Centro De Ciencias Económicas y Administrativas.

Departamento De Sistemas De Información.

Licenciatura en Informática y Tecnologías Computacionales.

Diseño Para Móviles.

Examen Parcial 2: AnimeDraw.

Profesor(a):

Margarita Mondragón Arellano.

Alumnos:

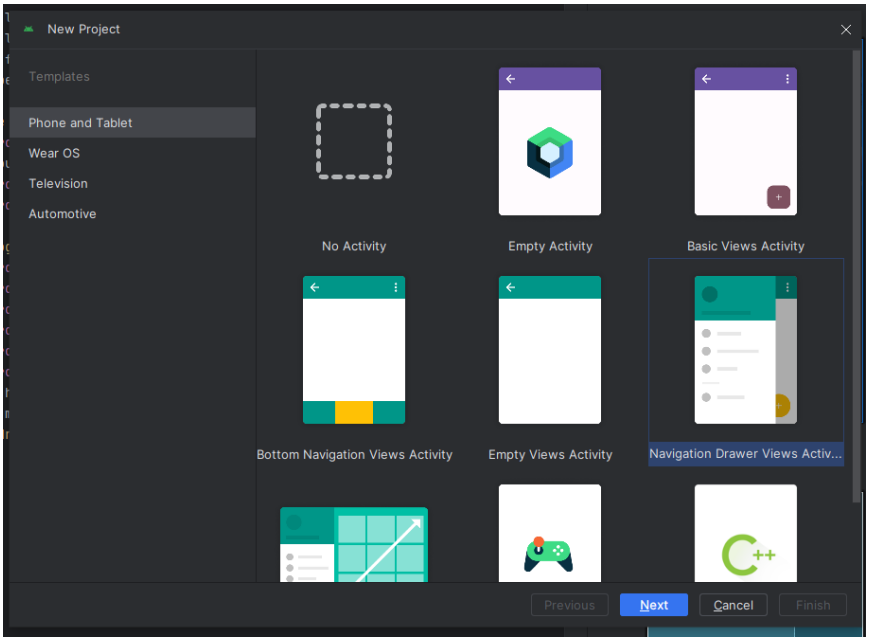
José Henry González Herrada.

Aguascalientes, Ags.

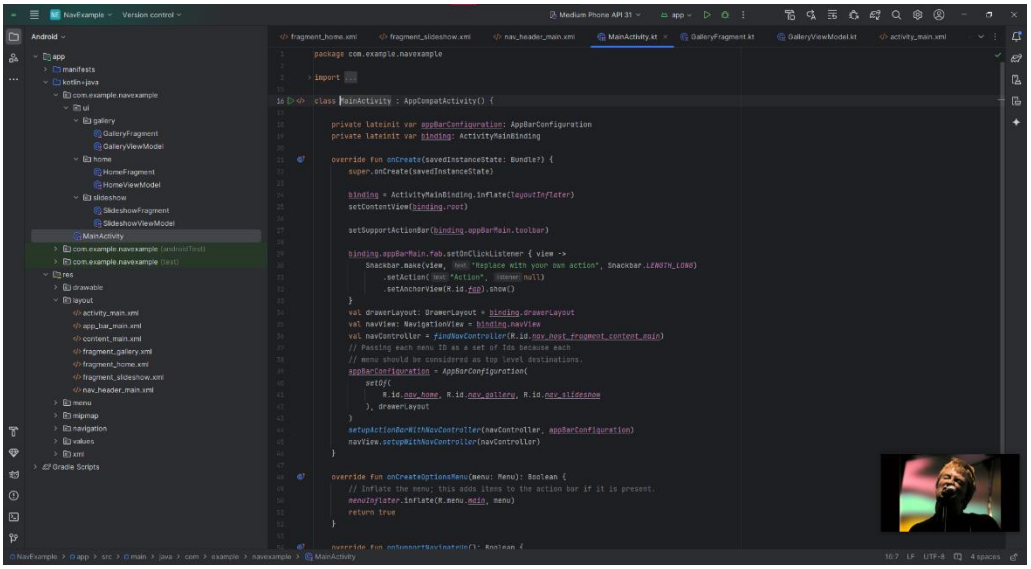
06 de mayo de 2024.

Estructura.

La estructura general del proyecto se define mediante la plantilla de “Navigation Drawer Views Activity”, con ella podemos comenzar a dar forma a nuestra aplicación con la temática de nuestra preferencia seleccionada.

