicaciones? ¿Qué medidas debes implementar?
- ¿Se puede identificar si se ha violado la seguridad de una aplicación?

Tercera parte: Importancia y orígenes del cifrado.

Sugerencia de tiempo invertido: 4 horas.

La participante identificará el concepto de criptografía y revisará algunos ejemplos de códigos existentes. Utilizando el recurso *Un viaje por la criptografía* que se puede encontrar en enlaces. En el que deberá revisar todo lo que contiene la unidad llamada *Criptografía antigua*.

Siguiendo la línea de investigación sobre criptografía, se sugiere ver la película *Código enigma* o investigar acerca de Alan Turing y el código enigma que desarrolló, para después contestar las siguientes preguntas:

- Si viste la película (alternativamente puedes investigar en internet):

- ¿Cuál era el nombre de la única mujer del equipo de Turing?
 - Investiga un poco más acerca de ella (mínimo media cuartilla).
 - ¿Qué tan marcada era la diferencia entre hombres y mujeres? Da algunos ejemplos de la película que respalden tus argumentos.
 - ¿Cuál fue la debilidad de enigma?
 - ¿Qué tan diferentes crees que hubieran sido las cosas de no haberse descifrado el código?
 - Con tus propias palabras explica qué es y cuál es la importancia de la criptografía.
 - ¿Se sigue usando hoy en día?
 - ¿Habías escuchado de la existencia del código enigma?
- Alguna vez has tratado de comunicarte con alguien sin que nadie más se entere de la conversación. ¿Qué mecanismo has utilizado para proteger el secreto?
 - Escribe un mensaje cifrado para una mujer importante en tu vida, puedes auxiliarte de la página *cryptii* que se anexa en los *Links de apoyo*. Revisa las diferentes opciones de códigos que ofrece la página, escoge al menos dos y escribe el mensaje en las dos elecciones.
 - PILARES tiene el siguiente mensaje para ti, pero es sumamente secreto. Se encuentra codificado en código César cuyo desplazamiento se encuentra entre 1 y 50. (Recuerda que puedes hacer uso de *cryptii*).
“U ftuzw, ftmf eayqfuyqe uf ue ftq bqabxq ita za azq uymsuzqe mzkftuzs ar, ita pa ftq ftuzse za azq... omz uymsuzq.”
 - Genera tu propio código secreto y compártelo con alguien, reta a alguna otra persona para que trate de descifrarlo. También es posible brindar pequeñas pistas a la persona que quiera descifrar el código.
 - En los *Links para aprender más*, se encuentra un desafío sobre un misterio más avanzado y también se anexa un ejemplo de cómo trasladar los procesos de encriptación a *MIT App Inventor*. (No es obligatorio realizar estas actividades).

Cuarta parte: Mecanismos (Autenticación, Autorización, Cifrado).

Sugerencia de tiempo invertido: 30 minutos.

La participante:

1. Visitará los links provistos para esta sección, para identificar el concepto de autorización, autenticación y cifrado.

2. Elaborará una tabla en la que ubicará el mecanismo de protección y dará tres ejemplos de aplicaciones o sitios web que lo implementen.

Mecanismo	Ejemplo 1	Ejemplo 2	Ejemplo 3

Quinta parte: Añadiendo seguridad a mi app.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante reflexionará sobre las mejoras que puede implementar a su aplicación para garantizar la seguridad de la información.

Se recomienda responder las siguientes preguntas guía:

- ¿Cómo te impacta la seguridad al ser desarrollador de aplicaciones? ¿Qué medidas debes considerar?
- ¿Todos los usuarios tendrán acceso a los mismos datos en tu aplicación?
- ¿Todos los usuarios serán capaces de realizar las mismas acciones?
- ¿De qué manera puedes proteger sus datos?

Basándose en las respuestas a las preguntas anteriores y de lo aprendido a lo largo de toda la actividad, la participante elaborará un documento explicando las mejoras de seguridad que implementará a su aplicación (únicamente involucra la reflexión, no es necesario que las implemente). Dicho documento será similar al punteo de una actualización, destacando las mejoras en las que se basará y los posibles problemas que puede resolver.

Links de apoyo:

Primera parte:

- Permisos de aplicaciones.
ChicaGeek. (2020). *¿Qué APPS tienen ACCESO a tu CÁMARA, UBICACIÓN, FOTOS... EN TU MÓVIL? | ChicaGeek.*[video]
<https://www.youtube.com/watch?v=oJCr0GNVR0Y> (Julio, 2020).

Segunda parte:

- Importancia de tus datos y su protección.
TEDx Talks. (2015). *¿Por qué me vigilan, si no soy nadie? | Marta Peirano | TEDxMadrid.* [video]
<https://www.youtube.com/watch?v=NPE7i8wuupk> (Julio, 2020).
Ginatost. (2018). *La verdad sobre la privacidad en Internet y la GDPR | Gina Tost.* [video]
https://www.youtube.com/watch?v=Lcrgby_8E8 (Julio, 2020).
- Detectar malware.
BBVA. (2020). *Cómo evitar 'malware', 'apps' falsas y otros ciberataques en tu móvil.* [video]
<https://www.youtube.com/watch?v=VtJic0c7Cso> (Julio, 2020).
Empey, Charlotte. (2020). *Cómo saber si una aplicación de Android es segura para instalar.*
<https://blog.avast.com/es/check-android-app-safety> (Julio, 2020).
- Seguridad informática.
EDteam. (2020). *¡La seguridad informática es para todos!* [video]
<https://www.youtube.com/watch?v=UBh1XPQuVIM> (Julio, 2020).

Tercera parte:

- Crear tu propio código. Códigos ya existentes.
- Learnincolor. (2016). *7 Secret Spy Codes for Kids with Printable List | Cryptography for kids.*
<https://learnincolor.com/secret-spy-codes-for-kids.html> (Julio, 2020).
- Cifrar y descifrar mensajes.
cryptii.com. (Sin fecha). *Cryptii.*
<https://cryptii.com/> (Julio, 2020).
- Plantilla para los códigos.
ClayMaze. (Sin fecha). *Spy Decoder Wheel-Custom Code Version*
<https://www.pinterest.com.mx/pin/129548926766414896/> (Julio, 2020).

- Un viaje por la criptografía.
Khan Academy. (Sin fecha). *Ciencias de la computación. Unidad: Criptografía.*
<https://es.khanacademy.org/computing/computer-science/cryptography> (Julio, 2020).
- Historia sobre el abecedario y código César.
CuriosaMente. (2020). *¿Quién inventó el ABECEDARIO?-CuriosaMente 211.* [video]
<https://www.youtube.com/watch?v=wkORjpAriY> (Julio, 2020).

Para aprender más:

- khan Academy. (Sin fecha). *Computación < Ciencias de la computación < Criptografía < Desafío introductorio de criptografía. Introducción.*
<https://es.khanacademy.org/computing/computer-science/cryptography/cryptochallenge/a/cryptochallenge-introduction>
(Julio, 2020).
- *Codificar y decodificar*
Villalpando, J.. (Sin fecha). *60.-Codificar un texto.*
<http://kio4.com/appinventor/29codificatexto.htm> (Julio, 2020).

Cuarta parte:

- That C# guy. (2017). *Autenticación y autorización.* [video]
https://www.youtube.com/watch?v=LhxWy_I3EKM (Julio, 2020).
- ComputerHoy.com. (2020). *¿Qué es Cifrado de Extremo a Extremo?* [video]
<https://www.youtube.com/watch?v=d8uUWWDOazk> (Julio, 2020).
- Whatsapp. (Sin fecha). *Información sobre el cifrado de extremo a extremo.*
<https://faq.whatsapp.com/general/security-and-privacy/end-to-end-encryption?lang=es> (Julio, 2020).
- Laboratorio feminista de derechos digitales. *Compilatorio: ¿Qué sucede con derechos en los ámbitos digitales?*
<https://www.facebook.com/LAMFEMDD/> (Julio, 2020).
- Vega, A. (2019). *Ciberviolencia contra las mujeres y el discurso de odio sexista.*
http://portal.iedf.org.mx/biblioteca/descargasC.php?id=412&fbclid=IwAR1l__qBPr9ArDpKgXxrKibDnvyvp_ndmmXv2DJ5yhP47ae-TWiko2rOGG8 (Julio, 2020).
- Luchadoras. (Sin fecha). *Toolkit de cuidados digitales.*

<https://luchadoras.mx/toolkit-de-cuidados-digitales/> (Julio, 2020).

Actividad 14: Probando el funcionamiento de mi aplicación con mi red de amigas e Instalando mi aplicación en un teléfono móvil

Aprendizaje esperado: Probar el funcionamiento de la aplicación a través de mensajería (SMS, <i>WhatsApp</i> , etc). Identificar los pasos a seguir para la instalación de mi aplicación móvil en mi teléfono con los recursos de <i>MIT App Inventor</i> .		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> Otros teléfonos con sistema Android y con la aplicación MIT AI2 Companion instalada. 	Evidencia <ol style="list-style-type: none"> Instalación de la aplicación desarrollada en la actividad A12 en uno o varios teléfonos móviles y prueba de su funcionalidad. Reflexionar sobre la habilidad de elaboración de aplicaciones web aprendida hasta ahora y cómo hacer de ésta un actividad remunerada. 	Retroalimentación <ul style="list-style-type: none"> Se debe verificar la funcionalidad de todos los componentes de la aplicación instalada en otro teléfono. Se debe verificar la publicación de la aplicación creada por la participante.
Desarrollo de la actividad:		
Primera parte: ¿Qué debe cumplir mi app para salir al mundo? Sugerencia de tiempo invertido: 1 hora.		
<ol style="list-style-type: none"> La participante deberá hacer una búsqueda acerca de los distintos sitios de publicación de aplicaciones móviles y deberá revisar específicamente los requisitos que se necesitan para publicar una aplicación. Posteriormente, responderá las siguientes preguntas: <ul style="list-style-type: none"> ¿Qué ventajas y desventajas encuentras en estos sitios? ¿Tu aplicación cumple con los criterios que establecen dichos sitios? ¿Tu aplicación será gratis o tendrá algún costo? ¿Dónde sería conveniente publicar tu aplicación para su distribución (gratuita o con costo)? ¿Cómo podría hacer que mi aplicación sea una fuente de trabajo? 		

Segunda parte: La prueba de mi app.

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá instalar su aplicación en distintos dispositivos móviles, para probar su funcionamiento.
2. Para que esto sea posible necesitará contar con la aplicación *MIT I2 Companion*.
3. Cuando se utiliza *MIT App Inventor* con un teléfono o tableta, el dispositivo se comunica con el software de *MIT App Inventor* que se ejecuta en el navegador del ordenador. Esta comunicación se administra a través de la *AI2 Companion* funcionando en el dispositivo.
4. La comunicación entre el ordenador y el teléfono o la tableta requiere instalar el programa *aiStarter*.
5. Se deberá iniciar el programa *aiStarter* con el comando / usr / google / appinventor / command-for-Appinventor / aiStarter & . La forma precisa de hacerlo depende de la distribución de GNU / Linux que esté utilizando.
6. Descargará e instalará la *App MIT AI2 Companion* en el teléfono, ya sea directamente de la tienda *PlayStore* o manualmente por *Fichero APK*.
7. Únicamente se necesitará instalar una vez y a partir de ahí ejecutarla cada vez que se use *MIT App Inventor*.
8. Se necesitará un escáner de códigos QR, se recomienda obtener uno en la Play Store (ZXing).
9. Abrirá el proyecto en *MIT App Inventor* que se desea instalar en los distintos teléfonos (el producto de la *MMT1A12*).
10. En *MIT App Inventor* se deberá elegir *Connect* y *AI Companion* en el menú principal, aparecerá un diálogo con un código QR.
11. En el dispositivo, se ejecutará la app *MIT App Companion* (tal como se haría con cualquier otra app). A continuación hará click sobre el botón *Scan QR code* y se escaneará el código.
12. En unos segundos, se deberá poder ver la app que se está construyendo en el dispositivo. Si existen problemas al escanear el código QR, se puede introducir a mano en la caja de texto de la app *Companion* y pulsar *Connect with code*.
13. Una vez que la participante haya instalado la aplicación en todos los dispositivos que considere necesarios deberá hacer algunas pruebas del funcionamiento de aquella. Se recomienda llevar un control por si se presentan fallos para que estos puedan ser corregidos, así como recibir *feedback* de quienes usaron la app, es decir, si consideran que es sencilla de manejar, atractiva, eficaz y opiniones extras que refieran a su experiencia.

Tercera parte: Conociendo App Gallery.

Sugerencia de tiempo invertido: 2 horas.

MIT App Inventor Gallery es el sitio principal para compartir las aplicaciones creadas en *MIT App Inventor* y los bloques específicos de programación con toda una comunidad. Es como *Google Play*, exceptuando el código abierto: todas las aplicaciones cargadas tienen código fuente (bloques) que se pueden estudiar y mezclar.

1. La participante podrá usar su cuenta de gmail para iniciar sesión y publicar su aplicación.
2. Deberá hacer clic en *Mis aplicaciones* y luego en *Agregar nueva aplicación*, se deberán llenar los siguientes datos:
 - Título. El nombre que tiene la app.
 - Descripción. Enlistar de manera breve lo que realiza la aplicación o con qué enfoque fue creada.
 - Categoría . Elegir uno de la lista.
 - Archivo fuente. Este es un archivo ZIP que contiene todos los bloques y medios en la aplicación. Se podrá descargar de *MIT App Inventor*, deberá abrir el administrador de proyectos y seleccionar la opción *Más acciones / Descargar fuente*.
 - Imagen o ícono. Este se mostrará en la *Galería* como la imagen de la aplicación. Por lo general, será una captura de pantalla de la aplicación o el ícono de la aplicación.
 - Etiqueta. Las etiquetas pueden ser cualquier cosa para ayudar a las personas a encontrar la aplicación. Si lo desea la participante, puede dar a todas sus aplicaciones la misma etiqueta para que pueda encontrarlas fácilmente.
3. Después de ingresar todos los campos, deberá hacer clic en *Guardar aplicación* para enviar la aplicación.
4. Para finalizar, hará clic en la página de inicio y se deberá ver la aplicación en la lista de *Aplicaciones más nuevas*.
5. La participante podrá hacer clic en su aplicación para poder visualizar la página de la aplicación pública.
6. Más tarde, se puede editar la descripción de la aplicación y cargar una nueva versión (archivo zip) volviendo a la página *Mis aplicaciones*.

Cuarta parte: Celebración de mi proyecto.

Sugerencia de tiempo invertido: 2 horas.

1. Terminar un proyecto representa un gran logro para la participante y la tallerista, ya que se invirtió tiempo, esfuerzo y compromiso para aprender los conceptos y habilidades necesarios para desarrollar un proyecto de este tipo, por lo tanto se sugiere realizar una actividad de celebración donde la participante pueda invitar a personas de su elección (amigas, compañeras y/o familiares).
2. Se sugiere aprovechar la celebración para presentar la aplicación que se desarrolló y otorgar el reconocimiento que la

participante merece como creadora de una aplicación funcional.

3. Si es posible la tallerista puede realizar una presentación en power point que exponga a mujeres desarrolladoras y en ella agregué el nombre y/o foto de la participante, para visibilizar su identidad en el gremio.

Taller 2: Mis primeros pasos en Kotlin

Este taller se desarrolla en honor a **Hipatia (355 d. C. - 415 d. C.)** Primera matemática reconocida en la historia. No existe mucha información sobre ella, sin embargo, se sabe que nace en Alejandría durante el proceso de los inicios de la intelectual, durante este periodo, el imperio romano veía en las matemáticas y las ciencias: herejía y no progreso.

Sus obras fueron escritas por sus discípulos, entre estos apuntes están: la “Aritmética” de Diofanto, “Geometría de las cónicas” de Apolonio, “Elementos de geometría” de Euclides y el “Tratado matemático” de Ptolomeo. También se le conoce por el diseño en el astrolabio plano y el aparato para destilación de agua.

Debido a sus contribuciones y posicionamiento político que era el paganismo y el racionalismo científico fue asesinada por los monjes fanáticos de la Iglesia De San Cirilo de Jerusalén.

Competencia esperada:

La participante aprenderá el lenguaje de programación Kotlin, este lenguaje, con ayuda del IDE (Entorno de Desarrollo Integrado) Android Studio, serán las herramientas fundamentales para el desarrollo de aplicaciones móviles a la medida.

En este taller se aplicarán los conocimientos para crear pequeños proyectos en donde se vea claramente la funcionalidad de todos los elementos que conforman la estructura de un programa, como son el uso las variables y constantes, tipos de datos, operadores, sentencias condicionales y de control de flujo, se aplicarán los conceptos de clases y objetos y la importancia de reutilizar código con el uso de funciones.

En el taller uno de este módulo se vio cómo crear e instalar aplicaciones móviles con la herramienta App Inventor de una forma modular, con los que se elaboran aplicaciones usando componente básicos que permiten el acceso a diferentes funcionalidades del teléfono móvil, el lenguaje de programación es una herramienta potente para crear aplicaciones personalizadas.

Actitudes: Curiosidad, disposición, constancia, persistencia, apertura a la incorporación de nuevos aprendizajes, capacidad de análisis, apertura al diálogo, escucha, trabajo en equipo, intercambio de opiniones, participación activa, interés y apertura a incorporar en las actividades la Perspectiva de Género para el logro de un bien común.

Actividad 1: Mis primeros pasos con Kotlin

Aprendizaje esperado: Identificar las partes fundamentales del lenguaje de programación <i>Kotlin</i> , así como describir las reglas y principios generales para la creación de un programa.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> ● Computadora PILARES. ● Conexión a internet. ● Editor de textos. ● Anexo <i>MMT2A1_Anexos.pdf</i> 	Evidencias: <ol style="list-style-type: none"> 1. Presentación que contenga las características más sobresalientes del lenguaje de programación <i>Kotlin</i>. 2. Archivo de texto con el desarrollo de los tópicos: Diferencia entre software libre y comercial, diferencia entre <i>MIT App Inventor</i> y <i>Kotlin</i>, programación funcional y características, y programación orientada a objetos y características. 3. Glosario de los conceptos utilizados en <i>Kotlin</i>. 4. Captura de pantalla de los códigos identificados y resaltados en colores. 	Retroalimentación: <ul style="list-style-type: none"> ● Revisar que la presentación contenga los elementos correspondientes desarrollados con información verídica. ● En la presentación deberá tocar los siguientes puntos: <ul style="list-style-type: none"> ○ Ventajas de <i>Kotlin</i> sobre <i>MIT App Inventor</i> para la creación de aplicaciones personalizadas. ○ Mención de las características de programación funcional y orientada a objetos. ● Revisar que el glosario de conceptos de <i>Kotlin</i> contenga los siguientes elementos: <ul style="list-style-type: none"> ○ Definición de paquetes e importaciones. ○ Punto de entrada del programa. ○ Funciones. ○ Variables y constantes. ○ Comentarios. ○ Plantillas de cadena. ○ Expresiones condicionales. ○ Valores anulables y comprobaciones nulas.

		<ul style="list-style-type: none"> ○ Verificaciones de tipo y selecciones automáticos. ○ Estructuras de control. ○ Rangos. ○ Colecciones. ○ Clases e instancias. <ul style="list-style-type: none"> ● Revisar la captura de pantalla con los códigos identificados en el color correspondiente a las partes reconocidas con base en su concepto.
Desarrollo de la actividad:		
<p>Primera parte: Conociendo nuevos ambientes.</p> <p>Sugerencia de tiempo invertido: 2 horas.</p> <ol style="list-style-type: none"> 1. Deberá abordar de manera general el lenguaje de programación <i>Kotlin</i> y sus características principales, para esto la participante realizará una búsqueda en internet y responderá las siguientes preguntas: <ul style="list-style-type: none"> ● ¿Qué es <i>Kotlin</i>? ● ¿Para qué se puede utilizar? ● ¿Cómo funciona? ● ¿Cuáles son las características principales del lenguaje? ● ¿Qué relación tiene Android Studio con Kotlin? 2. La información obtenida deberá ser integrada en una presentación (se recomienda utilizar la plataforma <i>Genially</i>). 3. Terminada la presentación, la tallerista deberá visualizarla para retroalimentar a la participante sobre sus hallazgos. Asimismo, es crucial corroborar que se hayan colocado las fuentes utilizadas para la investigación en la diapositiva final de la presentación. 4. Posteriormente, se deberá compartir en alguna red social. 		

Segunda parte: Solidificando conocimientos.

Sugerencia de tiempo invertido: 1 hora.

1. Es importante que la participante tenga una serie de conocimientos previos, que serán de gran utilidad a lo largo de este taller, si la participante ya los domina esta parte de la actividad puede ser sólo un repaso que lleve a cabo junto con la tallerista.
2. Se utilizará la plantilla A (disponible en el anexo *MMT2A1_Anexos.pdf*) la cual contiene una breve trivia de 10 preguntas. La tallerista jugará con la participante con el objetivo de identificar el nivel de conocimientos que posee.
3. Con base en las preguntas que se hayan contestado de manera errónea o incompleta, le sugerirá ver algunos videos que aborden esos temas para complementar el conocimiento. En caso de que la participante responda de manera correcta toda la trivia, podrán pasar a la tercera parte de la actividad.
4. Para finalizar, solicitará realizar un archivo de texto donde se desarrollen de forma breve los siguientes tópicos.
 - Diferencia entre software libre y comercial.
 - Diferencia entre *MIT App Inventor* y *Kotlin*.
 - Programación funcional y características.
 - Programación orientada a objetos y características.

Tercera parte: Mi glosario de *Kotlin*.

Sugerencia de tiempo invertido: 1 hora.

1. Una vez que la participante se haya relacionado con *Kotlin* y con algunos conceptos básicos respecto a la programación del módulo, deberá revisar el funcionamiento de *Kotlin* en algunas cuestiones específicas.
2. Se le proporcionarán a la participante los links correspondientes para ahondar en los siguientes conceptos:
 - Paquetes e importaciones.
 - Punto de entrada del programa.
 - Funciones.
 - Variables y constantes.
 - Comentarios.
 - Plantillas de cadena.
 - Expresiones condicionales.
 - Valores anulables y comprobaciones nulas.
 - Verificaciones de tipo y selecciones automáticos.

- Estructuras de control.
 - Rangos.
 - Colecciones.
 - Clases e instancias.
3. Posteriormente, utilizará la plantilla B (disponible en el anexo *MMT2A1_Anexos.pdf*) donde deberá llenar una tabla con las definiciones de cada concepto, para crear un pequeño glosario de términos utilizados en *Kotlin*.

Cuarta parte: Las partes del código.

Sugerencia de tiempo invertido: 1 hora.

1. La participante revisará los códigos de *Kotlin* para identificar los conceptos que trabajó en la parte anterior de la actividad.
2. Tomará captura de pantalla a los diferentes códigos que elija y con diferentes colores remarcará las partes que reconozca, esas capturas deberá mostrarlas a la tallerista para obtener la retroalimentación correspondiente.

Links de apoyo:

Paquetes e importaciones:

- Chike Mgbemena (2017). *Kotlin Desde Cero: Paquetes y Funciones Básicas*.
<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-packages-basic-functions--cms-29445> (Agosto, 2020).
- Digital Guide IONOS by 1&1. (2019). *Kotlin: tutorial sobre el nuevo lenguaje de programación*.
<https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/kotlin-tutorial/> (Agosto, 2020).

Funciones:

- Universidad Politécnica de Valencia (Sin fecha) *Funciones en Kotlin*.
<http://www.androidcurso.com/index.php/913> (Agosto, 2020).

Variables y constantes:

- Chike Mgbemena (2017) *Kotlin Desde Cero: Variables, Tipos Básicos y Arreglos*.
<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-variables-basic-types-arrays-type-inference-and-comments--cms-29328> (Agosto, 2020).

Comentarios:

- Aprende el lenguaje de programación Kotlin.

developers (Sin fecha). *Guía de estilo de Kotlin*.

<https://developer.android.com/kotlin/style-guide?hl=es> (Agosto, 2020).

Plantillas de cadena:

- RIP Tutorial (Sin fecha). *Kotlin. String Templates*.

<https://riptutorial.com/es/kotlin/example/26589/plantillas-de-cadena> (Agosto, 2020).

Expresiones condicionales:

- Cruz, A. (2018) *Los condicionales en Kotlin: if y when*.

<https://www.desarrollolibre.net/blog/android/los-condicionales-en-kotlin-if-y-when#.Xy91-i3mFN0> (Agosto, 2020).

Valores anulables y comprobaciones nulas:

- Kotlin Doc. Kotlin & Android. (Sin fecha). *Gestión de tipos nulos en Kotlin*.

<https://kotlindoc.blogspot.com/2019/04/gestion-de-tipos-nulos-en-kotlin.html> (Agosto, 2020).

- SO Documentation. (Sin fecha). *Kotlin Null Safety. Nullable and Non-Nullable types*.

<https://sodocumentation.net/es/kotlin/topic/2080/seguridad-nula> (Agosto, 2020).

Verificaciones de tipo y selecciones automáticos:

- Olivares, A. (2017). *Verificación y Conversión de Tipos en Kotlin*.

<https://medium.com/aldominium-com/verificación-y-conversión-de-tipos-en-kotlin-f8b7f4f6cc9a> (Agosto, 2020).

Estructuras de control:

- Universidad Politécnica de Valencia. (Sin fecha). *Estructuras de control en Kotlin*.

<http://www.androidcurso.com/index.php/912> (Agosto, 2020).

Rangos:

- Chike Mgbemena. (2017). *Kotlin Desde Cero: Rangos y Colecciones*.

<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-ranges-and-collections--cms-29397> (Agosto, 2020).

Colecciones:

- Universidad Politécnica de Valencia. (Sin fecha). *Colecciones en Kotlin: introducción*.

<http://www.androidcurso.com/index.php/99-kotlin/924-colecciones-en-kotlin-introduccion> (Agosto, 2020).

Clases e instancias:

- Chike Mgbemena. (2017). *Kotlin Desde Cero: Clases y Objetos*.

<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-classes-and-objects--cms-29590> (Agosto, 2020).

- Cruz, A. (2018). *Clases en Kotlin: clases vacías, constructores y propiedades*.

<https://www.desarrollolibre.net/blog/android/clases-en-kotlin-clases-vacias-constructores-y-propiedades#.Xy93Zi3mFN0> (Agosto, 2020).

Actividad 2: Un vistazo a Android Studio

Aprendizaje esperado: Ejecutar el IDE Android Studio y revisar de forma general el ambiente de trabajo que este software ofrece para la creación de programas con <i>Kotlin</i> .		Duración de la actividad: 8 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Editor de imágenes. • Editor de textos. • Anexo <i>MMT2A2_Anexos.pdf</i> 	Evidencia: <ul style="list-style-type: none"> • Capturas de pantalla. 	Retroalimentación: Capturas de pantallas consistentes de: <ul style="list-style-type: none"> • Captura de pantalla de ejecución del IDE Android Studio. • Captura de pantalla de ventana de inicio y comienzo de un nuevo proyecto. • Captura de pantalla de configuración básica del proyecto. • Captura de pantalla de ambiente de trabajo inicial. • Captura de pantalla de abrir, guardar y renombrar un proyecto. • Captura de pantalla de la aplicación funcionando.
Desarrollo de la actividad: Primera parte: Creando un nuevo proyecto. Sugerencia de tiempo invertido: 3 horas. <ol style="list-style-type: none"> 1. La participante realizará el codelab de Google para la creación de un nuevo proyecto. Deberá centrarse en todos los pasos, excepto el paso 6. 2. Revisará además los recursos recomendados, o bien la tallerista podrá elegir otros que considere convenientes. 3. Después de haber consultado los materiales, la participante deberá ser capaz de conocer los pasos involucrados en la creación de un nuevo proyecto y deberá tener un entendimiento básico de los distintos campos que se deben llenar. 4. Como evidencia se pedirá a la participante, crear un nuevo proyecto con las características descritas a continuación: <ul style="list-style-type: none"> • Uso de la plantilla <i>Empty Activity</i>. 		

- El nombre de la aplicación será “Mi Aplicación”.
 - El proyecto deberá estar contenido en una carpeta con el nombre de la participante.
 - El lenguaje usado será *Kotlin*.
 - Estará disponible para el 95% de los dispositivos.
5. Deberá documentar el proceso de creación, incluyendo capturas y breves explicaciones de cada paso.
 6. Por último, realizará una breve investigación centrada en el nivel de api tanto de su dispositivo como el de otros usuarios. Para ello consultará su versión de Android y el de otras cinco personas.
 7. Con la información recabada contestará lo siguiente:
 - ¿Tu aplicación funcionó en todos los dispositivos que investigaste?
 - ¿Cuál sería una ventaja o desventaja de elegir una API muy nueva?
 - ¿Cuál sería una ventaja o desventaja de elegir una API muy vieja?
 - ¿Por qué crees que se lanzan nuevas versiones?
 - Investiga cada cuando sale una nueva versión y la particularidad de los nombres.

Segunda parte: Conociendo la estructura de directorios.

Sugerencia de tiempo invertido: 2 horas y 30 minutos.

1. La participante ordenará la *Ley Federal para Prevenir y Eliminar la Discriminación* o La *Ley General para la Igualdad entre Mujeres y Hombres* en el tema del Lenguaje Incluyente y No Sexista que se encuentra en el anexo *MMT2A2_Anexos.pdf*. Puede realizarse de manera digital haciendo uso de documentos y carpetas o mediante un cuadro sinóptico.
2. Revisará los materiales referentes a la estructura de un proyecto en Android, que le ayudarán a identificar la función de cada carpeta.
3. Sintetizará la información en un cuadro sinóptico o esquema, en el que deberá ser identificable la jerarquía de carpetas y archivos, además de incluir una descripción de lo que almacenan y la ruta de ese recurso en su proyecto. (Para evitar confusiones, utilizar la vista Android para mostrar el proyecto dentro del IDE).

Las carpetas que se deberán incluir son las siguientes:

- gradle
- java
- manifests
- res (subcarpetas: drawable, layout, mipmap y values)

- app

Los archivos que se deberán incluir son los siguientes:

- AndroidManifest.xml
- strings.xml
- styles.xml
- colors.xml
- dimens.xml

4. Expondrá la relación que existe entre el proceso realizado y la estructura de carpetas que sigue un proyecto en Android Studio utilizando sus materiales elaborados.

Tercera parte: Visualizando mi aplicación.

Sugerencia de tiempo invertido: 30 minutos.

La participante ejecutará la aplicación en su dispositivo (es importante que sea en un dispositivo físico y no un AVD). Para llevar a cabo esta acción, seguirá los siguientes pasos:

1. Activar el modo desarrollador en el dispositivo.
2. Conectar el dispositivo a la computadora.
3. Permitir la opción Depuración por USB.
4. Desde el IDE, seleccionar el icono para correr la aplicación. Otra opción es hacerlo desde la pestaña Run > Run 'app'.

