Práctica 3.
Fragmentos y ViewModel

Resultados de aprendizaje y criterios de evaluación.

CE2a. Se ha generado la estructura de clases necesaria para la aplicación.

CE2g. Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones desarrolladas a partir de emuladores.

CE2h. Se han empaquetado y desplegado las aplicaciones desarrolladas en dispositivos móviles reales.

CE2i. Se han documentado los procesos necesarios para el desarrollo de las aplicaciones.

.

1. Parte 1. Trabajando con fragmentos.

Se crea una aplicación sencilla para ver los dragones de Juego de Tronos, en la que se tiene una única actividad, esta actividd posee un contenedor de fragmentos, que gestiona los siguientes fragmentos.

- Pantalla principal.
- Listado de dragones.
- Detalle de un dragón.
- Alta dragón (ampliación).

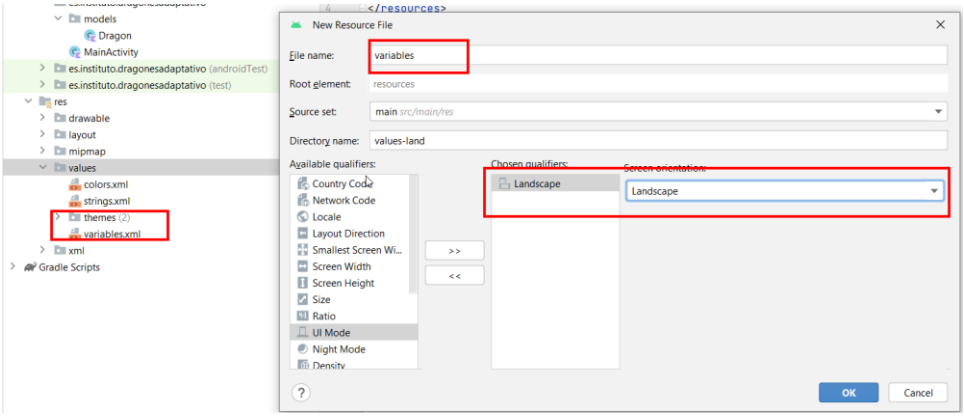
Si la aplicación se encuentra rotada, se puede ver la lista de dragones a la izquierda y a la derecha la ficha de este o crear nuevo dragón.

Se define la clase Dragón:

```
data class Dragon (var nombre:String, var longitud:Int,var anchura:Int,var descripcion:String,var edad:Int) {
}
```

Ahora se define un nuevo fichero de valores (en concreto se definen 2) que posee una variable booleana que indica si se encuentra en horizontal o vertical, y que se usará para tener la aplicación de tipo master/detail.

El primer recurso se crea sin cualificado, y el segundo para el caso en que se encuentre girada:



El contenido del fichero normal:

```
<resources>
    <bool                                    name="land">false</bool>
</resources>
```

Y el de orientación girado:

```
<?xml                version="1.0"                encoding="utf-8"?>
<resources>
    <bool                                    name="land">true</bool>
</resources>
```

Se procede ahora a configurar la actividad y crear los 4 fragmentos, en principio sin tener en cuenta la opción de giro:

Actividad principal:

Se registra la barra de herramientas, se activa el botón de desapilar y se reescribe el método que se ejecuta al pulsar en los botones de la barra de herramientas:

```
class        MainActivity        :        AppCompatActivity()        {
    private     lateinit     var     binding:     ActivityMainBinding
    override     fun     onCreate(savedInstanceState:     Bundle?)     {
        super.onCreate(savedInstanceState)
        binding     =     ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        setSupportActionBar(binding.toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
    }

    override  fun  onOptionsItemSelected(item:  MenuItem):  Boolean  {
        if(item.itemId==                              android.R.id.home){
            onBackPressedDispatcher.onBackPressed()
        }
        return                          super.onOptionsItemSelected(item)
```

```
    }
}
```

Fragmento principal:

```
class HomeFragment : Fragment() {

    private lateinit var v:View
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        v= inflater.inflate(R.layout.fragment_home, container,
false)

v.findViewById<ImageView>(R.id.icono_principal).setOnClickListener {
            val fm: FragmentManager = parentFragmentManager
            fm.commit {
                replace(R.id.fragmentContainerView,
ListFragment.newInstance())
                addToBackStack("replacement")
            }
            true
        }
        return v
    }

    companion object {

        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() =
            HomeFragment().apply {

            }
    }
}
```

