Proyecto: Interfaz de usuario 1

Inidce

Trasladar el contenido de la “MainActivity” y del “CreditActivity” en dos fragmentos que deben permitir la técnica de “view binding”. Desde este momento todos los fragmentos que se creen deben usar el “view binding”. (0,5)

```kotlin
package com.marisma.fase1recu

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.findNavController
import androidx.navigation.ui.NavigationUI
import com.marisma.fase1recu.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Configurar la ActionBar
        setSupportActionBar(binding.toolbar)

        // Obtener el NavController
        val navController = findNavController(R.id.nav_host_fragment)

        // Configurar la ActionBar con el NavController
        NavigationUI.setupActionBarWithNavController( activity: this, navController)
    }

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

```kotlin
package com.marisma.fase1recu.adapter

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.marisma.fase1recu.R
import com.marisma.fase1recu.databinding.FragmentCreditsBinding

class CreditsFragment : Fragment() {

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

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment using view binding
        _binding = FragmentCreditsBinding.inflate(inflater, container,  attachToParent: false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Set click listener for the back button
        binding.backButton.setOnClickListener {  it: View!
            findNavController().navigate(R.id.action_creditsFragment_to_menuFragment)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        // Clear the binding reference to avoid memory leaks
        _binding = null
    }
}
```

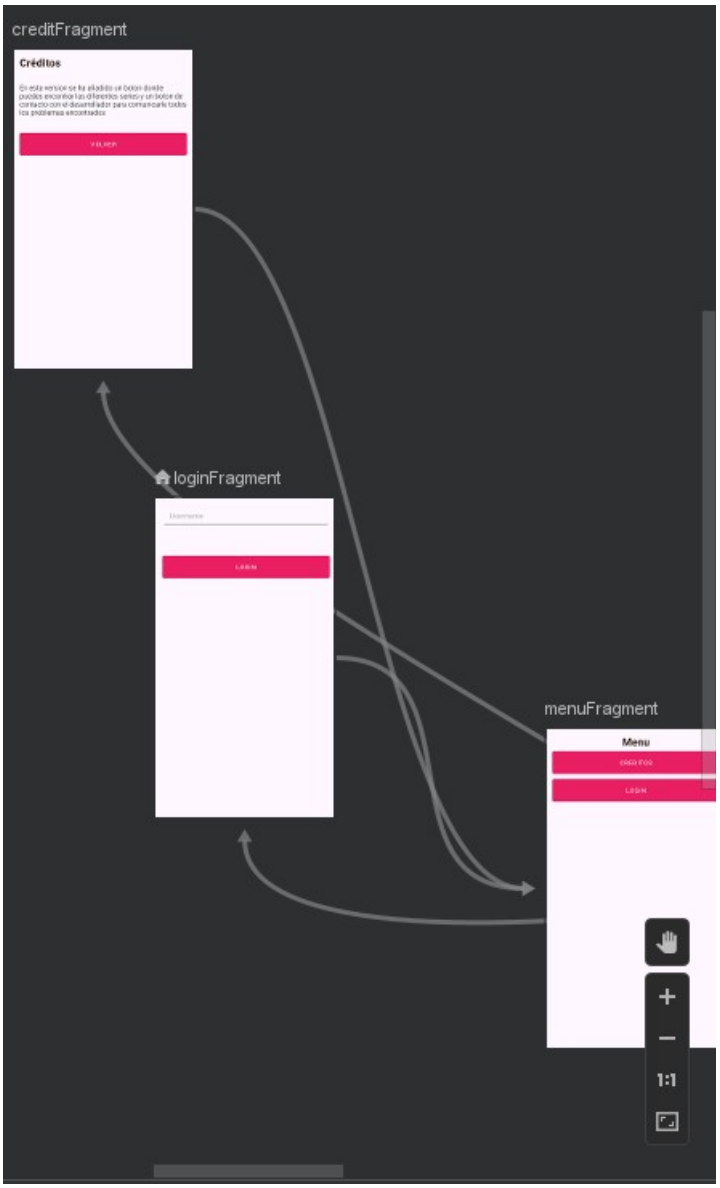
Con estos pasos, has trasladado el contenido de MainActivity y CreditActivity a fragmentos y configurado view binding para los fragmentos

Crear un sistema de navegación de modo que la aplicación se inicia con el “SplashActivity” (queda fuera de la navegación) y a continuación viaja al “MainActivity”, que por defecto carga el contenido del fragmento “LoginFragment”. En el fragmento de Login se podrá introducir el nombre del usuario y llegar a un fragmento de Menú (MenuFragment), desde este fragmento se podrá navegar hasta el de créditos y viceversa. Desde el MenuFragment se podrá navegar hasta el de Login si se pulsa en el botón de “Salir” (que debe crearse). (1)

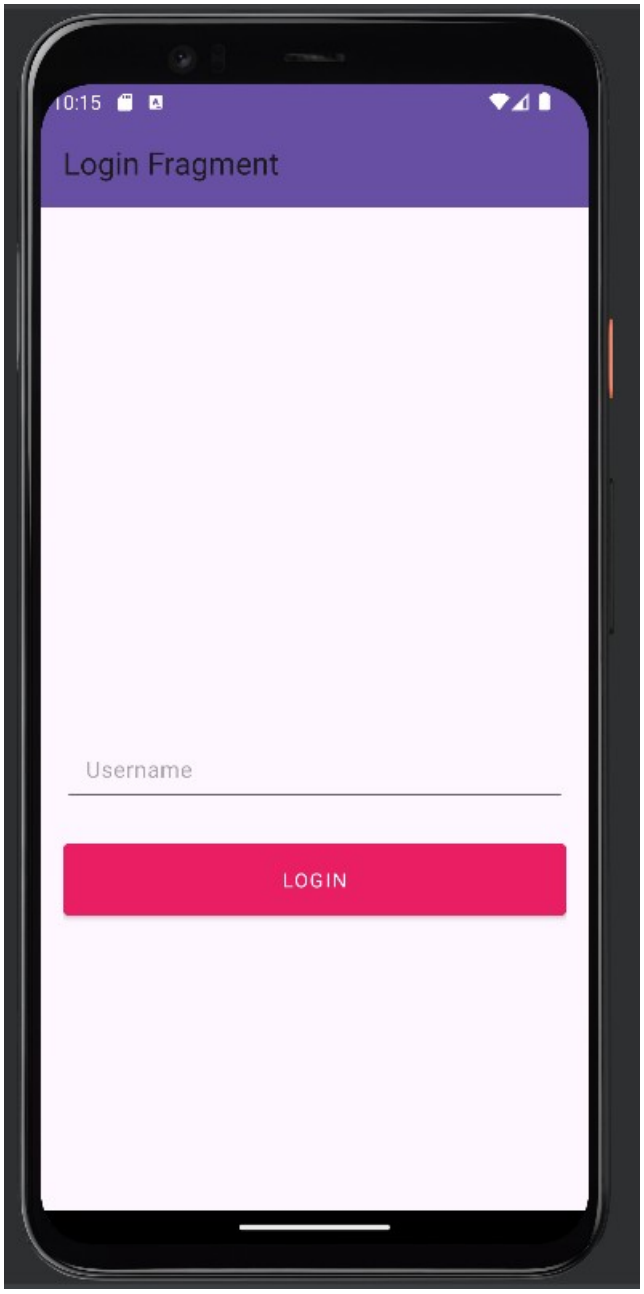
```kotlin
package com.marisma.fase1recu