Deberá adjuntar como evidencia una captura de la aplicación en funcionamiento.

Cuarta parte: Personalizando mi aplicación.

Sugerencia de tiempo invertido: 2 horas.

Para profundizar y poner en práctica el conocimiento referente a la estructura de carpetas, se solicitará a la participante realizar en su proyecto lo siguiente:

- Modificar el texto "¡Hola Mundo!" por "¡Hola (nombre de la participante)!".
- Cambiar los colores de la aplicación apoyándose de la herramienta *Color Tool* de *Material Design* o utilizando el selector provisto por Android Studio.
- Identificar la carpeta que contiene el archivo con el icono de la aplicación.

- Modificar el color de fondo de la pantalla `activity_main.xml` (buscar o agregar la propiedad *background* dentro de *Constraint Layout*)
- Cambiar el icono de la aplicación mediante los siguientes pasos:
 1. Buscar un icono en alguna página que provea imágenes libres. (En los enlaces del documento se agrega el enlace a icons8, las imágenes de este sitio pueden usarse gratis, siempre y cuando se mencione de dónde provienen).
 2. Descargar la imagen y moverla a la carpeta correspondiente en su proyecto de Android Studio. Si esta acción se realiza desde el IDE, es posible modificar el nombre (es importante que el nombre sólo contenga números, letras minúsculas y guiones bajos).
 3. Localizar dentro del archivo `AndroidManifest.xml` la propiedad `android:icon` y cambiar la referencia por la de la imagen añadida.

Notas para la actividad:

Para la segunda parte de la actividad

- La estructura de directorios hace referencia al ordenamiento de carpetas bajo un sistema definido, dicho orden no es el mismo en todos los sistemas operativos.

Para la cuarta parte de la actividad

- Para mostrar el selector sólo es necesario presionar los cuadrados con color que aparecen a la izquierda del texto, a lado del número de línea.

Links de apoyo:

Codelab: Fundamentos de Android y Kotlin.

- developer.android.com. (Sin fecha). *Android Kotlin Fundamentals: Get started. 7. Task: Run your app on a physical device.* <https://codelabs.developers.google.com/codelabs/kotlin-android-training-get-started/index.html?index=..%2F..index#6> (Febrero, 2021).
Introducción a Android Studio.
- Yo Androide (Sin fecha). *CURSO COMPLETO ANDROID STUDIO 2021 1: CREAR PRIMER PROYECTO DE APLICACIÓN.* [video]<https://www.youtube.com/watch?v=lg0umrTO-Hw> (Febrero, 2021).
Estructura de un proyecto.
- Reina, C. (Sin fecha). *Arquitectura de una aplicación para Android.* <https://platzi.com/blog/arquitectura-android-app/> (Febrero, 2021).

- codigofacilito (2015). 3.- *Curso Introducción a Android - Estructura de un proyecto*. [video].
<https://www.youtube.com/watch?v=9a6p8Mzbf7A> (Febrero, 2021).

icons8

- Icons8 (sin fecha). *Iconos, ilustraciones, fotos, música y herramientas de diseño*.
<https://iconos8.es/> (Febrero, 2021).

Actividad 3: Elementos que permanecen y que van cambiando en mi programa

Aprendizaje esperado: Crear y configurar un nuevo proyecto que permita diferenciar el uso de variables y constantes, además de ejecutar el primer programa.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Anexo MMT2A3_Anexos.pdf 	Producto: <ol style="list-style-type: none"> 1. Creación de un nuevo proyecto relacionado a una lista de contactos usando solo cadenas, haciendo uso de variables y constantes usando la nomenclatura camel-case. 2. Ejecutar el programa en el emulador de Android Studio. 3. Archivos <i>Declaraciones.kt</i> y <i>Conversiones.kt</i>. 	Evaluación: <ul style="list-style-type: none"> • Mostrar el primer programa en Kotlin. • Verificar su funcionamiento resaltando las diferencias entre constantes y variables. • Uso de la nomenclatura camel-case. • Verificar que los archivos <i>Declaraciones.kt</i> y <i>Conversiones.kt</i> contengan todas las modificaciones y se ejecuten de forma exitosa.
Desarrollo de la actividad:		
<p>Primera parte: Conocimientos previos.</p> <p>Sugerencia de tiempo invertido: 2 horas.</p> <ol style="list-style-type: none"> 1. La participante deberá dominar los conceptos de constante y variables, si existen dudas se recomienda hacer una búsqueda en internet para aclararlas, puesto que son dos conceptos clave en el desarrollo de esta actividad. 2. Dominados los conceptos anteriores deberá realizar un cuadro comparativo donde explique las diferencias entre estos términos y en qué casos se usa cada uno. Se recomienda revisar los <i>Links de apoyo</i> donde se aborda de forma concisa el uso y manejo de variables o constantes en <i>Kotlin</i>. <p>Segunda parte: Nomenclatura camel-case.</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <ol style="list-style-type: none"> 1. La participante deberá responder las siguientes preguntas correspondientes al tema de la nomenclatura camel-case. 		

- ¿Qué es la nomenclatura camel-case?
 - ¿Para qué se usa?
 - ¿Qué son las palabras reservadas en Kotlin java?
 - ¿Cuáles son los identificadores en Kotlin?
 - ¿Kotlin es *key sensitive*?
2. Posteriormente, resolverá la plantilla A (disponible en el anexo *MMT2A3_Anexos.pdf*) que contiene diversos ejemplos de usos correctos de lenguaje incluyente y no sexista en los distintos tipos de nomenclatura camel-case.
 3. Es importante que la participante decida qué tipo de nomenclatura camel-case usará en sus programas ya que es considerado una buena práctica y es un aspecto que la tallerista evaluará en los siguientes programas que se presenten.

Tercera parte: Mi primer programa.

Sugerencia de tiempo invertido: 2 horas.

1. Deberá crear un nuevo proyecto. Para realizarlo abrirá Android Studio, elegirá un *Empty Activity* y hará clic en el botón *Next* para continuar.
2. En la siguiente ventana colocará el nombre a la aplicación, cuyo nombre será “AgendaApp”, posteriormente en la lista de Lenguajes (*Language*) seleccionará *Kotlin* y dará clic en el botón *Finish* para que Android Studio empiece a crear el proyecto. Deberá esperar sin mover nada hasta que termine la creación del proyecto.
3. Una vez que Android Studio termine de crear el proyecto se mostrarán 2 archivos abiertos *activity_main.xml* y *MainActivity.kt* este último archivo tendrá la extensión *.kt* que representará al formato de archivo *Kotlin*, deberá dirigirse al archivo *activity_main.xml*.
 - En el archivo *activity_main.xml* podrá visualizar que hay un *TextView* con el texto en inglés *Hello World!*, deberá dirigirse a la parte derecha en los atributos y cambiar el texto al nombre de la app.
 - Posteriormente, en el código fuente la participante deberá realizar una lista de contactos usando sólo cadenas, variables y constantes de la nomenclatura camel-case.
 - Al imprimir en pantalla se deberá ver la lista de contactos y el nombre de las personas con el número registrado para cada una de ellas.
 - Para probar el proyecto en el emulador de Android Studio se presionarán las teclas SHIFT + F10, podrá seleccionarse en qué tipo de equipo desean realizarse las pruebas.

Links de apoyo:

Variables y Constantes con *Kotlin*

- DDPADMIN. (2020). *Variables y Constantes con Kotlin en Android (Tema 2)*.
http://iqanansoft.es/ddp_web/variables-constantes-con-kotlin-en-android/ (Agosto, 2020).
- developers. (Sin fecha). *Aprende el lenguaje de programación Kotlin*.
<https://developer.android.com/kotlin/learn?hl=es> (Agosto, 2020).
- Digital Guide IONOS by 1&1.(2019). *Kotlin: tutorial sobre el nuevo lenguaje de programación*.
<https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/kotlin-tutorial/> (Agosto, 2020).

Actividad 4: Esos elementos que permanecen y cambian son de varios tipos

Aprendizaje esperado: Diferenciar los tipos de datos que pueden ser usados en el lenguaje de programación Kotlin para su correcta asignación de valores.		Duración de la actividad: 6 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.	Producto: <ol style="list-style-type: none">1. Creación de un programa relacionado con un horario de actividades diarias, haciendo uso de los diferentes tipos de datos numéricos y tipos de datos no numéricos.2. Crear constantes y variables de forma implícita y explícita que puedan ser convertidas a otro tipo de datos.3. Archivo <i>conversiones.kt</i>	Evaluación: <ul style="list-style-type: none">• Mostrar funcionando el programa en el emulador de Android Studio verificando el uso correcto de los diferentes tipos de datos, su declaración y conversión de tipo.
Desarrollo de la actividad:		
<p>Primera Parte: Vida cotidiana y programación.</p> <p>Sugerencia de tiempo invertido: 30 minutos.</p> <p>Introducción: Al igual que los lenguajes como el español o el inglés (conocidos también como naturales), los lenguajes de programación poseen una estructura y un significado, llamados de forma técnica sintaxis o gramática y semántica, respectivamente. Para ambos casos, existen reglas que deben seguirse para la conformación de frases o sentencias, una diferencia en el caso de los lenguajes naturales, es que las reglas gramaticales o las reglas sociales pueden no seguirse para atender a problemáticas referentes al Lenguaje Incluyente y No Sexista, mientras que en los lenguajes de programación se deben seguir las reglas en todo momento para evitar errores en la construcción de un programa.</p> <p>Es importante hacer llegar esta pequeña introducción a las participantes y promover la relación entre los tipos de lenguajes mencionados.</p>		

1. La participante identificará nuevas directrices para la promoción del Lenguaje Incluyente y No Sexista dentro del lenguaje natural y su implementación en la vida cotidiana, para ello se sugiere que se complete la tabla presentada a continuación, utilizando su experiencia personal.

Directriz	Ejemplo	¿Qué sucede?	Experiencia personal (¿has presenciado algo similar?)
Usar formas impersonales	No incluyente: Invitamos a todos a participar en los cursos impartidos en los PILARES. Incluyente: Les invitamos a participar en los cursos impartidos en los PILARES.		
Usar genéricos más incluyentes	No incluyente: Los ciudadanos, los indígenas o los trabajadores. Incluyente: La ciudadanía, la población indígena o el personal.		
Desdoblar los términos. Si no se encuentra un genérico inclusivo, se recomienda desdoblar el término en su	No incluyente: Todos, padres de familia , los niños. Incluyente: Todas y todos, las madres y padres, las niñas y niños.		

femenino y masculino.			
Usar el femenino de las profesiones y oficios.	<p>No incluyente: El gobernador, abogados, médico, ingeniero, diseñador, etc.</p> <p>Incluyente: La gobernadora, abogada, médica, ingeniera, diseñadora, etc.</p>		

2. La participante seleccionará una de las siguientes preguntas detonadoras para su discusión con la tallerista u otras participantes y realizará una relatoría corta de lo discutido (no más de 300 palabras), puede ser a mano o en documento digital.
- ¿Consideras que el Lenguaje Incluyente y No Sexista puede impactar de alguna manera en la vida diaria?
 - Usar el Lenguaje Incluyente y No Sexista es algo nuevo ¿por qué consideras que ha sido así?
 - ¿Crees importante utilizar el Lenguaje Incluyente y No Sexista en tu vida personal? ¿Por qué?

Segunda parte: Tipos de datos.

Sugerencia de tiempo invertido: 1 hora.

1. La participante reconocerá los tipos de datos básicos que utiliza el lenguaje de programación *Kotlin* y los relacionará con experiencias de su vida diaria. Para esto, se apoyará de la siguiente tabla:

Clasificación	Tipo de dato	Descripción
	Entero	Cuando hablamos de números enteros nos referimos a números completos , como: 1, 2, 3, 4, 5 ...

	Numérico		<p>También pueden ser negativos como: -1, -2, -3 ...</p> <p>Un ejemplo es la cantidad de dedos que tenemos, es decir 10.</p> <p>Otro ejemplo es la cantidad de ojos u oídos, que sería igual a 2.</p>	
		Punto flotante	<p>Son los números que poseen parte fraccionaria o decimal. Es muy común encontrar estas cantidades en la vida cotidiana. Por ejemplo, cuando pedimos kilo y medio de tortillas.</p> <p>Si lo queremos expresar con números, el número al que nos estamos refiriendo es 1.5.</p> <p>Otro ejemplo es la temperatura corporal, que para considerarse normal debe estar entre 36.1 y 37.2 grados centígrados.</p> <p>También hay números de punto flotante negativos, el requisito es poseer punto decimal.</p>	
	Texto	Carácter	<p>Hace referencia a una sola letra, dígito o símbolo.</p> <p>Algunos ejemplos serían: 'A', 'a', '1', '-', '!', etc.</p>	
			<p>Se refiere a un conjunto de caracteres, lo que nos permite formar palabras, cantidades, etc.</p> <p>Aunque la definición más pura es la de conjunto de</p>	

		Cadena	caracteres. Ejemplos de cadena son: “Hola”, “123”, “hsvjhsb”, etc.
	Lógicos	Booleano	Este tipo de dato sólo puede tener 2 valores: verdadero o falso . Por ejemplo, tenemos la siguiente pregunta: ¿Terminaste la tarea? Las respuestas posibles son sí y no. Y dependiendo de la respuesta sabemos que la interrogante puede ser igual a verdadero o falso.

- Una vez que la participante haya terminado de revisar las descripciones de todos los tipos, elaborará una trivia de 10 preguntas sobre datos curiosos, en el que cada respuesta involucrará alguno de los tipos de datos presentados. (2 preguntas para cada tipo).

Ejemplo: Tipo booleano. “¿El color verde se forma con la combinación del color azul y el amarillo?”. La respuesta involucra verdadero o falso.

Tercera parte: Hay 10 tipos de personas...

Sugerencia de tiempo invertido: 2 horas.

- La participante distinguirá conceptos relacionados a la representación de la información en una computadora. Se recomienda consultar los recursos propuestos en los *Links de apoyo*.
- Para poner en práctica los conocimientos adquiridos en este rubro, la participante realizará lo siguiente:

- Completará la siguiente tabla, que hace referencia al tamaño en memoria de los enteros en Kotlin. Se podrá apoyar de las siguientes recomendaciones para el llenado (en caso de ser necesario, la tallerista podrá apoyarla en el llenado de la tabla):
 - Observa las casillas conocidas de la tabla.
 - Realiza las operaciones y comprueba que los resultados asentados son correctos.
 - Toma en cuenta que con 8 bits es posible representar 256 números, esto se obtiene de la expresión 2^8 . Si quisiéramos representar sólo números positivos estos serían del 0 al 255.
 - ¿Qué pasa en este caso? ¿Se representan 256 números?
 - Comprueba que se representan 256 números, sumando los 128 números negativos y los 127 positivos. ¿Cuál es el total?
 - ¿Por qué el total es 255 y no 256?
 - ¿En cuál de los 2 conjuntos estará el 0?
 - Realiza de nuevo la operación, esta vez contemplando el 0. ¿Se representan 256 números?
 - ¿Por qué crees que el número máximo a representar es -128 y 127, en lugar de 255?
 - Para representar el signo de un número necesitamos un bit, de esta manera:

0 -> representaría negativo

1 -> representaría positivo

En un inicio teníamos 8 bits, si restamos el del signo, nos daría un total de 7 bits. Realizando la operación $2^7 = 128$, observamos que podemos representar 128 números.

Tipo	Tamaño en bits	Valor mínimo	Valor máximo
Byte	8	-128 (-2^7)	127 ($2^7 - 1$)
Short	16	-32768	32767

		(-2^{15})	$(2^{15} - 1)$
Int	32	-2,147,483,648 (-2^{31})	
Long	64		9,223,372,036,854,775,807 $(2^{63} - 1)$

Las respuestas de la tabla se pueden encontrar en la documentación oficial, disponible en los *Links de apoyo*.

- Basándose en los datos de la tabla anterior, responderá lo siguiente:
 - ¿Por qué crees que existan tantos tamaños para representar enteros?
 - ¿Consideras que es importante conocer el tamaño que ocupa un dato en memoria? ¿Por qué?
 - Supongamos que en tu programa requieres guardar tu edad, ¿Qué tipo de dato elegirías?
 - Lo recomendable sería no ocupar más recursos de los que necesitas. En este caso al elegir un Short, Int o Long. ¿Es viable que la edad de una persona pueda ser mayor a 127?
 - Comenta tus respuestas con la tallerista.
- La siguiente tabla muestra el tamaño para los números de punto flotante (aquellos que tienen decimales).

Tipo	Tamaño en bits	Dígitos decimales
Float	32	6-7
Double	64	15-16

Realiza lo siguiente para saber más sobre el funcionamiento de estos tipos de datos:

- Visita la página Kotlin Playground.
- Deberías encontrar un contenido parecido al siguiente:
 - ◆ fun main(){
 -
 - }
- Coloca todo lo que se te pedirá a continuación dentro de las llaves ({ - }).
- Genera una constante llamada float.
- Dale el siguiente valor 1.123456f y utiliza println(<< pon aquí el nombre de tu constante>>) para ver su valor. La f después del número indica que el número será de tipo float. Se puede usar “f” o “F” para indicarlo
- Cambia ahora el valor de float por 1.12345678f e imprímelo.
- ¿Tu número se imprimió completo?
- ¿Por qué crees que fue así?
- Es importante revisar los límites permitidos para no perder la precisión de los valores. Retira la f del número e imprímelo. ¿Qué sucede?
- Deberá notar que el tipo de dato ha cambiado a *Double* y ahora hay más espacio para representar el número.
- Una vez más, ¿Consideras que es importante conocer el tamaño que ocupa un dato en memoria? ¿Por qué? (orienta tu respuesta a los datos de tipo flotante).
- Dedicar algunos minutos para seguir experimentando con estos 2 tipos de datos, alternando los valores para ver qué ocurre si aumentamos la parte entera y disminuimos los decimales, o cualquier otra cosa que se te ocurra.

Cuarta parte: Declaraciones y conversiones entre tipos.

Sugerencia de tiempo invertido: 2 horas.

La participante identificará cómo llevar a cabo declaraciones implícitas y explícitas en Kotlin, así como conversiones entre los diferentes tipos de datos. Trabjará en el IDE de Android Studio o en Kotlin Playground, haciendo uso de los archivos provistos. Como evidencia se pedirá a la participante entregar los archivos modificados con todos los pasos ejecutados. Incluyendo su nombre y anotaciones que considere pertinentes, a partir de comentarios.

La ejecución variará dependiendo del entorno, así que podrá brindar apoyo a la participante.

- Declaraciones

Para esta sección se trabajará con el archivo *declaraciones.kt*, en donde viene indicado el lugar donde la participante deberá escribir su código.

- Implícitas

Identificará la parte marcada para las declaraciones implícitas, en donde ya hay un ejemplo de cómo se desarrollará el ejercicio. Ejecutará el código y observará la salida.

Después llevará a cabo las siguientes acciones y contestará las preguntas:

1. ¿Qué tipo de dato tiene la variable “número”? ¿Te parece correcto?
2. Declara una nueva variable llamada “short” y asígnale el valor que quieras, que esté en el rango del tipo Short.
3. Imprime el tipo de dato de la variable *short*.
4. ¿Qué observas?
5. Por defecto Kotlin infiere el tipo de nuestras variables, mientras no sobrepasen el rango de Int, el tipo que obtendremos será siempre ese.
6. Comprueba lo anterior creando una nueva variable llamada “long” asignándole un valor mayor al que acepta un Int (puedes apoyarte de la tabla).
7. ¿Fue posible demostrar el comportamiento?
8. Crea una nueva variable llamada “inicial” y dale el valor de la primera letra de tu apellido materno. (Es importante que envuelvas la letra en comillas simples, de otra forma no funcionará. Aquí hay un ejemplo -> ‘A’).
9. Revisa el tipo al que pertenece la variable *character*.
10. Intenta poner como valor las 2 primeras letras esta vez.
11. ¿Qué ocurrió? (Recuerda que la definición de carácter aplica sólo para un elemento).
12. Arregla tu código para que funcione nuevamente.
13. Genera una variable llamada “nombre” y dale el valor de tu nombre completo. (Esta vez envuelve el valor en comillas dobles. Aquí hay un ejemplo -> “Hola mundo!”)
14. Infiere el tipo de la variable, ¿Cuál es?
15. Por último agrega una variable llamada “pi” y dale el valor de 3.1416, e infiere el tipo.

16. ¿Cuál fue?

17. Al igual que con los tipos enteros, el tipo por defecto que infiere Kotlin para las variables de tipo punto flotante es `Double`.

- Explícitas

Identifica el área para las declaraciones explícitas, descomenta las 2 líneas que se encuentran comentadas, ejecuta el programa y observa la salida. Contesta y realiza lo siguiente:

1. ¿De qué tipo es la variable `shortExp`?
2. ¿Qué fue lo que cambió en la salida y en la notación?
3. Utiliza la misma notación para declarar una variable de tipo `Float` con el valor de raíz de 0.75, nombrala como quieras.
4. Infiere el tipo ¿Cuál es?
5. Crea otra variable, esta vez como en la sección de declaraciones implícitas y dale el mismo valor de 0.75, pero esta vez coloca una `f` después del 5.
6. ¿De qué tipo es la variable?
7. Cambia la `f` minúscula por mayúscula ¿Cambió el tipo?
8. Se puede indicar que un número será de tipo `Float` agregando al final una `f`, ya sea mayúscula o minúscula. Lo mismo ocurre para el tipo `Long` sólo que al final se coloca una `L`.
9. Comprueba que el comportamiento funciona para las variables de tipo `Long`.

- Conversiones

Para esta sección la participante trabajará con el archivo `conversiones.kt`, en el que probará las funciones para conversiones de tipos.

En el archivo se encuentra declarada una variable de cada tipo, la participante verificará las conversiones que son posibles aplicándolas a cada variable. Para aplicar las funciones conversoras bastará poner un punto después del valor y después el nombre de la función correspondiente, seguida de paréntesis. Por ejemplo, para convertir un entero a cadena:

```
val entero : Int = 0.toString()
```

Con base en los resultados obtenidos llenará la tabla de conversiones con las siguientes recomendaciones y nomenclatura:

Abreviatura	Tipo	Función conversora
B	Byte	toByte()
Sh	Short	toShort()
I	Int	toInt()
L	Long	toLong()
F	Float	toFloat()
D	Double	toDouble()
C	Char	toChar()
St	String	toString()

Para Char, probar con sólo una letra/número.

Para el llenado de la tabla se dará color a las casillas de la siguiente manera:

Fue posible realizar la conversión	Verde
Arrojó una excepción	Rojo

Tabla de conversiones

Tip o	Función							
	B	Sh	I	L	F	D	C	St
B								
Sh								
I								
L								
F								
D								
C								
St								

Por último, la participante aprenderá sobre otra forma de realizar conversiones leyendo el artículo *Verificación y Conversión de Tipos en Kotlin*, a partir de éste entregará los pedazos de código que el artículo menciona en un archivo llamado "conversiones2.kt".

Quinta parte: Mi calendario de actividades.

Sugerencia de tiempo invertido: 1 hora.

La participante:

1. Diseñará el bosquejo para un calendario de actividades.
2. Definirá los campos y tipos de sus variables para después transportarlo a Kotlin.
3. Identificará al menos cinco datos que debe contener un planeador, definirá el tipo de dato con el que se debe representar e identificará si el dato es constante o variable.
4. La participante puede apoyarse de una tabla como la siguiente para reunir la información que utilizará su programa:

Tabla de ejemplo

Nombre del campo	Tipo de dato	¿Este dato podría cambiar con el tiempo?	Da un ejemplo de valor que podría almacenar
Nombre de la actividad	String	Sí	Reunión de trabajo

5. Una vez identificados los campos, se deberá pasar la información de la tabla a sintaxis de Kotlin, definiendo una variable con las características identificadas y añadiendo como valor, el identificado en el ejemplo.

Links de apoyo:

Para la tercera parte de la actividad

- Educar Portal. (2019). Microaprendizaje: *¿Cómo funciona una computadora?* [video]
<https://www.youtube.com/watch?v=oYxE3L-6-a8&t=171s> (Julio, 2020).

Concepto de bit y byt:

- Code.org. (2018). *How Computers Work: Binary & Data*. [video]
<https://www.youtube.com/watch?v=USCBCmwMCDA> (Julio, 2020).
- codigofacilito (2017). *Bits y Bytes explicados en 2 minutos* 🕒. [video]
<https://www.youtube.com/watch?v=thoGwqjPHRM> (Julio, 2020).
- Cosas del internet. (2017). *Diferencias entre Bit y Byte*. [video]
<https://www.youtube.com/watch?v=uECdAxiHAvY> (Julio, 2020).

¿Qué es el código binario?

- Derivando. (Sin fecha). *El código binario | Explicación*. [video]
<https://www.youtube.com/watch?v=f9b0wwhTmeU> (Julio, 2020).

Tipos básicos, Documentación oficial.

- Kotlin v1.4.30. (Sin fecha). *Basic types*.
<https://kotlinlang.org/docs/reference/basic-types.html> (Julio, 2020).
- Kotlin. (Sin fecha). *Kotlin Playground is an online sandbox to explore Kotlin programming language. Browse code samples directly in the browser*.
<https://play.kotlinlang.org/> (Julio, 2020).

Para la cuarta parte de la actividad

- Olivares, A. (2017). *Verificación y Conversión de Tipos en Kotlin*.
<https://medium.com/aldominium-com/verificaci%C3%B3n-y-conversi%C3%B3n-de-tipos-en-kotlin-f8b7f4f6cc9a> (Julio, 2020).

Actividad 5: ¿Cómo le digo a mi programa lo que debe de hacer con los datos?

Aprendizaje esperado: Manipular los datos de un programa mediante los operadores matemáticos, relacionales y lógicos.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Editor de textos. • Anexo <i>MMT2A5_Anexos.pdf</i> 	<p>Evidencia:</p> <ol style="list-style-type: none"> 1. Un archivo de texto que contenga los operadores usados en Kotlin. <p>Producto:</p> <ol style="list-style-type: none"> 1. Un programa que al introducir mediante el teclado el valor del lado de un triángulo equilátero, calcule su área, su perímetro e imprima los valores en la pantalla. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> • En un archivo de texto mostrar lo siguiente: <ul style="list-style-type: none"> • Operadores matemáticos. • Operadores relacionales. • Operadores lógicos. <p>Evidencia:</p> <ul style="list-style-type: none"> • En el emulador de Android Studio verificar la estructura del programa creado y los datos de entrada y salida del programa.
Desarrollo de la actividad:		
<p>Primera parte: Tipos de operadores.</p> <p>Sugerencia de tiempo invertido: 2 horas.</p> <ol style="list-style-type: none"> 1. La participante deberá dominar el manejo y uso de los distintos tipo de operadores que existen en Kotlin, es probable que ya haya tenido algún tipo de acercamiento con este tipo de operadores previamente pero es importante que aprenda sobre su uso en específico. 2. Deberá realizar una búsqueda en internet de los tipos de operadores (matemáticos, de asignación, lógicos y de comparación o relacionales). 3. Posteriormente, resolverá la plantilla A (disponible en el anexo <i>MMT2A5_Anexos.pdf</i>). 4. Para finalizar, responderá en un archivo de texto las siguientes preguntas: <ul style="list-style-type: none"> • ¿Qué es un operador? • ¿Cómo funciona? 		

- ¿Qué estructura tiene cada tipo de operador?

Segunda parte: Categorías de los operadores.

Sugerencia de tiempo invertido: 1 hora.

1. La participante buscará códigos en Kotlin con ejemplos de cada categoría que integra cada tipo de operador (ejemplo: Operadores matemáticos; suma, resta, multiplicación, división, etc.).
2. De cada código tomará una captura de pantalla y subrayará con colores diferentes la parte donde se ocupan los operadores.
3. En cada operador redactará de manera breve qué realiza cada uno.

Tercera parte: Cálculo de área y perímetro en triángulos equiláteros.

Sugerencia de tiempo invertido: 2 horas.

1. Se creará un nuevo proyecto en Kotlin nombrado “AreaYPerimetroApp”, para calcular el área y el perímetro de triángulos equiláteros a partir de datos introducidos por el usuario.
2. El programa deberá detectar qué es y qué no es un triángulo equilátero y enviar (si no lo es) un mensaje que indique que las medidas no corresponden a ese tipo de triángulo.
3. Deberá probar su programa en un emulador de Android Studio.

Links de apoyo:

Operadores:

- Fábrica de software. (Sin fecha). *Curso Kotlin: Operadores*.
<https://fabrica-software.blogspot.com/2018/12/curso-kotlin-operadores.html#Introduccion-Operadores> (Agosto, 2020).
- DDPADMIN. (2020). *Trabajar con Números y Operaciones Matemáticas en Android con Kotlin (Tema 4)*.
https://iqanansoft.es/ddp_web/trabajar-con-numeros-y-operaciones-matematicas-en-android-con-kotlin/ (Agosto, 2020).

Actividad 6: Tomando decisiones en mi programa

Aprendizaje esperado: Aplicar la sentencia condicional “if” cuando es necesario tomar una decisión dentro del programa.		Duración de la actividad: 3 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.• Dispositivo móvil.	<p>Producto:</p> <ol style="list-style-type: none">1. Desarrollar un programa que permita ingresar el sueldo de una persona, si supera los \$10,000 pesos mostrar un mensaje en pantalla indicando que debe agregar 15% de impuestos, imprimir la cantidad que debe de sumar y la cantidad total de esta suma. Mostrar resultados en pantalla.2. Desarrollar un programa que permita ingresar 2 valores enteros (positivos o negativos). Si el primero es menor al segundo calcular la suma y resta, sino calcular el producto y división, mostrar el resultado. En cuestión de la división considerar el caso en que se indique dividir entre cero, y mandar un mensaje de “no se puede hacer la división para este caso”.	<p>Evaluación:</p> <ul style="list-style-type: none">• Mostrar el funcionamiento de los programas creados.• Presentar los resultados en pantalla.• Verificar que la solución de los problemas propuestos sea correcta.
Desarrollo de la actividad:		

Primera parte: Si estudio en la escuela de código, entonces...

Sugerencia de tiempo invertido: 30 minutos.

