PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

PROYECTO: INTERFAZ DE USUARIO 2

android

REALIZADO POR:

Miguel Figuerola Alonso

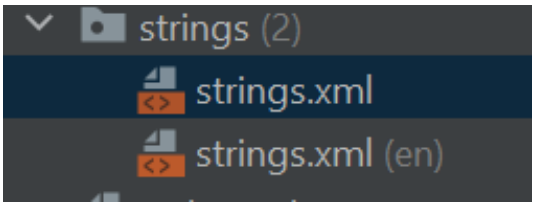
ÍNDICE

1. Realizar que la aplicación soporte como mínimo dos idiomas. Todos los textos y opciones se deben poder traducir. (No realizar las traducciones de los textos de prueba, es decir, de objetos temporales). (1)

Dentro del directorio de values nos encontraremos con el archivo strings.xml, en él podremos traducir todos los textos que tengamos de forma directa abriendo el editor de traducción que nos ofrece Android Studio.

Key	Resource Folder	Untranslatable	Default Value	English (en)
app_name	app/src/main/res	☐	SocialCooking	SocialCooking
usuario	app/src/main/res	☐	Usuario	User
aceptar	app/src/main/res	☐	Aceptar	Accept
iniciar	app/src/main/res	☐	Iniciar	Start
creditos	app/src/main/res	☐	Creditos	Credits
salir	app/src/main/res	☐	Salir	Exit
errorUsuario	app/src/main/res	☐	Introduce el nombre del usuario.	Enter the username.
bienvenidoUsuario	app/src/main/res	☐	Bienvenido	Welcome

Al traducir todos los strings que tengamos al idioma que seleccionemos, se nos creará de forma automática otro archivo strings.xml pero esta vez con nuestros textos ya traducidos.



2. Incluir el UpButton en la aplicación. (1)

Hemos incluido un FloatingActionButton para realizar la función de UpButton siguiendo la lógica de nuestro NavigationController.

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/upButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="true"
    android:src="@android:drawable/ic_menu_revert"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.045"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0" />
androidx.constraintlayout.widget.ConstraintLayout>
```

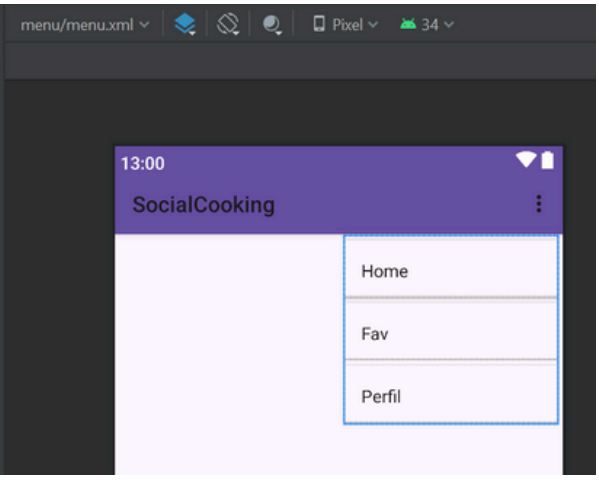
```
binding.upButton.setOnClickListener{ it: View!
    findNavController().navigate(R.id.action_inicioFragment_to_menuFragment)
}
```

3. Crear un menú inferior en la aplicación con las opciones mínimas de: (1)

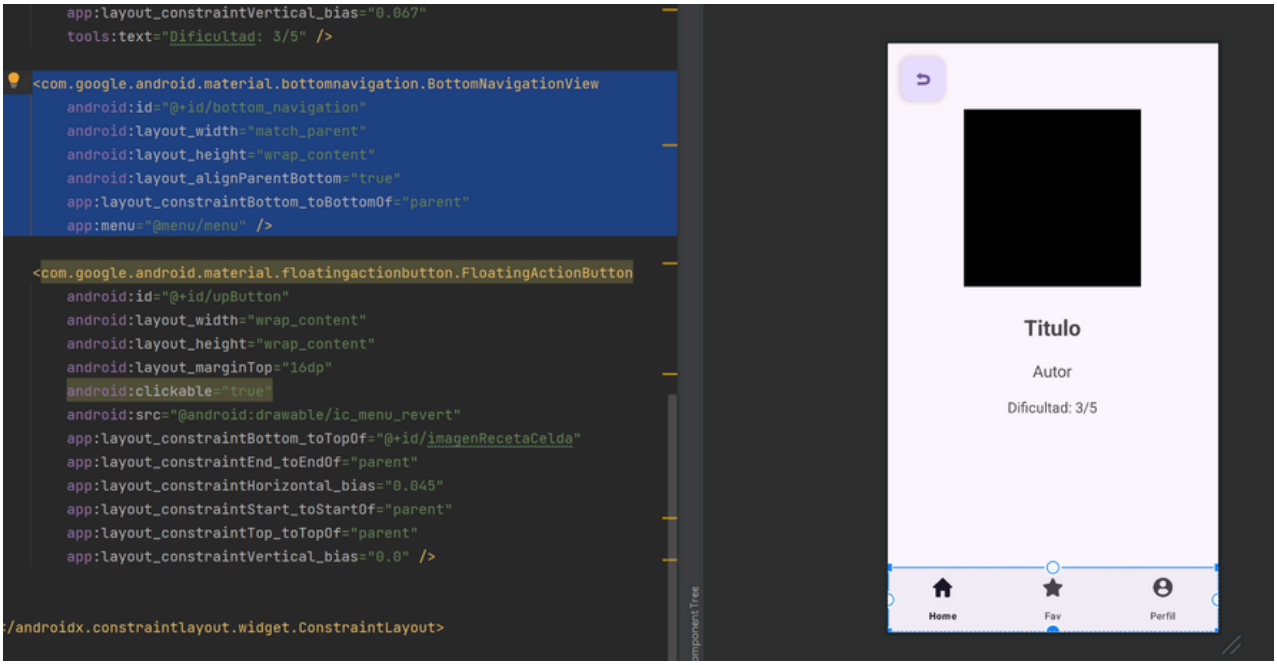
- Descubrir: Lleva a ItemListFragment.
- Favoritos: Lleva a FavItemListFragment.
- Personal: Lleva a UserInfoFragment.

Para crear un menú inferior, lo primero que tenemos que hacer sería crear un nuevo directorio llamado menu, dentro de nuestra carpeta de recursos. Una vez creado tendremos que crear un archivo .xml e implementar el menú, como veremos a continuación.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/androi
    <item
        android:id="@+id/navigation_home"
        android:icon="@drawable/hogar"
        android:title="Home" />
    <item
        android:id="@+id/navigation_fav"
        android:icon="@drawable/favorito"
        android:title="Fav" />
    <item
        android:id="@+id/navigation_perfil"
        android:icon="@drawable/iconperfil"
        android:title="Perfil" />
</menu>
```