import ...

class SplashActivity : AppCompatActivity() {
    @SuppressLint("ResourceType")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Set the content view to your splash screen layout
        setContentView(R.navigation.nav_graph)

        // Delay the transition to MainActivity
        Handler(Looper.getMainLooper()).postDelayed({
            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
            finish()
        }, delayMillis: 2000) // 2 segundos de retraso
    }
}
```

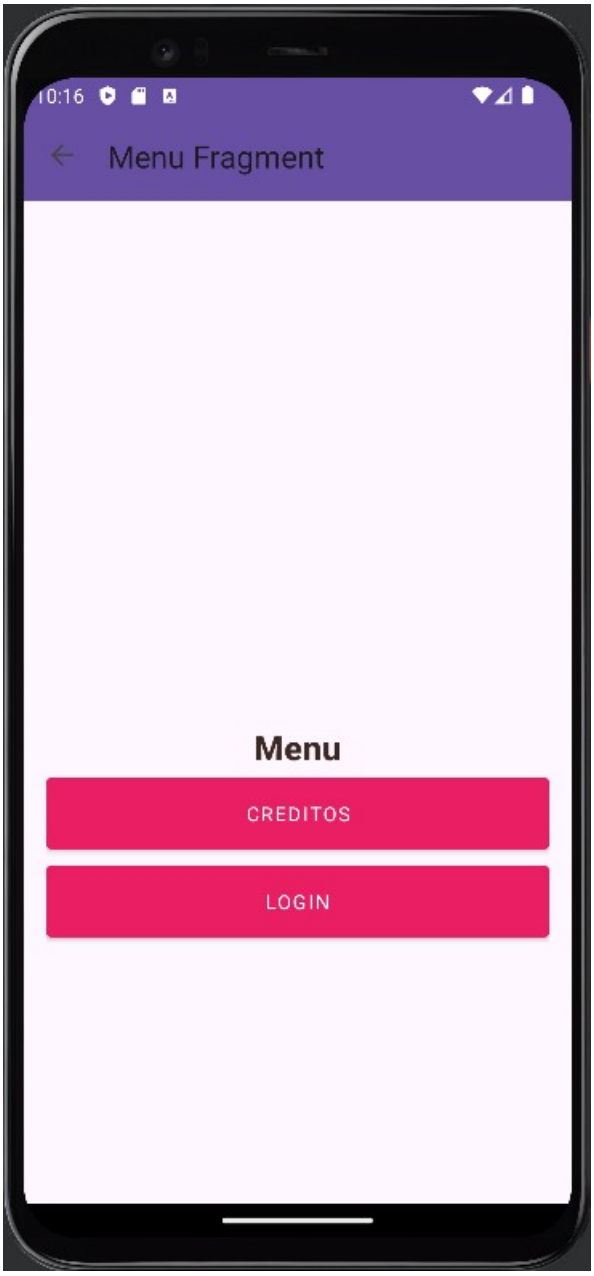
```kotlin
package com.marisma.fase1recu.adapter

import ...

class LoginFragment : Fragment() {

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

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment using view binding
        _binding = FragmentLoginBinding.inflate(inflater, container, attachToParent: false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Set click listener for the login button
        binding.loginButton.setOnClickListener { it: View!
            // Here you can add authentication logic
            findNavController().navigate(R.id.action_loginFragment_to_menuFragment)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        // Clear the binding reference to avoid memory leaks
        _binding = null
    }
}
```

```kotlin
import ...

