UNIVERSIDAD GERARDO BARRIOS.

Facultad: Ciencia y Tecnología.

Carrera: Ingeniería en Sistemas y Redes.

Asignatura: Programación Computacional II.

Docente: Inga. Gisela Espinoza

Actividad: Investigación de "Componentes de un proyecto".

Estudiante:

- Josué Amílcar López Benítez SMIS935820

Fecha de entrega:

Domingo 24 / Enero / 2021

DESARROLLO DE LA INVESTIGACIÓN

Primeramente debemos resaltar la idea fundamental detrás de esta investigación, y es que si nos ponemos a pensar un momento, nos daremos cuenta de la realidad tecnológica en la que vivimos inmersos día a día y como la tecnología ha formado parte "necesaria" de nuestra sociedad, forjando con ello nuevas aptitudes en el desarrollo del ser humano mismo.

A su vez esto ha convertido a los sistemas operativos que son destinados a ser utilizados por "celulares o dispositivos móviles" en los sistemas de mayor frecuencia en la actualidad, todo gracias a sus diversas características que lo hacen tener una gran influencia en el área socio-cultural del cual somos participantes y como bien conocemos esto abre las puertas al ambiente tecnológico que es ya de por sí muy explotado con el desarrollo de nuevas tecnologías de comunicación… Siendo aún más centralistas, podemos observar el amplio campo de las oportunidades laborales que trae consigo el aprovechamiento de estas herramientas, en donde su adaptabilidad, portabilidad, mecanismos e incursión, eficacia y eficiencia son de gran importancia para la sociedad que sigue buscando esa innovación entorno a la gran curiosidad tecnológica y lo que esto trae consigo…

Entrando en materia de la investigación, podemos justificarla con el hecho de que debemos desarrollar conocimientos que nos servirán como nuestra base fundamental para los aspectos teóricos y prácticos, los cuales estaremos cursando con el desarrollo de las actividades ligadas a la construcción de software nativo o hibrido para las diversas plataformas móviles que existen, con ello aumentando las oportunidades en las áreas relacionadas al ambiente académico y laboral.

Así sin dilatarme más de lo esperado, es necesario que tengamos un trasfondo del área de estudio en el cual emprenderemos dicho aprendizaje; Hablando de "Android" podemos describirlo de una forma simple como: 'Un ambiente (Sistema Operativo) potenciador para los programadores por su entorno de desarrollo integrado, que puede abarcar una alta gamma de herramientas y lenguajes de programación, lo que ayuda a pulir conceptos y aprender más temas relacionados a la programación de aplicaciones móviles'. De esta forma pasamos a la descripción de los siguientes conceptos que nos servirán como soporte al momento de recibir las clases y desarrollar sus respectivas prácticas….

1. **Activity:**

La clase Activity es un componente clave de una App para Android, y la forma en que se inician y se crean las actividades son una parte fundamental del modelo de aplicación para el ambiente de desarrollo Android. A diferencia de los paradigmas de programación en los que las Apps se inician con un método main(), el sistema Android inicia el código en una instancia de "Activity" invocando métodos de devolución de llamada específicos que corresponden a las etapas específicas de su ciclo de vida. Entonces de cierto modo una Activity en Android se corresponde con una pantalla de nuestra App que sirve como punto de entrada para que el mismo Sistema Operativo pueda cargarla en cualquier momento y se compone de:

- **Una clase:** normalmente se extiende de la AppCompatactivity y es donde definimos el código de lo que queremos que haga nuestra aplicación móvil.
- **Layout:** Su función es identificar la apariencia de la vista y el diseño.
- **Una definición de uso:** Realizada comúnmente en el AndroidManifest.