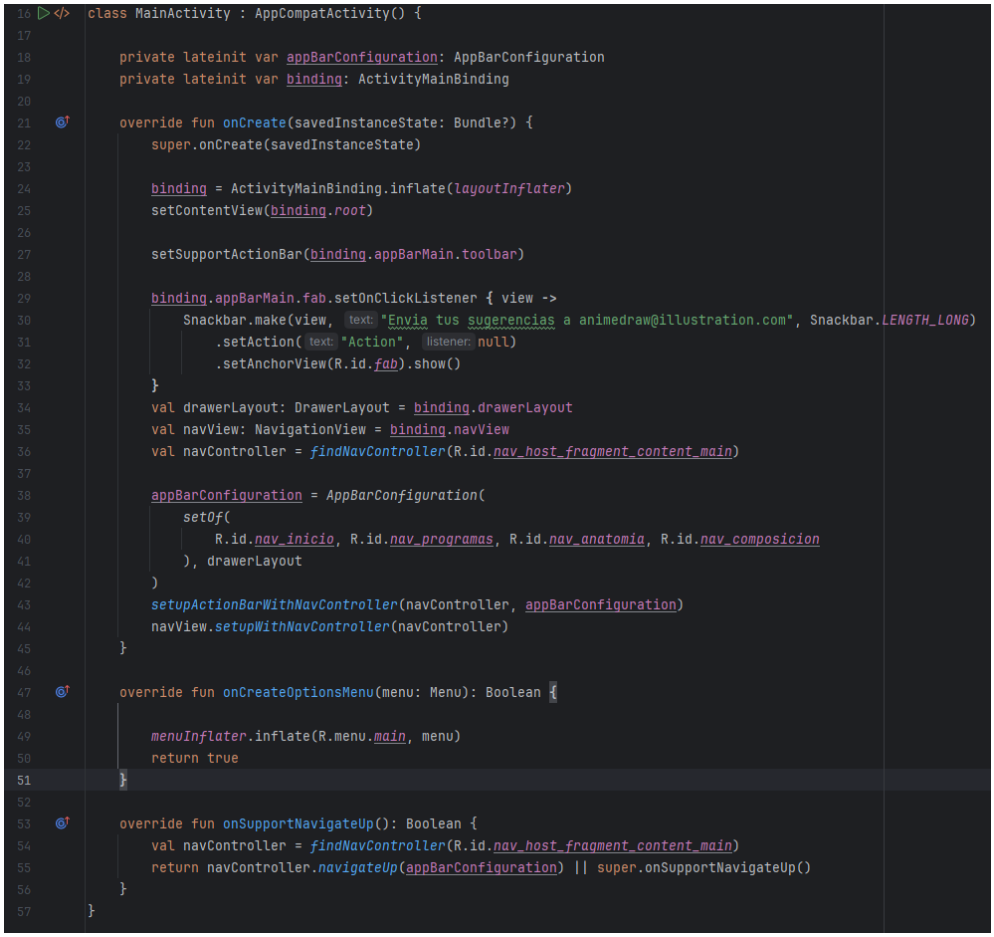


“Navigation Drawer Views Activity” nos proporciona una estructura básica para dar comienzo con el desarrollo de un RecyclerView dentro de nuestro proyecto.



MainActivity.

“MainActivity” configura la interfaz de usuario principal de la aplicación, que incluye una barra de herramientas, un botón de acción flotante, un “Navigation Drawer” y la navegación entre diferentes secciones de la aplicación.



```kotlin
class MainActivity : AppCompatActivity() {

    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityMainBinding

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

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

        setSupportActionBar(binding.appBarMain.toolbar)

        binding.appBarMain.fab.setOnClickListener { view ->
            Snackbar.make(view, text: "Envia tus sugerencias a animedraw@illustration.com", Snackbar.LENGTH_LONG)
                .setAction( text: "Action",  listener: null)
                .setAnchorView(R.id.fab).show()
        }
        val drawerLayout: DrawerLayout = binding.drawerLayout
        val navView: NavigationView = binding.navView
        val navController = findNavController(R.id.nav_host_fragment_content_main)

        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.nav_inicio, R.id.nav_programas, R.id.nav_anatomia, R.id.nav_composicion
            ), drawerLayout
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
        navView.setupWithNavController(navController)
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {

        menuInflater.inflate(R.menu.main, menu)
        return true
    }

    override fun onSupportNavigateUp(): Boolean {
        val navController = findNavController(R.id.nav_host_fragment_content_main)
        return navController.navigateUp(appBarConfiguration) || super.onSupportNavigateUp()
    }
}
```

InfActivity.

Muestra información específica (título, texto e imagen) tomados de los layouts pertenecientes a los fragments de la aplicación.

```kotlin
class InfActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_inf)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets ^setOnApplyWindowInsetsListener
        }

        // Recuperar los datos del Intent
        val titulo = intent.getStringExtra( name: "titulo")
        val informacion = intent.getStringExtra( name: "informacion")
        val imagen = intent.getIntExtra( name: "imagen",  defaultValue: 0)

        // Obtener referencias a los elementos del layout
        val imgEncabezado = findViewById<ImageView>(R.id.img_encabezado)
        val txvTitulo = findViewById<TextView>(R.id.txv_titulo)
        val txvTexto = findViewById<TextView>(R.id.txv_texto)

        // Asignar los datos a los elementos del layout
        titulo?.let { txvTitulo.text = it }
        informacion?.let { txvTexto.text = it }
        imagen.takeIf { it != 0 }?.let { imgEncabezado.setImageResource(it) }
    }
}
```

CardView.