class MenuFragment : Fragment() {

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

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment using view binding
        _binding = FragmentMenuBinding.inflate(inflater, container,  attachToParent: false
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Set click listener for the credits button
        binding.creditsButton.setOnClickListener { it: View!
            findNavController().navigate(R.id.action_menuFragment_to_creditsFragment)
        }
        // Set click listener for the logout button
        binding.logoutButton.setOnClickListener { it: View!
            findNavController().navigate(R.id.action_menuFragment_to_loginFragment)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        // Clear the binding reference to avoid memory leaks
        _binding = null
    }
}
```


creditFragment
Créditos
loginFragment
menuFragment
Menu
1:1

Realizar pruebas de uso en el emulador, indicando si encuentra fallos y sus soluciones. (0,25)



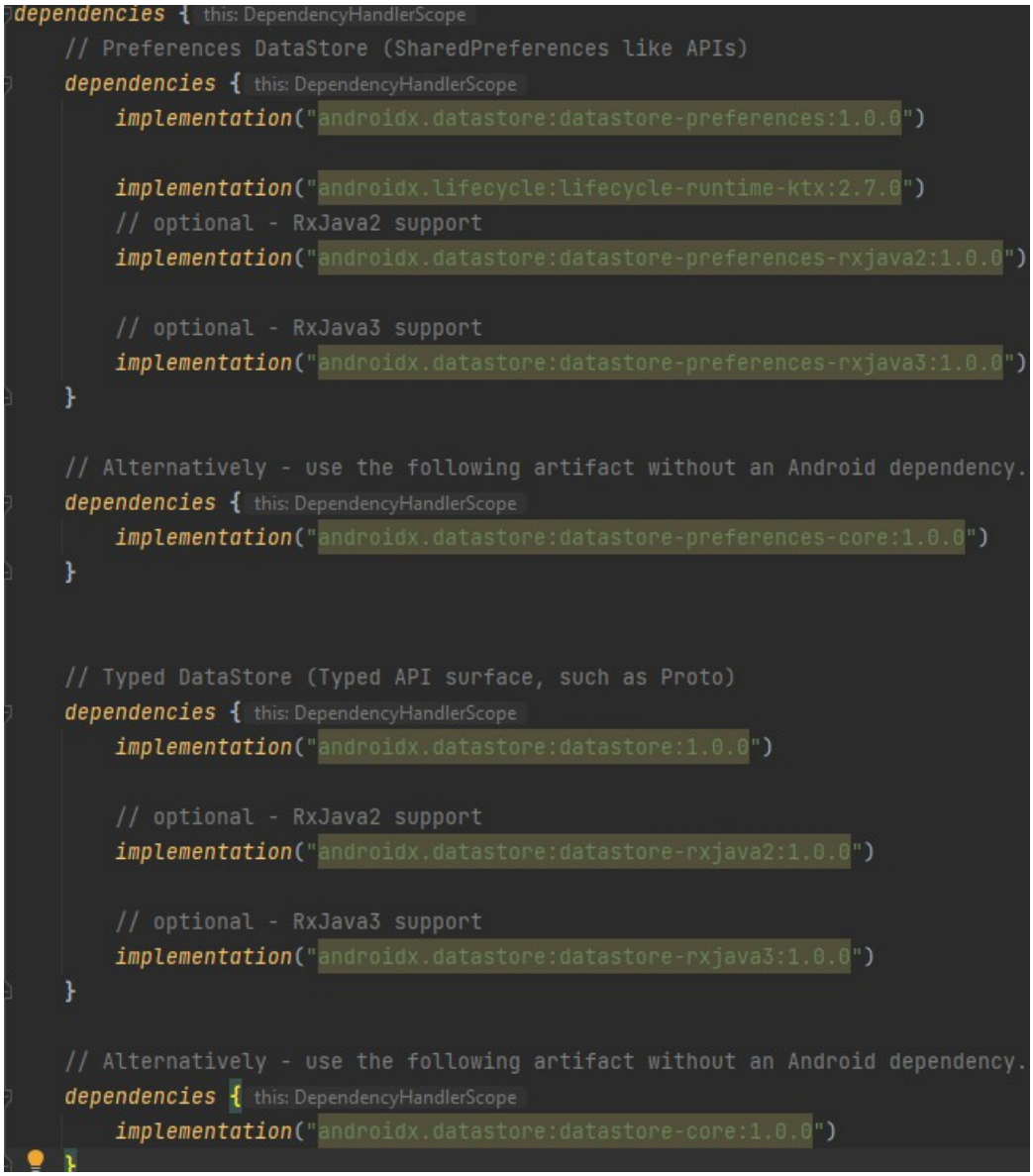


Personalizar completamente los colores de la aplicación, tanto para el modo estándar como para el oscuro. (1)

```xml
<!-- Colores de tema claro -->
<color name="seed">#E91E63</color>
<color name="md_theme_light_primary">#E91E63</color>
<color name="md_theme_light_onPrimary">#FFFFFF</color>
<color name="md_theme_light_primaryContainer">#F8BBD0</color>
<color name="md_theme_light_onPrimaryContainer">#880E4F</color>
<color name="md_theme_light_secondary">#9C27B0</color>
<color name="md_theme_light_onSecondary">#FFFFFF</color>
<color name="md_theme_light_secondaryContainer">#E1BEE7</color>
<color name="md_theme_light_onSecondaryContainer">#4A148C</color>
<color name="md_theme_light_tertiary">#BA68C8</color>
<color name="md_theme_light_onTertiary">#FFFFFF</color>
<color name="md_theme_light_tertiaryContainer">#E1BEE7</color>
<color name="md_theme_light_onTertiaryContainer">#4A148C</color>
<color name="md_theme_light_error">#D32F2F</color>
<color name="md_theme_light_errorContainer">#FFCDD2</color>
<color name="md_theme_light_onError">#FFFFFF</color>
<color name="md_theme_light_onErrorContainer">#B71C1C</color>
<color name="md_theme_light_background">#FFF3E0</color>
<color name="md_theme_light_onBackground">#3E2723</color>
<color name="md_theme_light_surface">#FFF3E0</color>
<color name="md_theme_light_onSurface">#3E2723</color>
<color name="md_theme_light_surfaceVariant">#F3E5F5</color>
<color name="md_theme_light_onSurfaceVariant">#6A1B9A</color>
<color name="md_theme_light_outline">#BDBDBD</color>
<color name="md_theme_light_inverseOnSurface">#FFFFFF</color>
<color name="md_theme_light_inverseSurface">#3E2723</color>
<color name="md_theme_light_inversePrimary">#E1BEE7</color>
<color name="md_theme_light_shadow">#000000</color>
<color name="md_theme_light_surfaceTint">#E91E63</color>
<color name="md_theme_light_outlineVariant">#F3E5F5</color>
<color name="md_theme_light_scrim">#000000</color>
```

```xml
    <!-- Colores de tema oscuro -->
    <color name="md_theme_dark_primary">#E1BEE7</color>
    <color name="md_theme_dark_onPrimary">#4A148C</color>
    <color name="md_theme_dark_primaryContainer">#6A1B9A</color>
    <color name="md_theme_dark_onPrimaryContainer">#E1BEE7</color>
    <color name="md_theme_dark_secondary">#BA68C8</color>
    <color name="md_theme_dark_onSecondary">#4A148C</color>
    <color name="md_theme_dark_secondaryContainer">#7B1FA2</color>
    <color name="md_theme_dark_onSecondaryContainer">#E1BEE7</color>
    <color name="md_theme_dark_tertiary">#D1C4E9</color>
    <color name="md_theme_dark_onTertiary">#4A148C</color>
    <color name="md_theme_dark_tertiaryContainer">#BA68C8</color>
    <color name="md_theme_dark_onTertiaryContainer">#D1C4E9</color>
    <color name="md_theme_dark_error">#FFB4AB</color>
    <color name="md_theme_dark_errorContainer">#93000A</color>
    <color name="md_theme_dark_onError">#690005</color>
    <color name="md_theme_dark_onErrorContainer">#FFDAD6</color>
    <color name="md_theme_dark_background">#3E2723</color>
    <color name="md_theme_dark_onBackground">#FFF3E0</color>
    <color name="md_theme_dark_surface">#3E2723</color>
    <color name="md_theme_dark_onSurface">#FFF3E0</color>
    <color name="md_theme_dark_surfaceVariant">#6A1B9A</color>
    <color name="md_theme_dark_onSurfaceVariant">#D1C4E9</color>
    <color name="md_theme_dark_outline">#BDBDBD</color>
    <color name="md_theme_dark_inverseOnSurface">#3E2723</color>
    <color name="md_theme_dark_inverseSurface">#FFF3E0</color>
    <color name="md_theme_dark_inversePrimary">#E91E63</color>
    <color name="md_theme_dark_shadow">#000000</color>
    <color name="md_theme_dark_surfaceTint">#E1BEE7</color>
    <color name="md_theme_dark_outlineVariant">#6A1B9A</color>
    <color name="md_theme_dark_scrim">#000000</color>
    <color name="Header">#E91E63</color>
</resources>
```

Incorporar una o varias fuentes de datos para la aplicación.