La participante distinguirá el funcionamiento de la sentencia if, asociándola con la toma de decisiones en su vida diaria. Para lo cual, en un documento realizará lo siguiente:

1. Identificará tres razones que la llevaron a inscribirse a la escuela de código.
2. Mencionará tres habilidades o conocimientos adquiridos hasta ahora en los módulos cursados.
3. Nombrará alguno de los proyectos que haya llevado a cabo en la escuela de código.
4. Con la información identificada en los puntos anteriores, la participante llenará la siguiente tabla:

	Condición	Consecuencias de cumplirse la condición		Consecuencias de no cumplirse la condición
Si	asisto a la escuela de código, entonces		En caso contrario	

5. La participante propondrá otros cinco enunciados en los que el cumplimiento de una acción dependa de que se efectúe una condición. Plasmará los ejemplos siguiendo la estructura:

Si <condición>, entonces <consecuencia>

6. Partiendo de los 5 enunciados del punto anterior, la participante ahora incluirá lo que puede ocurrir en caso de que la condición no se cumpla. Deberá añadir esta información en sus enunciados para seguir la estructura mostrada a continuación:

Si <condición>, entonces <consecuencia>. En caso contrario, <consecuencia de no cumplirse la condición>

Segunda parte: Toma de decisiones en un lenguaje.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante identificará las condiciones del Lenguaje Incluyente y No Sexista respecto a los siguientes enunciados y colocará la letra correspondiente en cada uno de ellos. En caso de tener dificultades para completar la relación de columnas, puede hacerse uso del link provisto al final del documento.

Condición	Ejemplo
<ol style="list-style-type: none">a. Desdobla los términos.b. Usa el femenino de las profesiones.c. Usa genéricos más incluyentes.d. Usa formas impersonales.	<p>() Las niñas y niños aprenden mientras juegan, comparten y se divierten.</p> <p>() Las ingenieras entre otras cosas, desarrollan, programan y diseñan aplicaciones móviles.</p> <p>() La ciudadanía ha tenido que tomar medidas de sana distancia durante sus actividades diarias.</p> <p>() Les invitamos a participar en los diferentes cursos impartidos en los PILARES.</p>

2. La participante discutirá con la tallerista o compañeras, las siguientes preguntas:
 - a. ¿Qué crees que son las condiciones?
 - b. ¿Las condiciones son exclusivas de los lenguajes naturales?
 - c. ¿Cómo crees que pueden expresarse las condiciones en los lenguajes de programación?
 - d. ¿Qué uso podrían tener las condiciones en los lenguajes de programación?

Tercera parte: “If” en acción.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante revisará los dos recursos propuestos para reforzar la explicación sobre la sentencia `if` y observar su uso en la resolución de un problema.
2. Después, revisará el link de la documentación oficial para conocer la sintaxis en Kotlin.
3. Por último, transformará las oraciones del punto 6 de la primera parte de la actividad, a la sintaxis de Kotlin y las adjuntará como punto 7 de su documento.

Cuarta parte: Agregando condiciones a mis programas.

Sugerencia de tiempo invertido: 2 horas.

La participante desarrollará dos programas en *Kotlin* que incluirán la toma de decisiones, para poner en práctica el concepto y sintaxis de la expresión `if`. La participante puede que genere un nuevo proyecto en *Android Studio* o se haga uso de *Kotlin Playground*. A continuación se enuncian las características que deberán poseer los programas:

- Programa 1
 - Definir en una variable de tipo `Double` el sueldo de una persona.
 - Si éste supera los \$10000 se deberá agregar un 15% de impuestos.
 - El programa deberá imprimir el sueldo original, la cantidad que se va a sumar de impuestos y el total.
- Programa 2
 - El programa tendrá como entrada 2 números, A y B.
 - Si A es menor que B, se deberá calcular la suma y resta de los números.
 - En caso de que A no sea menor que B, se deberá calcular la multiplicación y división de los números.
 - Se deberá tomar en cuenta que un número no puede ser dividido entre 0, por lo que de presentarse esta situación, se imprimirá el mensaje: "No se puede hacer la división para este caso".

Se deberán adjuntar en el documento capturas de pantalla tanto del funcionamiento de los programas como de los códigos.

Nota:

- Los recursos provistos para la tercera parte de la actividad se encuentran en inglés, por lo que se deberán activar los subtítulos al español en Youtube.

Links de apoyo:

Segunda parte

QROO.gob.mx. (2016). *LENGUAJE INCLUYENTE NO DISCRIMINATORIO Y NO SEXISTA. GUÍA PARA FACILITAR EL USO DEL LENGUAJE EN LAS COMUNICACIONES ESCRITAS Y ORALES.*

<https://qroo.gob.mx/seq/lenguaje-incluyente-no-discriminatorio-y-no-sexista-guia-para-facilitar-el-uso-del-lenguaje-en>
(Julio, 2020).

Tercera parte

- Code.org. (2013). *Hour of Code-Bill Gates explains If statements.* [video]
<https://www.youtube.com/watch?v=m2Ux2PnJe6E> (Julio, 2020).
- *If Statements| Computers Programming | Khan Academy.* [video]
<https://www.youtube.com/embed/f80BDI8N-6c> (Julio, 2020).
- Kotlin v1.4.30.(Sin fecha). *Conditions and loops.*
<https://kotlinlang.org/docs/reference/control-flow.html> (Julio, 2020).

Actividad 7: Tomando una decisión entre varias opciones

Aprendizaje esperado: Aplicar la sentencia de control de flujo “when” cuando es necesario tomar una decisión entre más de dos opciones.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Anexo <i>MMT2A7_Anexos.pdf</i> 	<p>Producto:</p> <ol style="list-style-type: none"> 1. Realizar un programa que permita ingresar el peso (en kilogramos) de distintos productos de una tienda. El proceso termina cuando ingresamos el valor 0. Se debe informar: <ul style="list-style-type: none"> a) Cuántas piezas tienen un peso entre 9.8 Kg. y 10.2 Kg., cuántas con más de 10.2 Kg., y cuántas con menos de 9.8 Kg. b) La cantidad total de piezas procesadas. 2. Realizar el ejercicio anterior usando la condicional “if”. 	<p>Evaluación:</p> <ul style="list-style-type: none"> • Mostrar el número correcto de piezas que se encuentren entre los valores de peso requerido y el número total de piezas procesadas. • Mostrar que el resultado de ejecutar los dos programas es el mismo usando distintas estructuras.
Desarrollo de la actividad:		
<p>Primera parte: Sentencias de control de flujo. Sugerencia de tiempo invertido: 1 hora.</p> <ol style="list-style-type: none"> 1. La participante deberá realizar una búsqueda en internet acerca de la sentencia de control de flujo <i>when</i>. Posteriormente, responderá las siguientes preguntas: <ul style="list-style-type: none"> • ¿Qué son las sentencias de control de flujo? (If, if-else, for, while, when). • ¿Para qué sirven la sentencia de control de flujo <i>when</i>? • ¿Cómo funciona? • ¿Cómo se declara cada una de las sentencias? (Hacer un ejemplo para cada una). 		

- ¿Para qué se ocupan usualmente?
 - ¿Cuál es la diferencia entre las sentencias de control y las estructuras de repetición?
2. Con base en la búsqueda realizada, en la plantilla A (disponible en el anexo *MMT2A7_Anexos.pdf*) deberá relacionar las sentencias con los enunciados que les correspondan siguiendo las reglas del juego tripas de gato (no se pueden cruzar las líneas entre sí).

Segunda parte: Practicando las sentencias de control de flujo.

Sugerencia de tiempo invertido: 2 horas.

1. La participante realizará un pequeño ejercicio para practicar las sentencias de control de flujo.
2. Deberá crear un programa por cada sentencia de control de flujo que resuelva los siguientes problemas:
 - Un programa que compare dos números ingresados por el usuario (Variable A y Variable B) e indique cuál es mayor.
 - Un programa que indique si el número ingresado es negativo o positivo.
 - Un programa que lea dos números enteros y escriba todos los números mayores que el primero y menores que el segundo.
 - Programa que muestre de forma descendente los números del uno al diez.
3. En caso de que la participante tuviera problemas al realizar sus códigos se recomienda revisar los *Links de apoyo* donde encontrará la explicación correspondiente a cada uno.

Tercera parte: Diferencias entre *if* y *when*.

Sugerencia de tiempo invertido: 2 horas.

1. Para reforzar la diferencia entre las sentencias de control, la participante realizará dos versiones del siguiente programa:
 - Un programa que permita ingresar el peso (en kilogramos) de distintos productos de una tienda (al menos 6) y que finalice el proceso si se ingresa un producto que pese 0kg.

Se deberá informar:

- a. ¿Cuántas piezas tienen un peso entre 9.8 Kg. y 10.2 Kg?
 - b. ¿Cuántas piezas hay con más de 10.2 Kg?
 - c. ¿Cuántas piezas hay con menos de 9.8 Kg?
 - d. La cantidad total de piezas procesadas.
 - e. Mostrar en forma de lista los productos y su peso.
2. Se deberá realizar primero con la sentencia *when* y posteriormente con la sentencia *if*.
 3. Al finalizar la participante contestará las siguientes preguntas y si es posible las discutirá con la tallerista y/o sus compañeras.
 - ¿Hubo alguna diferencia en el desarrollo del programa entre las distintas estructuras de control? ¿Cuál?
 - ¿Cuál código es más eficiente, cuando se usa *when* o al usar *if*? ¿Por qué?
 - ¿Lo intentarías con algún otro tipo de estructura de control?

Links de apoyo:

Secuencias de control de flujo:

- IBM Knowledge Center. (Sin fecha). *Sentencias de control de flujo*.
https://www.ibm.com/support/knowledgecenter/es/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/adminuse/terminal_scriptflowcontrol_tep.htm (Agosto, 2020).
- Kotlin para Android. (2019). *Control de Flujo: if, when, for y while*.
<https://kotlin.desarrollador-android.com/basico/control-de-flujo/> (Agosto, 2020).
- Salas, I. (2017). *LOS CONDICIONALES IF, ELSE Y WHEN EN KOTLIN*.
<https://programandoointentandolo.com/2017/11/los-condicionales-if-else-when-kotlin.html> (Agosto, 2020).
- Kotlin Doc Kotlin & Android. (Sin fecha). *Instrucciones condicionales I: if*.
<https://kotlin.doc.blogspot.com/2019/04/instrucciones-condicionales-i-if.html> (Agosto, 2020).

Actividad 8: Repetir acciones hasta alcanzar el objetivo

Aprendizaje esperado: Aprender la forma en el que <i>Kotlin</i> permite realizar la misma acción varias veces de forma automática.		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. 	<p>Evidencia:</p> <ol style="list-style-type: none"> 1. Captura de pantalla del funcionamiento de <i>ciclo_for.kt</i> 2. Captura de pantalla de programa 1 y programa 2 de parte 5. <p>Producto:</p> <ol style="list-style-type: none"> 1. Realizar un programa que imprima en pantalla los números pares entre 1 y 100 de manera inversa. 2. Desarrollar un programa que le pregunte al usuario 10 valores y muestre posteriormente la suma de los valores ingresados y su promedio. 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> • Verificar que las capturas de pantalla muestren el correcto funcionamiento del código. <p>Evaluación:</p> <ul style="list-style-type: none"> • Comprobar la correcta ejecución del contador. • Verificar la interacción entre el programa y el usuario, demostrando que las operaciones sean correctas.
Desarrollo de la actividad:		
<p>Primera parte: Los ciclos están en todas partes.</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <p>La participante inferirá el concepto básico y las características de un ciclo, a partir de la asociación con experiencias personales o conocimientos previos. Para ello, desarrollará una redacción corta que compartirá con la tallerista y compañeras a partir de los siguientes puntos:</p> <ul style="list-style-type: none"> • ¿Alguna vez has escuchado la palabra ciclo? ¿En dónde? • ¿Sabes a qué hace referencia? • Define la palabra ciclo. 		

- ¿Cuáles consideras que son sus características principales?

Algunos ejemplos de ciclo son:

- Ciclo del agua.
- Ciclo de la vida.
- Ciclo de la Luna.

Además de los ejemplos antes mencionados, existen ciclos que están más relacionados con nuestra rutina o nuestro cuerpo, por ejemplo:

- Ciclo menstrual.
- Ciclos circadianos.
- Día.
- Año.
- ¿Te afectan de manera directa?
- Para conocer las características de los ciclos, investiga más a detalle sus etapas, menciona el nombre, tiempo estimado y qué ocurre en cada una de ellas. Con la información recabada completa la siguiente tabla:

Nombre	¿Las etapas siguen un orden?	Cuando se llega a la última etapa ¿se repite?	¿Cada cuánto se repite?
Ciclo menstrual			
Ciclos circadianos			
Día			
Año			

La tercera columna puede no tener una respuesta definida, como en el caso del ciclo del agua o el de la vida.

- ¿Todos tienen una duración entre cada repetición definida?
- ¿En algún momento se detienen?

Es importante aclarar que tener una duración establecida no es necesariamente una característica de un ciclo, algunos podrán detenerse en un punto porque determinada condición ocurrió y otros podrán seguir “indefinidamente”.

- ¿El reciclaje podría considerarse un ciclo?
- Propón al menos otros 3 ejemplos de fenómenos, sucesos o acciones que ocurran de forma cíclica y otros 3 ejemplos de procesos que impliquen repetición en tu vida cotidiana.

Previo a la segunda parte:

Como se revisó en la primera parte “Los ciclos están en todas partes”, un elemento fundamental de los ciclos es la repetición, esta situación es clave en la vida cotidiana, ya que, por medio de la repetición se crean hábitos y costumbres que permiten el desarrollo individual.

Segunda parte: De hábitos y costumbres.

Sugerencia de tiempo invertido: 30 minutos.

La presencia de las mujeres en ámbitos profesionales, cargos y oficios masculinizados como las STEM, ha traído como consecuencia que a las mujeres se reconozca como parte de estos grupos, y para ello, el lenguaje se ha convertido en una herramienta aliada no sólo para reconocerlas, sino también para eliminar estereotipos y evitar la discriminación que se puede generar al invisibilizar a las mujeres a través de no nombrarlas.

A continuación, presentamos una tabla donde se muestra que el hábito y la costumbre de no nombrar a las mujeres puede modificarse mediante el uso y aprendizaje del Lenguaje Incluyente y No Sexista. Sugerimos que la participante complete la tabla con dos situaciones en las que haya identificado un entorno donde no se nombrara a las mujeres. Algunos ejemplos de nombramientos femeninos son: doctora, abogada, licenciada, ingeniera, programadora, panadera, la chef, artesana, etcétera.

Situación	Modificación
-----------	--------------

Karla es el ingeniero que se hace cargo de desarrollar aplicaciones para una empresa en crecimiento.	Karla es la ingeniera que se hace cargo de desarrollar aplicaciones para una empresa en crecimiento.
Nosotros, en la Escuela de Código, tenemos experiencias teóricas y prácticas en programación.	Nosotras en la Escuela de Código, tenemos experiencias teóricas y prácticas en programación.

Tercera parte: Definiendo un objetivo y avanzando hasta alcanzarlo.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante relacionará la programación con el concepto de ciclo, lo comparará con el ejercicio que realizó en la primera parte “Los ciclos están en todas partes”, de manera que identificará las similitudes o diferencias que pudieran existir.
2. Se sugiere revisar los recursos proporcionados para abordar la relación entre los ciclos y la programación (disponibles en los *Links de apoyo*).
3. Después de revisar los recursos, discutirá con otras participantes o la tallerista las siguientes preguntas:

- Partiendo del concepto que observaste en la primera parte ¿hay alguna diferencia con los ciclos en programación y los que ya conocemos? ¿Cuál?
- ¿Será posible escribir en código los ejemplos de la primera parte?

4. Explicado el concepto de ciclo en la programación, se identificará al ciclo *for* junto con sus características. Los siguientes puntos ayudarán a conducir la discusión:

- Selecciona 2 de los ejemplos mencionados en la primera parte o propón otros, considera que el periodo de repetición del ciclo se encuentre bien definido o exista un aproximado. Por ejemplo: Un año se repite cada 365 días aproximadamente.
- Identifica el inicio y fin de los ciclos elegidos en el punto anterior y completa la tabla:

Ciclo	Inicio	Fin
Año	Día 1 Mes 1	Día 365 Mes 12

- Puede llegar a ocurrir que el inicio y el fin del ciclo se represente en más de una forma, como se indica en el ejemplo. Si ese es el caso, deberá elegirse solo una forma.
- El ciclo *for* define un punto de partida, un objetivo al que quiere llegar y el tamaño de los pasos que dará en cada repetición. Si no se ha cumplido el objetivo el ciclo *for* se continuará repitiendo y avanzando hacia el objetivo según el tamaño de pasos definidos. Véase la siguiente expresión como ejemplo:

Si (variable entre punto_de_partida y objetivo) dar un paso de tamaño “x” y repetir.

Con la palabra *for*, lo anterior se expresaría:

```
for(variable entre punto_de_partida y objetivo dar un paso de tamaño x)
```

Si quisiéramos representar un año, podríamos ponerlo en palabras usando la estructura anterior de la siguiente manera:

```
for(día entre 1 y 365 dar 1 paso de tamaño 1)
```

De esta forma estamos diciendo que, empezando en uno y hasta no llegar a 365 avanzaremos un paso cada vez (similar a recorrer día por día el año).

Otro punto importante es que llamaremos rango a la parte del enunciado “entre 1 y 365”. Ya que estamos delimitando los valores que puede tomar.

- Sigue la estructura anterior con los 2 ciclos que elegiste.

Cuarta parte: Para cada ciclo con límites definidos usa *for*.

Sugerencia de tiempo invertido: 2 horas.

1. La participante investigará la sintaxis que utiliza el lenguaje de programación *Kotlin* para el ciclo *for* (se recomienda visitar la página de la documentación oficial).
2. Una vez que la participante reconozca la estructura de un ciclo *for*, transformará los enunciados de la segunda parte de la actividad a *Kotlin*.

Para el ejemplo de año:

```
for(dia in 1..365)
```

3. Posteriormente, la participante creará un nuevo proyecto en *Android Studio* o abrirá la consola de *Kotlin Playground* para probar su funcionamiento.
 - Primero, copiará el contenido del archivo *ciclo_for.kt* (disponible en los anexos), para ver un ejemplo en funcionamiento.
 - Luego de ello, deberá incluir al código sus 2 enunciados y verificar que trabajen correctamente. De ser así, adjuntará una captura de pantalla del código en funcionamiento.
4. La participante experimentará con las palabras reservadas: *until*, *downTo* y *step*, a fin de comprender mejor su funcionamiento. Se recomienda seguir los siguientes pasos:
 - a. Comienza desde cero un proyecto en *Android Studio* o *Kotlin Playground*.
 - b. Genera el código para representar un año pero esta vez con meses.
 - c. Copia y pega el código anterior y modifica los 2 puntos que están entre el inicio y fin de tu ciclo por la palabra *until*.
 - d. Copia y pega el código que acabas de crear pero esta vez cambia la palabra *until* por *downTo* e intercambia el lugar de inicio y fin de tu ciclo.
 - e. ¿Cuál es el propósito de la palabra *downTo*?
 - f. Si quisiéramos generar un código para el ciclo de la Luna, debemos tomar en cuenta que:
 - i. El ciclo comienza en el día 1, por lo que inicio = 1
 - ii. La duración total del ciclo es de 28 días, por lo que fin = 28
 - iii. Contemplando sólo 4 fases lunares, aproximadamente cada fase tiene una duración de 7.38

Así que el código debería verse más o menos así:

```
for(dia in 1..28 step 7.38) { println("La Luna cambio de fase") }
```

Copia esta instrucción en tu código y observa la salida

- g. ¿Cuál es el error?
- h. Los números que declaremos como rangos y pasos, siempre deben ser de tipo entero.
- i. Aunque perderemos exactitud modifica el valor de `step` a 8.
- j. ¿Cuál es el propósito de `step`?

Quinta parte: Perseverar para alcanzar.

Sugerencia de tiempo invertido: 2 horas.

La participante elaborará 2 programas con el propósito de integrar los conocimientos de la actividad. Las características de los programas son:

- **Programa 1**

- Imprimir los números pares entre 1 y 100 de manera inversa.

- **Programa 2**

- Generar el promedio de 10 números.
- El programa deberá recibir datos del usuario, haciendo uso de la instrucción `readLine()`.
- Deberá convertir el dato obtenido por `readLine()` a entero, ya que en otro caso será de tipo cadena.
- Con cada iteración se irá sumando al total el número que se leyó del usuario.
- Al final se deberá imprimir el mensaje "El promedio es: " así como el dato obtenido.

La tallerista deberá verificar que los programas cumplan las características requeridas. Y una vez que trabajen de manera correcta, la participante adjuntará las capturas de pantalla.

Nota:

- En la cuarta parte de la actividad deberá resaltar que por defecto el tamaño del paso es 1.

Links de apoyo:

Para la segunda parte de la actividad

- CNDH. (2016). “3.1 Hacer invisibles a las mujeres: el derecho a ser nombrada” y “3.2 Si eres mujer, nómbrate” en *Guía para el uso del Lenguaje Incluyente y no Sexista en la CNDH*.
<http://www.cdhezac.org.mx/TRANSPARENCIA/vinculos/GuiaLenguajeIncluyente.pdf> (Julio, 2020).
- Guichard, Claudia. (2018). CAPÍTULO 5 ¿Por qué nombrar a las mujeres? en *Manual de comunicación no sexista: hacia un lenguaje incluyente*.
http://cedoc.inmujeres.gob.mx/documentos_download/101265.pdf. (Julio, 2020).

Para la tercera parte de la actividad (todos se encuentran en idioma inglés, sin embargo, es posible activar los subtítulos al español)

- Codecademy. (2019). *Intro to Programming: Loops*. [video]
<https://www.youtube.com/watch?v=wxds6MAtUQ0> (Julio, 2020).
- GCFLearnFree.org. (2018). *Computer Science Basics: Sequences, Selections and Loops*. [video]
https://www.youtube.com/watch?v=eSYeHlwDCNA&list=LLiETdo_W-qIgaLVGvrZ_Vhw&index=3&t=0s (Julio, 2020).
- Kodable (2020). *What are Loops? Coding for kids* | Kodable. [video]
https://www.youtube.com/watch?v=r3Ti5Xp9W8A&list=LLiETdo_W-qIgaLVGvrZ_Vhw&index=1 (Julio, 2020).
- Code.org. (2013). *Hour of Code - Mark Zuckerberg teaches Repeat Loops*. [video]
<https://www.youtube.com/watch?v=mgooqyWMTxk> (Julio, 2020).

Sintaxis ciclo for:

- Eldhuset, A. (2018). *Kotlin for Python developers*.
<https://kotlinlang.org/docs/tutorials/kotlin-for-py/loops.html> (Julio, 2020).

Actividad 9: Repetir acciones mientras el objetivo esté vigente.

Aprendizaje esperado: Aprender otra forma en que <i>Kotlin</i> ejecuta instrucciones o conjunto de instrucciones de forma repetitiva.		Duración de la actividad: 6 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.• Anexo <i>MMT2A9_Anexos.pdf</i>	Producto <ul style="list-style-type: none">• Realizar un programa que imprima los números del 1 al 50.• Una fábrica de barras de hierro posee un lote de 'n' piezas. Desarrollar un programa que pida ingresar la cantidad de piezas a procesar y luego ingrese la longitud de cada barra; sabiendo que las piezas cuya longitud esté comprendida en el rango de 1.20 m y 1.30 m son aptas. Imprimir la cantidad de piezas aptas que hay en el lote.	Evaluación <ul style="list-style-type: none">• Comprobar la correcta ejecución del contador.• Verificar que el total de piezas aptas se encuentren en el rango de 1.20 m y 1.30 m.
Desarrollo de la actividad:		
Primera parte: Los ciclos en Kotlin. Sugerencia de tiempo invertido: 2 horas.		
<ol style="list-style-type: none">1. La participante deberá realizar una búsqueda en internet acerca del ciclo <i>while</i> en <i>Kotlin</i> y responderá las siguientes preguntas:<ul style="list-style-type: none">• ¿Qué es el ciclo <i>while</i> y <i>do-while</i>?• ¿Para qué sirven?• ¿Cómo funcionan?• ¿Cómo se estructuran? Buscar un ejemplo.		

- ¿Cuál es la diferencia entre el ciclo *for*, *while* y *do-while*?
 - ¿Qué es un bucle en *Kotlin*?
 - ¿Para qué se utilizan los bucles?
 - ¿Cómo salir de un bucle (*break* y *continue*)?
2. Con base en la búsqueda realizada completará los enunciados de la plantilla A y posteriormente, en la plantilla B ordenará las piezas del código de ejemplo (ambas plantillas se encuentran disponibles en el anexo *MMT2A9_Anexos.pdf*).

Segunda parte: Eligiendo la estructura.

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá de realizar lo siguiente:
 - Un programa que pida números (mientras no se escriba un número negativo) y que al terminar escriba la suma de los números introducidos.
 - Un programa que pida dos números enteros y escriba ¿qué números son pares y cuáles impares? desde el primero hasta el segundo.
 - Un programa que pida un número entero mayor que cero y que escriba sus divisores.
 - Un programa que muestre en la pantalla cuántos múltiplos de 3 existen entre el 10 y el 70.
 - Un programa que calcule la suma de los primeros diez números naturales y muestre el resultado en pantalla.
 - Un programa que imprima los números del 1 al 50.
2. Deberá usar la sentencia que crea correspondiente para dar solución.
3. Al finalizar, explicará ¿qué estructura decidió utilizar para cada programa?, ¿por qué eligió esa en específico? y si considera que su elección fue la mejor o si podría optimizar su código con alguna otra estructura.
4. Además deberá tomar captura de pantalla a sus códigos y en un color diferente subrayar las estructuras *for*, *while* y *do while* que haya usado.

Tercera parte: Requerimientos de un programa.

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá leer detenidamente el siguiente problema:

- Una fábrica de barras de hierro que posee un lote de “N” piezas, necesita que desarrolle un programa que pida ingresar manualmente la cantidad de piezas a procesar y posteriormente, la longitud de cada barra. El programa deberá saber que las piezas cuya longitud esté comprendida en el rango de 1.20 m y 1.30 m son aptas. Al final deberá imprimir la cantidad de piezas aptas que hay.
2. Posteriormente, deberá responder las siguientes preguntas:
 - ¿Qué estructura se debe de usar? Justificar la respuesta.
 - ¿Es conveniente ayudarse del uso de bucles?
 - ¿Cómo se podría optimizar alguna parte del programa para ingresar en una sola entrada el número de barras y la longitud? Describir el procedimiento.
 3. Contestadas las preguntas, deberá realizar un programa que resuelva el problema.
 4. Por último, verificará las respuestas de las preguntas planteadas para comprobar si el razonamiento del programa fue el solicitado.

Links de apoyo:

Ciclos for,while y do-while:

- Salas, I. (2017). *BUCLES EN KOTLIN (FOR, WHILE Y DO WHILE)*.
<https://programandoointentandolo.com/2017/11/bucles-kotlin-for-while-y-do-while.html> (Agosto, 2020).
- tutorialesprogramacionya.com. (Sin fecha). 9. *Estructura repetitiva while*.
<https://www.tutorialesprogramacionya.com/kotlinya/detalleconcepto.php?punto=9&codigo=9&inicio=0> (Agosto, 2020).
 - 11. *Estructura repetitiva for y expresiones de rango*.
<https://www.tutorialesprogramacionya.com/kotlinya/detalleconcepto.php?punto=11&codigo=11&inicio=0> (Agosto, 2020).

Bucles en kotlin:

- SO Documentation. (Sin fecha). *Loops in Kotlin*.
<https://sodocumentation.net/es/kotlin/topic/2727/bucles-en-kotlin> (Agosto, 2020).

Actividad 10: Almacenando y recuperando datos del mismo tipo de elemento

Aprendizaje esperado: Manejar un conjunto de datos del mismo tipo con ayuda de una estructura llamada arreglo.		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. 	<p>Evidencias:</p> <ol style="list-style-type: none"> 1. Documento <i>MMT2A10_evidencias.txt</i> 2. Capturas de pantalla del funcionamiento del código. <p>Producto:</p> <ol style="list-style-type: none"> 1. Crear un programa que pida la estatura de 5 personas y las almacene en un arreglo. <ol style="list-style-type: none"> a) Obtener el promedio de las mismas. b) Contar cuántas personas son más altas que el promedio. c) Contar cuántas personas son más bajas que el promedio. d) En cada caso imprimir las alturas. 	<p>Retroalimentación:</p> <ol style="list-style-type: none"> 1. Revisar los puntos del documento <i>MMT2A10_evidencias.txt</i> 2. Verificar las capturas de pantalla. <p>Evaluación:</p> <ul style="list-style-type: none"> • Verificar el uso correcto de arreglos. • Demostrar que los datos impresos sean correctos en cada caso.
Desarrollo de la actividad:		
<p>Primera parte: ¿Alguna vez has coleccionado objetos?</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <p>La participante interpretará el concepto de arreglo en programación, sus características e identificará algunos ejemplos en la vida real.</p> <ul style="list-style-type: none"> • En algún momento de tu vida ¿has coleccionado objetos? • Si tu respuesta fue sí, ¿qué coleccionabas? • ¿Cuáles crees que sean las colecciones más comunes? • Realiza una breve búsqueda de imágenes de colecciones. 		

- ¿Cuál crees que sea el primer paso para iniciar una colección?
- ¿Cuál crees que es la característica principal de una colección?
- Aparte de una colección, piensa en otros 2 ejemplos en donde se agrupen objetos del mismo tipo.
- Revisa los recursos propuestos para esta primera parte, en el siguiente orden:
 - Arreglos en programación
 - Explicación (ver los primeros 26 segundos).
 - Arreglos 1: ¿qué es un arreglo?
 - Explicación (ver hasta el minuto 1:52).
- ¿Cómo los arreglos nos ayudan a reducir código?
- Otra característica de los arreglos es que su tamaño una vez definido no puede cambiar, únicamente podemos cambiar el valor de sus elementos, ¿esto supone una similitud o diferencia con las colecciones de objetos que tenemos las personas?
- ¿Aún así sería posible transportar el concepto a la programación?
- ¿Cuáles crees que sean los tipos de datos permitidos para un arreglo en *Kotlin*?

Segunda parte: Mis colecciones.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante describirá una colección de sí misma, basándose en: las actividades que suele realizar, gustos, características, etc. En la descripción no debe olvidar priorizar el uso del femenino, por ejemplo: corredora, bailarina, artesana, jugadora de videojuegos, lectora, etcétera.
2. La colección puede ser presentada a la tallerista o a compañeras participantes, comenzando de la siguiente manera “la colección (Nombre de la participante), contiene: ...”.

Tercera parte: Llevando las colecciones al código.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante revisará la sintaxis para definir y manipular arreglos en *Kotlin*. Primero se sugiere realizar una breve búsqueda en internet, que abarque tanto la creación de un arreglo como las funciones *get* y *set*.