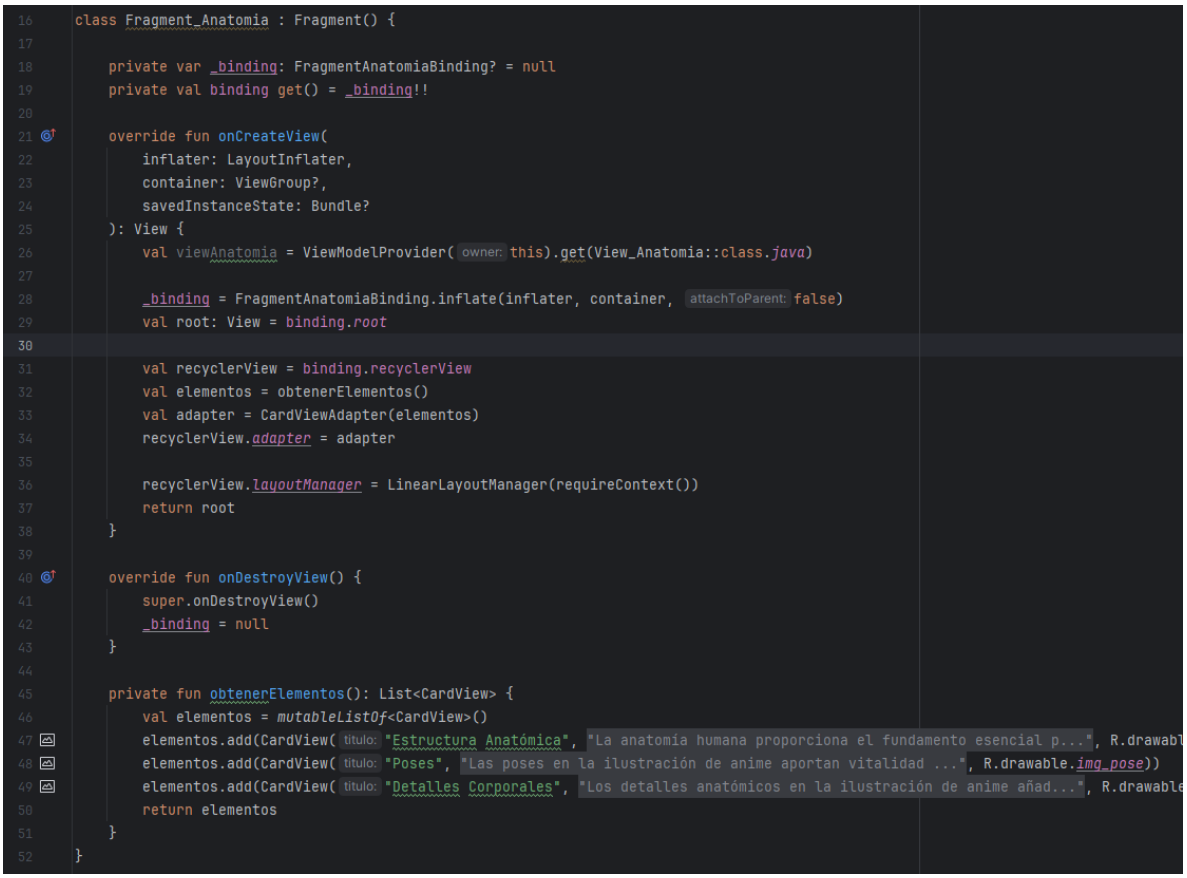
El CardViewAdapter define un adaptador personalizado con el mismo nombre que se utiliza para vincular datos a elementos de una RecyclerView en Android. La RecyclerView muestra una lista de elementos de tipo CardView, también maneja la navegación a una nueva actividad (InfActivity) cuando se hace clic en un elemento de la lista.

```kotlin
package com.example.animedraw.ui.model

class CardView(
    var titulo: String? = null,
    var descripcion: String? = null,
    var imagenDrawable: Int? = null
)
```

```
class CardViewAdapter(private val elementos: List<CardView>) :
    RecyclerView.Adapter<CardViewAdapter.ViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view: View = LayoutInflater.from(parent.context).inflate(R.layout.elemento_cardview, parent, attachToRoot: false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val elemento: CardView = elementos[position]
        holder.tituloTextView.text = elemento.titulo
        holder.descripcionTextView.text = elemento.descripcion
        holder.imagenImageView.setImageResource(elemento.imagenDrawable ?: R.drawable.img_banner)

        holder.itemView.setOnClickListener { it: View!

            val titulo = elemento.titulo
            val informacion = elemento.descripcion
            val imagen = elemento.imagenDrawable

            val intent = Intent(holder.itemView.context, InfActivity::class.java)

            intent.putExtra( name: "titulo", titulo)
            intent.putExtra( name: "informacion", informacion)
            intent.putExtra( name: "imagen", imagen)

            holder.itemView.context.startActivity(intent)
        }
    }

    override fun getItemCount(): Int {
        return elementos.size
    }

    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val tituloTextView: TextView = itemView.findViewById(R.id.titleTextView)
        val descripcionTextView: TextView = itemView.findViewById(R.id.descriptionTextView)
        val imagenImageView: ImageView = itemView.findViewById(R.id.imageView)
    }
}
```