```
dependencies { this: DependencyHandlerScope
    // Preferences DataStore (SharedPreferences like APIs)
    dependencies { this: DependencyHandlerScope
        implementation("androidx.datastore:datastore-preferences:1.0.0")

        implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")
        // optional - RxJava2 support
        implementation("androidx.datastore:datastore-preferences-rxjava2:1.0.0")

        // optional - RxJava3 support
        implementation("androidx.datastore:datastore-preferences-rxjava3:1.0.0")
    }

    // Alternatively - use the following artifact without an Android dependency.
    dependencies { this: DependencyHandlerScope
        implementation("androidx.datastore:datastore-preferences-core:1.0.0")
    }


    // Typed DataStore (Typed API surface, such as Proto)
    dependencies { this: DependencyHandlerScope
        implementation("androidx.datastore:datastore:1.0.0")

        // optional - RxJava2 support
        implementation("androidx.datastore:datastore-rxjava2:1.0.0")

        // optional - RxJava3 support
        implementation("androidx.datastore:datastore-rxjava3:1.0.0")
    }

    // Alternatively - use the following artifact without an Android dependency.
    dependencies { this: DependencyHandlerScope
        implementation("androidx.datastore:datastore-core:1.0.0")
    }
```

Creación de estilos personalizados, al menos para los siguientes elementos: (0,75)

Textos.

Títulos.

Botones.

```
    <!-- Estilo personalizado para textos -->
    <style name="CustomText" parent="TextAppearance.MaterialComponents.Body1">
        <item name="android:fontFamily">@font/font_family</item>
        <item name="android:textColor">@color/md_theme_light_onBackground</item>
        <item name="android:textSize">16sp</item>
    </style>

    <!-- Estilo personalizado para titulos -->
    <style name="CustomTitle" parent="TextAppearance.MaterialComponents.Headline6">
        <item name="android:fontFamily">@font/font_family</item>
        <item name="android:textColor">@color/md_theme_light_onBackground</item>
        <item name="android:textSize">24sp</item>
        <item name="android:textStyle">bold</item>
    </style>

