Guía de ejercicios - Ciclo de vida de componentes (III)

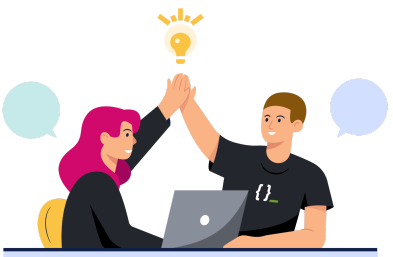
¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos



¡Comencemos!

Actividad guiada N°1: Adapter completamente funcional paso a paso

En esta actividad guiada aprenderemos cómo crear un Adapter, específicamente un RecyclerView.Adapter.

Para completar esta actividad necesitamos crear:

- 1 Activity con un RecyclerView
- 1 Layout el cual será la vista de diseño del ViewHolder
- 1 RecyclerView.Adapter
- 1 ViewHolder
- Una clase que representará al objeto que mostraremos en el RecyclerView

Adicionalmente, aprenderemos a interactuar con el adapter, por ejemplo, cuando el usuario haga click en uno de los ítems

¡Manos a la obra!

Crea un proyecto con un Empty Activity y dale el nombre que desees.
En la vista XML del MainActivity, agrega un RecyclerView, e identifícalo como **rv_contact**

```
<androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_contact"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/section_label"
        tools:listitem="@layout/item_contact" />
```

Dentro del directorio layout, crea una nueva vista y llámala **item_contact.xml**

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

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#F1F1F1F1"
    android:paddingBottom="8dp">

    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/appCompatImageView"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:src="@drawable/ic_outline_account_circle"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/appCompatImageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/appCompatImageView"
        app:layout_constraintTop_toTopOf="@+id/appCompatImageView">

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/tv_contact_name"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="12dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:text="Samuel Taylor Coleridge" />

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/tv_contact_phone_number"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            app:layout_constraintEnd_toEndOf="@+id/tv_contact_name"
            app:layout_constraintStart_toStartOf="@+id/tv_contact_name"
            app:layout_constraintTop_toBottomOf="@+id/tv_contact_name"
            tools:text="+56 92661 61 98" />
```

```
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ahora necesitamos escribir el Adapter. Para esto crearemos una clase llamada ContactAdapter y extenderá de RecyclerView.Adapter, esto generará un error al principio, ya que requiere que se pase como parámetro el ViewHolder, el cual no hemos creado aún.

```
class ContactAdapter() : RecyclerView.Adapter<>() {}
```

Dentro de `ContactAdapter` podemos crear el ViewHolder, como sabemos que ocuparemos el ViewHolder solamente en este adapter, entonces lo crearemos como inner class.

`ContactViewHolder` debe extender de `RecyclerView.ViewHolder`, y ya que usaremos ViewBinding, este recibirá como parámetro el root del View Binding object.

```
inner class ContactViewHolder(private val binding: ItemContactBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun onBind(contact: Contact) {
        }
    }
```

Antes de terminar el adapter, vamos a crear el objeto que mostraremos en el ViewHolder, se trata de una clase llamada Contact la cual guardará datos de contactos y lo pasaremos al Adapter como una lista.

```
data class Contact(
    val id: Int,
    val name: String,
    val phoneNumber: String
```

Ahora podemos terminar el Adapter

```
class ContactAdapter(
    private val contacts: List<Contact>
) : RecyclerView.Adapter<ContactAdapter.ContactViewHolder>() {
```

```
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ContactViewHolder {
        val binding =
ItemContactBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return ContactViewHolder(binding)
    }

    override fun onBindViewHolder(holder: ContactViewHolder, position:
Int) {
        holder.onBind(contacts[position])
    }

    override fun getItemCount(): Int = contacts.size

    inner class ContactViewHolder(private val binding:
ItemContactBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun onBind(contact: Contact) {
            with(binding) {
                contact.run {
                    tvContactName.text = name
                    tvContactPhoneNumber.text = phoneNumber
                }
            }
        }
    }
}
```

Necesitamos llamar el adapter desde el MainActivity, el constructor por defecto del adapter requiere que se pase una lista de objetos Contact. Para efecto de demostrar cómo funciona el adapter, crearemos una lista de objetos de prueba.