2. Para observar y entender mejor cómo funcionan los arreglos en *Kotlin*, la participante realizará la siguiente secuencia de acciones, ya sea en un nuevo proyecto o en la consola de *Kotlin Playground*.

- Creará una variable “arreglo” que almacene el nombre de los meses del año. Apoyándose del método *arrayOf()* e indicando las cadenas con los nombres de cada mes, separados por comas, dentro de los paréntesis.
- Obtendrá el elemento en la posición 1 y lo imprimirá en pantalla, con ayuda de la función *get()* y *println()*. Recordar que la posición que se quiere obtener debe ir dentro de los paréntesis de la función *get()*.
- Posteriormente, responderá las siguientes preguntas ¿Qué mes obtuviste? ¿Es lo que esperabas obtener?
- Ahora cambiará la posición por la 0.
- Si el arreglo comienza en la posición 0, ¿en qué posición estará el último mes?
- Modificará el contenido de la localidad que almacene el mes de su cumpleaños por el mensaje “Mes de mi cumpleaños”, haciendo uso de la función *set* e imprimirá el nuevo valor.
- En los meses restantes ubicará eventos que considere importantes y reemplazará el contenido por un mensaje representativo para reconocer el evento. Por ejemplo: en diciembre el mensaje puede ser “Navidad”.
- Creará un nuevo arreglo con el nombre “dias” que almacenará, el día exacto de los eventos que reconoció en su otro arreglo. Pero esta vez usando el método *intArrayOf()*.
- Imprimirá el tamaño de sus 2 arreglos con la función *size*.
- Creará un último arreglo vacío, con ayuda del método *arrayOfNulls()*, será del mismo tamaño que los arreglos anteriores y almacenará datos de tipo *Double*. Recuerda que la sintaxis es la siguiente:

```
var arreglo = arrayOfNulls<tipo_de_dato>(tamaño)
```
- Verificará que el tamaño corresponda al que declaró, imprimiéndolo en pantalla.
- Responderá, ¿cuál crees que sea el contenido de cada localidad del arreglo?
- Por último, imprimirá cualquier elemento para observar su contenido, con ayuda de *get()*.

Cuarta parte: Observando los elementos del arreglo.

Sugerencia de tiempo invertido: 2 horas.

La participante aprenderá a recorrer y visualizar los elementos de un arreglo, haciendo uso de ciclos.

1. Reconocerá la necesidad de aplicar ciclos para recorrer e imprimir los elementos de un arreglo. En un nuevo proyecto o en la consola de *Kotlin Playground* creará un arreglo que almacenará el nombre de los días de la semana y lo imprimirá con el método *println()*. A partir de lo anterior, colocará sus observaciones en el documento “MMT2A10_evidencias.txt”, basándose en las siguientes cuestiones:
 - a. ¿Cuál es el resultado en pantalla?
 - b. ¿Se parece al contenido de tu arreglo?
 - c. ¿Qué crees que signifique?
 - d. ¿De qué manera entonces se podrían imprimir todos los elementos de un arreglo?
 - e. Si se debe imprimir elemento por elemento, ¿este proceso involucrará repetición?
 - f. ¿Se te ocurre alguna forma para facilitar el proceso?
2. Hará uso de ciclos de repetición para la impresión de un arreglo. En la carpeta de anexos se encuentra un archivo llamado *ciclos_arreglos.kt* dentro de *MMT1A10_anexos.zip* con el código que permite imprimir cada elemento del arreglo haciendo uso de ciclos.

La participante deberá adaptar el código del archivo para imprimir el arreglo con los días de la semana, aplicando el ciclo *for* y *while*. Después comentará sus observaciones en el documento de evidencias “MMT2A10_evidencias.txt” y contestará las siguientes preguntas:

 - a. ¿El resultado es el mismo utilizando ambos ciclos?
 - b. ¿Cuál te parece más sencillo de comprender?
 - c. Escribe con tus propias palabras el proceso que se sigue con el ciclo *for*.
 - d. Escribe con tus propias palabras el proceso que se sigue con el ciclo *while*.
3. La participante imprimirá los 3 arreglos realizados en la segunda parte de la actividad, en un nuevo archivo. La tallerista deberá verificar que el programa funcione correctamente y una vez aprobado, la participante adjuntará capturas de su código como evidencia.

Quinta parte: Trabajando con arreglos.

Sugerencia de tiempo invertido: 2 horas.

1. La participante integrará los conocimientos adquiridos en la actividad, a partir de la elaboración de un programa con las características enunciadas a continuación:
 - El programa deberá pedir la altura de 5 personas y las almacenará en un arreglo (hacer uso de ciclos de repetición).
 - Calculará el promedio de las alturas ingresadas.
 - Imprimirá la cantidad total de personas que son más altas que el promedio junto con sus alturas (deberá usar ciclos y condicionales).
 - Imprimirá la cantidad total de personas que son más bajas que el promedio junto con sus alturas (deberán usar ciclos y condicionales).
2. La tallerista verificará que se cumplan los puntos establecidos y de ser así, la participante podrá adjuntar las capturas y comentarios correspondientes a el documento “*MMT2A10_evidencias*”.

Links de apoyo:

Primera Parte: ¿Alguna vez has coleccionado objetos?

- Herrera, F. (2018). *Arreglos en programación*. [video]
<https://www.youtube.com/watch?v=xC-kySYWTd0&t=33s> (Julio, 2020).
- Marin, D.(2020). *Arreglos 1: ¿Qué es un arreglo?* [video]
<https://www.youtube.com/watch?v=nY5Hhofa0RM&t=274s> (Julio, 2020).

Actividad 11: Una llave para recuperar un elemento de valor

Aprendizaje esperado: Manejar una estructura llamada mapas para agrupar datos no importando el orden en el cual son ingresados o recuperados.		Duración de la actividad: 8 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Anexo <i>MMT2A11_Anexos.pdf</i> 	Producto: <ul style="list-style-type: none"> • Desarrollar un programa que pida el nombre de 5 países como clave y como valor el número de habitantes. <ol style="list-style-type: none"> a) Imprimir el listado completo (clave-valor). b) La cantidad de elementos del mapa. c) Verificar si un país en específico se encuentra en el mapa, si es así, imprimir el país con su respectivo número de habitantes, si no, imprimir que el país no se encuentra almacenado. d) Imprimir la cantidad total de habitantes de los 5 países. 	Evaluación: <ul style="list-style-type: none"> • Verificar el uso correcto de mapas. • Mostrar la importancia del uso de mapas y su diferencia con arreglos. • Demostrar que los datos impresos sean correctos en cada caso.
Desarrollo de la actividad:		
<p>Primera parte: Herramientas que necesito conocer.</p> <p>Sugerencia de tiempo invertido: 2 horas.</p> <p>Es importante que la participante adquiera o refuerce sus conocimientos acerca de las estructuras para agrupar datos. Como en todos los lenguajes de programación, en <i>Kotlin</i> se manejan sintaxis específicas para cada una de estas estructuras, por lo cual la participante deberá hacer una investigación que aborde los siguientes puntos:</p>		

- Rango en *Kotlin*. ¿Cómo se utilizan los operadores y las funciones para poder crear rangos?
- Colección *JavaMap* en *Kotlin*. ¿Cómo funciona la estructura de datos *Maps*?, ¿cómo se declaran?, ¿qué propiedades y funciones están disponibles en esta interfaz (*map*)? y ¿qué funciones son compatibles con *Maps* e implementan el uso de éste? Se deberá anexar las sintaxis de cada una.
- *Arrays* o arreglos y las listas. ¿Para qué se usan los arrays?, ¿cómo se define un *array* en *Kotlin*? y ¿qué métodos se usan para declarar arrays en *Kotlin*?
- Estructuras del tipo clave/valor. ¿Qué son?

Segunda parte: Aprendiendo conceptos.

Sugerencia de tiempo invertido: 1 hora.

1. A partir de la investigación anterior, la participante deberá identificar los siguientes conceptos:
 - *size*
 - *isEmpty()*
 - *containsKey(key: K)*
 - *containsValue(value: V)*
 - *get(key: K)*
 - *keys*
 - *values*
 - *mapOf()*
2. En la plantilla A (disponible en el anexo *MMT2A11_Anexos.pdf*) deberá relacionar los conceptos con los enunciados que expliquen cada uno.

Tercera parte: ¿Maps o Arrays?

Sugerencia de tiempo invertido: 3 horas.

1. La participante deberá dar solución a los siguientes programas utilizando ambas estructuras para agrupar datos (*Maps* y *Arrays*), es decir, realizará cada programa dos veces:

- Crear un programa de 10 posiciones de números con valores pedidos por teclado. Mostrar por consola el índice y el valor al que corresponde. Realizar dos métodos, uno para rellenar valores y otro para mostrar.
 - Crear un programa que permita almacenar 5 artículos, utilizar como clave el nombre de productos y como valor el precio del mismo. Desarrollar además las funciones de:
 1. Imprimir en forma completa el mapa.
 2. Mostrar la cantidad de artículos con precio superior a 20.
2. Una vez la participante haya resuelto los programas, deberá analizar qué estructura fue la correcta para cada caso y justificar su respuesta. Se espera que compare de forma práctica el uso de mapas con arrays e identifique los casos óptimos para el uso de cada uno.
 3. Finalmente, podrá discutir sus ideas con la tallerista para retroalimentar el aprendizaje obtenido.

Cuarta parte: Uso de Maps.

Sugerencia de tiempo invertido: 2 horas.

1. Se solicitará a la participante realizar un programa que use exclusivamente la estructura *Maps* y que realice lo siguiente:
 - En el bloque principal del programa definir un *Map immutable* que almacene los nombres de países como clave y como valor la cantidad de habitantes de dicho país, (deberán ser al menos 5 países).
2. El programa deberá:
 - Imprimir el listado completo (clave-valor).
 - Mostrar la cantidad de elementos del mapa.
 - Verificar si un país en específico se encuentra en el mapa, si es así, imprimir el país con su respectivo número de habitantes, en caso contrario, imprimir el país que no se encuentra almacenado.
 - Imprimir la cantidad total de habitantes de los 5 países.

Links de apoyo:

Maps:

- Kotlin v1.4.30.(Sin fecha). *Map*.
<https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-map/> (Agosto, 2020).

- tutorialesprogramacionya.com. (Sin fecha). *46-Colecciones-Map y MutableMap*.
<https://www.tutorialesprogramacionya.com/kotlinya/detalleconcepto.php?punto=46&codigo=46&inicio=45> (Agosto, 2020).
- Chike Mgbemena. (2017). *Kotlin Desde Cero: Rangos y Colecciones*.
<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-ranges-and-collections--cms-29397> (Agosto, 2020).
- Universidad Politécnica de Valencia (Sin fecha). *Colecciones en Kotlin: List, Set y Map*.
<http://www.androidcurso.com/index.php/99-kotlin/925-colecciones-en-kotlin-list-set-y-map> (Agosto, 2020).

Arrays:

- Cruz, A. (Sin fecha). *Los arrays y listas en Kotlin: Primeros pasos estas estructuras mutables e inmutables*.
<https://www.desarrollolibre.net/blog/android/los-arrays-y-listas-en-kotlin-primeros-pasos-estas-estructuras-mutables-e-inmutables#.Xz8jmS2b5N0> (Agosto, 2020).

Actividad 12: No llevar a la escuela una mochila sin útiles

Aprendizaje esperado: La participante establecerá que una de las características más sobresalientes de <i>Kotlin</i> es que no acepta valores nulos, a menos que se indique explícitamente.		Duración de la actividad: 3 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.	<p>Evidencias:</p> <p>1. Captura de pantallas de resultado de código.</p> <p>Producto:</p> <p>Crear un programa que haga lo siguiente:</p> <ol style="list-style-type: none">a) Generar un error de forma intencionada para ejemplificar la asignación de un valor nulo a una variable no anulable.b) Usar el operador <code>?</code> para que una variable tenga la posibilidad de aceptar nulos.c) Verificar con la sentencia <i>if</i> en tiempo de compilación si una variable es nula.d) Usar el operador <code>?.</code> para acceder de forma segura a un valor anulable sin recurrir al <i>if</i>.e) Usar el operador <code>?:</code> para proporcionar un valor predeterminado a una variable en lugar de un nulo.f) Usar el operador <code>!!</code> para tratar con valores de tipo anulable y afirmar que un tipo es, con total seguridad, no nulo.	<p>Retroalimentación:</p> <p>Verificar la ejecución de código en captura de pantallas.</p> <p>Evaluación:</p> <p>Mostrar la correcta ejecución del programa explicando el funcionamiento de cada operador en cada inciso.</p>

	g) Usar la función "let" para facilitar el manejo de expresiones anulables.	
--	---	--

Desarrollo de la actividad:

Primera parte: El error de mil millones de dólares.

Sugerencia de tiempo invertido: 1 hora.

La participante reconocerá el concepto *null* en la programación, comprenderá su función, modo de empleo así como problemas asociados a su utilización. Para lo cual deberá realizar una búsqueda en internet, basándose en las siguientes preguntas guía:

- ¿Alguna vez habías escuchado la palabra nulo?
- ¿A qué hace referencia?
- Investiga el significado de *null* en programación.
- Revisa el artículo *How Kotlin Solves the "Billion-Dollar Mistake"!* y contesta lo siguiente:
 - ¿Por qué la creación de la referencia *null* se considera un "error de mil millones de dólares"?
 - ¿Quién fue su creador?
 - ¿Qué es una excepción en programación?
 - ¿Cuál es la causa de un *NullPointerException*?
 - ¿Cómo deberíamos declarar una variable en *Kotlin* llamada "saludo" de tipo *String* si quisiéramos que admitiera valores nulos?
 - Verifica lo que ocurre cuando a una variable que no admite nulos le asignamos esa referencia.
 - ¿Consideras que es una ventaja conocer cuándo una variable puede ser nula y cuándo no?
 - ¿Qué recomendación nos da el artículo cuando tenemos variables que admiten la referencia *null*?

Segunda parte: ¡Kotlin al rescate!

Sugerencia de tiempo invertido: 2 horas.

La participante conocerá las diversas formas en que *Kotlin* permite trabajar las referencias nulas de forma segura. Para ello revisará los recursos propuestos o realizará una búsqueda en Internet con el fin de contestar y realizar lo siguiente:

Siempre que se mencione la ejecución de un código, se deberá tomar una captura de pantalla del resultado obtenido.

- ¿Cuáles son las posibles causas de un *NullPointerException* en *Kotlin*?
- ¿Cuál es la sintaxis de una llamada segura (*safe call*) en *Kotlin*?
- Ve a la página de la documentación *Null Safety* y ejecuta el ejemplo para llamadas seguras.
- Con base en tus observaciones, ¿cómo funcionan las llamadas seguras?
- En el siguiente ejemplo:

```
bob?.department?.head?.name
```

- ¿Qué ocurrirá si el departamento es nulo?
- ¿Consideras que es relevante en estos casos contar con las llamadas seguras?
- Observa el siguiente código que describe el funcionamiento del enunciado anterior en caso de no existir las llamadas seguras:

```
if(empleado != null)
    if(departamento != null)
        if(jefe != null)
            empleado.departamento.jefe.nombre
        else
            null
    else
        null
else
    null
```

- ¿Qué opción te parece más legible?
- Ubica el apartado de la documentación dentro de *Safe calls* en el que se localiza el siguiente código y ejecútalo:

```
val listWithNulls: List<String?> = listOf("Kotlin", null)
for (item in listWithNulls) {
```



```
        item?.let { println(it) } // prints Kotlin and ignores null
    }
}
```

- Ahora prueba estas 2 opciones de código en *Kotlin Playground*, después describe las similitudes y diferencias con el punto anterior (tanto a sintaxis, legibilidad como a resultados obtenidos):

- **Código 1**

```
fun main() {
    val listWithNulls: List<String?> = listOf("Kotlin", null)
    for (item in listWithNulls) {
        if(item != null)
            println(item)
    }
}
```

- **Código 2**

```
fun main() {
    val listWithNulls: List<String?> = listOf("Kotlin", null)
    for (item in listWithNulls) {
        println(item)
    }
}
```

- Con base en tus observaciones, ¿para qué sirve la función *let*?
- ¿Cuál es la función del operador *Elvis*?
- ¿Por qué se le llama operador *Elvis*?
- ¿Podría asociarse el funcionamiento del operador *Elvis* con una sentencia condicional?
- ¿Cuál es la diferencia entre el operador *Elvis* y las llamadas seguras?
- ¿Para qué sirve el operador *!!*?
- ¿Cuál será el resultado de ejecutar el siguiente código?

```
val valorNulo : String? = null
println(valorNulo!!.length)
```

- Explica por qué se obtiene esa salida.
- Por último, a manera de resumen, concentra tus respuestas en la siguiente tabla:

Nombre	Sintaxis	Funcionamiento	Ejemplo
Llamadas seguras(Safe calls)			
Operador Elvis			
Operador de aserción no nula (non-null assertion operator)			
Función let			

Tercera parte: Seamos como Kotlin.
Sugerencia de tiempo invertido: 30 minutos.

La participante identificará situaciones de la vida diaria en donde la característica de no reconocimiento de valores nulos de *Kotlin* podría ser aplicable, para ello se sugiere utilizar la secuencia que se presenta a continuación:

- Los estereotipos son una forma de estigmatizar a la gente, ya que usan generalizaciones que no respetan ni representan a una persona.

- Los roles de género, junto a los estereotipos de género pueden entenderse como el “[...] conjunto de ideas impuestas por la sociedad y la cultura acerca de lo que se considera el deber hacer y deber ser de cada sexo [...]”. (INMUJERES, 2018, p 173).
- Si el valor Nulo (estereotipo) de una Variable (persona) genera errores en el funcionamiento del Código (vida cotidiana), ¿cómo se puede ser como Kotlin y evitar los valores nulos?
- Completar la siguiente tabla haciendo uso de los ejemplos y experiencias personales.

Situación con estereotipo de género/valor nulo	Se expresa	Alternativa no sexista (característica <i>Kotlin</i>)
Se asume que los doctores son hombres y las mujeres enfermeras.	Los doctores y las enfermeras son especialistas en la salud, con diversas ramas de especialización.	El personal médico y de enfermería son especialistas en la salud, con diversas ramas de especialización.
Los juguetes como muñecas o bebés son para niñas mientras que, los coches y pelotas son juguetes de niños.	Las niñas juegan con muñecas y no pueden jugar con balones de fútbol porque ese es un juguete de niños.	Niñas y Niños juegan con los juguetes que les gustan para divertirse y aprender.
Hay más hombres programadores porque hacen mejor esta actividad que las mujeres.		
Las mujeres son mejores para el diseño gráfico.		
Los hombres son mejores con el manejo de números que las mujeres.		

Referencias:

- Guichard, Claudia. (2018). *Manual de comunicación no sexista, hacia un lenguaje incluyente*.
http://cedoc.inmujeres.gob.mx/documentos_download/101265.pdf (Agosto, 2020).

Links de apoyo:

- Cassingena, Estefania. (2018). *How Kotlin Solves the “Billion-Dollar Mistake”!*
<https://medium.com/techmacademy/how-kotlin-solves-the-billion-dollar-mistake-a6c0cb8dbc88> (Agosto, 2020).
- GeeksforGeeks. (2019). *Kotlin Null Safety*.
<https://www.geeksforgeeks.org/kotlin-null-safety/> (Agosto, 2020).
- Kotlin Doc Kotlin & Android. (Sin fecha). *Gestión de tipos nulos en Kotlin*.
<https://kotlinlang.org/docs/reference/null-safety.html> (Agosto, 2020).
- Kotlin v1.4.30. (Sin fecha). *Null Safety*.
<https://kotlinlang.org/docs/reference/null-safety.html> (Agosto, 2020).

Actividad 13: La reutilización es una buena práctica, ahorra tiempo y esfuerzo

Aprendizaje esperado: Crear funciones que permitan a la participante reutilizar código.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. 	<p>Evidencia:</p> <ol style="list-style-type: none"> 1. Llamar desde “main” las funciones, en cada caso imprimir el resultado. <p>Producto:</p> <ol style="list-style-type: none"> 1. Desarrollar un programa con cinco funciones: <ol style="list-style-type: none"> a. Presentación del programa y requerimiento de ingreso de dos enteros. b. Suma de los dos números. c. Resta de los dos números (el mayor menos el menor). d. Multiplicación de los dos números. e. División de los números (el mayor entre el menor). 	<p>Retroalimentación:</p> <ul style="list-style-type: none"> • Verificar que el resultado de cada función sea correcto. <p>Evaluación:</p> <ul style="list-style-type: none"> • Mostrar el uso de funciones de cada inciso.
Desarrollo de la actividad:		
<p>Primera parte: Introducción a las funciones. Sugerencia de tiempo invertido: 1 hora.</p> <ol style="list-style-type: none"> 1. En esta actividad se abordarán las funciones en <i>Kotlin</i>, también llamadas métodos en otros lenguajes. 2. La participante realizará una búsqueda en internet acerca del tema funciones en <i>Kotlin</i>, se recomienda revisar los <i>Links de apoyo</i>. 3. Posteriormente, deberá responder las siguientes preguntas: 		

- ¿Qué es una función?
 - ¿Para qué sirven?
 - ¿Cómo se llama a una función y su sintaxis general? Buscar un ejemplo y describir los pasos (el nombre de la función no puede tener espacios en blanco ni comenzar con un número).
 - ¿Qué es un parámetro de entrada?
 - ¿Qué es un parámetro de salida?
 - ¿Por qué en *Kotlin* siempre comienza en la función *main*?
4. Con base en la información obtenida, la participante realizará en un archivo de texto un escrito en el cual enuncie las ventajas que encuentra al utilizar funciones para desarrollar un programa.

Segunda parte: Más sobre funciones.

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá realizar los siguientes programas para practicar el uso de funciones:
- Un programa que reciba dos números enteros, los valide e indique cuál es el mayor.
 - Un programa que haga la suma de dos números dados por el usuario e imprima el resultado.
 - Un programa que haga la resta de dos números (el mayor menos el menor) dados por el usuario e imprima el resultado.
 - Un programa que haga la multiplicación de dos números dados por el usuario e imprima el resultado.
 - Un programa que haga la división (el mayor entre el menor) de dos números dados por el usuario e imprima el resultado.

Tercera parte: Reinventar la rueda.

Sugerencia de tiempo invertido: 2 horas.

1. Se le preguntará a la participante si alguna vez ha utilizado un código ya elaborado y si lo ha hecho en el código que está desarrollando.
2. Posteriormente, se le explicará que al programar reutilizar código es una estrategia válida, puesto que muchos programas ya se encuentran resueltos, por tal, pueden ser tomados como guía o utilizarse como base (siempre y cuando tengan lo

que se necesita) para no construirlos desde cero.

3. Luego de ello, se le solicitará que reflexione la siguiente pregunta, ¿por qué reinventar la rueda cuando puedes mejorar el mundo? y escriba en un archivo de texto las conclusiones a las que haya llegado.
4. Con base en lo anterior, la participante realizará una calculadora, deberá utilizar los códigos de la segunda parte de la actividad.
5. Desde el *main* se deberá llamar a todas las funciones y en cada caso imprimir el resultado; pero además deberá implementar lo siguiente en su programa:
 - Potencia: La lógica de esta operación nos dice que debemos multiplicar el número de veces del segundo parámetro al primer parámetro.
 - Factorial: es resultado de multiplicar a un número por todos los números enteros positivos menores a él.
6. La participante deberá probar su calculadora y asegurarse que esta funcione.
7. Deberá comentar con la tallerista y/o compañeras ¿qué opina acerca de reutilizar códigos o implementarlos para su uso personal?

Nota:

- Si existieran dudas respecto al álgebra o declaración de las funciones se recomienda consultar el enlace: *Calculadora científica en Kotlin*, el cual contiene ejemplos de códigos de los cuales puede auxiliarse si le son útiles.

Links de apoyo:

Funciones:

- tutorialesprogramacionya.com. (Sin fecha). *14 - Concepto de funciones*.
<https://www.tutorialesprogramacionya.com/kotlinya/detalleconcepto.php?punto=14&codigo=14&inicio=0> (Agosto 2020).
- cursokotlin.com.(2017).Curso Kotlin Para ANDROID. *Capítulo 6 – Funciones en Kotlin*.
<https://cursokotlin.com/capitulo-6-funciones-kotlin/> (Agosto 2020).
- tutorialesprogramacionya.com. (Sin fecha). *Kotlin Ya*.

<https://www.tutorialesprogramacionya.com/kotlinya/index.php?inicio=15> (Agosto 2020).

Calculadora en *Kotlin*:

- Palencia, N. (Sin fecha). *Calculadora científica en Kotlin*.
<https://devcode.la/tutoriales/calculadora-cientifica-en-kotlin/> (Agosto 2020).

Actividad 14: Elementos robustos de mi programa

Aprendizaje esperado: Usar las clases y objetos para crear programas más robustos con funcionalidades más complejas.		Duración de la actividad: 6 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.	Producto: Desarrollar un programa con una clase que cargue los lados de un triángulo y tenga los siguientes métodos: a) Inicializar las propiedades (ingreso de cada lado) b) Imprimir el valor del lado mayor. c) Mostrar si el triángulo es equilátero o no. Cargar datos de tres triángulos diferentes.	Evaluación: <ul style="list-style-type: none">• Mostrar la importancia del uso de clases, métodos y objetos.• Verificar que el funcionamiento y resultado de cada método sea correcto.
Desarrollo de la actividad:		
Primera parte: Otras formas de programar. Sugerencia de tiempo invertido: 2 horas. <ol style="list-style-type: none">1. La participante conocerá el concepto de paradigma de programación, para ello realizará una búsqueda en internet, apoyándose de los recursos propuestos y enunciará algunos ejemplos. Se recomienda utilizar los siguientes puntos para conducir la investigación:<ol style="list-style-type: none">a. ¿Qué es un paradigma de programación?b. ¿Por qué existen diversos paradigmas?c. ¿Con qué paradigmas has trabajado?d. ¿Cuántos paradigmas soporta un lenguaje de programación?e. Menciona 3 ejemplos de paradigmas de programación y describe brevemente en qué consisten.		

2. Una vez que la participante conozca el concepto de paradigma, reconocerá a detalle los conceptos relacionados con el paradigma orientado a objetos y su importancia, además identificará las diferencias entre la programación orientada a objetos y la programación estructurada.
3. Realizará una búsqueda en la web, para lo cual se recomienda el uso de las preguntas que se enuncian a continuación:
 - a. ¿En qué se diferencia la programación estructurada de la programación orientada a objetos?
 - b. ¿Por qué razones usamos el paradigma orientado a objetos, en lugar del estructurado?
 - c. ¿Qué es una clase?
 - d. ¿Qué es un objeto?
 - e. ¿Qué es un atributo?
 - f. ¿Qué es un método?
 - g. ¿Qué es una abstracción?
 - h. Pensará en 3 objetos del mundo real y realizará la abstracción de ellos, tendrá que identificar al menos 3 atributos y 3 métodos de cada uno. Posteriormente, colocará la información identificada en la siguiente tabla:

Objeto	Atributos	Métodos

- i. Pide a la tallerista o a otras participantes que realicen la abstracción de uno de los objetos de tu tabla, ¿las respuestas fueron similares?
- j. Por último, realiza la abstracción de un usuario de *Twitter* y compara tus respuestas al igual que en el punto anterior.
- k. ¿Cambiaría mucho la abstracción si en su lugar modeláramos un usuario de *Facebook*?
Pese a que podemos abstraer diferentes características para un mismo objeto, dependiendo del contexto unas serán más relevantes que otras.

Segunda parte: Clases y objetos en Kotlin.

Sugerencia de tiempo invertido: 3 horas.

La participante identificará la sintaxis utilizada por *Kotlin* para la declaración de clases y objetos. Para abordar estos conceptos se sugiere revisar los recursos proporcionados en los *Links de apoyo* y posteriormente, realizar lo siguiente:

- Elegir 2 de los objetos modelados en la primera parte de la actividad y crear su clase correspondiente en *Kotlin*, es importante que todos los atributos sean inicializados desde el constructor principal y la clase sea lo más concisa posible.

La declaración de las clases se deberá realizar en archivos distintos del proyecto.

- Luego de ello, deberá instanciar un objeto de cada clase, imprimirá sus atributos y llamará al menos uno de sus métodos.

La participante mostrará su código en funcionamiento a la tallerista, y en caso de que cumpla con los requisitos establecidos, adjuntará como evidencia los archivos correspondientes a sus clases.

Tercera parte: Mi primer programa utilizando objetos.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Al resolver el siguiente problema la participante pondrá en práctica los conocimientos adquiridos a lo largo de la actividad. Deberá elaborar un programa en el que incorpore los elementos de la programación orientada a objetos y que resuelva el enunciado descrito a continuación:

- Una cadena de pizzas desea mejorar el control de calidad de sus productos, para ello requiere que todas las rebanadas de sus pizzas sean exactamente triángulos equiláteros.

Genera una clase triángulo, que tenga atributos por cada uno de los lados. Posteriormente define 2 métodos, uno para imprimir los 3 lados del triángulo y otro que determine si un triángulo es o no equilátero.

En la función principal, crea 3 objetos que simulen las rebanadas y determina si pasaron el control de calidad o no (imprimiendo un mensaje).

La tallerista revisará el correcto funcionamiento del programa y una vez que cumpla con todas las características, la participante adjuntará los archivos creados y modificados como evidencia.

Links de apoyo:

- EDteam. (Sin fecha). *¿Qué son los paradigmas de programación?* [video]
<https://www.youtube.com/watch?v=hcuvB58hwIE> (Agosto, 2020).
 - (2019). *¿Qué es la programación orientada a objetos?* [video]
https://www.youtube.com/watch?v=DlphYPc_HKk&t=152s (Agosto, 2020).
- Absolute. (2020). *La Lógica de la Programación Orientada a Objetos explicada por Minecraft.* [video]
<https://www.youtube.com/watch?v=l848HdWjLMo> (Agosto, 2020).
- Kotlin v1.4.31 (Sin fecha). *Inheritance.*
<https://kotlinlang.org/docs/reference/classes.html> (Agosto, 2020).
- Chike Mgbemena. (2017). *Kotlin desde cero: Clases y objetos.*
<https://code.tutsplus.com/es/tutorials/kotlin-from-scratch-classes-and-objects--cms-29590> (Agosto, 2020).

Actividad 15: Uniendo cada parte para la creación de mi aplicación personalizada.