    <!-- Estilo personalizado para botones -->
    <style name="CustomButton" parent="Widget.MaterialComponents.Button">
        <item name="android:fontFamily">@font/font_family</item>
        <item name="android:textColor">@color/md_theme_light_onPrimary</item>
        <item name="android:backgroundTint">@color/md_theme_light_primary</item>
        <item name="android:padding">16dp</item>
        <item name="android:minHeight">48dp</item>
    </style>
</resources>
```

Tras haber estudiado los diferentes “Layouts” y algunos de sus elementos auxiliares como las ventanas deslizantes (ViewPager2) o las tarjetas (CardView) es el momento de crear los diferentes fragmentos que va a contener la aplicación: (CE 2A y CE 2B)

Los fragmentos a crear son:

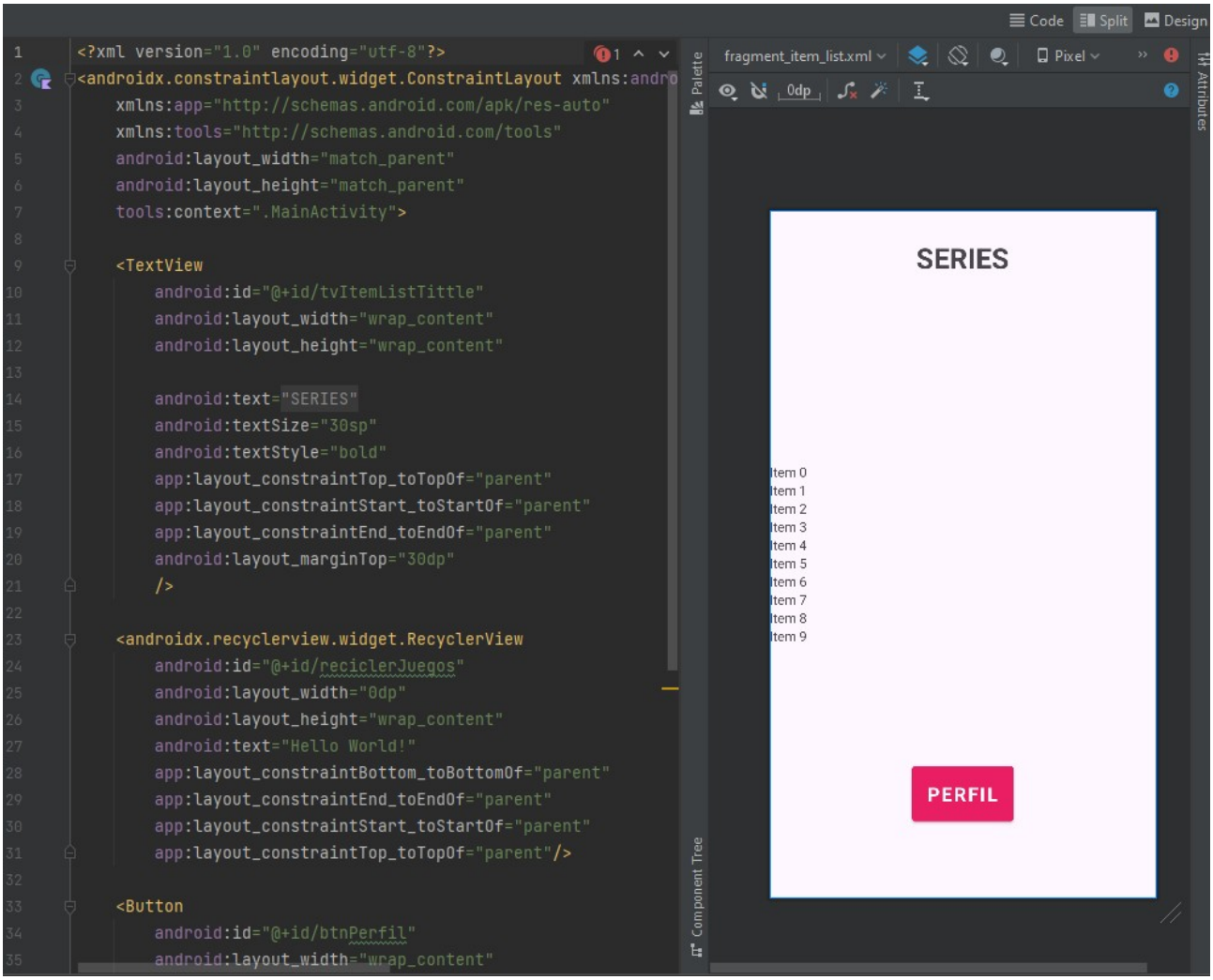
ItemListFragment: Fragmento que contendrá la lista de elementos de los que trata la aplicación. Debe contener un “botón” para incluir el elemento en “Favoritos”. (1)
DetailItemFragment: Fragmento con información más detallada de un elemento de los que trata la información. (1)

UserInfoFragment: Fragmento con información del usuario que se encuentra en la aplicación. (1)

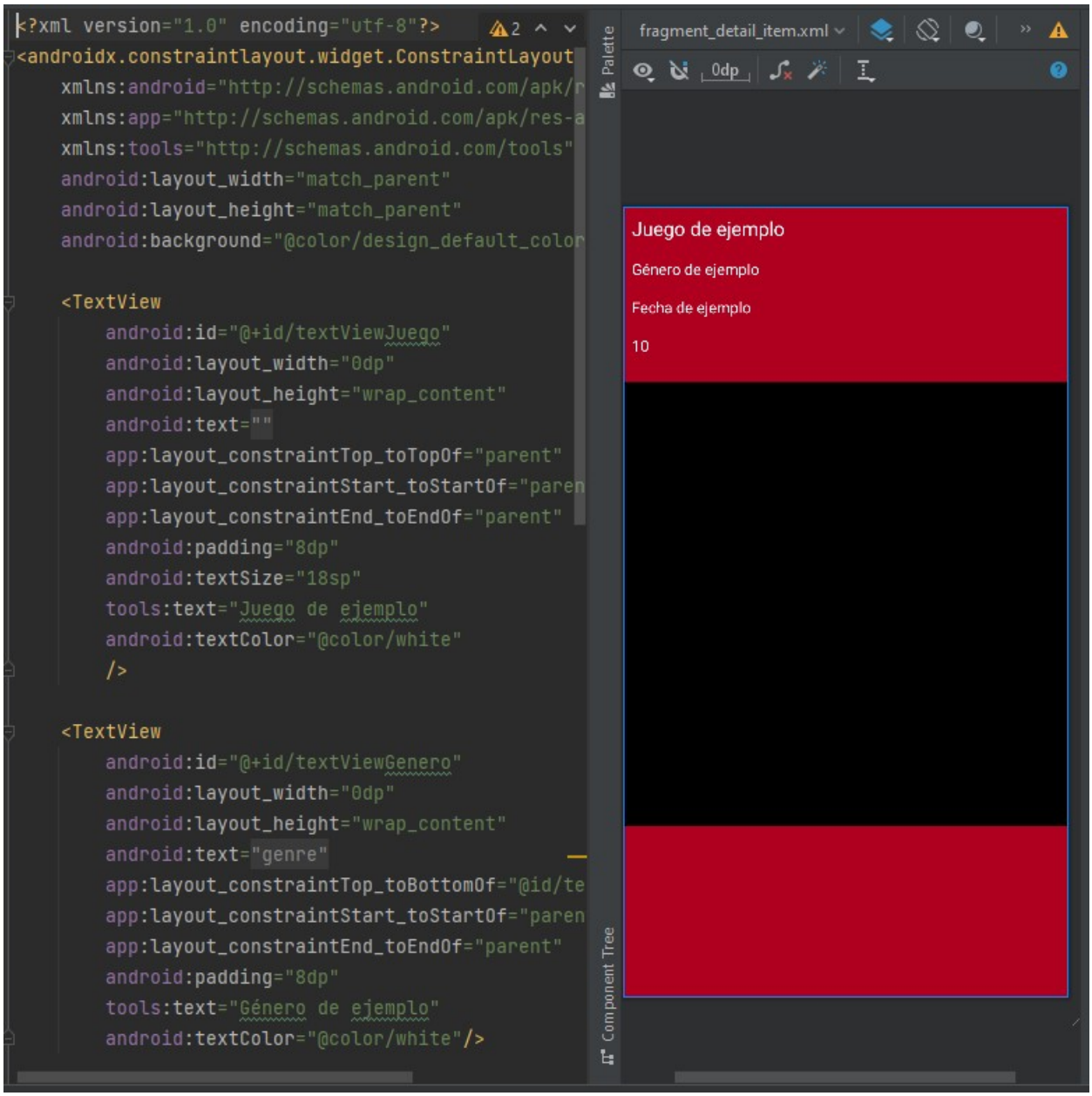
FavItemListFragment: Fragmento con la lista de elementos favoritos del usuario. Debe tener un botón para eliminar el elemento de la lista de favoritos. (0,5)