La siguiente función retorna una lista de `Contacts`

```
fun dummyData(): List<Contact> = listOf(
    Contact(
        id = 1,
        name = "Samuel Taylor Coleridge",
        phoneNumber = "+56 234 1231"
    ),
```

```
    Contact(
        id = 2,
        name = "Edgar Alan Poe",
        phoneNumber = "+56 234 1134"
    ),
    Contact(
        id = 3,
        name = "H.P. Lovecraft",
        phoneNumber = "+46 264 1134"
    ),
)
```

Para empezar a utilizar el Adapter en un Activity o Fragment, podemos hacer lo siguiente:

Definimos un `lateinit var`

```
lateinit var adapter: ContactAdapter
```

Luego, dentro del método `onCreate()` en el caso de un Activity o dentro de `onViewCreated()` en el caso de un Fragment, asignamos un valor a Adapter, lo pasamos los datos de prueba y lo asignamos al recyclerView que creamos en vista XML del activity.

```
adapter = ContactAdapter(dummyData())
binding.rvContact.layoutManager = LinearLayoutManager(context)
binding.rvContact.adapter = adapter
```

Finalmente podemos correr la app y veremos como en el recyclerView se cargan 3 contactos distintos, como ejercicio extra, puedes crear más datos de prueba y ver cómo responde el adapter.

Necesitamos manejar el interactuar del usuario con la app, para esto podemos crear una variable en el adapter

```
var onClick: ((String) -> Unit)? = null
```

Luego podemos hacer que cuando el usuario presione, por ejemplo, el numero de telefono, este envie el dato al activity o fragment

```
tvContactPhoneNumber.setOnClickListener {
   onClick?.invoke(phoneNumber)
}
```

Luego en el activity o en el fragment hacemos del valor que enviamos desde el adapter:

```
adapter.onClick= {
    Log.d("ContactsFragment", "adapter.onClick: $it")
}
```

Ahora ya podemos interactuar entre los datos del Adapter y el Activity o Fragment

¡Manos a la obra! - ViewPager Adapter

ViewPager, es un tipo de adaptador muy especial, permite mostrar una lista de Fragments, los cuales usualmente están alineados de forma horizontal.
Siguiendo el código explicado en la clase anterior, crea un proyecto nuevo con un empty activity e implementa un ViewPager.

1. Crea 2 Fragments y nombralos `HomeFragmemt()`, `ContactFragment()`.

2. Crea una clase para el adaptador y nombrala `SectionsPagerAdapter` y que extienda de `FragmentStatePagerAdapter` sigue el código mostrado durante la clase *Android Adaptadores*.

3. Implementa los métodos requeridos por `FragmentStatePagerAdapter`.

4. Completa el código para el método `getItem()`, el cual dependiendo de la posición retorna un Fragment distinto.

5. Completa el código para `getPageTitle()` el cual, dependiendo de la posición, retorna un título distinto.

6. Completa el código para `getCount()` el cual retorna la cantidad de fragments distintos.

7. En la vista de diseño del Activity crea un ViewPager y nombralo **`view_pager`**

8. Dentro del metodo onCreate del activity, crea una instancia del **`SectionsPagerAdapter`**, recuerda que este requiere como parámetro **`supportFragmentManager`**.

9. Asigna la instancia de **`SectionsPagerAdapter`** al ViewPager creado en la vista de diseño.

10. Compara con el código final.

¡Manos a la obra! - ConcatAdapter

En el siguiente ejercicio implementaremos un ConcatAdaper, para esto haremos uso del adaptador creado en el ejercicio paso a paso.

Empecemos!