Fragments.

Los fragments muestran una lista de elementos de tarjeta relacionados con las temáticas de la aplicación, como ejemplo, el fragment de anatomía, utilizando el RecyclerView mencionado, muestra elementos que contienen información sobre un tipo específico de estructura anatómica. Los fragments de Composición, Inicio y Programas muestran la misma estructura.



```
class Fragment_Anatomia : Fragment() {

    private var _binding: FragmentAnatomiaBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val viewAnatomia = ViewModelProvider( owner: this).get(View_Anatomia::class.java)

        _binding = FragmentAnatomiaBinding.inflate(inflater, container, attachToParent: false)
        val root: View = binding.root

        val recyclerView = binding.recyclerView
        val elementos = obtenerElementos()
        val adapter = CardViewAdapter(elementos)
        recyclerView.adapter = adapter

        recyclerView.layoutManager = LinearLayoutManager(requireContext())
        return root
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }

    private fun obtenerElementos(): List<CardView> {
        val elementos = mutableListOf<CardView>()
        elementos.add(CardView( titulo: "Estructura Anatómica", "La anatomía humana proporciona el fundamento esencial p...", R.drawabl
        elementos.add(CardView( titulo: "Poses", "Las poses en la ilustración de anime aportan vitalidad ...", R.drawable.img_pose))
        elementos.add(CardView( titulo: "Detalles Corporales", "Los detalles anatómicos en la ilustración de anime añad...", R.drawable
        return elementos
    }
}
```

Views.

Tomando como ejemplo nuevamente la sección de anatomía dentro de la aplicación, el ViewModel del mismo tema proporciona un único dato observable llamado text, que contiene el texto "Fragmento Anatomia". Esto permite que las clases externas, como fragmentos o actividades, observen los cambios en el texto y actualicen la interfaz de usuario en consecuencia.

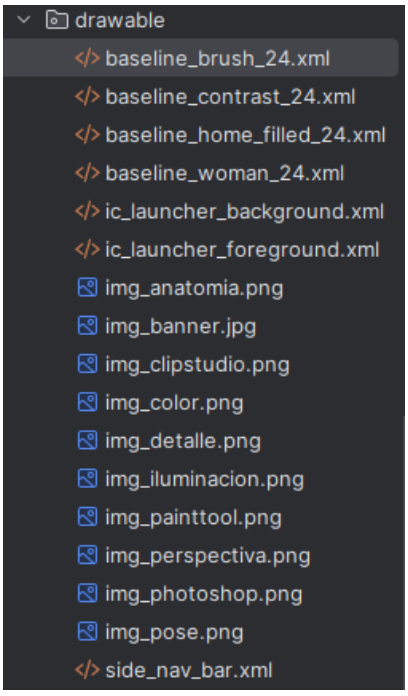
```
package com.example.animedraw.ui.view

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class View_Anatomia : ViewModel() {

    private val _text = MutableLiveData<String>().apply { this: MutableLiveData<String>
        value = "Fragmento Anatomia"
    }
    val text: LiveData<String> = _text
}
```

Diseño.

Como distintivo general de la aplicación en cuanto al diseño, se optó por seleccionar una paleta de colores atractiva que combinara con la idea que se busca transmitir con la temática.