DetailFavItemFragment: Fragmento con información extra que el usuario puede añadir sobre los elementos de la aplicación, como por ejemplo sus anotaciones personales, su puntuación, etc. En este fragmento debe existir un botón flotante para incluir nuevos comentarios “privados” en este ítem. (0,5)

ItemListFragment:

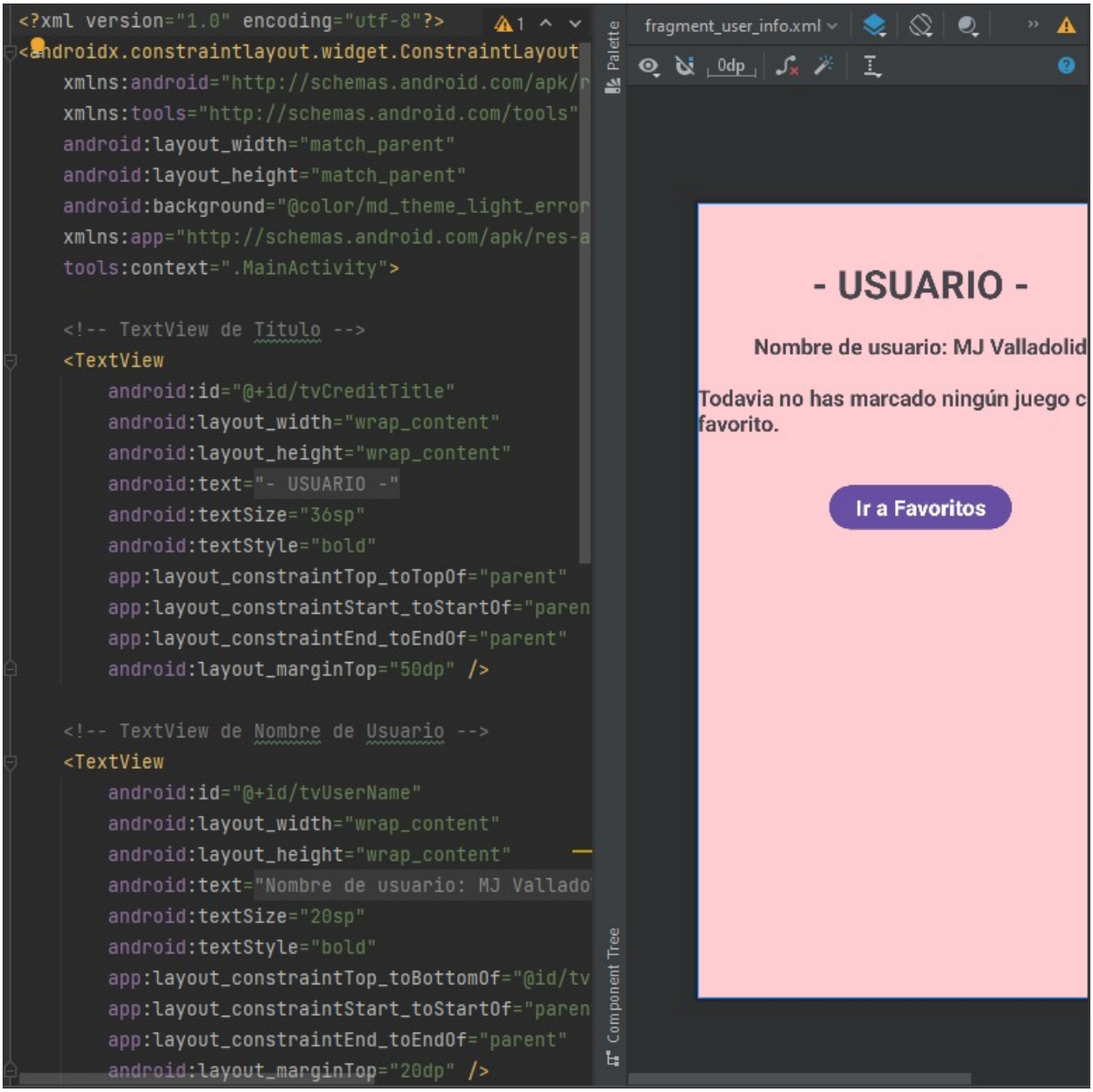


DetailItemFragment:

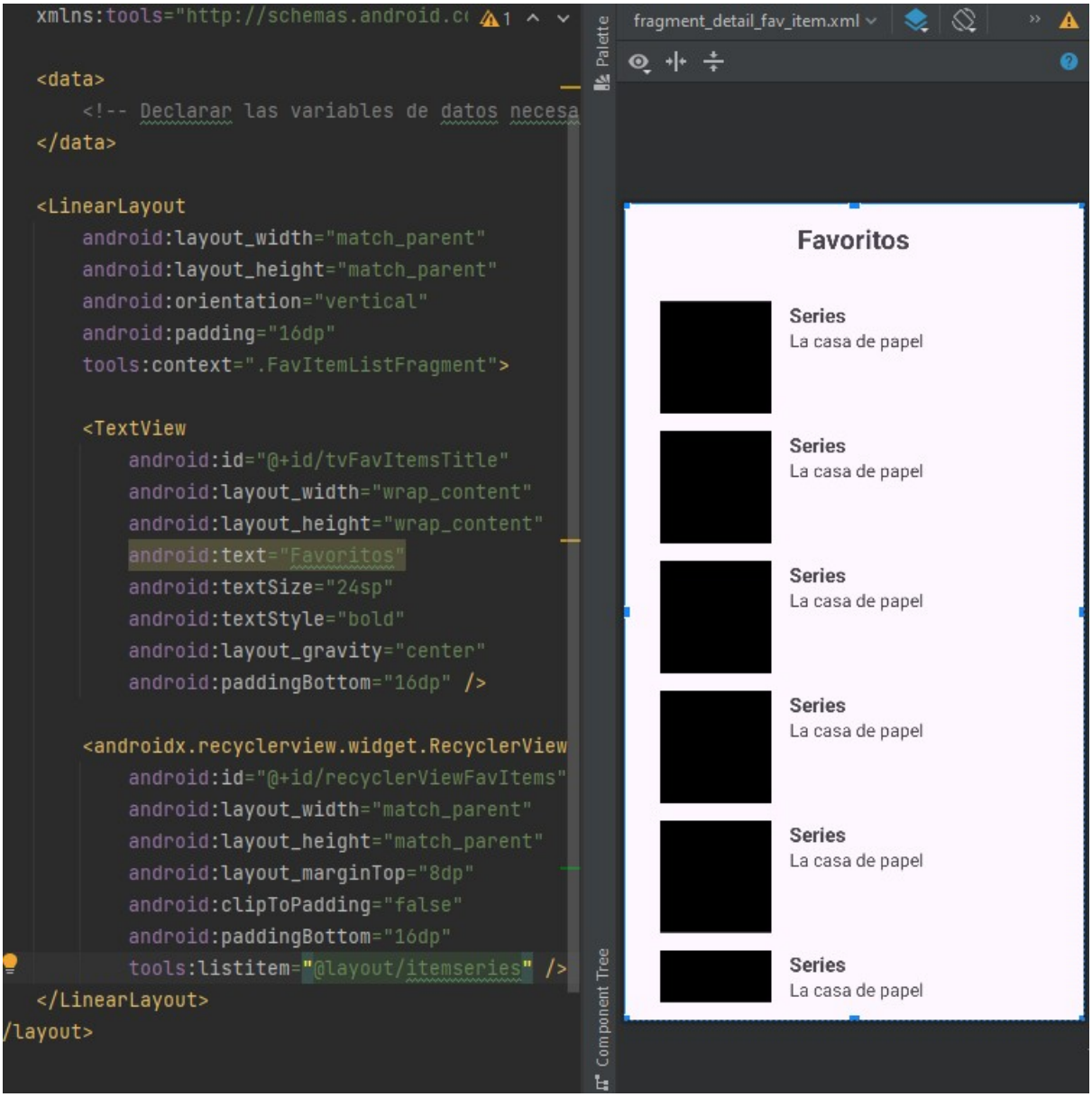



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/r
    xmlns:app="http://schemas.android.com/apk/res-a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/design_default_color

    <TextView
        android:id="@+id/textViewJuego"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text=""
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="paren
        app:layout_constraintEnd_toEndOf="parent"
        android:padding="8dp"
        android:textSize="18sp"
        tools:text="Juego de ejemplo"
        android:textColor="@color/white"
        />

    <TextView
        android:id="@+id/textViewGenero"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="genre"
        app:layout_constraintTop_toBottomOf="@id/te
        app:layout_constraintStart_toStartOf="paren
        app:layout_constraintEnd_toEndOf="parent"
        android:padding="8dp"
        tools:text="Género de ejemplo"
        android:textColor="@color/white"/>
```