1. Crea dos instancias del Adapter creado en la siguiente sección del código.

```
adapter = ContactAdapter(dummyData())
binding.rvContact.layoutManager = LinearLayoutManager(context)
binding.rvContact.adapter = adapter
```

Nombra las instancias **`adapter1`** y **`adapter2,`** respectivamente.

2. En el archivo build.gradle de la app, agrega: `androidx.recyclerview:recyclerview:1.2.1`

3. Crea una variable val, nombrala `concatAdapter` y asígnale un `ConcatAdater` el cual debe recibir como parámetros `adapter1` y `adapter2.`

4. Asigna la variable `concatAdapter` a tu recyclerView.

5. Compara el código final.

Solución - ViewPager Adapter

1. HomeFragmemt(), ContactFragment()

2.

```
class SectionsPagerAdapter(fm: FragmentManager) :
    FragmentStatePagerAdapter(fm, BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT)
{
```

3.

```
override fun getCount(): Int {
    TODO("Not yet implemented")
}

override fun getItem(position: Int): Fragment {
    TODO("Not yet implemented")
}
```

4.

```
override fun getItem(position: Int): Fragment = when (position) {
        0 -> HomeFragment()
        1 -> ContactsFragment()
        else -> HomeFragment()
    }
```

5.

```
override fun getPageTitle(position: Int): CharSequence= when (position){
        0 -> "Hone"
        1 -> "Contacts"
        else -> "Home"
    }
```

6.

```
override fun getCount(): Int = 2
```

7.

```
<androidx.viewpager.widget.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        />
```

8.

```
val sectionsPagerAdapter = SectionsPagerAdapter(supportFragmentManager)
```

9.

```
binding.viewPager.adapter = sectionsPagerAdapter
```

10.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val sectionsPagerAdapter =
SectionsPagerAdapter(supportFragmentManager)
        binding.viewPager.adapter = sectionsPagerAdapter
    }
}
```

```
class SectionsPagerAdapter(fm: FragmentManager) :
    FragmentStatePagerAdapter(fm, BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT)
{

    override fun getItem(position: Int): Fragment = when (position) {
        0 -> HomeFragment()
        1 -> ContactsFragment()
        else -> HomeFragment()
    }

    override fun getPageTitle(position: Int): CharSequence = when
(position) {
        0 -> "Hone"
        1 -> "Contacts"
        else -> "Home"
    }
```

```
    override fun getCount(): Int = 2
}
```

Solución - ConcatAdapter

1.

```
val adapter1 = ContactAdapter(dummyData())
val adapter2 = ContactAdapter(dummyData())
```

2.

```
implementation "androidx.recyclerview:recyclerview:1.2.1"
```

3.

```
val concatAdapter = ConcatAdapter(adapter1, adapter2)
```

4.

```
binding.rvContact.adapter = concatAdapter
```

5.

```
val adapter1 = ContactAdapter(dummyData())
val adapter2 = ContactAdapter(dummyData())

val concatAdapter = ConcatAdapter(adapter1, adapter2)
binding.rvContact.adapter = concatAdapter
```

Preguntas de proceso

Reflexiona:

- A partir de lo aprendido hasta aquí ¿Qué contenido se me ha hecho más difícil? ¿Y cuál más sencillo? ¿Por qué crees que se da esa diferencia?
- ¿Existe algún ejercicio de los resueltos previamente que deba repetir o desarrollar otra vez para poder dominarlo mejor?
- Nombra al menos un uso que pueda darle a lo aprendido hasta ahora.

Preguntas de cierre

- ¿Qué es lo te parece más complicado de los adaptadores?
- Revisa cuáles son las Top 5 de aplicaciones que más usas, ¿hay alguna de ellas que no haga uso de un Adapter?, comenta
- ¿Te parece intuitivo la forma en que los ítems en el recyclerview interactúan con el resto de las vistas?
- ¿Con lo aprendido hasta ahora, crees que puedas desarrollar una aplicación como un Chat?

Referencias bibliográficas

https://developer.android.com/docs