La clase Activity está diseñada para facilitar el paradigma generado cuando una App realiza una llamada invocando una actividad en otra, en lugar de la App en sí mismo y de esta forma la actividad sirve como punto de entrada para la integración de la aplicación con el usuario. Además de ello generalmente una actividad puede generar una pantalla en una App y de igual forma las aplicaciones pueden incluir dentro de sí, varias actividades que trabajan en conjunto a fin de crear una experiencia al usuario que sea coherente con el software al cual tienen acceso. En cuánto a la forma de declarar una actividad se debe hacer agregándolo en el archivo de "manifiesto" (AndroidManifest) y agrega el elemento <activity> como objeto secundario del elemento <application>.

Activity a su vez tendrá un único atributo obligatorio el cual es android:name y servirá para especificar el nombre de la clase de la actividad. A continuación se detalla el ciclo de vida de una actividad (Activity):

- OnCreate(): Es el primer paso, por lo tanto, se activa cuando el sistema crea tu actividad.

- OnStart(): La actividad pasa al estado Iniciada y se vuelve visible para el usuario. Esta devolución de llamada contiene los preparativos finales de la actividad para pasar al primer plano y convertirse en interactiva.
- OnResume(): Se utiliza cuando la actividad se reanuda. onResume() se puede llamar cuando algo se pone por encima de nuestra activity (por ejemplo un diálogo), pero la activity se sigue viendo.
- OnPause(): El sistema llama a onPause() cuando la actividad pierde el foco y pasa al estado Detenida. Este estado se produce, por ejemplo, cuando el usuario presiona el botón Atrás o Recientes.
- OnStop(): El sistema llama a onStop() cuando la actividad ya no es visible para el usuario, lo cual puede ocurrir porque se está eliminando la actividad, porque se inicia una nueva o porque una ya existente pasa al estado Reanudada y cubre la actividad que se detuvo.
- OnRestart(): El sistema invoca esta devolución de llamada cuando una actividad en estado Detenida está por volver a iniciarse. onRestart() restaura el estado de la actividad desde el momento en que esta se detuvo.
- OnDestroy():El sistema invoca esta devolución de llamada antes de que se elimine una actividad

2. Intent:

Los Intents son un tema fundamental para los desarrolladores que trabajan bajo el ambiente de Android, ya que es imposible construir aplicaciones Android sin entrar en contacto con Intents, en donde su característica principal consiste en ayudar a que los componentes de android pueden solicitar funcionalidades de otros componentes Android. Por ejemplo Cuando abres la aplicación Instagram en tu teléfono y la usas para tomar una foto, has hecho uso de un intent ya que solicitas a una aplicación la funcionalidad de otra App; De esta misma forma los Intents se pueden utilizar para ayudar a comunicar las partes de una misma aplicación y el ejemplo más claro se puede ver aplicado en el movimiento de una pantalla (actividad) a otra, lo cual es posible gracias al enlace de conexión que

proporcionan los Intents y estos a su vez se encargan de contrarrestar el “aislamiento” de cada componente que conforma la App.

De una forma más técnica podemos definir a estos elementos como: ‘aquellos elementos que pueden intercambiar datos entre aplicaciones o componentes de aplicaciones, mismos datos que pueden ser usados para iniciar actividades o servicios, solicitando al sistema que realice una determinada acción con ciertos datos y luego el Sistema Operativo Android se encargaría de buscar la información más cualificada para realizar el trabajo’. Por lo tanto, los Intents se convierten en un mecanismo para la transmisión de mensajes que puede ser utilizado tanto en el seno de una única aplicación como para comunicar aplicaciones entre sí. Los posibles usos de los Intents son:

- Solicitar que se inicie una actividad o servicio.
- Anunciar al sistema que se ha producido un determinado evento.
- Desacoplamiento de componentes de la aplicación para la sustitución con otros componentes que realicen de forma más fácil la realización de una tarea determinada.

Los Intents pueden ser utilizados para iniciar o navegar entre actividades. Para iniciar una actividad debemos hacer una llamada a la función startActivity, pasando como parámetro un Intent: `startActivity(intent);`

El Intent puede o bien especificar de manera explícita el nombre correspondiente a la clase de la actividad que queremos iniciar, o bien indicar una acción que queremos que sea resuelta por una actividad, sin especificar cuál. En este segundo caso el sistema escogerá la actividad a ejecutar de manera dinámica en tiempo de ejecución, buscando aquella que permita llevar a cabo la tarea solicitada de la manera más apropiada.