Aprendizaje esperado: Crear una aplicación que funcione como una agenda la cual incorpore elementos vistos de Kotlin en las actividades previas.		Duración de la actividad: 10 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.• Dispositivo móvil.• Anexo <i>MMT2A15_Anexo.pdf</i>	Producto: 1. Desarrollar una aplicación que funcione como una agenda teniendo como operaciones mínimas las siguientes: <ul style="list-style-type: none">a. Añadir contacto.b. Borrar contacto.c. Modificar contacto.d. Visualizar (completa, o un solo contacto).	Evaluación: <ul style="list-style-type: none">• Demostrar el buen funcionamiento de la aplicación en el dispositivo móvil, haciendo uso de todas las herramientas que se utilizaron en las actividades anteriores.• Verificar que el tipo de datos sea el correcto en cada campo de un contacto.• Verificar que la aplicación sea robusta y sólida, es decir, que sea resistente a errores del usuario.• El programa anterior debe ejecutarse en un dispositivo móvil.
Desarrollo de la actividad:		
Primera parte: La construcción del lenguaje. Sugerencia de tiempo invertido: 1 hora. 1. Leerá a la participante el siguiente texto (también disponible en el anexo <i>MMT2A15_Anexo.pdf</i>): Un lenguaje es, según la RAE, un “Conjunto de signos y reglas que permite la comunicación” ya sea entre las personas o entre las computadoras, el lenguaje se construye para interpretar el mundo real o abstracto y permitir la comunicación de ideas, órdenes o reglas. Así como el lenguaje que se utiliza en programación, el lenguaje incluyente y no sexista es una construcción humana que ha surgido por la necesidad de nombrar lo que antes no era nombrado.		

En programación existen diversas alternativas de lenguajes para utilizar en casos específicos; lo mismo sucede en la vida cotidiana, el lenguaje incluyente y no sexista surge como una alternativa al lenguaje cotidiano (que muchas veces resulta ser discriminatorio) y ayuda a visibilizar a las personas que anteriormente no se tomaban en cuenta; las mujeres, las personas con discapacidad, las personas de la diversidad sexual, a los grupos indígenas, a las personas migrantes y a cualquier persona que se ha visto invisibilizada por el lenguaje no incluyente y sexista.

El lenguaje es cambiante y también se aprende, así como los lenguajes de programación se desarrollan y/o actualizan.

2. Al finalizar, se le solicitará a la participante que en un archivo de texto conteste de manera breve la pregunta: ¿Qué relación encuentras entre aprender un lenguaje de programación y aprender a utilizar el lenguaje incluyente y no sexista?

Segunda parte: El diseño va primero.

Sugerencia de tiempo invertido: 2 horas.

1. La participante creará un nuevo proyecto en *Android Studio* o en *Kotlin Playground*.
2. Usará la plantilla *Empty Activity* y nombrará la app como “Mi Agenda personal”.
3. Deberá realizar el diseño de la aplicación considerando lo siguiente:
 - De la *paleta* deberá usar tres *edit text*, el primero del tipo *Plain text* que se usará para agregar el nombre del contacto, el segundo de tipo *number* para agregar el número y el tercero de tipo *multiline* que permitirá agregar múltiples líneas de texto. Los tres componentes se arrastrarán a la pantalla principal.
 - Se le deberá asignar el correspondiente *ID* a cada *edit text*, al primero (el que contendrá el nombre) se le asignará “txt_nombre”, al segundo (donde se agregaran los datos del contacto) se le asignará “txt_datos” y al tercero se le asignará “txt_ocupación”.
4. En la sección de *widgets* se agregarán cuatro botones a la pantalla, que se utilizaran para guardar, buscar, modificar y eliminar contacto.
5. En la vista *Blue print* deberá establecerse la distancia entre los controles y el *activity*.
6. En la vista diseño se modificarán las medidas de los controles, en la sección de *atributos*, la participante deberá tomar en cuenta las dimensiones de la pantalla para el tamaño de aquéllos.
7. Se deberá añadir un nuevo atributo “*TextStyle*” al *edit text* que corresponde al nombre de contacto y se seleccionará *bold* o *negritas* para que el texto aparezca en diferente formato.

Tercera parte: Funcionamiento interno.

Sugerencia de tiempo invertido: 3 horas.

1. Ya que el diseño de la app haya quedado listo, será momento de hacer funcionar los botones y componentes para la agenda.
2. La participante escribirá su código en *MainActivity*.
3. Creará tres objetos del tipo *private* que sean *EditText*: *et_nombre*, *et_datos*, *et_numero*.
4. Dentro del método *Override* se creará la relación de la parte gráfica con la parte lógica, haciendo el *casting* entre los *objetos* y los *plain text*, por ejemplo:

```
et_nombre = (EditText) findViewById(R.id.txt_nombre)
```

5. Se deberá hacer lo mismo con cada objeto creado de acuerdo a los nombres asignados.
6. Para hacer funcionar los botones, se deberá trabajar con métodos, la participante colocará a modo de comentario el nombre del método que esté declarando, para que queden separados y pueda identificarse cuál es cuál en caso de algún error.
7. Para comenzar con el botón “guardar”, se iniciará con *public void* y se agregará el objeto *view* y el nombre del objeto.
8. Para este botón se necesitará crear dos variables que permitirán almacenar el nombre del contacto así como los datos que se agreguen del mismo.
 - La primera, un variable de tipo *string*, la cual será el “nombre” y será igual a *et_nombre.getText().toString()*;
 - La segunda, la variable “datos” que recuperará la información de *et_datos*.
9. Deberá crear un objeto de la clase *SharePreference* cuyo nombre será “preferencias”, será igual al método *GetSharePreference*. Por defecto, pedirá asignar un nombre al archivo el cual será “Agenda” *context.modePrivate*.
10. Deberá crear otro objeto que permita editar el archivo, para ello usará el método *edit*.
11. Con un *obj_editor* y el método *putString* deberá delimitar los parámetros para poder buscar un contacto a partir de su nombre.
12. Usará un *Toast* que indique al usuario que añadió su contacto con éxito a partir del siguiente mensaje “El contacto ha sido registrado”. Deberá recordar usar el método *SHOW* para que se muestre.
13. Para el botón “Buscar contacto” se creará el método “buscar” iniciará con *public void* y deberá agregar el objeto *view* y el nombre del objeto.
14. Es necesario crear una variable del tipo *string*. Usar el método *GetText* y *ToString* para recuperar el nombre del contacto que desea buscar.

15. Se creará un objeto de la clase *SharePreference* y un objeto “preferencias”, con ayuda del método *GetSharePreference* y el nombre del archivo con el se está trabajando “Agenda” *context. modePrivate*.
16. Deberá crearse una nueva variable tipo *string*, que almacenará los datos de contacto que se mostrarán al usuario al realizar una búsqueda en la agenda.
17. Se usará de nuevo el objeto *preferencias* y con ayuda del método *.GetString* se solicitarán dos parámetros para realizar la “búsqueda”, el primero será el nombre del contacto que es la variable asignada como “nombre”, el segundo parámetro serán los datos del contacto.
18. Para el botón “Eliminar contacto” deberá crear el método “borrar” el cual eliminará la cadena de información relacionada a ese contacto.
19. Para el botón “modificar” se deberá crear el método “editar” mismo que permitirá al usuario modificar el número o información del contacto.
20. Con la estructura condicional *If else* se deberá indicar si el nombre de contacto está guardado o no existe, si lo encontró deberá mostrar los datos y si no deberá enviar un mensaje al usuario de que el contacto no está registrado.
21. La participante podrá usar el método *.Length* para confirmar si la cadena de texto es igual a cero (caso en el que no encontró nada) y por consecuencia se deberá mandar un *toast* que muestre lo siguiente en pantalla “No se encontró ningún registro del contacto” seguido del método *show*.
22. En caso de que el programa encuentre algún valor, se usará el método *et_datos* y con ayuda del método *.setText* más el nombre “datos” se realizará la búsqueda de una cadena que corresponda .
23. La participante deberá regresar a la parte gráfica de la aplicación para relacionar cada uno de los botones con su respectivo método, en la sección de atributos, deberá seleccionar *onclick* y ahí elegir el método que corresponde al botón, repitiendo lo mismo con todos.

Cuarta parte: Añadiendo contactos.

Sugerencia de tiempo invertido: 1 hora.

1. La participante deberá probar su aplicación en el emulador o descargarla directamente en su teléfono.
2. Añadirá a sus contactos mujeres de confianza; deberá escribir el nombre en la primera casilla, su número en la segunda y en la casilla “datos” la ocupación, oficio, profesión o cargo de la persona.
3. En la tercera casilla se solicitará que la participante utilice el lenguaje incluyente y no sexista para referirse a la ocupación, oficio, profesión o cargo de la persona. Ejemplos:

Nombre	Teléfono	Ocupación, Oficio, Profesión/Cargo
María Elena Ruiz	0000000000	Chef
Beatriz Muñoz	0000000000	Ingeniera
Patricia Sepúlveda	0000000000	Médica
Leticia Vargas	0000000000	Trabajadora del hogar
Eva García	0000000000	Directora

- Deberá guardar todos los contactos, podrá editarlos o eliminarlos si así lo requiere con ayuda de los botones.

Quinta parte: Ajustando los engranes.

Sugerencia de tiempo invertido: 1 hora.

- La participante se asegurará que su app sea resistente a los errores de usuario, se sugieren revisar la aplicación y responder las siguientes preguntas:
 - ¿Qué pasa si sólo agregan un número sin nombre o un nombre sin número?
 - ¿Qué sucede si en la casilla de “nombre” agregan números, se permitirá guardar el contacto?
 - ¿Qué sucede si en la casilla de “número” agregan letras, se permitirá guardar el contacto?
 - ¿Algún campo acepta caracteres especiales? ¿Qué sucede si el usuario decide registrar alguno?
- Si la participante encontrara fallos al revisar los puntos anteriores se sugerirá revisar de nuevo la programación de la app para corregir el fallo.

Sexta parte: Celebración por mi nueva app.

Sugerencia de tiempo invertido: 2 horas.

1. Terminar una aplicación representa un gran logro para la participante y la tallerista, ya que se invirtió tiempo, esfuerzo y compromiso para aprender los conceptos y habilidades necesarios para desarrollar un proyecto de este tipo; por lo tanto se sugiere realizar una actividad de celebración donde la participante pueda invitar a personas de su elección (amigas, compañeras y/o familiares) a celebrar con ella que ha completado su proyecto.
2. Se sugiere aprovechar la celebración para presentar la aplicación que se desarrolló y otorgar el reconocimiento que la participante merece como creadora de una aplicación funcional.

Nota

- En la segunda parte de la actividad se emplea un diseño básico de la app, por ende, si la participante desea hacer un diseño más específico modificando colores o agregando iconos a cada botón, es libre de hacerlo.

Taller 3: Mis primeras aplicaciones con Kotlin

En honor la **Dra. Hanna Oktaba** quien obtuvo su doctorado en Informática por la Universidad de Varsovia, Polonia. Es profesora del área de Computación de la Facultad de Ciencias y del Posgrado en Ciencia e Ingeniería en Computación de la UNAM. Experta en el área de Ingeniería de Software, fundadora de la Asociación Mexicana para la Calidad en Ingeniería de Software (AMCIS) y de la Sociedad Mexicana en Ciencia de la Computación (SMCC). Estuvo a cargo del proyecto que culminó en la norma mexicana MNX-I-059-NYCE dirigido a las pequeñas organizaciones de software. Representó a México en el WG 24 de *ISO JTC1/SC7 Software and System Engineering* en la creación del estándar ISO/IEC 29110 para *Very Small Entities (VSEs)* y participó en la propuesta ESSENCE 1.0 que fue publicada como estándar de Organization Management Guide. Autora de múltiples artículos de investigación.

Competencia esperada:

La participante podrá desarrollar aplicaciones móviles en el *IDE* (Entorno de Desarrollo Integrado) *Android Studio* con el lenguaje de programación *Kotlin*, integrando conocimientos de los talleres uno y dos de este mismo módulo.

Actitudes: Curiosidad, disposición, constancia, persistencia, apertura a la incorporación de nuevos aprendizajes, capacidad de análisis, apertura al diálogo, escucha, trabajo en equipo, intercambio de opiniones, participación activa, interés y apertura a incorporar en las actividades la Perspectiva de Género para el logro de un bien común.

Actividad 1: Un paseo por el código de mi aplicación

Aprendizaje esperado: Conocer el entorno de desarrollo integrado (IDE) de Android Studio		Duración de la actividad: 3 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia <ol style="list-style-type: none"> 1. Creación de un nuevo proyecto. 2. Identificación de los elementos importantes de los archivos <i>MainActivity</i> y <i>activity_main</i>, localización, estructura. 3. Función del panel de <i>Debug</i> y <i>Errores</i>. 4. Configuración del Emulador. 	Retroalimentación <ul style="list-style-type: none"> • Verificar que la participante navegue entre los elementos importantes al generar un nuevo proyecto para el desarrollo de una aplicación. • Verificar que la vista de diseño sea visible en el emulador o el teléfono móvil.
Desarrollo de la actividad:		
<p>Primera parte: Hola, soy IDE Android Studio...</p> <p>Sugerencia de tiempo invertido: 1 hora y 30 minutos.</p> <ol style="list-style-type: none"> 1. La participante revisará en la documentación oficial, la descripción general del <i>IDE Android Studio</i> para conocer las características y ventajas que incorpora (revisar recurso en la sección de <i>Links de apoyo</i>). 2. Navegará entre las carpetas, archivos y ventanas disponibles en el <i>IDE</i> al crear un nuevo proyecto e identificará la secuencia que sigue el entorno de trabajo. 3. Generará las instrucciones necesarias para acceder a cada una de las siguientes secciones: <p>Archivos:</p> <ul style="list-style-type: none"> • <i>MainActivity.kt</i> • <i>activity_main.xml</i> • <i>Android_Manifest.xml</i> • <i>build.gradle</i> (aplicación y proyecto) 		

Ventanas de herramientas:

- *Logcat*
- *Proyecto*
- *TODO*
- *Resource Manager*
- *Debug*

Segunda parte: Recorriendo atajos.

Sugerencia de tiempo invertido: 30 minutos.

Los comandos facilitan la comunicación con un ordenador, simplificando el proceso para indicar que se ejecute una acción. La participante explorará las distintas combinaciones de teclas y su funcionamiento (comandos) e identificará cinco que faciliten su actividad como programadora, pueden retomar aquellos referentes a las ventanas de herramientas vistas en la primera parte de la actividad.

Se sugiere utilizar el recurso provisto para esta parte de la actividad (disponible en los *Links de apoyo*).

Sección	Descripción	Comando Windows/Linux
General		
Navegación y Búsqueda dentro de Studio		
Visualización de diseños		
Herramientas de diseño: editor de diseño		
Herramientas de diseño: editor de navegación		

Tercera parte: Separando vista y diseño.

Sugerencia de tiempo invertido: 30 minutos.

La participante distinguirá la relación entre los archivos *MainActivity.kt* y *activity_main.xml*. Identificará su estructura, componentes y función, para ello abrirá un proyecto en *Android Studio* y completará los siguientes pasos:

1. Investigará el concepto de *activity* en *Android*.
2. Identificará el archivo perteneciente al diseño y el archivo perteneciente al funcionamiento.

MainActivity.kt

- ¿Cuál es el nombre de la clase?
- ¿Posee constructor principal?
- ¿Cuál es el nombre del único método definido?
- ¿Cuál es el parámetro de la función *setContentView*?
- ¿A qué crees que hace referencia este parámetro?

activity_main.xml

- ¿Qué es *XML*?
- Visualizará el archivo *activity_main.xml* en la ventana de editor.
- Ubicará en la parte superior derecha los botones *Code*, *Split* y *Design*, selecciona cada uno de ellos y describe sus diferencias.
- En la sección *Code* ubica la línea *tools:context*. ¿A qué hace referencia su valor?

Relación entre los archivos

- A partir de lo observado hasta este punto, responderá: ¿Consideras que existe una relación entre los dos archivos descritos?
- Modificará el texto que se muestra en *activity_main.xml* por "Hola tu_nombre".

- Probará la actividad en el emulador o en un dispositivo físico.
- Creará un nuevo recurso de tipo *layout* con el nombre *vista.xml* y colocará como texto “¡Hola mundo! desde otro layout”.
- Modificará el parámetro *setContentView* de *MainActivity.kt* por *R.layout.vista*
- Volverá a cargar la actividad en el emulador y comentará sus observaciones.
- Responderá: ¿Qué tan estrecha consideras que es la relación entre los 2 archivos?

Cuarta parte: Con las herramientas en el cinturón, empecemos.

Sugerencia de tiempo invertido: 1 hora.

1. Una vez que la participante identifique las herramientas ofrecidas por *Android Studio* para facilitar la generación de código, practicará dónde utilizarlas, para ello, revisará el enlace *Conceptos básicos del flujo de trabajo* (disponible en los *Links de apoyo*), correspondiente a la documentación oficial.
2. Después, identificará los siguientes vínculos:
 - Cómo depurar tu app.
 - Cómo compilar y ejecutar una app.
 - Cómo escribir tu app.
3. En cada uno revisará la *Descripción general* e identificará las Ventanas de herramientas o secciones del *IDE* utilizadas para completar el siguiente formato:

Paso del flujo de trabajo	Ventana o sección que utiliza	Atajo (en caso de existir)
Escribir app		
Compilar y ejecutar app		
Depurar app		

Nota:

Se sugiere mencionar a la participante la importancia y utilidad de los *Links de apoyo* (sobre todo los provistos para la cuarta parte de la actividad) tanto para otras actividades del taller como en sus futuras funciones como programadora.

Links de apoyo:

- developers. (Sin fecha). *Combinaciones de teclas*.
<https://developer.android.com/studio/intro/keyboard-shortcuts> (Agosto, 2020).
 - *Introducción a Android Studio*.
<https://developer.android.com/studio/intro> (Agosto, 2020).
- resources.jetbrains.com. (Sin fecha). *IntelliJ IDEA*.
https://resources.jetbrains.com/storage/products/intellij-idea/docs/IntelliJIDEA_ReferenceCard.pdf (Agosto, 2020).
- developers (Sin fecha). *Conceptos básicos del flujo de trabajo para desarrolladores*.
<https://developer.android.com/studio/workflow> (Agosto, 2020).

Actividad 2: Diseñando mi primera App

Aprendizaje esperado: Manipular los elementos necesarios básicos mediante código para establecer el diseño de la pantalla de una aplicación.		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia: <ol style="list-style-type: none"> 1. Diseño de los elementos gráficos de la aplicación. 2. Manipulación y cambio de dimensiones y parámetros de los elementos del diseño de la aplicación. 	Retroalimentación: <ul style="list-style-type: none"> • Verificar que al hacer cambios en los parámetros de los elementos de la aplicación estos se vean reflejados correctamente en la aplicación.
Desarrollo de la actividad:		
<p>Primera parte: Conocimientos del diseño.</p> <p>Sugerencia de tiempo invertido: 2 horas.</p> <ol style="list-style-type: none"> 1. La participante deberá de leer el artículo <i>Estructura de un proyecto en Android Studio</i> (disponible en los <i>Links de apoyo</i>). 2. Posteriormente, deberá realizar una búsqueda complementaria en internet que responda las siguientes preguntas: <ul style="list-style-type: none"> • ¿Qué es el diseño de estilo <i>Drag and Drop</i>? • ¿Qué es un <i>Layout</i>? • ¿Cómo se define un <i>Layout</i>? • ¿Qué tipos de <i>Layout</i> existen y para qué se utiliza cada uno? • ¿Qué es <i>RelativeLayout</i> y para qué se usa? • ¿Qué es <i>View</i>? • ¿Qué es <i>ViewGroup</i>? • ¿Cuáles son los atributos de view? • ¿Qué es un <i>EditText</i> y para qué sirve? • ¿Qué es un <i>Button</i> y para qué sirve? 		

- ¿Qué es un *medium Text* y para qué sirve?
 - ¿Qué es un *checkbox* y para qué sirve?
 - ¿Qué tipos de botones tiene *Android Studio* y para qué sirven? (*Radio button*, *toggle button*).
 - ¿Qué es *Constraint Layout* y para qué se utiliza?
 - ¿Cómo se personalizan los elementos?
 - ¿Qué es el archivo *r.java*?
 - ¿Cuál es el propósito del archivo *r.java*?
3. Se deberá considerar que estos conocimientos estarán enfocados en su uso en *Android Studio*.
 4. La información obtenida deberá redactarse en un archivo de texto que en la posteridad le servirá como un “manual”.
 5. La tallerista deberá revisar el manual y corregir o complementar la información obtenida por la participante.

Segunda parte: Conociendo el ambiente.

Sugerencia de tiempo invertido: 2 horas.

Android Studio utiliza el Diseño de interfaz de usuario declarativo, esta característica ubica el diseño de interfaz de usuario en un alto nivel, evitando la escritura de código abrumador. Se usa el estilo declarativo para crear la forma de la interfaz y luego usar el estilo programático para implementar los eventos sobre la interfaz. La combinación de ambas formas desata un poder de flexibilidad increíble.

1. Es importante que la participante conozca todos los componentes que comprenden a *Android Studio*.
2. Se creará una pequeña aplicación con un *Relative Layout* y los siguientes tres *views*: un *EditText*, un *TextView*, y un *Button*.
3. El objetivo general es que el usuario digite algún escrito en la caja de texto y al presionar el botón la aplicación muestre la cantidad de caracteres que tiene, por lo cual deberá crear un nuevo proyecto y realizar las siguientes acciones:
 - a. Abrir el archivo *Layout*. Abrir el archivo de recursos que guarde el *layout* de la actividad principal.
 - b. Arrastrar y Soltar *Views*. Presionar la pestaña *Design* en la parte inferior del editor, la herramienta que se visualiza es el panel de diseño de interfaz de *Android Studio*.
 - En la parte central deberá visualizarse un lienzo que muestre la interfaz actual en un dispositivo.
 - En la parte superior se tendrá una barra que permite configurar el dispositivo elegido para la previsualización, herramientas de *zoom*, la versión de *Android* a visualizar, el modo táctil (*Landscape* o *Portrait*), etc.
 - A la izquierda se verá la paleta o estante de *views* para construir la actividad.

- Al lado derecho se encontrará el *Component Tree* o árbol de componentes, esta ventana se representa en forma jerárquica la relación de los *views*.
 - Debajo del *Component Tree* se encontrará el panel de propiedades. En él se verán todos los atributos que tiene un *view* seleccionado, los cuales se podrán editar con facilidad.
- c. A continuación arrastrar al *layout* el *Multiline Text* de la sección *Text Fields* y ubicarlo en la parte superior de forma centrada y horizontal.
 - d. Agregar un botón, éste se encuentra en la sección de *widgets*.
 - e. Por último, agregar un *medium text* que se encuentra de igual forma en la sección *widgets*.

Tercera parte: Configurar atributos de los *views*.

Sugerencia de tiempo invertido: 1 hora.

1. Se asignarán los *IDs*, las *dimensiones* y las *posiciones* de los tres componentes en el *layout*.
2. Para ello la participante seleccionará primero el *textView* en el *Component Tree*. Luego se dirigirá a sus propiedades, buscará el *atributo id* y empleará un nombre significativo para el componente, por ejemplo “main_textview”.
3. Deberá hacer lo mismo para los demás *views* y visualizará cómo cambian sus nombres en el *Component Tree*.
4. Confirmará que el *editText* configurado inicialmente tenga su límite superior alineado al límite superior del *Relative Layout*. Esto se podrá visualizar si se revisan las propiedades del atributo *layout:alignParent* seleccionado en la opción «top».
5. Visualizará que al seleccionar el botón esté ubicado en la parte inferior del *editText*, con una margen superior de *41dp* y alineado horizontalmente. Estas características se representan con los atributos *layout:margin*, *layout:alignComponent* y *layout:centerInParent*.
6. La participante deberá asignarle a su límite superior, el límite inferior del botón y además añadirle un margen superior de *41dp*.
7. Deberá modificar un atributo especial llamado *hint* del *editText*. *Hint* indicará a través de un texto informativo, el tipo de texto que el *edittext* puede recibir. En este caso se deberá poner «Mensaje».
8. Modificará los atributos *text* del *textView* y el botón, asignará una cadena vacía al primero y la cadena «Procesar» al segundo.

Cuarta parte: Añadir las cadenas relevantes al archivo *strings.xml*.

Sugerencia de tiempo invertido: 1 hora.

1. La participante deberá identificar las cadenas que son estáticas y que no se modificarán a lo largo de la ejecución de la aplicación. Estas deberán ser *texto del botón* al igual que el *Hint del editText*. El texto del *textView* está destinado a ser dinámico así que quedaría descartado.
2. Definido lo anterior, se deberán agregar estas cadenas al archivo *strings.xml*. Para ello se buscará en el panel de propiedades el atributo *text* del botón y *hint* del *edittext* y se presionará el botón «...».
3. Se desplegará una ventana que permitirá elegir y navegar a través de todos los recursos del proyecto en la pestaña «Project».
4. Ya que el recurso que se necesita usar aún no existe, se deberá crear. Para ello hará clic en «New Resource» y seleccionará «New String Value...».
5. A continuación aparecerá un formulario con dos campos que describen al *string*, *Resource name* y *Resource content*. El primero hace referencia al nombre del recurso, el segundo al contenido del *string*. En este caso como nombre se le asignará *main_button* al texto del botón y para el *hint* del *edittext* se usará «*main_hint*».
6. Se deberá guardar y así el recurso estará creado. Si se desea, podrá abrir el archivo *strings.xml* y comprobar la existencia de los nodos creados.

Quinta parte: Programar los eventos.

Sugerencia de tiempo invertido: 1 hora.

1. Con las cuatro primeras partes de esta actividad la participante habrá acabado con toda la parte de diseño e interfaz. Lo que seguirá será programar lo que se quiere que la aplicación realice si el botón es presionado.
2. El algoritmo a seguir será:
 ¿Botón presionado?:
 SI
 -Obtener valor del *editText*
 -Poner valor del *editText* en el *textView*
3. La participante deberá usar una Interfaz que escuche los taps del usuario en un *view*. (Los *listeners* o escuchas en español, son Interfaces implementadas en *Java* para participar en las recepciones de eventos).

4. Para los botones se usará una escucha especial llamada *OnClickListener*. Solo bastará con añadir una relación entre el botón y la escucha para reportar el evento.
5. Se deberá crear un objeto del tipo *Button* para guardar la instancia del botón.
6. Dentro del método *callback onCreate()* se llamará al método de la clase *Activity findViewById()*. Este método retornará en los *views* que se encuentran en el *layout* a través de una referencia. Puesto que devuelve un objeto de tipo *View*, será necesario realizar el *casting* pertinente.
7. La referencia del botón se obtendrá a través del atributo *id* de la *clase R*.
8. Con este proceso, lo siguiente será modificar de forma análoga los otros dos *views*:
9. Deberá anidar una nueva instancia de la escucha al botón con el método de la clase *Button setOnClickListener()*.
10. *OnClickListener* permite sobrescribir su método de escucha de taps *onClick()*. Dentro de él se agregarán las acciones que se ejecutarán al presionar el botón.
11. Se deberá hacer transferencia del mensaje desde el *editText* hasta el *textView*. Para ello se usarán los métodos *getters* y *setters* del atributo *text* que ambos poseen. El método *getText()* de *EditText* no retorna en la cadena como tal del componente, si no en un objeto del tipo *Editable*, por ello luego se deberá castear a *String* con *toString()*.
12. Finalmente, ejecutará el proyecto en *Android Studio*.
13. La participante deberá revisar que los componentes se mantengan de acuerdo a los parámetros que se establecieron, que el funcionamiento de la aplicación sea el programado y que no cuente con errores.

Nota:

Todo lo realizado en la tercera parte de la actividad se puede especificar a través de la *pestaña Text* del Editor. Esta pestaña muestra la descripción *XML* sobre lo construido en la pestaña de Diseño. Dicho archivo muestra los componentes de cada *View* y los atributos que le han asignado (también hay otros atributos configurados intrínsecamente por *Android Studio*).

Links de apoyo:

Estructura de un proyecto en Android Studio:

- develou.com. (Sin fecha) *Estructura De Un Proyecto En Android Studio*.
<http://www.hermosaprogramacion.com/2014/08/android-studio-proyecto-en/#concepto-layout> (Septiembre, 2020).

Constraint Layout:

- developers (Sin fecha) *Cómo brindar compatibilidad con diferentes tamaños de pantalla.*
<https://developer.android.com/training/multiscreen/screensizes?hl=es-419> (Septiembre, 2020).

Componentes de vistas personalizadas:

- developers (Sin fecha) *Componentes de vistas personalizadas.*
<https://developer.android.com/guide/topics/ui/custom-components?hl=es> (Septiembre, 2020).

Botones:

- developers (Sin fecha) *Radio Buttons.*
<https://developer.android.com/guide/topics/ui/controls/radiobutton> (Septiembre, 2020).

Archivo r.java:

- develou.com (Sin fecha) *Estructura De Un Proyecto En Android Studio.*
<http://www.hermosaprogramacion.com/2014/08/android-studio-proyecto-en/#concepto-r> (Septiembre, 2020).

Actividad 3: Uso de listas

Aprendizaje esperado: Configuración de una colección de objetos para su mapeo en una interfaz gráfica.		Duración de la actividad: 8 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">● Computadora PILARES.● IDE Android Studio.● Dispositivo móvil y/o Emulador	Evidencia: 1. Mostrar una lista en la pantalla del teléfono, relacionando cada elemento de la lista (negocio donde participen mujeres) con una imagen y/o con una descripción breve.	Retroalimentación: <ul style="list-style-type: none">● Verificar la correspondencia entre los elementos de la lista y su imagen y/o descripción.
Desarrollo de la actividad:		
<p>Primera parte: Las listas están en todas partes. Sugerencia de tiempo invertido: 30 minutos.</p> <ol style="list-style-type: none">1. La participante realizará una búsqueda en Internet para conocer los componentes <i>RecyclerView</i> y <i>CardView</i> e identificará al menos tres aplicaciones descargadas en su dispositivo móvil que hagan uso de estos componentes (en conjunto).2. En un documento escribirá el nombre de las aplicaciones y adjuntará una captura de pantalla del lugar en el que se esté utilizando. En ese mismo documento responderá de manera breve las siguientes preguntas:<ul style="list-style-type: none">● ¿Qué permite visualizar <i>RecyclerView</i>?● ¿Es común que <i>RecyclerView</i> y <i>CardView</i> vayan siempre juntos?● ¿Consideras difícil encontrar aplicaciones que utilicen dichos componentes?● ¿Cuál consideras que es su importancia? <p>Segunda parte: Una solución reutilizable. Sugerencia de tiempo invertido: 1 hora.</p>		

La participante conocerá el concepto de patrón de diseño y analizará los patrones utilizados por *RecyclerView*. Para ello realizará una investigación, basándose en las siguientes preguntas guía:

Patrones de diseño

- ¿Qué es un patrón de diseño?
- ¿Cuál es la clasificación de los patrones de diseño?
- ¿Cuáles son las ventajas de utilizar patrones de diseño?