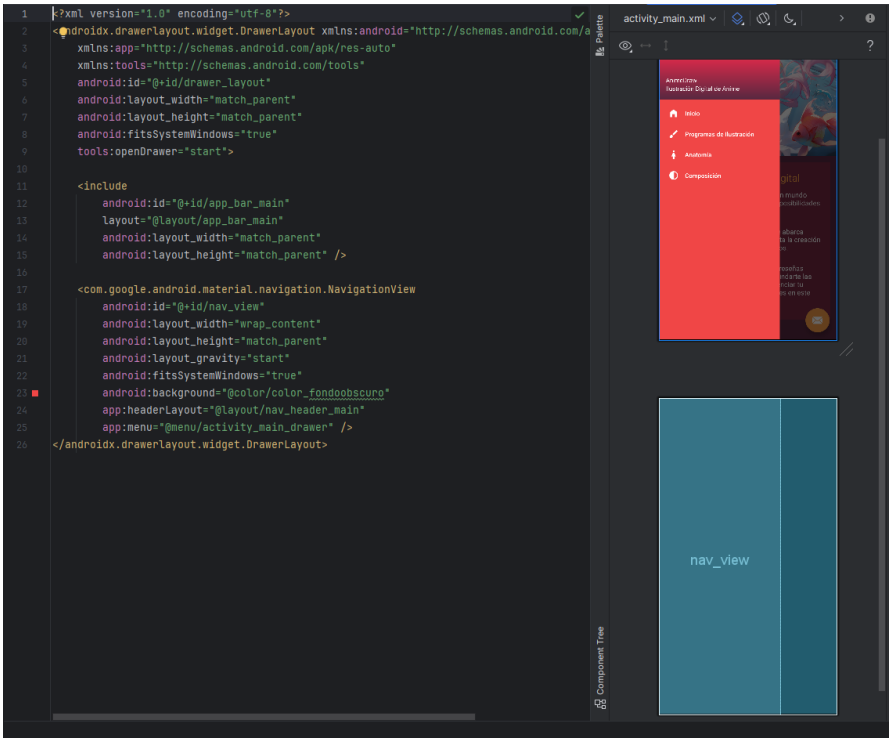

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#f89f49</color>
    <color name="white">#FFFFFFFF</color>

    <color name="color_cardview">#ce2949</color>
    <color name="color_titulo">#f89f49</color>
    <color name="color_texto">#ffffff</color>

    <color name="color_principal">#f89f49</color>
    <color name="color_secundario">#ce2949</color>
    <color name="color_terciario">#582840</color>

    <color name="color_fondo">#582840</color>
    <color name="color_fondoobscuro">#f04646</color>
    <color name="color_fondotransparente">#40ce2949</color>

    <color name="colorprueba">#FF3700B3</color>
    <color name="mi_color_de_texto_normal">#FFFFFF</color> <!-- Blanco -->