En el código anterior destaca el siguiente fragmento:

```
v.findViewById<ImageView>(R.id.icono_principal).setOnClickListener {
    val fm: FragmentManager = parentFragmentManager
    fm.commit {
        replace(R.id.fragmentContainerView,
ListFragment.newInstance())
        addToBackStack("replacement")
    }
    true
}
```

Que obtiene el manejador de fragmentos del padre y apila el nuevo fragmento

El XML:

```
<?xml                version="1.0"                encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeFragment">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:text="Juego              de              Tronos"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"              />

        <ImageView
            android:id="@+id/icono_principal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/baseline_grass_24"
            app:layout_constraintEnd_toEndOf="parent"
```

```
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView"    />
    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```

ListFragment:

Contiene un grid de dragones, con el índice, el nombre y un botón para entrar en el detalle, además necesida un adaptador para mostra transformar:

DragonRecyclerViewAdapter:

```
class                                    DragonRecyclerViewAdapter(
    private            val            values:            List<Dragon>
)  :  RecyclerView.Adapter<DragonRecyclerViewAdapter.ViewHolder>()  {
    var    click:    ((Int,    Dragon)    ->    Unit)?    =    null
    override  fun  onCreateViewHolder(parent:  ViewGroup,  viewType:
Int):                          ViewHolder                          {

        return                                                ViewHolder(
            FragmentItemBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                false
            )
        )

    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int)
{
        val            item            =            values[position]
        holder.idView.text           =           position.toString()
        holder.contentView.text             =             item.nombre
        holder.button.setOnClickListener                              {

   this.click?.let  {  it1  ->  it1(position,  values[position])  }

        }
    }

    override    fun    getItemCount():    Int    =    values.size

    inner    class    ViewHolder(binding:    FragmentItemBinding)    :
RecyclerView.ViewHolder(binding.root)                                 {
        val      idView:      TextView      =      binding.itemNumber
        val      contentView:      TextView      =      binding.content
```

```
        val button: Button = binding.button
        override fun toString(): String {
            return super.toString() + " '" + contentView.text + "'"
        }
    }

}
```

Se establece una función lambda que se ejecutara al hacer click en el elemento.

El XML de un item (fragment_item.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/item_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:layout_weight="1"
        android:textAppearance="?attr/textAppearanceListItem" />

    <TextView
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:layout_weight="1"
        android:textAppearance="?attr/textAppearanceListItem" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```

El código del fragmento del Listado:

```
class ListFragment : Fragment() {

    private lateinit var v: View
    private var dragones: MutableList<Dragon>

    init {
        this.dragones = mutableListOf()
        this.dragones.add(
            Dragon(
                "Arrax",
                5,
                6,
                "Color blanco perla con cresta y ojos dorados que
expelía llamas amarillas",
                12
            )
        )
        this.dragones.add(
            Dragon(
                "Vhagar",
                150,
                170,
                "Hembra cuya mandíbula era lo suficientemente grande
como para que pasara un caballo y su jinete",
                181
            )
        )

    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        v = inflater.inflate(R.layout.fragment_list, container,
false)
        val recyclerView: RecyclerView =
v.findViewById<RecyclerView>(R.id.recyclerview)
        var adaptador = DragonRecyclerViewAdapter(this.dragones)
```

```
        adaptador.click = { position, dragon ->
            run {

            val fm: FragmentManager = parentFragmentManager
                    fm.commit {
                        replace(R.id.fragmentContainerView,
DetailFragment.newInstance())
                        addToBackStack("replacement")
                    }

            }
        }
        val layoutManager = GridLayoutManager(this.context, 2)
        recyclerView.layoutManager = layoutManager
        recyclerView.adapter = adaptador
        return v
    }

    companion object {

        @JvmStatic
        fun newInstance() =
            ListFragment().apply {

            }
    }
}
```

Se le pasa al adaptador una función lamda que recibe dos parámetros, la posición y el dragón, apilando un nuevo fragmento en la pila.

DetailFragment:

Se crea un nuevo fragmento simple, con el nombre del dragón, su longitud, anchura, edad y descripción:

```
class DetailFragment : Fragment() {
    private lateinit var v: View

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View? {
        v = inflater.inflate(R.layout.fragment_detail, container,
false)
        // Inflate the layout for this fragment
        return v //inflater.inflate(R.layout.fragment_detail,
container, false)
    }

    companion object {
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() =
            DetailFragment().apply {