Patrón *Adapter*

- ¿A qué clasificación pertenece?
- ¿Cuál es la problemática que resuelve?
- ¿Cómo la resuelve?

Patrón *ViewHolder*

- ¿Qué problemática resuelve?
- ¿Cómo la resuelve?

Tercera parte: El directorio de mujeres.
Sugerencia de tiempo invertido: 2 horas.

1. La participante realizará una lista (directorio) en la que reconocerá a las mujeres de su colonia, amigas y/o familiares que tengan locales de oficios/profesiones, deberá procurar reconocerlas desde el mismo nivel de importancia. Ejemplo: la dentista, la estilista, la tendera, etc.
2. Entrevistará de forma breve a dichas mujeres, con el objetivo de recabar la información necesaria para la creación de un directorio. Ejemplo: servicio que se ofrece, ubicación, teléfono, redes sociales, una historia o reseña breves, etc.

Cuarta parte: Eligiendo el estilo de los elementos de mi lista.
Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. A partir de la información recabada en la tercera parte de la actividad: *El directorio de mujeres*, la participante elegirá el orden en el que ubicará los datos en cada elemento de su lista.
2. Puede realizar primero el diseño en papel o en algún *software* de diseño para posteriormente implementarlo en *Android*. El diseño será libre y la participante podrá basarse en vistas ya existentes que se encuentren en la web.
3. Una vez se tenga definido el diseño, creará un nuevo proyecto y lo nombrará “MiDirectorio”. Dentro del proyecto creará un nuevo archivo de tipo *xml* en la carpeta *layout* del proyecto (se recomienda llamar al archivo *item_view.xml*).
4. El elemento raíz de este archivo será de tipo *CardView* y se utilizará *Constraint Layout* para la definición de las reglas de posicionamiento.
5. La participante también definirá en *activity_main.xml* un elemento de tipo *RecyclerView* que abarque toda la pantalla.
6. Es importante recordar asignar un *id* a todos los elementos de las vistas y elegir un nombre lo suficientemente descriptivo para poder localizarlos desde cualquier otra parte del proyecto.

Quinta parte: Generando la clase.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante creará una nueva clase en su proyecto que hará referencia a los datos de un elemento del directorio, se sugiere nombrarla “ItemDirectorio”.
2. Para cada atributo, se deberá seleccionar un nombre descriptivo, definir su tipo de dato correspondiente y colocarlo dentro del constructor principal. En caso de que se requiera guardar una imagen, el tipo de dato que se debe elegir es *Int*.

Sexta parte: Aplicando los patrones *Adapter* y *ViewHolder*.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

Definido el diseño y la información para el proyecto, la participante aprenderá a desplegar la información de la clase en la interfaz, se deberá hacer hincapié al uso de los patrones descritos en la segunda parte de la actividad.

El proceso para la implementación del adaptador es el siguiente:

1. Creará una clase llamada *DirectorioViewHolder*, definirá un atributo llamado *itemView* de tipo *View* en el constructor principal y hará extender a la clase de *RecyclerView.ViewHolder*.
2. Creará una clase llamada *DirectorioAdapter* y en su constructor principal definirá un atributo llamado “directorio”, el cual será de tipo arreglo y los elementos que almacenará serán de tipo “ItemDirectorio”.
3. Deberá hacer que la clase herede de “RecyclerView.Adapter” y pasará el *ViewHolder* definido en el paso anterior. La parte referente a la herencia quedaría de la siguiente forma:

RecyclerView.Adapter<DirectorioViewHolder>
4. Dentro de *DirectorioAdapter* implementará los métodos requeridos y definirá la funcionalidad adecuada en cada uno de ellos:
 - a. *onCreateViewHolder*
 - b. *getItemCount*
 - c. *onBindViewHolder*
5. Una vez aplicado el método *onCreateViewHolder* (que establece qué archivo *xml* se utiliza para el despliegue de los elementos), definirá dentro de la clase *DirectorioViewHolder* los atributos para cada uno de los elementos definidos en el *CardView* (*TextView*, *ImageView*) y encontrarlos en el layout, partiendo de *itemView*.
6. Es importante brindar apoyo a la participante en el proceso de comprender qué está realizando en cada uno de los métodos, un buen primer enfoque se puede realizar verificando el nombre de los métodos traducidos al español.

Séptima parte: Asignando el adaptador.

Sugerencia de tiempo invertido: 1 hora.

1. La participante asignará el adaptador a la vista correspondiente. Verificará que el *RecyclerView* que definió en *activity_main.xml* posea un *id*.
2. Dentro del método *onCreate* del archivo *MainActivity.kt* definirá un nuevo arreglo que almacenará objetos *ItemDirectorio* (se agregarán tantos objetos como registros tenga nuestro directorio).
3. Si se quieren agregar imágenes, estas deberán guardarse en *drawable* y llamarse por medio de la clase *R* desde *MainActivity*.
4. Definida la lista, la participante creará un objeto de tipo *DirectorioAdapter* al que se le pasará la lista, después se deberá llamar al *RecyclerView* del *xml* por medio de su *id* (basta con colocar el nombre) y asignarle en su propiedad *adapter*, el adaptador que acabamos de instanciar.

5. Se elegirá el modo de posicionamiento de la lista, el cual puede ser: *GridLayoutManager* o *LinearLayoutManager*. Los datos se requiere agregar varian dependiendo de la opción elegida, a *GridLayout* se le deberá pasar el número de columnas mientras que a *LinearLayout* la orientación (vertical/horizontal).
6. El modo de asignar el *LayoutManager*, es muy parecido al del *Adapter*, primero se crea el objeto y posteriormente se llama a la propiedad *layoutManager* para igualarla al *LayoutManager* definido.
7. La participante deberá probar los dos modos de posicionamiento, así como explorar sus parámetros. Al final podrá elegir el que le parezca más adecuado.

Octava parte: Probando mi aplicación.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante verificará el correcto funcionamiento de su aplicación y en caso de que existan errores podrá consultarlos en la ventana *Logcat*. Podrá brindarle apoyo a la participante en caso de ser necesario.
2. La aplicación podrá probarse tanto en un dispositivo físico como en un emulador.
3. Se recomienda que el proyecto se suba a la plataforma *GitHub*.

Notas:

Para la tercera parte de la actividad:

- El uso de la información brindada por las mujeres entrevistadas podrá ser usada únicamente con el consentimiento de las mismas.
- Puede hacer uso de la información recopilada durante las distintas actividades del taller 1 del *Módulo Desarrollo de Aplicaciones Móviles* bajo el consentimiento de las entrevistadas y recopilando información adicional.

Actividad 4: Usando mi cámara

Aprendizaje esperado: Implementar en <i>Kotlin</i> el acceso a la cámara del teléfono móvil para obtener fotografías.		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">● Computadora PILARES.● IDE Android Studio.● Dispositivo móvil y/o Emulador.	Producto: 1. Una aplicación que muestre un botón en la pantalla para “tomar una foto”, y al tomar la foto se elija entre los botones de “guardar”, “descartar”.	Evaluación: <ul style="list-style-type: none">● Verificar la correcta funcionalidad de la aplicación revisando las fotos guardadas en la galería.
Desarrollo de la actividad:		
Primera parte: Los permisos en una app. Sugerencia de tiempo invertido: 1 hora. 1. La participante deberá realizar una búsqueda de los siguientes temas, se sugiere consultar la documentación (disponible en los <i>Links de apoyo</i>): <ul style="list-style-type: none">● Permisos de la <i>app</i> en <i>Android Studio</i>.● Manifiesto de una app.● Funcionamiento de los permisos.● Cómo agregar permisos al manifiesto.● Solicitar permisos.● Permisos a nivel <i>API</i>.● Principios básicos del trabajo con permisos de <i>Android</i>.● Permisos que se requieren para acceder a cámara.● <i>API</i> de la cámara.		

2. Con la información obtenida deberá realizar una presentación sintética que exponga el tema. Se recomienda utilizar la plataforma *Genially* o *Prezi*.

Segunda parte: La cámara en mi *app*.

Sugerencia de tiempo invertido: 1 hora.

Android proporciona acceso completo al *hardware* de la cámara del dispositivo para que se puedan crear una amplia gama de *apps* basadas en la cámara o en la visión. De igual forma, si sólo se necesita una manera de permitir que el usuario capture una foto, simplemente se puede realizar una solicitud a una *app* de cámara existente para que capture una foto y la muestre.

1. Para esta aplicación se usará la clase *Camera*, para desarrollar una aplicación que muestre un botón en la pantalla para abrir la cámara y “tomar una foto”, además de dos botones: “guardar” y “eliminar”.
2. Como paso inicial la participante deberá crear un nuevo proyecto en *Android Studio*, lo nombrará “Fotograpp” y seleccionará un *activity* vacía.
3. Deberá buscar dos imágenes libres de *copyright*, una para ser el icono de la cámara, se deberá guardar con el nombre “cámara” y la otra para funcionar como “vista previa” se deberá guardar con el nombre de “previo”.
4. Una vez descargadas deberá copiarlas de la carpeta donde se encuentre y pegarlas en el directorio *drawable* (dentro de *Android Studio*). Deberá verificar que la imagen aparezca en el directorio.

Tercera parte: Componentes del diseño.

Sugerencia de tiempo invertido: 2 horas.

1. En el diseño de la *app* se utilizarán dos controles, para el primer control se agregará un *imageview*, para ello desde el menú *project* se seleccionará la imagen que se subió previamente en el directorio de *drawable* bajo el nombre de “previo”, se deberá configurar el tamaño del *imageview* además de considerar las distancias con los bordes.
2. Para el segundo control se utilizará un *ImageButton*, el cual se deberá arrastrar para agregar y se le asignará la imagen “cámara” del directorio de *drawable*, para que sea el ícono.
3. La participante deberá determinar los bordes del botón y márgenes.
4. De igual forma es libre de agregar más componentes de diseño para hacer su *app* más llamativa si es que lo desea.

Cuarta parte: Permisos de la cámara.

Sugerencia de tiempo invertido: 2 horas.

1. Ya que el diseño haya quedado listo, lo siguiente será hacer funcionar los componentes.
2. Primero se deberán agregar los permisos necesarios en el *AndroidManifest.xml*; el primero deberá permitir el acceso a la cámara y el segundo permitirá guardar las imágenes en el dispositivo.
3. Dentro de *AndroidManifest.xml* también se deberá colocar la etiqueta *provider* la cual permitirá el acceso a la cámara por medio de un archivo *XML* (dentro de la documentación de *Android Studio* se encuentra esta etiqueta ya declarada con el acceso a cámara, si la participante lo desea puede tomar ese código, el enlace se encuentra en los *Links de apoyo*) este código deberá ir entre las etiquetas *activity* y *provider*.
4. Se deberán presionar las teclas *ALT + ENTER* para crear el archivo *XML*, lo cual generará un nuevo *activity*, dentro del cual (en la pestaña *text*) se creará una nueva etiqueta y se le nombrará *paths*.
5. A la etiqueta *paths* se le deberá asignar el código de la etiqueta *PreferenceScreen* que se encuentra por default en el archivo *XML*, posteriormente se deberá borrar la etiqueta *PreferenceScreen*.
6. A continuación, se colocará donde se guardará la aplicación y a su vez las imágenes que se capturen, la participante deberá crear el directorio “my images” y la ruta correspondiente para la aplicación (si la desconoce puede encontrarla al desplegar la carpeta java).
7. En *Mainactivity* deberá declarar un objeto del tipo *imageview* con el nombre “img”.
8. Con el método *onCreate* deberá crear la relación entre la parte lógica y la parte gráfica; agregará la validación de los permisos necesarios para poder acceder a la cámara y guardar las fotos.
9. Se deberá agregar un método que permita asignar un nombre único a cada fotografía.
10. El método “tomarFoto” deberá permitir tomar la fotografía y guardar el archivo, este método deberá relacionarse con el *imagebutton*.
11. Se deberá asignar otro método el cual permita mostrar la vista previa de la foto tomada.
12. En el *ActivityMain* verificará que los métodos correspondientes estén asignados.

Quinta parte: Prueba de mi *app* con ayuda de mis alianzas.

Sugerencia de tiempo invertido: 1 hora.

1. La participante deberá probar el funcionamiento de su aplicación.
2. Se solicitará que realice un ejercicio de memoria donde recuerde a las mujeres con las cuales ha entablado algún tipo de alianza en su vida y que en la actualidad siga teniendo contacto con ellas. (Pueden ser mujeres: que la haya cuidado a lo largo de su vida, con las que haya hecho trabajos en equipo, con las que haya establecido un vínculo de amistad o compañerismo, etc.)
3. Una vez identificadas las mujeres, deberá contactarlas para platicar acerca de la aplicación que está desarrollando y solicitar su apoyo y autorización para tomarles una fotografía que ayude a probar el funcionamiento de su *app*.
4. A las mujeres que hayan dado su consentimiento deberá tomarles una fotografía y verificar que éstas se guarden en el carrete de foto del dispositivo.

Nota:

- Si la participante ya domina los conocimientos teóricos, podrá saltar la primera parte de la actividad y comenzar desde la segunda.

Links de apoyo:

- *Genially*.

<https://www.genial.ly/es> (Septiembre, 2020).

Documentación *Android* – Acceso a cámara:

- developers. (Sin fecha). *Cómo tomar fotos*.
<https://developer.android.com/training/camera/photobasics> (Septiembre, 2020).

Permisos de las apps:

- developers. (Sin fecha). *Recomendaciones sobre permisos de apps*.
<https://developer.android.com/training/permissions/usage-notes?hl=es-419> (Septiembre, 2020).

Cámara:

- developers. (Sin fecha). *Cámara*.
<https://developer.android.com/training/camera?hl=es-419> (Septiembre, 2020).

Actividad 5: Obtener mi ubicación actual

Aprendizaje esperado: Elaborar una aplicación que obtenga las coordenadas actuales de ubicación.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">● Computadora PILARES.● <i>IDE Android Studio</i>.● Dispositivo móvil y/o emulador.	Producto: 1. Una aplicación que muestre un botón en la pantalla para “obtener ubicación” y que al obtenerla muestre las coordenadas en la pantalla.	Evaluación: <ul style="list-style-type: none">● Verificar que las coordenadas obtenidas sean correctas, y que al cambiar de ubicación éstas cambien.● Verificar mediante impresiones de pantalla cada caso.
Desarrollo de la actividad:		
Primera parte: Los permisos de ubicación. Sugerencia de tiempo invertido: 1 hora. 1. La participante deberá realizar una búsqueda de los siguientes temas (se sugiere consultar la documentación disponible en los <i>Links de apoyo</i>): <ul style="list-style-type: none">● Permisos en <i>Android Studio</i>.● Permiso de <i>App</i> personalizado.● Solicitar permisos de ubicación: Ubicación en primer plano y ubicación en segundo plano.● Solicitud y validación de los permisos.● Proceso para realizar solicitudes incrementales.● Objeto tipo “Location”.● Clase <i>Geocoder</i>.● Método <i>getFromLocation()</i>.● Método <i>getLastLocation()</i>.● Objeto <i>Geocoder</i>.● <i>API</i> de <i>Google Maps</i> para <i>Android</i>.● ¿Qué es y cómo se configura r los servicios de ubicación en el <i>SDK</i> de <i>Google play services</i>?		

- Implementación de la ubicación en una *app*.

2. Con la información obtenida deberá responder de manera breve la siguiente pregunta: ¿Qué utilidad le encuentro a los temas en el desarrollo de mi *app*?

Segunda parte: La ubicación en mi *app*.

Sugerencia de tiempo invertido: 1 hora.

Una de las funciones exclusivas de las aplicaciones móviles es la detección de ubicaciones. Los usuarios llevan sus dispositivos móviles a todas partes, por lo que el conocimiento de la ubicación le permite a la *app* brindar una experiencia más contextual. Las *API* de ubicación disponibles en los Servicios de *Google Play* permiten agregar conocimiento de la ubicación a la *app* con seguimiento automático de la ubicación, geovallado y reconocimiento de actividades.

1. Con base en los conceptos abordados en la primera parte de esta actividad, la participante habrá descubierto que existen diferentes formas o métodos para poder obtener la ubicación de un usuario.
2. Dependiendo del enfoque o necesidades que su aplicación contenga, deberá hacer una lista de las distintas formas para obtener las coordenadas de un usuario.
3. Especificará qué métodos, objetos, clases, *API*, etc; utilizará en cada una.
4. Finalmente, deberá seleccionar la forma que le parezca más eficiente y justificar por qué lo es.

Tercera parte: Creando mi *app* con ubicación.

Sugerencia de tiempo invertido: 2 horas.

1. La participante deberá desarrollar una aplicación que muestre un botón en la pantalla para “obtener ubicación” y que al obtenerla muestre las coordenadas en la pantalla.
2. Deberá crear un nuevo proyecto en *Android Studio*, nombrará a la *app* “Mi gps”.
3. En *Android SDK Manager* deberá asegurarse que *Google play services* esté instalado, para poder cargar los mapas.
4. Se creará una clase que herede la clase base *Activity* y que implemente la interfaz *ActivityCompat.OnRequestPermissionsResultCallback*:

5. Deberán declararse tres variables de tipo *TextView*, que recogerán los valores de la posición del dispositivo: Latitud, longitud y altura.
6. Se declarará una variable de tipo *LocationManager*, la cual se encargará de proporcionar acceso al servicio de localización del sistema.
7. Se declarará una variable de tipo *Location*.
8. La *app* deberá solicitar los permisos para que el usuario seleccione si desea permitir el acceso al *GPS* del dispositivo y comprobar si se han concedido los permisos para mostrar los datos de latitud, longitud, altura y precisión del dispositivo.
9. Se le asignará a la clase *LocationManager*, el servicio a nivel de sistema a partir del nombre y se le asignará a la variable de tipo *Location* que accederá a la última posición conocida proporcionada por el usuario.
10. Finalmente se asignarán los valores de la última posición a cada variable de tipo *TextView*.
11. Además se deberá declarar en el fichero *AndroidManifest.xml* el permiso a la *API* para determinar la ubicación más precisa posible a través de los proveedores de ubicaciones disponibles, entre los que se incluyen el sistema de posicionamiento global (*GPS*), los datos de *Wi-Fi* y los datos móviles.
12. La participante deberá probar su aplicación y el funcionamiento de ésta. Tomará capturas de pantalla que muestre las coordenadas del *GPS* y su funcionamiento.

Cuarta parte: Todos los caminos llevan a Roma.

Sugerencia de tiempo invertido: 1 hora.

El desarrollo de una aplicación como ésta puede realizarse de múltiples maneras, algunas más difíciles que otras, hay muchos caminos para llegar al mismo resultado.

1. Una vez que la aplicación de la participante haya quedado lista y funcione sin problemas; deberá realizar un cuadro comparativo en el cual enliste las diferencias entre el método que ella seleccionó en la segunda parte de la actividad y el método seguido en las instrucciones de la tercera parte.
2. Al finalizar, responderá la siguiente pregunta: ¿Consideras que es más sencillo implementar tu método o el método de las instrucciones de la actividad?

Nota:

- Si la participante ya domina los conocimientos teóricos, podrá saltar la primera parte de la actividad y comenzar desde la segunda.

Links de apoyo:

Permisos:

- developers. (Sin fecha) *Permisos en Android*.
<https://developer.android.com/guide/topics/permissions/overview?hl=es-419> (Septiembre, 2020).

Crear apps de conocimiento de la ubicación:

- developers. (Sin fecha) *Crear apps de reconocimiento de la ubicación*.
<https://developer.android.com/training/location?hl=es-419> (Septiembre, 2020).

Mostrar la ubicación de una dirección:

- developers. (Sin fecha) *Cómo conocer la ubicación más reciente*.
<https://developer.android.com/training/location/display-address?hl=es> (Septiembre, 2020).

SDK de Google play services para configurar los servicios de ubicación:

- develou.com. (Sin fecha) *¿Cómo Obtener La Ubicación De Tus Usuarios En Android?*
<http://www.hermosaprogramacion.com/2016/08/ubicacion-android-google-play-services/> (Septiembre, 2020).

Proyecto Geolocalización de android:

- academiaandroid (2015) *Proyecto Geolocalización Android*.
<https://academiaandroid.com/proyecto-geolocalizacion-android/> (Septiembre, 2020).

Actividad 6: Mostrar ubicación en mapa

Aprendizaje esperado: Mostrar ubicación en un mapa de Google.		Duración de la actividad: 4 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador 	<p>Evidencia:</p> <ol style="list-style-type: none"> 1. Una captura del mapa con tres marcadores de ubicación. <p>Producto:</p> <ol style="list-style-type: none"> 2. Una aplicación que muestre un botón en la pantalla para “obtener ubicación” y que al obtenerlas muestre las coordenadas en un mapa en la pantalla. 	<p>Retroalimentación:</p> <ol style="list-style-type: none"> 1. Verificar la captura del mapa. <p>Evaluación:</p> <ul style="list-style-type: none"> • Verificar que las coordenadas obtenidas sean correctas, y que al cambiar de ubicación éstas cambien. • Verificar mediante una impresión de pantalla cada caso.
Desarrollo de la actividad:		
<p>Primera parte: ¿Cómo puedo agregar mapas a mi aplicación?</p> <p>Sugerencia de tiempo invertido: 1 hora y 30 minutos.</p> <p>La participante realizará una investigación en internet para identificar las ventajas del uso de <i>APIs</i> dentro del desarrollo de <i>software</i>, así como conceptos relacionados a la <i>API</i> de <i>Google Maps</i>. Las preguntas para conducir la investigación, son las siguientes:</p> <ul style="list-style-type: none"> • ¿Para qué sirven las <i>APIs</i>? • ¿Cuáles son las <i>APIs</i> más populares actualmente? Menciona al menos cinco. • Agrega una breve descripción de las acciones que se pueden realizar con las <i>APIs</i> de la pregunta anterior. • Revisa la lista de <i>APIs</i> públicas y observa qué otras acciones e información hay disponible para incorporar en tus proyectos. Elige tres y escribe las razones por las que las elegiste. • Investiga el concepto de <i>API Key</i>. • ¿Cuál es la importancia de no compartir tu <i>API Key</i> públicamente? • Ingresa a la página de la <i>API</i> de <i>Google Maps</i> y navega a través de ella. • Escribe al menos tres acciones que se pueden llevar a cabo utilizando la <i>API</i> de <i>Google Maps</i>. 		

Segunda parte: ¿Qué es una plantilla?

Sugerencia de tiempo invertido: 1 hora.

La participante identificará el concepto y modo de empleo de plantillas en *Android Studio*. Los pasos para el desarrollo se describen a continuación:

1. Se agregará una plantilla de tipo *Google Maps Activity* tanto en un nuevo proyecto como en uno existente. La Galería con plantillas a elegir se despliega automáticamente cuando se crea un nuevo proyecto o bien en un proyecto existente, bastará con ir a la ruta

File>New>Activity>Gallery...

Es posible utilizar el proyecto de la actividad *MMT3A5* como “Proyecto existente” o bien, generar un nuevo proyecto para realizar las dos acciones requeridas en este punto.

2. Dentro del nuevo proyecto, se explorará el código generado automáticamente y se leerán las instrucciones contenidas en *google_maps_api.xml* (archivo que se encuentra en la carpeta *values*). Con base en las observaciones se contestarán las siguientes preguntas:
 - ¿Habías utilizado una plantilla antes?
 - ¿Qué archivos se vieron afectados por la plantilla *Google Maps Activity* en comparación con la plantilla *Empty Activity*?
 - ¿Cuál crees que sea la definición de plantilla?

Tercera parte: Obteniendo mi *API Key*.

Sugerencia de tiempo invertido: 1 hora.

1. La participante realizará la incorporación y restricción del *API Key* de *Google Maps* para su proyecto. Las instrucciones para llevar a cabo ambas acciones se encuentran en el enlace proporcionado en los *Links de apoyo*.

2. Para verificar la correcta incorporación de la clave, la participante ejecutará su aplicación en un emulador o dispositivo físico. En ellos, deberá observar un mapa en toda la pantalla.
3. Adicionalmente, la tallerista verificará que la *API Key* se haya restringido para su uso dentro del proyecto creado. Esto se puede verificar en *Credentials>Claves de API* dentro de la consola de *Google Maps* de la participante, en donde la clave creada deberá aparecer sin advertencias y en la sección *Restricciones* aparecerá el texto “Apps de Android”.
4. La participante discutirá con la tallerista u otras participantes las siguientes preguntas y registrará sus conclusiones.
 - ¿Consideras que agregar un mapa a tu aplicación fue sencillo?
 - ¿Cuánto tiempo, dificultad y dinero consideras que conlleva la creación de una plataforma como *Google Maps*?
 - ¿Qué aplicaciones (que no sean *Google Maps*) incorporan mapas de *Google*?
 - ¿Cómo crees que los incorporaron?
 - Una vez que agregaste la *API Key* a tu proyecto, ¿ésta ya contaba con un funcionamiento definido? ¿por qué?
 - ¿Cuáles consideras que son las ventajas del uso de plantillas?
 - ¿Existirán desventajas?
 - ¿Qué parte del código deberías modificar para mostrar otra ubicación en tu mapa?

Cuarta parte: Modificando el código de la plantilla.

Sugerencia de tiempo invertido: 1 hora.

1. La participante pensará en tres mujeres que han marcado su vida y ubicará los lugares donde conoció a cada una, estos espacios tienen que ser públicos.
Por ejemplo: La mejor amiga que conoció en la escuela secundaria.
2. Para ello modificará el código del método *onMapReady* contenido en *MainActivity.kt*.
3. Deberá ubicar la latitud y longitud de cada uno de los sitios en un objeto *LatLng*, para posteriormente colocarlos en el mapa con la función *addMarker*.
4. Como evidencia se adjuntará una captura del mapa con los tres marcadores.

Nota:

- En la tercera parte de la actividad es importante situar la *API* en el archivo *google_maps_api.xml*, en caso de no existir este archivo, se situará el dato directamente en el lugar descrito en el enlace.

Links de apoyo:

- Google Cloud. (Sin fecha). *Google Maps Platform*.
<https://cloud.google.com/maps-platform?hl=es> (Septiembre, 2020).
- Google Maps Platform. (Sin fecha). *Obtén una clave de API*.
<https://developers.google.com/maps/documentation/android-sdk/get-api-key?hl=es-419> (Septiembre, 2020).
- Why GitHub? (Sin fecha). *public APIs*.
<https://github.com/public-apis/public-apis> (Septiembre, 2020).

Actividad 7: Más pantallas

Aprendizaje esperado: Agregar pantallas a la aplicación para mostrar información.		Duración de la actividad: 8 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia: <ol style="list-style-type: none"> 1. Mostrar la creación de nuevas pantallas dentro de la aplicación de directorio. 2. Pantalla de bienvenida y pantallas anexas a la información del negocio. 3. Agregar una <i>toolbar</i> personalizada. 4. Agregar botón de “Atrás”. 5. Agregar botón de “Compartir”. 	Retroalimentación: <ul style="list-style-type: none"> • Verificar mediante impresiones que las nuevas pantallas se desplieguen correctamente. • Verificar el uso correcto de los botones creados.
Desarrollo de la actividad:		
<p>Primera parte: Pantallas en <i>Android</i>.</p> <p>Sugerencia de tiempo invertido: 1 hora y 30 minutos.</p> <p>La participante profundizará en el concepto de <i>Activity</i> y pondrá en práctica sus conocimientos para la creación de nuevas pantallas en su proyecto referente al directorio de mujeres. La secuencia de acciones que la participante realizará para alcanzar el objetivo, se describe a continuación:</p> <ol style="list-style-type: none"> 1. Responderá las siguientes preguntas: <ol style="list-style-type: none"> a. ¿Qué es un <i>Activity</i>? b. ¿Qué archivos se ven modificados o son creados cuando se crea una nueva actividad? c. Investiga el ciclo de vida de una actividad (coloca el esquema). d. ¿Qué ventajas tendrá conocer el ciclo de vida de una actividad? e. ¿Por qué crees que el único método implementado es <i>onCreate</i>? 		

2. En *MainActivity.kt*, implementará los métodos faltantes del ciclo de vida de una actividad y colocará dentro de cada uno de ellos, un *Toast* con el nombre del método. Después, ejecutará el proyecto y verificará lo que ocurre ante las siguientes situaciones:

- Rotación del dispositivo.
- Uso de los tres botones en la parte inferior de todos los dispositivos.

Comentará sus observaciones con la tallerista y borrará los *Toast* agregados.

3. Observará el archivo *AndroidManifest.xml* e identificará las características de la actividad principal.

- a. ¿Qué líneas crees que son las que indican que la actividad será la primera en mostrarse?
- b. ¿Se podrá indicar que se quiere comenzar con otra actividad?

4. Creará tres actividades nuevas, correspondientes a las siguientes funcionalidades: bienvenida, detalles de un contacto y agregar contacto.

5. Realizará las modificaciones correspondientes para hacer que la primera actividad lanzada sea la correspondiente a la bienvenida.

Segunda parte: Diseñando mis pantallas.

Sugerencia de tiempo invertido: 4 horas.

La participante desarrollará la parte gráfica de las tres actividades creadas en el punto anterior, las cuales deberán cumplir con las características siguientes:

- Bienvenida.
 - Diseñará el *logotipo* de la *app*.
 - Definirá el nombre de su *app*, para ello buscará reflejar el objetivo de la misma o puede utilizar alguna de las características planteadas con anterioridad (directorio de negocios/servicios emprendidos o atendidos por mujeres, alianzas, sororidad, visibilización, etc.)
 - Accederá a una plataforma libre para el diseño de su logotipo.

- Incorporará la descripción general de la *app* (puede redactar el objetivo, la misión y la visión de la *app* en este apartado).
 - Deberá contar con un botón con el mensaje “Empezar”.
- Detalles de contacto.
 - Deberán mostrar todos los datos identificados en el directorio, distribuidos a lo largo de toda la pantalla.
 - Deberá contar con un mapa que despliegue la ubicación del negocio.
- Agregar contacto.
 - Contará con los *EditText* y *TextView* necesarios para pedir la información, así como un botón “Enviar”.
 - De manera opcional se agregará un botón “Subir Imagen”.
- Mostrar contactos (*RecyclerView*).
 - Agregará un *Floating Action Button*.

Tercera parte: Comunicando mis pantallas.

Sugerencia de tiempo invertido: 1 hora.

La participante descubrirá el procedimiento para comunicar las pantallas de su aplicación, a partir de realizar la búsqueda del concepto en Intent. En los *Links de apoyo* se encuentra la información provista por la documentación oficial para iniciar nuevas actividades.