</resources>
```

Lo siguiente por considerer es, una vez que se tienen los elementos de los que se habla dentro de la temática, conseguimos los recursos, es decir, las imágenes representativas de cada elemnto en torno a la Ilustración digital de anime.



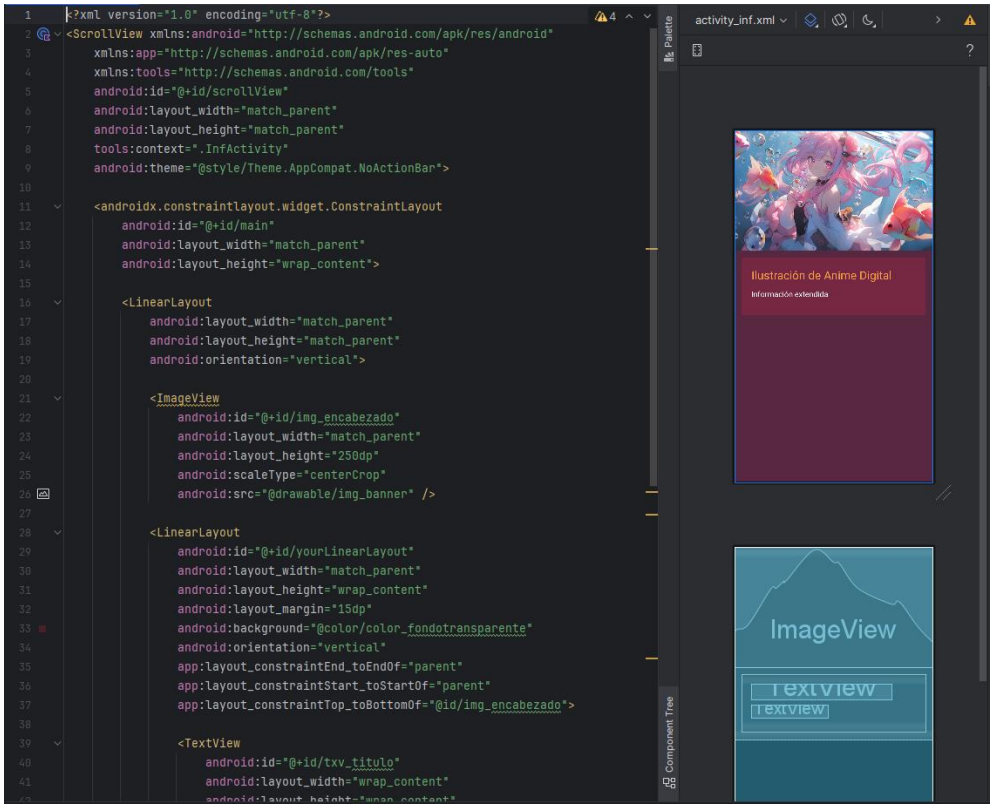
Activity_main.

Este xml se encarga del diseño del nav_view de la aplicación (color base y posicionamiento.



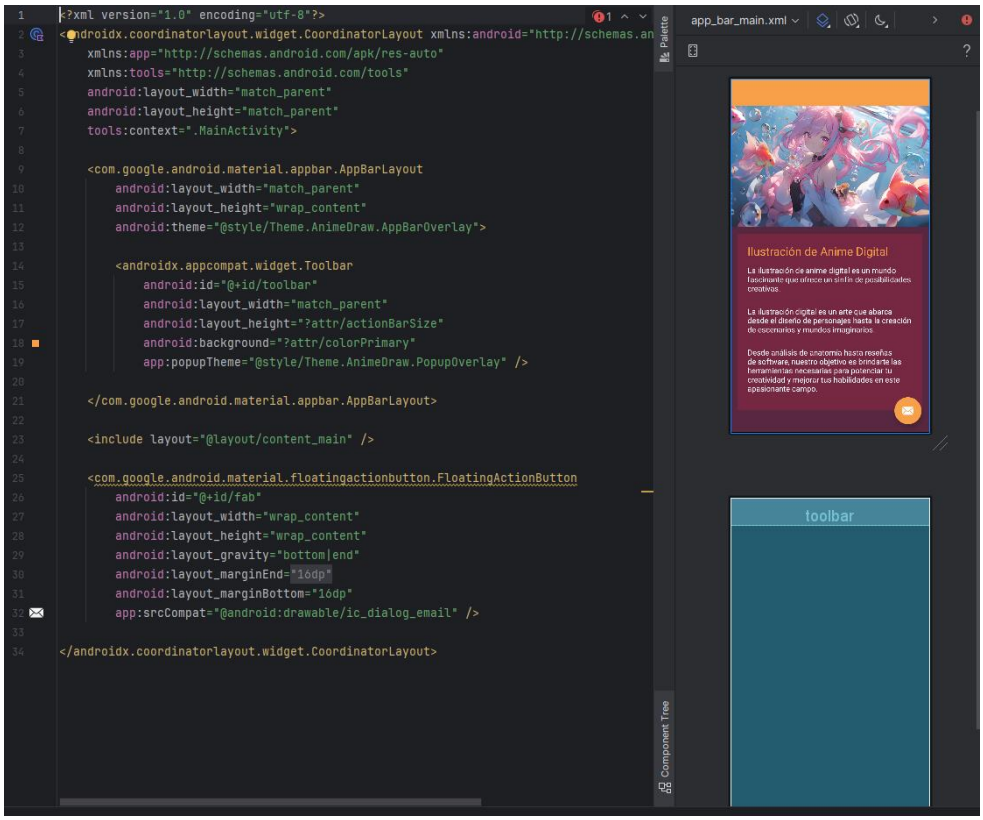
Activity_Inf.

Define la estructura y los elementos que se mostraran en la pantalla de inicio de la aplicación, aquella que habla de la temática general.