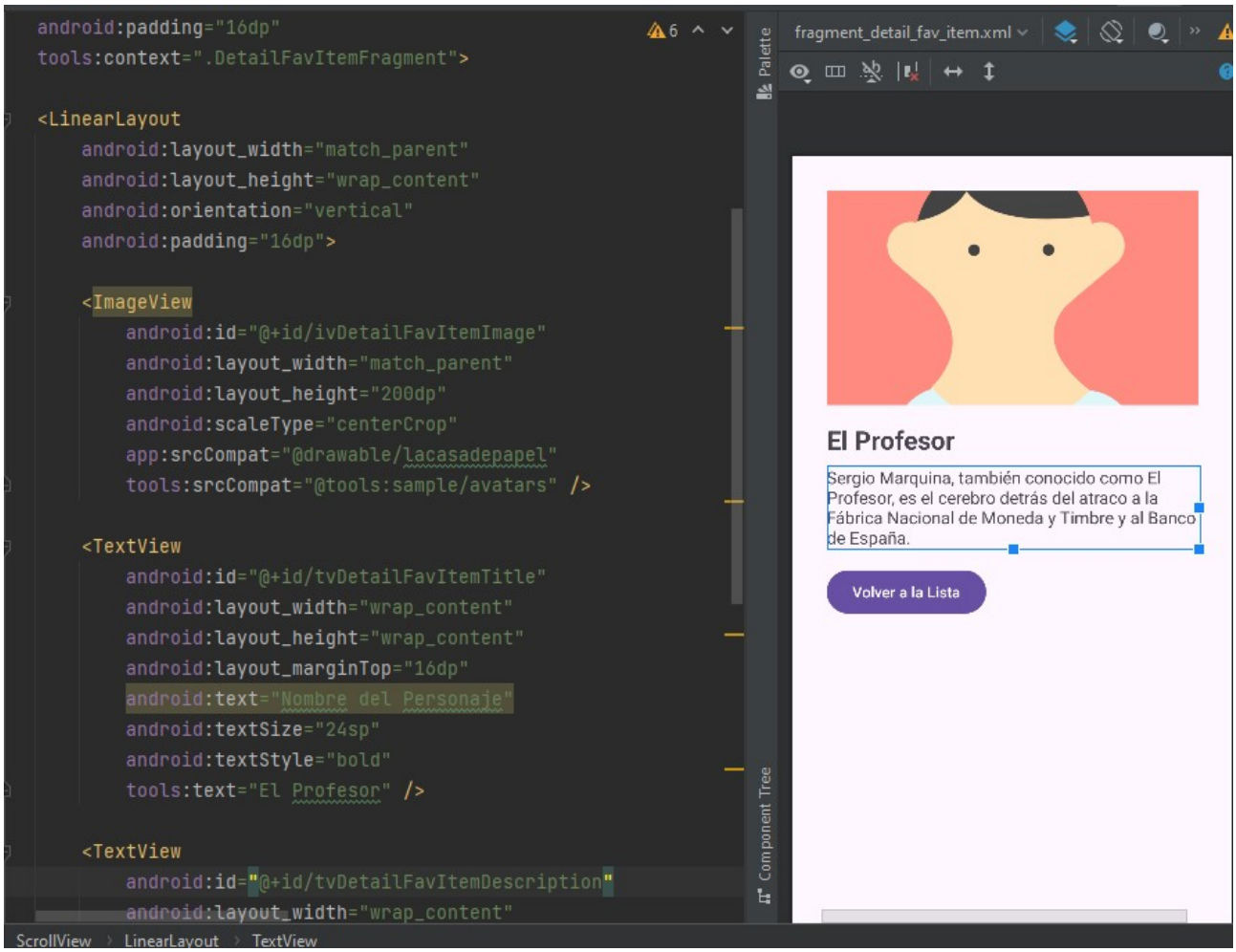
UserInfoFragment:



FavItemListFragment:



DetailFavItemFragment:



Transforma los “LinearLayout” de los fragmentos que estaban ya creados con anterioridad en “ConstraintLayout”. (0,25)

ItemListFragment:

DetailItemFragment:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res
    xmlns:app="http://schemas.android.com/apk/res-aut
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/design_default_color_

    <TextView
        android:id="@+id/textViewJuego"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text=""
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:padding="8dp"
        android:textSize="18sp"
        tools:text="Juego de ejemplo"
        android:textColor="@color/white"
        />

    <TextView
        android:id="@+id/textViewGenero"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="genre"
        app:layout_constraintTop_toBottomOf="@id/text
```

UserInfoFragment:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/md_theme_light_errorCo
    xmlns:app="http://schemas.android.com/apk/res-aut
    tools:context=".MainActivity">

    <!-- TextView de Título -->
    <TextView
        android:id="@+id/tvCreditTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="- USUARIO -"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="50dp" />

    <!-- TextView de Nombre de Usuario -->
    <TextView
        android:id="@+id/tvUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nombre de usuario: MJ Valladolid
```

FavItemListFragment

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FavItemListFragment">

    <TextView
        android:id="@+id/tvFavItemsTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Favoritos"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp" />

    <androidx.recyclerview.widget.RecyclerView
```

DetailFavItemFragment

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp">

    <ImageView
        android:id="@+id/ivDetailFavItemImage"
        android:layout_width="0dp"
        android:layout_height="200dp"
        android:scaleType="centerCrop"
        app:srcCompat="@drawable/lacasadepapel"
        tools:srcCompat="@tools:sample/avatars"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <TextView
        android:id="@+id/tvDetailFavItemTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Nombre del Personaje"
```

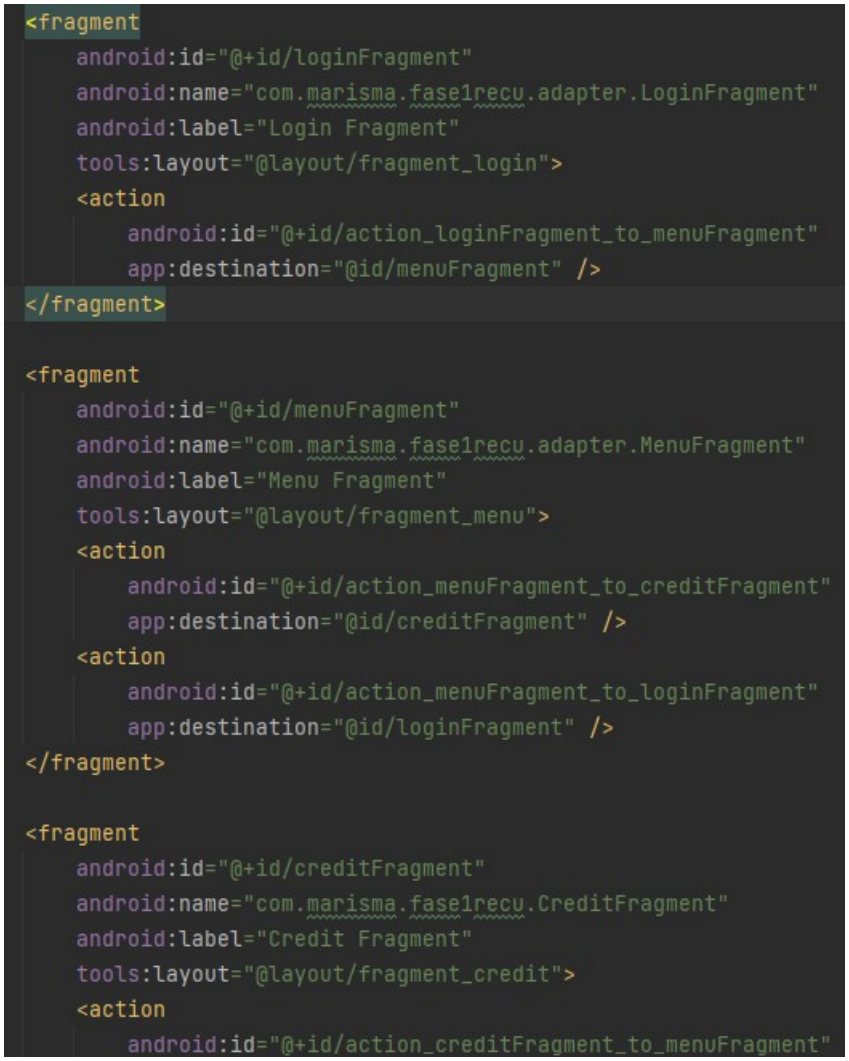
Crea un par de pestañas deslizantes (ViewPager2) con información sobre la aplicación que quieras resaltar. En la segunda pestaña debe existir el botón "Comenzar". Estas pestañas deslizantes se incluirán en la lógica de navegación tras la inclusión del usuario (LoginFragment → Pestañas ViewPager2) y al pulsar sobre el botón "Comenzar" navegará al menú principal (Pestaña ViewPager2 → MenuFragment). Si desde el menú principal se pulsa en "Salir", volvería directamente al fragmento de Login (MenuFragment → LoginFragment). (1)

```kotlin
package com.marisma.fase1recu

import ...

class ViewPagerAdapter(fragmentActivity: FragmentActivity) :
    FragmentStateAdapter(fragmentActivity) {

    override fun getItemCount(): Int {
        return 2
    }

    override fun createFragment(position: Int): Fragment {
        return when (position) {
            0 -> InfoFragment()
            1 -> StartFragment()
            else -> throw IllegalArgumentException("Invalid position")
        }
    }
}
```

```kotlin
package com.marisma.fase1recu

import ...

class StartFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        val view = inflater.inflate(R.layout.fragment_start, container, attachToRoot: false)
        // Set click listener for the startButton
        view.findViewById<Button>(R.id.startButton).setOnClickListener { it: View!
            // Navigate to the MenuFragment
            findNavController().navigate(R.id.action_startFragment_to_menuFragment)
        }
        return view
    }
}
```

```kotlin
package com.marisma.fase1recu

import ...

class InfoFragment : Fragment() {

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

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment using view binding
        _binding = FragmentInfoBinding.inflate(inflater, container, attachToParent: false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        // Perform any additional setup after the view is created
    }

    override fun onDestroyView() {
        super.onDestroyView()
        // Clear the binding reference to avoid memory leaks
        _binding = null
    }
}
```

Crear la navegación entre todos los fragmentos creados. La información que se muestran en los fragmentos se debe crear desde objetos incorporados y creados dentro de la lógica de la aplicación, en las propias clases o en un objeto (Singleton) que proporcione estos datos. En la próxima unidad, se tomarán los datos desde otras fuentes de datos. (0,5)



```
<fragment
    android:id="@+id/loginFragment"
    android:name="com.marisma.fase1recu.adapter.LoginFragment"
    android:label="Login Fragment"
    tools:layout="@layout/fragment_login">
    <action
        android:id="@+id/action_loginFragment_to_menuFragment"
        app:destination="@id/menuFragment" />
</fragment>

<fragment
    android:id="@+id/menuFragment"
    android:name="com.marisma.fase1recu.adapter.MenuFragment"
    android:label="Menu Fragment"
    tools:layout="@layout/fragment_menu">
    <action
        android:id="@+id/action_menuFragment_to_creditFragment"
        app:destination="@id/creditFragment" />
    <action
        android:id="@+id/action_menuFragment_to_loginFragment"
        app:destination="@id/loginFragment" />
</fragment>

<fragment
    android:id="@+id/creditFragment"
    android:name="com.marisma.fase1recu.CreditFragment"
    android:label="Credit Fragment"
    tools:layout="@layout/fragment_credit">
    <action
        android:id="@+id/action_creditFragment_to_menuFragment"
```