Las pantallas deberán comunicarse de la siguiente forma:

- Al presionar el botón “Empezar” en la pantalla de *Bienvenida*, se nos conducirá al *RecyclerView* con todos los contactos.
- Al seleccionar un contacto (revisar el recurso para implementar *onItemClickListener*) se nos mostrará la información detallada en la pantalla de *Detalles*.
- Al presionar el botón flotante en la pantalla del *RecyclerView*, se mostrará la actividad correspondiente a agregar un nuevo contacto.

Cuarta parte: Compartiendo información entre actividades.

Sugerencia de tiempo invertido: 30 minutos.

1. La participante investigará el funcionamiento y sintaxis de los métodos *putExtra* y *getExtra* de la clase *Intent*, después retomará el proyecto referente a la obtención de las coordenadas de una ubicación de la actividad *MMT3A6* y conectará la actividad con la de mostrar un mapa.
2. En la primera pantalla obtendrá las coordenadas y mediante un botón “Mostrar mapa” aquéllas deberán pasarse y mostrarse en otra actividad que será la que despliegue el mapa con el marcador.

Quinta parte: Compartiendo objetos entre actividades.

Sugerencia de tiempo invertido: 1 hora.

1. La participante descubrirá el procedimiento para comunicar objetos más complejos entre actividades, por medio de objetos parcelables. En los enlaces se encuentran los recursos propuestos para conocer qué son los objetos parcelables y cómo implementarlos en *Kotlin*.
2. Realizará las modificaciones correspondientes en su proyecto para pasar la información del *ItemDirectorio* seleccionado desde el *RecyclerView* a la actividad de detalles para desplegar su información.

Links de apoyo:

- developers. (Sin fecha). *Cómo iniciar otra actividad*.
<https://developer.android.com/training/basics/firstapp/starting-activity?hl=es-419> (Septiembre, 2020).
- Surwase. R. (2018). *Android RecyclerView onItemClickListener & getAdapterPosition (): A Better Way*.
<https://medium.com/hackernoon/android-recyclerview-onitemclicklistener-getadapterposition-a-better-way-3c789baab4db> (Septiembre, 2020).
- developers. (Sin fecha). *Objetos parcelables y paquetes*.
<https://developer.android.com/guide/components/activities/parcelables-and-bundles> (Septiembre, 2020).
- Rahman, Z. (2017). *Easy Parcelable in Kotlin*.

<https://medium.com/the-lazy-coders-journal/easy-parcelable-in-kotlin-the-lazy-coders-way-9683122f4c00> (Septiembre, 2020).

Actividad 8: Guardar información en una base de datos local

Aprendizaje esperado: Crear una base de datos local (<i>Room</i> o <i>SQLite</i>) para guardar la información recolectada en la aplicación.		Duración de la actividad: 8 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Producto: <ol style="list-style-type: none"> 1. Mostrar la creación de la base de datos haciendo consulta simples, por nombre o actividad. 2. Una captura de los archivos creados. 	Evaluación: <ul style="list-style-type: none"> • Verificar que al agregar o modificar información en la aplicación se vea reflejado en la base de datos haciendo consultas.
Desarrollo de la actividad:		
<p>Primera parte: Más y más patrones. Sugerencia de tiempo invertido: 2 horas.</p> <p>La participante retomará la importancia del uso de patrones para la creación de aplicaciones <i>Android</i>. Esta vez centrándose en otra clasificación: los patrones arquitectónicos.</p> <p>Se recomienda a la tallerista comentar con la participante las ventajas del uso de patrones y en caso de considerarlo necesario, dará la indicación para que se revise el documento producto de la actividad <i>MMT3A3</i>, con la finalidad de asegurarse que la participante cuente con los conocimientos previos necesarios.</p> <p>Los puntos guía que la participante investigará para introducir los nuevos conceptos, son:</p> <ul style="list-style-type: none"> • Consultar dos definiciones sobre arquitectura general e identificar las características en común. • ¿Qué es arquitectura de <i>software</i>? • ¿Qué es un patrón de arquitectura? • ¿En qué se diferencia un patrón de diseño con un patrón de arquitectura? • Revisa el recurso <i>Patrones Arquitectónicos en Android</i> y contesta (puedes visitar otros recursos): <ul style="list-style-type: none"> ○ ¿Por qué son importantes los patrones de arquitectura? ○ ¿Qué patrones soporta <i>Android</i>? 		

- ¿Cuál es la principal diferencia entre ellos?
- ¿Cuáles son los componentes del patrón *MVVM*? Describe brevemente su función.
- Revisa las siguientes secciones del recurso *Guía de arquitectura de apps*.
 - Experiencias del usuario de *apps* para dispositivos móviles.
 - Principios comunes de la arquitectura.
 - Separación de problemas.
 - Cómo controlar la IU a partir de un modelo.
 - Arquitectura de *app* recomendada.
 - Descripción general.
 - Cómo crear la interfaz de usuario.
- Contesta las siguientes preguntas con base en el contenido del artículo (puedes visitar otros recursos para complementar tu respuesta):
 - ¿Qué patrón arquitectónico se recomienda?
 - ¿Qué patrones de diseño se mencionan?
 - ¿Qué es *LiveData*? ¿Para qué sirve?
- ¿Los patrones de diseño y de arquitectura pueden trabajar en conjunto?
- Realizando una analogía con la construcción de casas y edificios, ¿cuál consideras que es la importancia de contar con una buena arquitectura?

Segunda parte: ¿Cómo conservar los datos?

Sugerencia de tiempo invertido: 2 horas y 30 minutos.

La participante identificará la importancia de la persistencia de datos y aprenderá a implementarla en su aplicación mediante una base de datos local.

1. Para plantear el problema que supone la pérdida de datos, la participante deberá agregar dos registros a su directorio, realizará las dos acciones que se describen a continuación y comentará sus observaciones (refiriéndose a los datos ingresados) con la tallerista u otras participantes:

- Rotación del dispositivo.

- Salida total de la aplicación y reingreso a ella.

En caso de no haber añadido la funcionalidad a su aplicación que permite agregar nuevos elementos a su directorio y desplegarlos en el *RecyclerView*, se seguirán primero estos pasos:

- 1) Añadirá un escuchador de eventos de tipo *onClick* en el botón adecuado, definido en la vista para agregar nuevos elementos.
 - 2) Dentro de la implementación del escuchador:
 - a) Creará un nuevo objeto de tipo *ItemDirectorio*, con los datos ingresados en los *EditText* y agregará este objeto a la lista del adaptador.
 - b) Después se llamará al método *notifyDataSetChanged* del adaptador, para notificar los cambios al *RecyclerView*.
 - 3) Por último, se recomienda probar el correcto funcionamiento de la aplicación, en un emulador o dispositivo físico.
2. Identificado el problema, los conceptos que se deberán abordar son los siguientes (para los tres últimos puntos se proporcionan recursos útiles en la sección de *Links de apoyo*):
- ¿Qué es una base de datos y para qué sirve?
 - ¿Qué es una llave primaria en bases de datos?
 - ¿Qué implica que una llave primaria sea autoincremental?
 - ¿Qué es *SQLite*? ¿Qué ventajas ofrece?
 - ¿Qué es una base de datos local?
 - ¿Qué es *Room* en *Android*?
 - ¿Qué ventajas ofrece *Room* sobre la *API* de *SQLite*?
 - Enumera los pasos para implementar una base de datos, haciendo uso de *Room*.
 - ¿Qué es una entidad?
 - ¿Qué es *DAO* en *Room*?

3. En caso de que la participante no haya cursado el *Módulo Web (MW)* de la Escuela de Código, podrá solicitar una investigación adicional respecto a la base de datos. Por otro lado, las participantes que sí hayan cursado el *MW*, podrán recurrir a los recursos provistos en las actividades *MWT2A16*, *MWT3A3* y *MWT3A11*.

Tercera parte: Almacenando los datos de mi aplicación.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante implementará una base de datos local en su aplicación de directorio, haciendo uso de *Room*. Para ello, retomará los pasos enumerados en la sección anterior de la actividad y realizará los cambios pertinentes en su proyecto.
2. La tallerista deberá verificar que los pasos identificados, se asemejan a los siguientes:
 - Agregar las dependencias correspondientes a *Room*, en el *archivo build.gradle* de la *app*.
 - Creación de la entidad.
 - a) Indicar que *ItemDirectorio* es de tipo *data class*.
 - b) Añadir la anotación (*@Entity*) a la clase *ItemDirectorio*.
 - c) Cambiar el nombre de la tabla, modificando la anotación anterior a *@Entity(tablename = nombre_tabla_entre_comillas)*.
 - d) Agregar un campo de tipo entero con nombre "id" en la entidad y anteponer la anotación *@PrimaryKey (autogenerated = true)*.
 - e) Cambiar el nombre de los otros campos de ser necesario con *@ColumnInfo*.
 - Creación de la interfaz *DAO*.
 - a) Crear una interfaz con el nombre sugerido *DirectorioDao* y añadirle la anotación *@Dao*.
 - Creación de la clase correspondiente a la base de datos.
 - a) Crear una clase abstracta con el nombre sugerido *DirectorioDatabase*, hacerla extender de *RoomDatabase* y añadirle la anotación *@Database(entities = arrayOf(ItemDirectorio.class), version = 1)*.
 - b) Declarar un método abstracto que devuelva un objeto de tipo *DirectorioDao*.
 - Instanciar la base de datos (en *MainActivity.kt*).

- a) Generar dentro de un *companion object*, la variable que guardará la referencia a la base de datos.
- b) En *onCreate* generar la instancia de la base de datos en la variable recién generada, con una instrucción semejante a la siguiente:

```
Room.databaseBuilder(this,ItemDirectorio::class.java,"directorio_database").allowMainThreadQueries().build()
```

3. Posteriormente, proveerá a la participante las instrucciones detalladas para la implementación, a su vez, la participante podrá revisar otros recursos que considere pertinentes, con el fin de alcanzar el objetivo planteado en esta sección.
4. Una vez realizados los cambios, el proyecto deberá compilar correctamente y funcionar como ya lo hacía, sin distinción. El único cambio realizado fue la creación de la base de datos. Así que la tallerista deberá verificar la correcta creación de los archivos y la participante deberá adjuntar una captura de estos mismos como evidencia.
5. Para acceder a los archivos correspondientes a la base de datos recién creada, estos se deberán localizar mediante el *Device File Explorer* (revisar recurso que explica cómo acceder a esta ventana), mientras se está ejecutando la aplicación.
6. En esta ventana, se debe localizar la carpeta *data* que contiene la referencia a todas las aplicaciones instaladas en el dispositivo. Dentro de esta, habrá que localizar la carpeta correspondiente a la aplicación mediante el nombre que se haya dado al proyecto.
7. La carpeta que corresponde a la aplicación deberá contener una carpeta llamada *databases*, en donde se encontrarán los archivos de los que se debe adjuntar captura (los cuales incluyen un archivo con terminación *.db*).

Cuarta parte: Siguiendo el patrón MVVM.

Sugerencia de tiempo invertido: 2 horas.

1. La participante realizará los cambios correspondientes a su proyecto para aplicar el patrón MVMM. Los pasos a seguir son los siguientes:
 - Repositorio
 - a) Crear una clase *Repositorio* (el nombre que se sugiere es “DirectorioRepositorio”).
 - b) Definir una variable que almacenará el DAO de la base de datos instanciada, la instrucción debe verse más o menos de la siguiente manera:

```
private val directorioDao = MainActivity.variable_en_companion_object.metodo_para_obtener_el_dao
```

c) Dentro de esta clase se definirán todos los métodos que se definan en el *Dao*, pero que se implementarán en la siguiente actividad.

- Injection

a) Crear un objeto en un nuevo archivo llamado *Injection* el cual definirá un único método para devolver el repositorio creado. La instrucción debe ser similar a:

```
fun proveerRepositorio() : DirectorioRepositorio = DirectorioRepositorio()
```

- ViewModel

a) Generar una nueva clase, de nombre *DirectorioViewModel* que herede de *ViewModel*.

b) Dentro, crear una variable llamada “repositorio” la cual almacenará el repositorio creado, obtenido a partir del objeto *Injection*.

2. Una vez efectuados los cambios, la participante probará su aplicación en un dispositivo o emulador, la cual no deberá arrojar errores. El resultado a esperar es que los datos aún no se guarden ya que no se han implementado los métodos que proveen de funcionamiento a la base de datos.
3. Sin embargo, la participante y la tallerista deberán verificar que las clases se relacionan de la forma en que lo indica el esquema referente a la arquitectura MVVM (revisar recurso).

Nota:

- En la tercera parte de la actividad se recomienda realizar la identificación de la base de datos en el emulador, ya que de probarse en un dispositivo físico puede ser difícil localizar los archivos debido a la cantidad de aplicaciones instaladas.

Links de apoyo:

- Vespa, L. (2019). *Patrones arquitectónicos en Android*.
<https://medium.com/@vespasoft/patrones-arquitect%C3%B3nicos-en-android-ded39f7a2c10> (Septiembre, 2020).
- developers. (Sin fecha). *Guía de arquitectura de apps*.
<https://developer.android.com/jetpack/docs/guide?hl=es-419> (Septiembre, 2020).

Room

- developers. (Sin fecha). *Guía de arquitectura de apps (Cómo conservar los datos)*.
<https://developer.android.com/jetpack/docs/guide?hl=es-419> (Septiembre, 2020).
 - *Cómo guardar contenido en una base de datos local con Room*.
<https://developer.android.com/training/data-storage/room?hl=es-419> (Septiembre, 2020).

Esquema de una aplicación usando MVVM.

- [Esquema] developer.android.com. (Sin fecha). Sin título.
Tomada de cursosdesdesarrollo.com. (2017). *Architecture Components Android*.
<https://developer.android.com/topic/libraries/architecture/images/final-architecture.png?hl=es-419> (Septiembre, 2020).
- developers. (Sin fecha). *Cómo ver archivos del dispositivo con el explorador de archivos de dispositivos*.
<https://developer.android.com/studio/debug/device-file-explorer?hl=es> (Septiembre, 2020).

Actividad 9: Agregar, editar, actualizar y borrar datos en la aplicación

Aprendizaje esperado: Administrar la información que se recopila en la aplicación y sus efectos en la base de datos.		Duración de la actividad: 7 horas y 30 minutos.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none">• Computadora PILARES.• IDE Android Studio.• Dispositivo móvil y/o Emulador.	Evidencia: <ol style="list-style-type: none">1. Hacer cambios en los datos de la aplicación y mostrar la actualización en la misma aplicación.2. Mostrar estos cambios en la base de datos.3. Una captura de su archivo DAO.4. Liga de repositorio del documento.	Retroalimentación: <ul style="list-style-type: none">• Verificar que los cambios hechos en la aplicación correspondan efectivamente a los cambios hechos en la base de datos haciendo consultas sencillas por nombre o actividad.• Revisar que el código en el repositorio esté completo.
Desarrollo de la actividad:		
<p>Primera parte: Operaciones CRUD. Sugerencia de tiempo invertido: 30 minutos.</p> <p>La participante realizará una breve investigación para saber qué significa CRUD y en qué contexto se aplica. Las preguntas para guiar la investigación son:</p> <ul style="list-style-type: none">• ¿A qué hace referencia CRUD?• ¿Cuál es el significado de las siglas?• ¿En dónde se aplica?• Menciona tres aplicaciones que implementen CRUD. <p>Segunda parte: Agregando funcionalidad a mi base de datos. Sugerencia de tiempo invertido: 1 hora y 30 minutos.</p> <ol style="list-style-type: none">1. La participante modificará su proyecto con el fin de permitir que su base de datos efectúe las cuatro operaciones CRUD.		

2. La tallerista deberá rectificar que la participante conozca el archivo principal que se debe modificar para definir los métodos que permiten acceder a la base de datos (DAO), en su defecto, dará la indicación de revisar nuevamente los recursos referentes a *Room*.
3. Los cambios que se realizarán en el proyecto son los siguientes:
 - DAO
En la clase correspondiente al DAO, generar cuatro nuevos métodos, cuyas características se mencionan a continuación:
 - CREATE
 - Nombre sugerido *addItem*.
 - Recibe un parámetro de tipo *ItemDirectorio*, cuyo valor es el que se desea agregar.
 - Se debe añadir la anotación `@Insert`.
 - READ
 - Nombre sugerido *getItems*.
 - Retornar un objeto de tipo *LiveData<List<ItemDirectorio>>*.
 - Se debe añadir la anotación `@Query("SELECT * FROM nombre_tabla")`.
 - UPDATE
 - Nombre sugerido *updateItem*.
 - Recibe como parámetro un variable *vararg* de tipo *ItemDirectorio* con los registros a actualizar.
 - Se debe añadir la anotación `@Update`.
 - DELETE
 - Nombre sugerido *deleteItem*.
 - Recibe un parámetro de tipo *ItemDirectorio*, cuyo valor es el que se desea eliminar.
 - Se debe añadir la anotación `@Delete`.
4. La participante deberá verificar que su proyecto compile de manera correcta y deberá adjuntar como evidencia una captura de su archivo DAO.

Tercera parte: Conectando la interfaz con el modelo.

Sugerencia de tiempo invertido: 3 horas.

La participante desplegará los datos obtenidos de su base de datos en la interfaz de su aplicación para que el usuario pueda visualizarla. Ya que el proyecto se está construyendo siguiendo la arquitectura MVVM, la participante deberá comentar con la tallerista qué archivos deberán verse modificados.

Una vez identificados los archivos, procederá a realizar los cambios en su proyecto, que para mayor claridad se mencionan a continuación:

- Repositorio

Debido a que el patrón nos indica que el repositorio debe ser el que se comunica con el *ViewModel*, en este se deberán generar métodos análogos a los del DAO, con parámetros y valores de retorno iguales.

- *ViewModel*

Desde *ViewModel* se hará referencia a los métodos del repositorio, así que de nuevo se definirán las acciones para crear, leer, actualizar y eliminar.

Se creará además una variable de tipo *allItems* cuyo valor inicial corresponderá al obtenido como resultado de realizar la consulta a la base de datos. Opcionalmente la variable *allItems* puede ser privada, de hecho constituye una mejor práctica.

- Interfaz

- **Actividad que muestra el *RecyclerView***

- READ

1. Crear una variable llamada "viewModel".
2. En el método *onCreate* asignarle el resultado de llamar a:

`ViewModelProvider(this).get(ClaseDelViewModel::class.java)`

3. En la creación del adaptador, pasarle como lista requerida, el valor de *allItems* que puede consultarse desde la variable *viewModel*.

4. Para observar los cambios en la interfaz, añadir un escuchador a la variable *allItems*, la instrucción debe ser similar a:

```
viewModel.getAllItems().observe(viewLifecycleOwner, Observer {  
    adaptador.notifyDataSetChanged()  
})
```

5. Lo anterior actualizará la interfaz, cuando un cambio de cualquier tipo ocurra.

- **Actividad que muestra los detalles de un contacto del directorio**

- DELETE

1. Crear una variable llamada “viewModel”.
2. En el método *onCreate* asignarle el resultado de llamar a:
`ViewModelProvider(this).get(ClaseDelViewModel::class.java)`
3. Añadir un botón “Eliminar” en la vista.
4. Definir un escuchador de tipo *onClick* para el botón eliminar.
5. Dentro de la implementación del escuchador, llamar al método correspondiente del *ViewModel* para eliminar el elemento.

- UPDATE

1. Añadir un botón “Actualizar”.
2. Añadir un escuchador de tipo *onClick* para el botón actualizar.
3. En la implementación del escuchador, cambiar a la actividad que permite agregar elementos, pasándole la información del elemento seleccionado y una bandera que permita identificar que se trabajará en modo actualizar.

- **Actividad que permite agregar nuevos elementos**

- Crear una variable llamada “viewModel”
 - En el método *onCreate* asignarle el resultado de llamar a:

```
ViewModelProvider(this).get(ClaseDelViewModel::class.java)
```

■ CREATE

1. Modificar la implementación del botón guardar elemento, el cual ahora hará referencia al método insertar de la variable *viewModel*.

■ UPDATE

1. Realizar la validación para el modo actualizar, recuperar el valor del registro.
2. Mostrar la información en los *EditText*.
3. Añadir la validación del modo editar, al presionar el botón guardar, de ser el caso se llamará al método que permite la actualización, contenido en el *ViewModel*.
4. Es importante que el id del elemento a actualizar sea el mismo del objeto recuperado en la actividad, de otro modo podría actualizarse otro registro o generar errores en tiempo de ejecución.

Cuarta parte: Probando mi aplicación.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

1. La participante verificará el correcto funcionamiento de su aplicación, probando las cuatro acciones recién implementadas.
2. De no existir errores, la participante deberá pedir a la tallerista que valide lo anterior, en caso contrario, se deberán corregir los errores hasta obtener el resultado deseado.
3. Al final, los cambios en el proyecto deberán reflejarse en *GitHub* ya que como evidencia de la actividad se deberá adjuntar la liga del repositorio en el documento.

Quinta parte: Áreas de mejora.

Sugerencia de tiempo invertido: 1 hora.

1. La tallerista deberá comentar que el uso de la cláusula *allowMainThreadQueries*, no es muy recomendable cuando los datos crecen en la aplicación. Invitará a la participante a saber el porqué de lo dicho anteriormente, ello a través de la lectura de la sección *Subprocesos* del artículo *Descripción general de los procesos y subprocesos* provisto en los *Links de apoyo*.

2. Por último, la tallerista le mencionará que el enfoque correcto para resolver el problema, es a partir del uso de llamadas asíncronas. Opcionalmente la participante podrá investigar un poco acerca del tema, pudiéndose sugerir también el tópico de *corutinas* en *Kotlin*.

Links de apoyo:

Room

- developers (sin fecha). *Guía de arquitectura de apps (Cómo conservar los datos)*.
<https://developer.android.com/jetpack/docs/guide?hl=es-419> (Septiembre, 2020).
 - *Cómo guardar contenido en una base de datos local con Room*.
<https://developer.android.com/training/data-storage/room?hl=es-419> (Septiembre, 2020).
 - *Descripción general de los procesos y subprocesos*.
<https://developer.android.com/guide/components/processes-and-threads> (Septiembre, 2020).

Actividad 10: Agregar barra de calificación y comentarios

Aprendizaje esperado: Agregar barra de calificación y caja de texto para comentarios breves en cada entrada del directorio.		Duración de la actividad: 5 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia: <ol style="list-style-type: none"> 1. Mostrar en la aplicación una barra de calificación (5 estrellas) en el apartado de la información del establecimiento, así como mostrar un cuadro de texto donde la o el usuario podrá escribir un comentario breve (establecer el límite de caracteres) sobre su experiencia al visitar el lugar. 	Retroalimentación <ul style="list-style-type: none"> • Verificar el funcionamiento adecuado de la barra de calificación cambiando la calificación para algún establecimiento. • Comprobar que los comentarios escritos se guarden correctamente. • Revisar que la información introducida se guarde y se pueda consultar después de cerrar y abrir la aplicación.
Desarrollo de la actividad:		
<p>Primera parte: Los <i>widgets</i>.</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <ol style="list-style-type: none"> 1. La participante leerá el artículo <i>Descripción general de los widgets de apps</i> (disponible en los <i>Links de apoyo</i>) para abordar los siguientes conceptos: <ul style="list-style-type: none"> • <i>Widgets</i>. • Tipos de <i>widgets</i>. • Limitación de los <i>widgets</i>. • Navegación desde <i>widgets</i>. • Consideraciones de diseño de <i>widgets</i>. • Configuración del <i>widget</i>. • Creación de un <i>widget</i> de la <i>app</i>. • La clase <i>AppWidgetProvider</i>. • Fijar <i>widgets</i> de <i>apps</i>. 		

- *RatingBar*.

2. Al finalizar, la participante deberá escribir de manera breve qué utilidad le encuentra a los *widgets* para el desarrollo de su aplicación.

Segunda parte: Cuantificando estrellas.

Sugerencia de tiempo invertido: 2 horas.

En *Android*, *RatingBar* es un control de interfaz de usuario que se usa para obtener la calificación del usuario; es una extensión de *SeekBar* y *ProgressBar* que muestra un punteo con base en estrellas y permite a las y los usuarios establecer el valor de calificación por medio del tacto o el clic directamente en las estrellas.

El *RatingBar* de *Android* siempre devolverá un valor de calificación como un número de punto flotante como 1.0, 2.0, 2.5, 3.0, 3.5, etc. A continuación, se muestra la representación gráfica del uso de *RatingBar* en aplicaciones de *Android*.



En *Android*, por medio del atributo *android: numStars* se puede definir el número de estrellas que se mostrarán en el *RatingBar*. Un ejemplo del uso de *RatingBar* es en sitios de películas o sitios de productos, puesto que se usa para recopilar la calificación de las y los usuarios sobre las películas o productos. Al usar el componente *android.widget.RatingBar* se permite mostrar la barra de clasificación con iconos de estrellas.

1. Como primer paso la participante deberá crear su propio *RatingBar* con cinco estrellas y un botón de envío. Siempre que una o un usuario haga clic en el botón, el valor del número total de estrellas y el valor de la calificación se mostrará mediante un botón en la pantalla.
2. Se deberá crear un nuevo proyecto con el nombre “Barra de Calificaciones”.
3. En *activity_main.xml* se abrirá un archivo *xml* y se agregará el código para mostrar una barra de calificación con cinco estrellas, un valor “0” para la calificación predeterminada y un botón de envío.

4. Se define una barra de calificación (<RatingBar>) con diferentes atributos, estos son los siguientes:

Atributo	Descripción
android: id	Se utiliza para identificar de forma única el control.
Android: numStars	Se utiliza para definir el número de estrellas que se mostrarán.
android: calificación	Se utiliza para establecer el valor de clasificación predeterminado para la barra de clasificación.

5. Se abrirá *MainActivity* para agregar el código e iniciar el *RatingBar* y el botón, posteriormente se agregará un evento al hacer clic en el botón y se mostrará el número total de estrellas y calificación.
6. En *styles.xml* deberá agregarse lo correspondiente para mostrar estrellas personalizadas para la calificación. Para hacer eso en el archivo *styles.xml*, deberá configurarse un archivo *xml* personalizado de *drawable* y configurar la altura y el ancho para ese elemento.
7. Creará un nuevo archivo XML *drawable* -> *customratingstars.xml*.
8. En el nuevo archivo XML dibujable, se deberán establecer los iconos para estrellas llenas y vacías; existen dos iconos diferentes configurados desde *drawable*.
9. Se necesitará cargar el recurso de diseño XML desde el método de devolución de llamada de actividad *onCreate ()*, para este archivo de actividad principal abierto >carpeta java path.
10. Se deberá probar que la aplicación funcione, para ello tendrá que dar clic o tocar diferentes estrellas y que al presionar el botón se muestre la calificación equivalente.

Tercera parte: Los comentarios importan.

Sugerencia de tiempo invertido: 1 hora.

1. Ahora que la barra ha quedado lista la participante deberá añadir un cuadro de texto donde la o el usuario pueda escribir algún comentario acerca del lugar o producto que está calificando, se deberán permitir máximo 150 caracteres. Este comentario deberá enviarse al mismo tiempo que la calificación de la barra de estrellas y con el mismo botón.
2. En el *layout* del proyecto “Barra de calificaciones” se deberá añadir un *EditText* en la parte inferior de la pantalla, configurando los márgenes de la pantalla.
3. Se añadirá un *TextView* al lado del *EditText* que indique la información a rellenar.
4. Se declararán las variables de tipo *EditText* y dentro del método *onCreate* se buscarán por su id.
5. Para guardar la información que la o el usuario colocó en la caja de texto, se usará un *String*. Una vez declarados los *Strings*, se usará una llamada a la propiedad *Text()* el cual permite capturar el texto.
6. Con el método *toString()* comprobará que el texto se guarde correctamente.
7. Una vez que pruebe que se obtiene el valor, se deberá subir a *Room* con un *setValue* a la referencia donde se vayan guardar los comentarios de las y los usuarios.
8. La participante deberá asegurarse que su casilla de comentarios funciona como se ha programado, para ello deberá escribir diferentes comentarios con distintas calificaciones y asegurarse que éstos queden registrados incluso después de cerrar la aplicación.

Cuarta parte: Conectando todo.

Sugerencia de tiempo invertido: 1 hora.

1. El fin de la barra de estrellas y la caja de comentarios es poder calificar un lugar o producto, la participante deberá reflexionar respecto a cómo esta nueva herramienta podría ayudar a mejorar las *apps* que ha creado.
2. Posteriormente, deberá responder las siguientes preguntas:
 - ¿Consideras útil poder calificar los lugares que visitas o productos que consumes?
 - ¿Has visto previamente este sistema de calificación en alguna otra *app*? ¿Lo tomas en cuenta al elegir?
 - ¿En qué *app* te gustaría implementar la barra de calificación y comentarios?

Notas para apoyar la actividad:

Si existen dudas, en la tercera parte de la actividad, revisar la actividad 8 y 9 del módulo que corresponde a Base de Datos, Taller ??)

Links de apoyo:

Widgets:

- developers. (Sin fecha). *Cómo crear un widget de la app*.
<https://developer.android.com/guide/topics/appwidgets?hl=es> (Septiembre, 2020).

RatingBar:

- developers. (Sin fecha). *RatingBar*.
<https://developer.android.com/reference/android/widget/RatingBar> (Septiembre, 2020).

Actividad 11: Compartiendo en redes sociales (Facebook, WhatsApp, Instagram)

Aprendizaje esperado: Compartir la información de la actividad, fotos, ubicación del establecimiento con los contactos de mis redes sociales.		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia: 1. Mostrar en la aplicación un recuadro donde se pueda escribir un mensaje de texto y un botón de “Compartir en” y al hacer clic muestre varias opciones de red social donde se pueda compartir la información del establecimiento con previa autorización de la propietaria.	Retroalimentación: <ul style="list-style-type: none"> • Verificar que la información compartida se muestre correctamente en las distintas redes sociales.
Desarrollo de la actividad:		
Primera parte: Intents y Widgets. Sugerencia de tiempo invertido: 2 horas.		
1. La participante deberá consultar la documentación sugerida (disponible en los <i>Links de apoyo</i>), para responder las siguientes preguntas: <ul style="list-style-type: none"> • ¿Qué son los <i>intents</i>? • ¿Para qué se usan los <i>intents</i>? • ¿Cómo se implementan los <i>intents</i> en una aplicación? • ¿Cuál es la diferencia entre los tres casos principales del uso de <i>intents</i>: iniciar una actividad, iniciar un servicio, transmitir una emisión? • ¿Qué tipos de <i>intents</i> existen? • ¿Cómo se crea un <i>intent</i>? • ¿Qué son los <i>intents</i> independientes? • ¿Qué son los <i>widgets</i>? 		