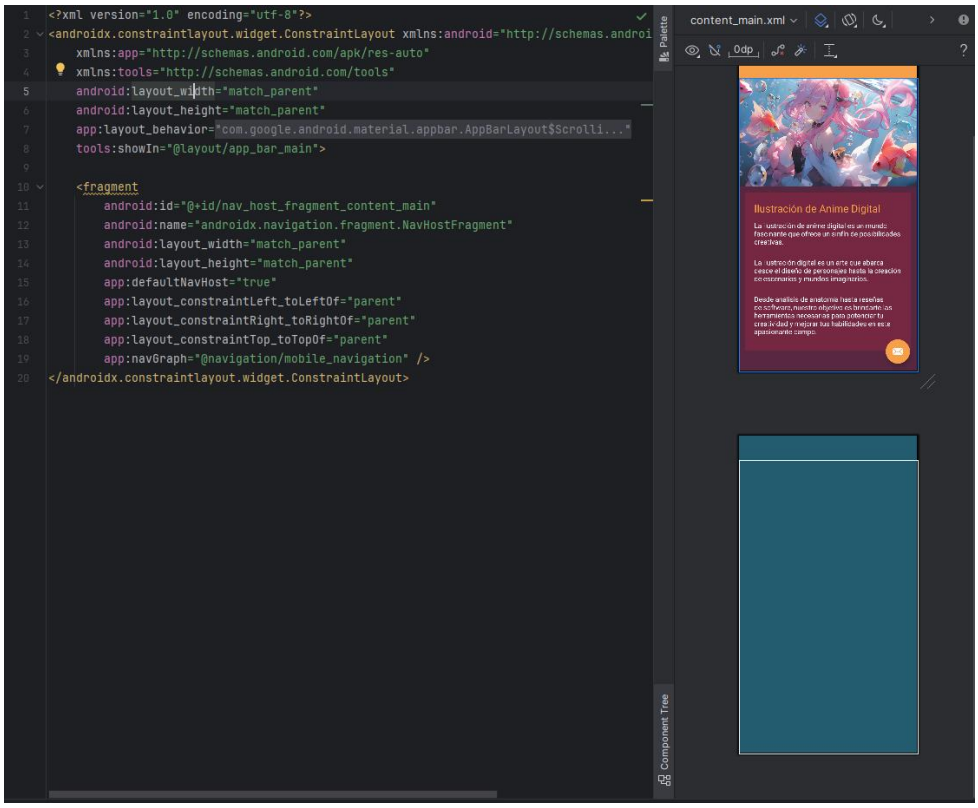
App_bar_main.

Aquí se define el diseño de la toolbar de la aplicación y el boton que despliega el mensaje (tamaño, anclaje, etc).



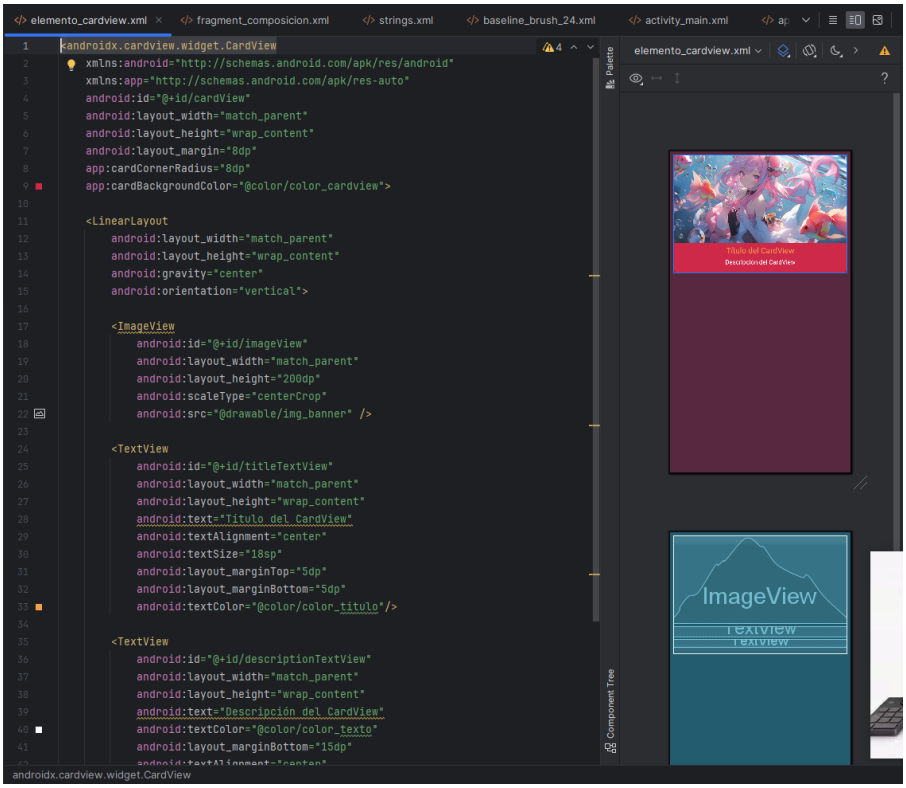
Content_main.

Define el espacio en el que se mostrara el contenido de cada fragment por debajo del toolbar.



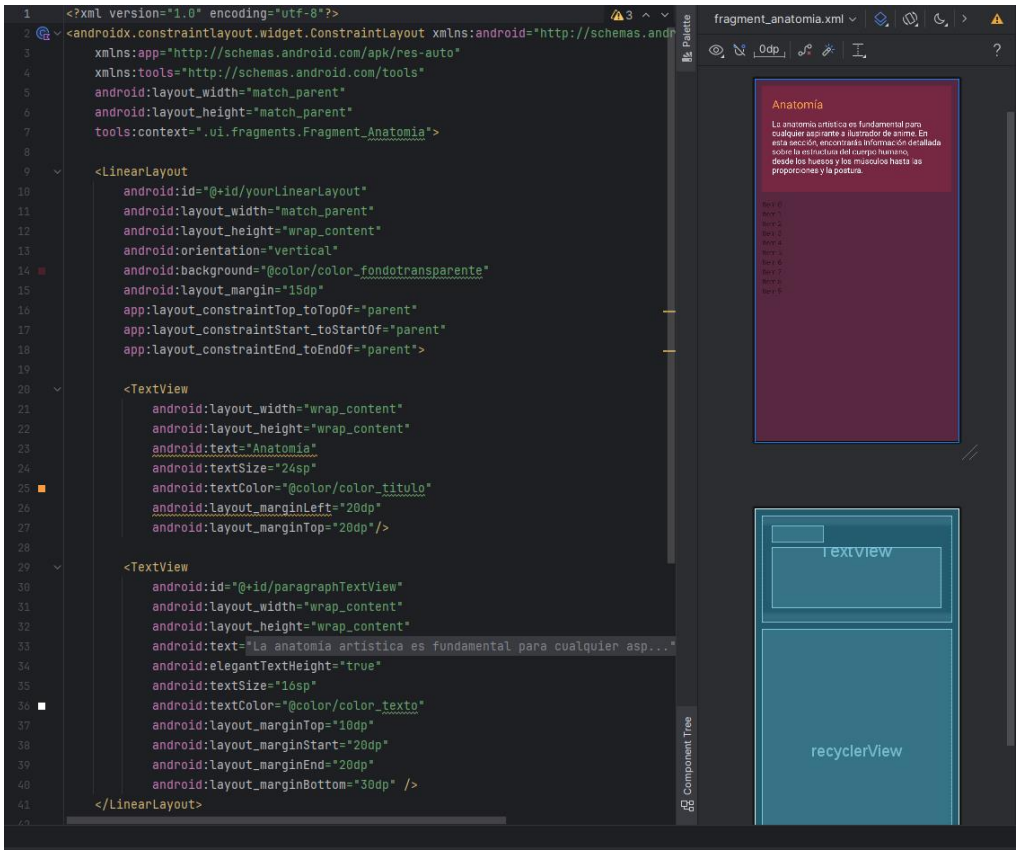
Elemento_Cardview.

Define el contenido que se mostrara de los cardview dentro de los fragments.



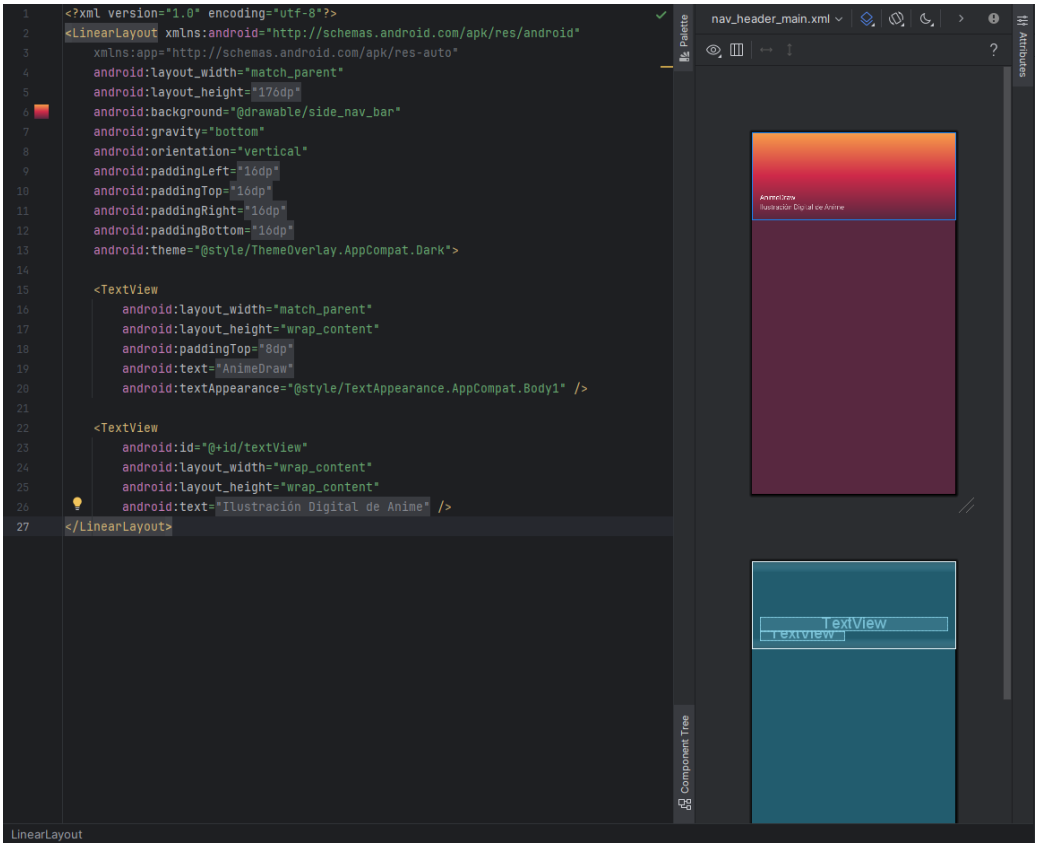
Fragments.

Cuando se toca el cardview el elemento deseado a visualizer de la temática, entonces se despliega la siguiente pantalla, que contiene la descripción.



Nav_header_main.

Para finalizar, aquí se define el encabezado del nav_view de la aplicación, con un color degradado y sujeto a los márgenes de este atributo.




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="176dp"
    android:background="@drawable/side_nav_bar"
    android:gravity="bottom"
    android:orientation="vertical"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="8dp"
        android:text="AnimeDraw"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Ilustración Digital de Anime" />
</LinearLayout>
```