Veamos en primer lugar como iniciar una Actividad de manera explícita. Para ello debemos crear un nuevo Intent al que se le pase como parámetro el contexto de la aplicación y la clase de la actividad a ejecutar:

```
Intent intent = new Intent(MiActividad.this, MiOtraActividad.class);
startActivity(intent);
```

La otra forma de iniciar una actividad es mediante un Intent implícito, en el que se solicita que un componente anónimo de una aplicación sin determinar se encargue de satisfacer una petición concreta. Para crear un Intent implícito indicamos la acción a realizar y opcionalmente, la URI de los datos sobre los que queremos que se lleve a cabo la acción. También es posible añadir datos adicionales al Intent, conocidos como extras. A continuación se presenta un Intent implícito que solicita que se lleve a cabo la tarea de realizar la llamada a un número de teléfono representado por una URI:

```
Intent intent = new Intent(Intent.ACTION_DIAL,Uri.parse("tel:123456789");
startActivity(intent);
```

En el caso en el que existan varias actividades que igualmente son capaces de llevar a cabo la tarea se le permitirá escoger al usuario cuál de ellas utilizar por medio de una ventana de diálogo. Los componentes y aplicaciones nativos de Android tienen la misma prioridad que el resto, por lo que pueden ser completamente reemplazados por nuevas actividades que declaren que pueden llevar a cabo las mismas acciones.

3. Tipos de Linear Layout:

Para poder comprender de una mejor manera es muy importante que definamos que es un Layout y entenderemos que este elemento es un contenedor de una o más vistas, en el cual se puede controlar su comportamiento y posición, destacando también que un Layout puede contener a otro Layout y que este es a su vez sería un descendiente de la clase View.

Ahora bien, si definimos que es un Linear Layout podemos decir que es la clasificación más utilizada en la práctica de estructuras para aplicaciones móviles dentro de la categoría de los Layout, todo gracias a que puede establecer los componentes visuales uno junto al otro, apilándolos ya sea de forma horizontal o vertical, siendo así un aspecto de suma importancia al momento de diseñar las perspectivas que utilizaremos para nuestra aplicación y con ello tener un mejor control en la organización de los elementos. Si lo comparamos con otro componente, podemos asimilarlo a los “Div” o “contenedores” que se utilizan en HTML para poder esquematizar y diseñar las interfaces de nuestros proyectos, en caso similar los Linear Layout cumplen la misma función de “alinear los

componentes" de nuestra aplicación para poder ayudar en el diseño de nuestras vistas o pantallas en las que pondremos todos nuestros controles que tendrán interacción con el usuario como EditText, Botones, etc...

4. Constraint Layout:

Con respecto a este elemento podemos empezar detallando que en "Android" siempre se tiene una dificulta al momento de armar vistas complejas ya que en ello influyen muchos factores que determinan la apreciación, eficacia y eficiencia de los componentes y las vistas mismas. A partir de esto nació Constraint Layout como una alternativa de solución para poder facilitar la tarea de construcción de perspectivas, dicho componente permite construir vistas complejas sin caer en el "error" de anidar muchas vistas una dentro de otra, anexo a ello forma parte de 'Android Jetpack', que es un conjunto de componentes cuyo uso forma parte de las buenas prácticas para el desarrollo de aplicaciones en Android. Entonces de cierto modo, cada Constraint representaría una conexión o alineamiento a otra vista, al Layout padre, o a una guía invisible, por lo tanto, cada Constraint define la posición de la vista en función a los ejes vertical u horizontal; por ello cada vista debe contar con al menos un Constraint por cada eje, aunque como es de esperarse a menudo será necesario definir más de uno para mantener el control sobre las vistas de la aplicación que estamos desarrollando.