- ¿Para qué se usan los *widgets*?
 - ¿Cómo se implementan los *widgets* en una aplicación?
 - ¿Cómo enviar datos simples a otra aplicación?
 - ¿Qué es *Android Share Sheet*?
 - ¿Cómo usar *Android Share Sheet*?
 - ¿Cómo enviar contenido de texto?
 - ¿Cómo enviar contenido binario?
 - ¿Cómo compartir varios contenidos?
2. Al finalizar la participante deberá escribir de manera breve qué *intents* o *widgets* desearía implementar para su aplicación y justificarlo.

Segunda parte: Probando *intents*.

Sugerencia de tiempo invertido: 3 horas.

En *Android*, la habilidad de enviar mensajes es posible por medio del objeto *Intent*. Con la ayuda de los *intents*, los componentes de *Android* pueden solicitar funcionalidad de otros componentes *Android*. Cuando se abre la aplicación *Instagram* en un teléfono y se usa para tomar una foto, se hace uso de un *intent*. Los *intents* también ayudan a comunicar entre partes de una *app*; el movimiento de una pantalla (actividad) a otra es posible mediante *intents*.

Desde esta perspectiva: todos los componentes (aplicaciones y pantallas) del dispositivo *Android* están aisladas. La única manera de comunicarse entre ellas es a través de *intents*.

1. La participante creará un nuevo proyecto en blanco al que nombrará «Petshow». La aplicación tendrá como finalidad practicar el uso de *intents*, para ello, la aplicación deberá mostrar una foto de una mascota o animal de su preferencia al momento de presionar un botón.
2. La aplicación se compondrá de dos actividades:
 - La primera será *Main.java*, la cual se producirá automáticamente al crear el nuevo proyecto, donde se deberá ubicar un botón.
 - La segunda actividad será creada por la participante. La idea es que se inicie al presionar el botón a través de un *Intent*.

3. A continuación creará una nueva actividad en blanco. Para ello irá a la raíz de sus archivos *java* y presionará «New». En ese nuevo menú presionará «Activity» y en seguida «Blank Activity»:
4. Se visualizará un asistente que proporcionará un formulario para personalizar la nueva actividad:
 - El primer parámetro será «Activity name.» El nombre que se pondrá a la actividad, será «Visor».
 - El segundo parámetro será «Layout Name». Se dejará el valor actual recomendado por *Android Studio*.
 - El tercer parámetro será «Title», es decir, el título que se mostrará al ejecutar la actividad en la parte superior. De igual forma se dejará el valor recomendado.
 - Para el cuarto parámetro se desmarcará la opción «Launcher Activity», ya que la actividad principal es *Main.java*.
 - El quinto parámetro será para indicar si la nueva actividad es *hija* de otra actividad. Esto significa que están relacionadas jerárquicamente en la navegación de la interfaz. Para seleccionar la actividad *padre* se pulsará «...» y se seleccionará a *Main.java*. Si se habilita esta opción se podrá hacer uso del *Up Button*. Este botón es autogenerado por las aplicaciones *Android* cuando se desea regresar de una actividad a otra. Normalmente se encuentra en la parte superior izquierda de la cabecera.
 - Y el sexto y último parámetro es el paquete donde se ubicará la actividad, por lo que se dejará el paquete por defecto. Luego se presionará «Finish» y la nueva clase será creada.
5. En el archivo *Android Manifest* se verá la nueva actividad agregada.
6. La actividad *Visor* tendrá un nuevo atributo llamado *android:parentActivityName*, el cual contendrá el nombre de la actividad *padre*. La cual será «.Main».
7. Para concluir esta relación se añadirá un elemento *<meta-data>* a la actividad y se especificará el nombre de la acción *PARENT_ACTIVITY* y el paquete donde se encuentra la clase «*com.herprogramacion.petmotion.Main*».
8. El siguiente paso será diseñar los *layouts* de ambas actividades. Para la actividad principal se agregará un botón con el identificador *show_pet_button* y el texto «Mostrar mascota».
9. Después se agregará un *ImageView* en *Visor*.
10. Cuando el usuario presione el botón *show_pet_button* de *Main*, será dirigido a *Visor* para visualizar la imagen; para esto será necesario añadir un *ImageView*.
11. Deberá ubicarse en la carpeta «*drawable-mdpi*» presionar clic derecho. Elegirá la opción «Show in Explorer»; a continuación, aparecerá la carpeta de los recursos (*res*) del proyecto.
12. Si se abre «*drawable-mdpi*» se encontrará el archivo «*ic_launcher.png*» que representa al icono de la aplicación. Junto a él deberá pegar la imagen de la mascota o animal que le guste.
13. Automáticamente *Android Studio* refrescará el proyecto y mostrará el archivo *pet1.jpg* en la carpeta de recursos.

14. Una vez ubicada la imagen, deberá situarse en su atributo *src* en el *panel de propiedades*. El atributo *src* representa la dirección del recurso (*source*) o imagen que se usará para visualizar en el *ImageView*; deberá presionar el botón «...».
15. En seguida se mostrará un asistente para pickear la imagen que se necesita ubicada en el proyecto. Se escogerá a la foto de la mascota o animal y presionará «OK».
16. Se deberá revisar cómo se visualiza el diseño.
17. El paso final será iniciar la actividad del *Visor* una vez sea presionado el botón *show_pet_button*. Para ello se usará el método *startActivity()* de la clase *Activity*. Este método recibe como parámetro de entrada un *Intent*. Este *intent* será el encargado de portar el mensaje para ejecutar la acción, que en este caso es iniciar una actividad.
18. El constructor de *intent* recibirá dos parámetros.
 - El primero será el contexto desde donde se desea enviar el mensaje, en este caso es la propia actividad.
 - El segundo hará referencia a la clase del componente receptor del mensaje de inicio, en este caso será la clase *Visor*.
19. Antes de iniciar la actividad se deberán adherir los datos al *intent* con el método *putExtra()*. Este método es polimórfico y es capaz de enviar varios tipos de datos.
20. Para mostrar su uso se agregará un *TextView* que mostrará la cadena en su atributo *text*. Esta cadena será el nombre de la imagen que se va a mostrar, que en este caso es «*pet1.jpg*».
21. Dentro de la actividad *Main* se declarará una constante de tipo *String* que represente el identificador único del dato *Extra* para el *Intent*. Esto le permitirá diferenciarse de otros pares.
22. Es importante que se encuentre una cadena que sea única para que no interfiera con otros datos, para ello se elegirá el nombre del paquete del proyecto para asegurar que no se repita. Posteriormente se implementará el método *putExtra()* antes de iniciar la actividad.
23. Se invocará el método *getIntent()* de la clase *Activity* y se obtendrá el *intent* que inició la actividad. Todas las actividades son iniciadas por un *intent*; se puede obtener el *Intent* de la actividad principal iniciado por el sistema.
24. Desde el método *onCreate()* de *Visor* se obtendrá el *intent*. A continuación, se extraerá la cadena del mensaje que se envió con *getStringExtra()* y se le asignará al atributo *text* del *TextView*:
25. Para finalizar, se probará el funcionamiento de la aplicación y se tomará una captura de pantalla que muestre el funcionamiento programado.

Tercera parte: Implementando mis *intents*.

Sugerencia de tiempo invertido: 1 hora.

1. Ya que la participante se haya relacionado con los *intents* deberá implementarlos para compartir información en sus redes sociales. En la actualidad, las y los usuarios de aplicaciones son cada vez más sociales y han desarrollado una necesidad por compartir aspectos de su vida privada en redes. Por lo que se ha hecho cada vez más común incluir la función de compartir contenido en redes sociales, en cada una de las aplicaciones que se desarrollan.
2. Las redes sociales que la participante decida usar para esta actividad son de libre elección, se sugiere *Facebook, Instagram, Twitter, Facebook, Messenger y Tik Tok*. Pero las que seleccione deben estar previamente instaladas en su teléfono.
3. Si lo desea puede crear un nuevo proyecto en *Android Studio* o usar la aplicación de *PetShow* que creó anteriormente y hacer las modificaciones correspondientes para compartir imágenes en sus redes sociales.
4. Será necesario que la aplicación cuente con un recuadro donde se pueda escribir un mensaje de texto y un botón de "Compartir en" para que al presionarlo se desplieguen las apps disponibles para que se comparta la imagen y el texto previamente escrito.
 - Para especificar la red social donde compartirá la información (si solo se requiere una en específico) el código correspondiente será: `intent.setPackage("com.whatsapp");`
 - Si se desea compartir en más de una red social solo será necesario agregar en los diferentes Package: `intent.setPackage("com.twitter.android")`
 - La aplicación deberá tener permisos de internet en el *manifest*.
5. La participante deberá verificar el funcionamiento de su app, al compartir distintos mensajes en sus redes sociales y compruebe que estos se visualicen.

Cuarta parte: Mi directorio.

Sugerencia de tiempo invertido: 1 hora.

1. La participante ha desarrollado una aplicación previamente (se comenzó en la actividad *MMT3A7*) la cual hace referencia a un directorio que guarda el contacto y negocio de las mujeres que conforman la red de confianza de las participantes.
2. Ahora bien, lo siguiente que deberá realizar será agregar los *intents* correspondientes para mejorar la *app* y poder compartir esos contactos.
3. Deberá abrir el proyecto que contiene la aplicación.
4. Dentro de la actividad *Main* se declarará una constante de tipo *String* que represente el identificador único del dato *Extra* para el *Intent*. Esto le permitirá diferenciarse de otros pares.

5. Se deberán agregar los *intents* necesarios para compartir la *app* a contactos específicos de sus redes sociales.
6. Añadirá un botón que se llame “compartir en” el cual desplegará una lista de opciones de las redes sociales de la participante donde puede compartir la información que desee (sea un contacto o la ubicación de un negocio).
7. Deberá revisar *Androidmanifest* para asegurarse que se tienen los permisos necesarios.
8. La participante tendrá que probar esta nueva implementación de la *app*, deberá al enviar diferentes contactos y negocios que son de su agrado a sus redes sociales con el objetivo de recomendarlos con su comunidad de alianzas (amigas, compañeras, colegas, etc.).

Nota:

Para la tercera parte de la actividad

- Es importante que la participante no comparta información en redes sociales que pueda ser de carácter sensible o que la pueda poner en alguna situación de riesgo a ella o a otras personas.

Links de apoyo:

Datos a otras apps:

- developers. (Sin fecha). *Cómo enviar datos simples a otras apps*.
<https://developer.android.com/training/sharing/send> (Septiembre, 2020).

Intents:

- developers (Sin fecha) *Intents comunes*.
<https://developer.android.com/guide/components/intents-common?hl=es-419> (Septiembre, 2020).
- Universidad de Alicante.Dept. Ciencia de la Computación e IA. (2012). *Experto en Desarrollo de Aplicaciones para Dispositivos Móviles. Intents y navegación entre actividades*.
<http://www.jtech.ua.es/dadm/restringido/android/sesion02-apuntes.html> (Septiembre, 2020).

Actividad 12: Privacidad y seguridad de los datos de mi aplicación

Aprendizaje esperado: Identificar en la aplicación desarrollada elementos básicos e importantes de seguridad de la aplicación y privacidad de los datos		Duración de la actividad: 7 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> • Computadora PILARES. • IDE Android Studio. • Dispositivo móvil y/o Emulador. 	Evidencia: <ol style="list-style-type: none"> 1. Mostrar una tabla donde se describa las ventajas y desventajas de otorgar permisos de uso de distintos componentes del teléfono celular como (camara, microfono, gps, compartir) en tres distintas aplicaciones considerando la aplicación desarrollada. 2. Establecer características importantes de la seguridad que proporciona <i>Android</i>. 	Retroalimentación: <ul style="list-style-type: none"> • Verificar las tablas generadas por la participante y comentar las distintas ventajas y desventajas que se piden. • Retroalimentar los conocimientos en las áreas de privacidad y seguridad de la aplicación.
Desarrollo de la actividad:		
<p>Primera parte: Seguridad y alianzas.</p> <p>Sugerencia de tiempo invertido: 1 hora.</p> <p>Diferenciar entre espacio público y privado se ha vuelto cada vez más difícil gracias a la incorporación del internet a la vida cotidiana; el surgimiento de las redes sociales ha facilitado el compartir aspectos privados de la vida en espacios que pueden ser poco seguros; ya que se puede exponer información sensible sin intención, lo que puede vulnerar la seguridad e integridad de las personas en especial de las mujeres.</p> <ol style="list-style-type: none"> 1. Se solicitará a la participante que realice un a breve reflexión personal entorno a la seguridad y que responda las siguientes preguntas: <ul style="list-style-type: none"> • ¿Qué es la seguridad para ti? • ¿Consideras que el internet puede ser un espacio no seguro? • ¿Qué información personal consideras que puede vulnerar tu seguridad si es expuesta en internet? 		

- ¿Conoces algún caso (cercano o no) en el que se haya vulnerado la seguridad de una mujer en internet?
- ¿Cómo consideras que las alianzas con otras mujeres pueden ayudarte a fortalecer tu seguridad tanto en internet como en la vida?
- ¿Qué aspectos consideras que debe tener la seguridad para las mujeres?

Segunda parte: Seguridad en las *apps*.

Sugerencia de tiempo invertido: 1 hora.

1. La participante deberá leer el artículo *Recomendaciones sobre seguridad de apps* (disponible en los *Links de apoyo*) para responder las siguientes preguntas:
 - ¿Qué prácticas recomendadas tienen un impacto positivo en la seguridad de una *app*?
 - ¿Qué son los permisos basados en firmas?
 - ¿Cuáles son las medidas de seguridad de tu red?
 - ¿Qué es un administrador de confianza?
 - ¿Qué tipo de permisos deben tener las *apps*?
 - ¿Cuáles son las principales funciones de seguridad?
 - ¿Qué son las validaciones de entrada?
 - ¿Cómo administrar credenciales?
 - ¿Cómo usar la criptografía?
 - ¿Qué es la biblioteca de seguridad de *Android Studio*?
 - ¿Qué es *malware*?
 - ¿Cuáles son las sugerencias de seguridad en *Android Studio*?
2. La participante deberá realizar una búsqueda en internet sobre las apps detectadas con *malware* dentro de *playstore*, deberá investigar qué daños causaron estas *apps* al estar instaladas en los dispositivos de los usuarios.
3. Discutirá con la tallerista y/o compañeras si considera que la seguridad es un punto importante en la creación de las aplicaciones.

Tercera parte: Elementos de seguridad en mi aplicación.

Sugerencia de tiempo invertido: 2 horas.

Si se aumenta la seguridad de una *app*, se ayuda a preservar la confianza de los usuarios y la integridad del dispositivo. *Android Studio* considera cuatro aspectos básicos durante el desarrollo de aplicaciones, los cuales también se consideran prácticas recomendadas de compilación de *apps* seguras para *Android*.

1. La participante deberá de verificar que su *app* cumpla con los siguientes puntos:
 - Almacena los datos de manera segura. Minimiza el uso de API sensibles y verifica los datos del almacenamiento externo antes de usarlos.
 - Implementa la comunicación segura. La *app* usa HTTPS/SSL para proteger los datos en la red.
 - Actualiza el proveedor de seguridad para protegerte contra vulnerabilidades de SSL. Actualiza automáticamente el proveedor de seguridad de un dispositivo para proteger contra vulnerabilidades.
 - Presta atención a los permisos. Solo usa los permisos necesarios y presta atención a los que sus bibliotecas puedan usar.
2. De no ser así, deberá implementarlos para volverla segura de utilizar.

Cuarta parte: Permisos y seguridad.

Sugerencia de tiempo invertido: 2 horas.

La mayor parte de los usuarios no toma precauciones a la hora de utilizar las redes sociales y suele compartir su ubicación en fotografías, vídeos, emisiones en *streaming* o publicaciones en sus plataformas favoritas. A fin de proteger la privacidad del usuario, las *apps* que usan servicios de ubicación deben solicitar permisos de ubicación, así como las de fotografía y video solicitan permisos de acceso a la cámara o micrófono.

Cada *app* de *Android* se ejecuta en una zona de pruebas con acceso limitado. Si una *app* necesita usar recursos o información ajenos a su propia zona de pruebas, debe solicitar el *permiso* correspondiente al usuario. Para declarar que la *app* necesita un permiso, se debe incluir en el manifiesto de la *app* y luego solicitar al usuario que apruebe cada permiso durante el tiempo de ejecución (en *Android 6.0* y versiones posteriores).

1. La participante deberá implementar los principios básicos de permisos en su *app*, para garantizar al usuario que sus datos están protegidos. Para ello la *app* deberá cumplir con los siguientes puntos:
 - Solicitar permisos en contexto cuando el usuario comience a interactuar con la función que lo requiere. Ejemplo: Permiso de ubicación si usará el GPS, permiso de acceso a cámara si tomara una foto.
 - No bloquear al usuario; proporcionar siempre la opción de cancelar un flujo de IU educativo relacionado con los permisos.

- Degradación elegante de la *app*; si el usuario rechaza o revoca un permiso que necesita una función, se debe realizar una degradación elegante de la *app* para que pueda seguir usándola; para ello se puede inhabilitar la función que requiere el permiso.
 - No dar por sentado ningún comportamiento del sistema.
 - No dejar los permisos abiertos en segundo plano, estos deben finalizar cuando se cierre la aplicación.
 - No solicitar permisos extras; la *app* solo debe solicitar los permisos que realmente sean indispensables para el funcionamiento de la aplicación.
2. La participante creará una tabla donde se describa las ventajas y desventajas de otorgar permisos de uso de distintos componentes del teléfono celular como (cámara, micrófono, gps, compartir) en su aplicación y en 2 aplicaciones diferentes.

Quinta parte: Seguridad de una aplicación en Android.

Sugerencia de tiempo invertido: 1 hora.

Android tiene funciones de seguridad integradas que reducen significativamente la frecuencia y el impacto de los problemas de seguridad de las aplicaciones. El sistema está diseñado con el objetivo de que se puedan compilar las *apps* con permisos predeterminados del sistema, esto le proporciona al usuario que no deba tomar decisiones difíciles sobre seguridad.

SafetyNet proporciona un conjunto de servicios y API que ayudan a proteger la *app* contra amenazas de seguridad, lo que incluye la manipulación del dispositivo, URL incorrectas, *apps* potencialmente dañinas y usuarios falsos.

1. La participante deberá consultar en la documentación de *Android Studio* las funciones de seguridad principales para compilar *apps* seguras (disponible en los *Links de apoyo*).
2. Posteriormente, redactará en un archivo de texto una respuesta argumentada que responda las siguientes preguntas:
 - ¿Por qué es importante que se conozcan las prácticas recomendadas de *Android*?
 - ¿Por qué se deben seguir estas prácticas como hábitos de codificación generales?
 - ¿Seguir las prácticas recomendadas de *Android* reduce las probabilidades de introducir de forma inadvertida problemas de seguridad que perjudiquen a los usuarios? ¿Por qué?

Links de apoyo:

Sugerencias de seguridad:

- developers. (Sin fecha). *Sugerencias de seguridad*.
<https://developer.android.com/training/articles/security-tips?hl=es-419> (Septiembre, 2020).

Recomendaciones sobre seguridad de apps:

- developers. (Sin fecha). *Recomendaciones sobre seguridad de apps*.
<https://developer.android.com/topic/security/best-practices?hl=es-419> (Septiembre, 2020).

Permisos:

- developers. (Sin fecha). *Cómo solicitar permisos de la app*.
<https://developer.android.com/training/permissions/requesting?hl=es-419> (Septiembre, 2020).

Seguridad con SafetyNet:

- developers. (Sin fecha). *Recomendaciones sobre seguridad de apps*.
<https://developer.android.com/topic/security/best-practices?hl=es-419> (Septiembre, 2020).

Actividad 13: Mejorando mi aplicación

Aprendizaje esperado: Uso de herramientas desarrolladas en actividades anteriores para mejorar la apariencia y funcionalidad de la aplicación.		Duración de la actividad: 9 horas.
Recursos	Evidencia/producto	Retroalimentación/Evaluación
<ul style="list-style-type: none"> ● Computadora PILARES. ● IDE <i>Android Studio</i>. ● Dispositivo móvil y/o Emulador. 	Evidencia: <ol style="list-style-type: none"> 1. Establecer cambios en el diseño y funcionamiento de la aplicación que mejoren su aspecto y funcionalidad, de acuerdo con el criterio de la participante. 2. Hacer una lista de los cambios hechos y su justificación. 3. Responder las distintas preguntas planteadas: <ol style="list-style-type: none"> a) ¿Qué necesito para publicar mi aplicación y que esté disponible para otros usuarios? b) ¿Qué funcionalidades necesitaría agregar en caso de que la aplicación fuera usada por muchos usuarios? c) Reflexionar sobre cómo me imagino que esta aplicación puede ayudar a las mujeres que participan en alguna actividad económica. d) ¿Qué otra aplicación se te ocurre que podría ser útil? 	Retroalimentación: <ul style="list-style-type: none"> ● Se deben revisar los cambios hechos en la aplicación que representen una mejora. ● Revisar la lista de la justificación de dichos cambios y retroalimentar. ● Revisar las respuestas de la participante y retroalimentar sobre el desarrollo y utilidad de la aplicación de acuerdo con su objetivo.

Desarrollo de la actividad:

Primera parte: Expandiendo horizontes.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante conocerá parte de un día común dentro de un equipo multidisciplinario de desarrollo de *software*, los miembros que lo integran y las áreas en las que se enfocan.

1. Seleccionará uno de los videos que se te presentan en los *Links de apoyo* y en un documento contestará: ¿Qué otras profesiones se ven envueltas en el desarrollo de aplicaciones?
2. Completará la siguiente lista de puestos en un equipo de desarrollo de *software* investigando características como: los salarios aproximados en bolsas de trabajo, la función que desempeña dentro del equipo, los conocimientos y habilidades con las que debe contar, etc.
 - a. UX/UI.
 - b. *Project Manager*.
 - c. Arquitecta de *Software*.
3. ¿Qué otras tecnologías se utilizan en la creación de aplicaciones? Realiza una breve búsqueda en la *web* e identifica sus características principales (*IOS, Flutter, Xamarin, React*).

Segunda parte: Cuidando el diseño de mi aplicación.

Sugerencia de tiempo invertido: 1 hora y 30 minutos.

La participante identificará las características de diseño en *apps* y el balance que tienen con su usabilidad, para ello seguirá los pasos que se presentan a continuación.

- *Material Design* y su relación con el diseño de mi aplicación.
 1. ¿Por qué es importante UX/UI?
 - a. ¿Hay alguna aplicación que hayas dejado de usar porque era difícil de entender?
 - b. ¿Consideras que la estética de una aplicación es importante para que la consumas?

- c. Anota tus tres aplicaciones favoritas y reflexiona respecto a su usabilidad y estética.
 - d. Investiga la importancia del desarrollo UX/UI en proyectos.
2. ¿Qué es *Material Design*?
 3. Investigar principios de *Material Design*.
 4. Ver aplicaciones muestra (consultar en los *Links de apoyo* el artículo: *Aplicaciones de muestra, Material Design > Example apps*).
 5. Ver ejemplos de UX/UI en aplicaciones móviles (vídeos o imágenes).
- Componentes de *Material Design*.
 1. Revisa en los *Links de apoyo* el recurso: *Componentes de Material Design*.
 2. Observa los elementos e identifica cuáles son familiares y enuncia la aplicación en la que los has visualizado.
 3. Investiga la función de aquellos que no reconozcas y en qué ocasiones se usan.
 4. ¿Qué elementos ya implementaste en tu *app*? ¿En dónde?
 5. ¿Cuáles componentes llaman tu atención? ¿Por qué?

Tercera parte: Mejora continua.

Sugerencia de tiempo invertido: 2 horas.

La participante identificará los puntos de mejora de la *app* que desarrolló durante el taller. A continuación, se presenta una serie de pasos que la participante deberá tomar en cuenta.

1. Hasta ahora el proyecto realizado trabaja con una base de datos local, ¿qué significa esto? Realiza los siguientes pasos para averiguarlo:
 - a. Instala la aplicación de directorio en 2 dispositivos diferentes.
 - b. Agrega 3 registros en cada uno de los dispositivos.
 - c. ¿Puedes ver los registros que se agregaron desde el otro dispositivo?
 - d. ¿Los dispositivos comparten la misma base de datos?
 - e. Define con tus propias palabras, ¿qué es una base de datos local?
 - f. ¿Es este el comportamiento deseable para tu aplicación?

2. ¿Qué funcionalidades necesitaría agregar en caso de que la aplicación fuera usada por muchos usuarios? Menciona al menos tres.
3. ¿Qué otras mejoras de diseño se podrían implementar en la aplicación actual sin realizar cambios en su funcionalidad? (Puede consultar el recurso: *Componentes de Material Design* disponible en los *Links de apoyo*).
4. ¿Qué nuevas mejoras, en cuanto a funcionalidad se podrían agregar? Menciona al menos tres siguiendo el objetivo de la *app*.
5. ¿Qué componentes de *Material Design* agregaría y para qué tarea? Completa la siguiente tabla con al menos tres ejemplos.
Ejemplo: Agregar comentarios, botones de redes sociales, compras, redirigir a las páginas, etc.

Nueva funcionalidad	Componente	¿Por qué es importante este cambio?

6. ¿Cómo me imagino que mi aplicación y otras similares pueden ayudar a las mujeres que participan en alguna actividad económica? Puedes discutir al respecto con la tallerista u otras participantes y anotar las conclusiones a las que se llegue.
7. ¿Qué otra aplicación podría ser útil para el mismo fin o uno similar? Realiza una breve búsqueda en la plataforma de *apps* de tu preferencia y haz mención de ellas.

Cuarta parte: Más recursos de diseño.

Sugerencia de tiempo invertido: 1 hora.

La participante explorará más alternativas que permitan la mejora de su aplicación, a través de la incorporación de librerías. Para ello seguirá los pasos que se presentan a continuación:

1. Investiga tres librerías que aporten al diseño de la aplicación.
2. Investiga tres librerías que añadan valor a una aplicación (no en el apartado visual, enfocado a la funcionalidad).
3. En caso de que las características se ajusten a las necesidades de la aplicación de directorio, se debe asociar la librería adecuada con una nueva funcionalidad o elemento de diseño a implementar. Con el fin de concentrar la información en un cuadro como el siguiente:

Nueva funcionalidad	Librería

Quinta parte: Animando mi *app*.

Sugerencia de tiempo invertido: 2 horas.

Con el objetivo de obtener un breve panorama sobre dos formas distintas de incorporar animaciones en una aplicación, la participante llevará a cabo la siguiente secuencia:

1. Leer el recurso *Introducción a las animaciones*, el cual le ayudará a conocer los distintos tipos de animaciones que puede agregar a su aplicación.
2. Explorar otra alternativa que facilita la incorporación de animaciones sencillas, a través de la librería *Lottie*. Se deberán contestar las siguientes preguntas:
 - a. ¿Qué es *Lottie*?
 - b. ¿Cuál es su principal ventaja?
 - c. ¿Cómo puedo crear animaciones para *Lottie*?
 - d. ¿Cómo puedo agregar una animación de *Lottie* a mi proyecto *Android*? (Esta pregunta cuenta con un recurso sugerido, mismo que debe cambiarse a español en la barra de *Google*).
3. Implementar una animación sencilla con *Lottie* para agregar un botón “favorito” a su directorio. En los recursos se encuentra el enlace que conduce a las animaciones relacionadas con el objetivo planteado, ahí seleccionará la que más se ajuste a sus gustos.
Después de seleccionar una animación, la participante retomará lo investigado en el inciso d) del punto 2 para efectuar los pasos que conduzcan a la implementación.
4. Opcionalmente, podrá explorar e implementar más animaciones, cuidando que estas agreguen valor a su aplicación.

La tallerista deberá destacar la importancia de agregar animaciones en las situaciones adecuadas. Agregar muchas animaciones, no siempre es la mejor opción (a veces menos es más), debe existir un balance entre la función y el diseño de los elementos contenidos en una aplicación.

Sexta parte: Publicando mi aplicación.

Sugerencia de tiempo invertido: 1 hora.

1. La participante investigará los lineamientos y pasos a seguir para llevar a cabo la publicación de una aplicación. Se recomienda consultar la documentación oficial, complementando la información con vídeos, tutoriales u otros recursos que se consideren convenientes.
2. La actividad está enfocada, únicamente a conocer el proceso. No es necesario seguir o completar ningún tutorial, ya que el cumplimiento total del proceso involucra la realización de un pago.
3. Como evidencia la participante deberá realizar una lista con los pasos identificados.

Links de apoyo:

Vida laboral en algunas empresas

- Life at Google. (2017). *Meet UX Designers at Google*. [video]
<https://www.youtube.com/watch?v=116sMd5U7UY> (Noviembre 2020).
 - (2018). *What's it like to work at Google?* [video]
https://www.youtube.com/watch?v=n_Cn8eFo7u8 (Noviembre 2020).
- BBC. (2019). *The reality of working for Facebook-BBC*. [video]
<https://www.youtube.com/watch?v=KgrNOhafmwo> (Noviembre de 2020).
- hunbuns (2020). *A day in the life of a UX Designers-what I do day to day*. [video]
<https://www.youtube.com/watch?v=gEYq6GFAFCs&t=249s>

Aplicaciones de muestra, Material Design.

- material.io. (Sin fecha). *Resources*.
<https://material.io/resources> (Noviembre 2020).

- NAM Desingn. (2020). *Best 20 Example UI/UX Design For Mobile App | UI/UX Animation Design*. [video] <https://www.youtube.com/watch?v=d6xn5uflUjg&t=31s> (Noviembre 2020).

Componentes de Material Design.

- material.io. (Sin fecha). *Components*. <https://material.io/components> (Noviembre 2020).

Ejemplo de librerías.

- Ertzil. (Sin fecha). *Librerías de diseño*. <https://baturamobile.com/blog/librerias-de-diseno/> (Noviembre 2020).
- developers. (Sin fecha). *Introducción a las animaciones*. <https://developer.android.com/training/animation/overview?hl=es-419> (Noviembre 2020).

Animación “favorito”.

- LottieFiles. (Sin fecha). *Search results for “fav”*. <https://lottiefiles.com/search?q=fav&category=animations> (Noviembre 2020).

Añadir animaciones a Android.

- airbnb.io. (Sin fecha). *Getting Started*. <http://airbnb.io/lottie/#/android> (Noviembre 2020).

Subir una app a la Play Store.

- Ayuda de Play Console. (Sin fecha). *Crear y configurar tu app*. <https://support.google.com/googleplay/android-developer/answer/113469?hl=es-419> (Noviembre 2020).



Escuela de Código para PILARES Descripción de actividades - Parte 4: Desarrollo de aplicaciones móviles (MM) por Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México se distribuye bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).