            }
    }
}
```

Y el XML:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DetailFragment">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TableLayout
            android:id="@+id/tableLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent">

            <TableRow
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
               >

                <TextView
```

```
            android:id="@+id/label_nombre"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="11"                                    />

        <TextView
            android:id="@+id/nombre"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="12"                                    />
    </TableRow>

    <TableRow
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <TextView
            android:id="@+id/label_edad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="21"                                    />

        <TextView
            android:id="@+id/edad"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="22"                                    />
    </TableRow>

    <TableRow
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"                      >

        <TextView
            android:id="@+id/label_descripcion"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:text="31"                                    />

        <TextView
            android:id="@+id/descripcion"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="2"
                android:text="32" />
        </TableRow>
    </TableLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
</FrameLayout>
```

Ahora es necesario pasar al fragmento de detalle el dragón seleccionado, se puede hacer con parámetros o con “ViewModel”, pasando el listado de dragones al “ViewModel” de forma que se quede de forma permanente (quitar del listado y pasar a ViewModel).

DragonViewModel:

```
class DragonViewModel: ViewModel() {
    private var _dragones: MutableList<Dragon>
    private var _selected:Dragon?=null
    val dragones:List<Dragon>
        get() = _dragones.toList()
    var selected:Dragon?
        get()=_selected
        set(item){ _selected=item}
    init {
        this._dragones = mutableListOf()
        this._dragones.add(
            Dragon(
                "Arrax",
                5,
                6,
                "Color blanco perla con cresta y ojos dorados que
expelía llamas amarillas",
                12
            )
        )
        this._dragones.add(
            Dragon(
                "Vhagar",
                150,
                170,
                "Hembra cuya mandíbula era lo suficientemente grande
como para que pasara un caballo y su jinete",
                181
            )
        )
```

```
    }
}
```

Posee una lista de dragones y un dragón seleccionado, se declara en la actividad:

Actividad:

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private val dragonviewmodel: DragonViewModel by viewModels()
```

En el fragmento del listado:

```
class ListFragment : Fragment() {

    private lateinit var v: View
    private val dragonviewmodel: DragonViewModel by
activityViewModels()

    class MainActivity : AppCompatActivity() {

    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

    }

    @SuppressLint("SuspiciousIndentation")
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        v = inflater.inflate(R.layout.fragment_list, container,
false)
        val recyclerView: RecyclerView =
v.findViewById<RecyclerView>(R.id.recyclerview)
        var adaptador =
DragonRecyclerViewAdapter(this.dragonviewmodel.dragones)
        adaptador.click = { position, dragon ->
            run {
                //se selecciona el dragón
            this.dragonviewmodel.selected=dragon
            val fm: FragmentManager = parentFragmentManager
                    fm.commit {
```

```
                        replace(R.id.fragmentContainerView,
DetailFragment.newInstance())
                        addToBackStack("replacement")
                    }

            }
        }
        val layoutManager = GridLayoutManager(this.context, 2)
        recyclerView.layoutManager = layoutManager
        recyclerView.adapter = adaptador
        return v
    }

    companion object {

        @JvmStatic
        fun newInstance() =
            ListFragment().apply {

            }
    }
}
```

Y por último en el de detalle:

```
class DetailFragment : Fragment() {
    private lateinit var v: View
    private val dragonviewmodel: DragonViewModel by
activityViewModels()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        v = inflater.inflate(R.layout.fragment_detail, container,
false)
        this.dragonviewmodel.selected?.let {
            v.findViewById<TextView>(R.id.nombre).setText(it.nombre)

v.findViewById<TextView>(R.id.edad).setText(it.edad.toString())
```

```
v.findViewById<TextView>(R.id.descripcion).setText(it.descripcion.to
String())
            //...resto de los campos
        }

        // Inflate the layout for this fragment
        return v //inflater.inflate(R.layout.fragment_detail,
container, false)
    }

    companion object {
        // TODO: Rename and change types and number of parameters
        @JvmStatic
        fun newInstance() =
            DetailFragment().apply {

            }
    }
}
```

1.1. Ejercicio propuesto.

A partir del ejemplo anterior, seleccionar algún tipo de personaje de juegos, películas, series (Harry Potter, Pokemon,...).... y gestionar usando fragmentos los 4 fragmentos indicados:

Portada.

Listado.

Detalle.

Edición

2. Parte 2. Diseño adaptativo

2.1. Ejemplo adaptativo.

2.2. Ejercicio propuesto.

3. Entrega.

Se realizará en Aules

- Código comentado.
- Pequeño manual explicando el proyecto.
- Proyecto comprimido y sin compilar en Aules