Si abarcamos la definición de Constraint Layout desde un aspecto técnico podemos decir lo siguiente: 'Constraint Layout es un ViewGroup que se introdujo en el SDK de Android en la versión 2.3 de Android Studio, y que logró hacer un gran cambio en la manera como se hacen las vistas. La idea principal de este contenedor es evitar el uso de Layouts anidados y así mejorar el alcance al momento de diseñar vistas complejas, entonces, un Constraint es un elemento que provee una relación de un objeto a otro, de una manera restrictiva o de referencia, que nos ayuda a posicionar los elementos en relación o restricción de otros.'

A continuación se definen algunas herramientas o particularidades que se pueden realizar utilizando Constraint Layout:

- **Convertir Layout:** si ya tenemos una vista que hace uso de algún otro Layout, podemos fácilmente migrar los elementos a Constraint Layout, Android Studio tiene una herramienta para esto en su editor visual.
- **Cadenas:** Este es un recurso muy común, ya que este comportamiento es parecido a cuando usamos Linear Layout y ponemos pesos en las vistas, para que se autoajusten al tamaño disponible del contenedor…. Para hacer una cadena, debemos seleccionar los elementos que queremos que sean parte de ella, dar clic derecho y seleccionar la opción Chains. Existen 4 modos con respecto a cómo funcionan las cadenas, y se aplican igual para horizontal o vertical, el tag que se utiliza es `layout_constraintHorizontal_chainStyle` para cadenas horizontales, y `layout_constraintVertical_chainStyle` para cadenas verticales, seguido del modo que se emplearía para ejecutar la cadena:
    1. Spread: se distribuye el espacio de igual manera.
    2. Spread_inside: se distribuye el espacio entre los elementos.
    3. Weighted: se indica un peso para cada elemento, como en Linear Layout.
    4. Packed: los elementos se juntan y el espacio se distribuye a los extremos.
- **Barrera y Guías:** Estas herramientas son de gran ayuda cuando necesitamos alinear varios elementos y pueden existir algunas dificultades en tamaños o contenido. Una barrera nos ayudará a que todo se vea “perfecto”, ya que el Constraint de un elemento C será a la barrera, y la barrera estará protegiendo los elementos A y B, sin importar su tamaño. Una guía tiene un comportamiento parecido a la barrera, solo que en vez de estar relacionada a elementos, lo hace en relación al contenedor, por ejemplo, a 33% del ancho disponible.
- **Grupos:** Hay ocasiones en donde para poder ocultar una sección de información, se suelen poner todos los elementos de la sección dentro de un contenedor y cambiar la visibilidad del mismo, pero para evitar el uso de Layouts anidados, Constraint Layout nos provee de un elemento no visual que es <android.support.constraint.Group />, no tiene dimensiones pero es necesario poner los tags de layout_width y layout_height; su uso se enfoca más en indicar los ids de

los elementos que deseamos cambiar su visibilidad, y ese cambio lo haremos sobre el id del grupo para ver el cambio en sus elementos.

- **Posición circular:** Este elemento nos permite hacer fácilmente en Layout lo que antes se tenía que hacer con librerías externas de vistas personalizadas y es que usando layout_constraintCircle podemos indicar que el Constraint es de forma circular, creando un elemento con forma esférica y podemos simplemente agregar un TextView con Constraint circular a un ImageView, dándole un toque personalizado a nuestras interfaces.

BIBLIOGRAFÍA

- https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es
- http://www.jtech.ua.es/dadm/restringido/android/sesion02-apuntes.html#:~:text=Los%20Intents%20son%20uno%20de,para%20iniciar%20actividades%20o%20servicios.
- https://www.tutorialesprogramacionya.com/javaya/androidya/androidstudioya/detalleconcepto.php?codigo=18&inicio=0
- https://www.mejorprogramacion.com/curso-android-basico/3-1-layout/
- https://medium.com/@diegop88/constraint-layout-en-modo-experto-6856db3d2ee4#:~:text=ConstraintLayout%20es%20un%20ViewGroup%20que,performance%20al%20pintar%20vistas%20complejas.