Lo único que tenemos que hacer sería llamar a nuestro menú en los fragments que queremos que aparezca, como veremos a continuación.

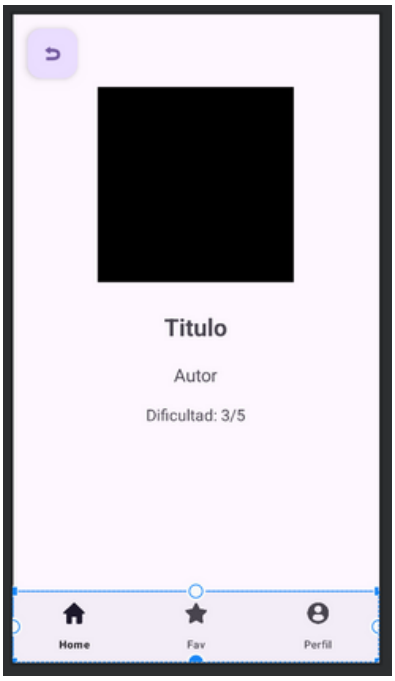


En nuestro navgraph.xml crearemos las acciones oportunas para poder desplazarnos por los diferentes fragments. Una vez hecho esto, sería crear una función que nos desplace hacia los fragments que queramos.

```
private fun llamarMenu(){
    binding.bottomNavigation.setOnNavigationItemSelectedListener { item ->
        when (item.itemId) {
            R.id.navigation_home -> {

                true ^setOnNavigationItemSelectedListener
            }
            R.id.navigation_fav -> {
                findNavController().navigate(R.id.action_inicioFragment_to_favsFragment)
                true ^setOnNavigationItemSelectedListener
            }
            R.id.navigation_perfil -> {
                // Acción para el ítem Notifications
                findNavController().navigate(R.id.action_inicioFragment_to_perfilFragment)
                true ^setOnNavigationItemSelectedListener
            }
            else -> false ^setOnNavigationItemSelectedListener
        }
    }
}
```

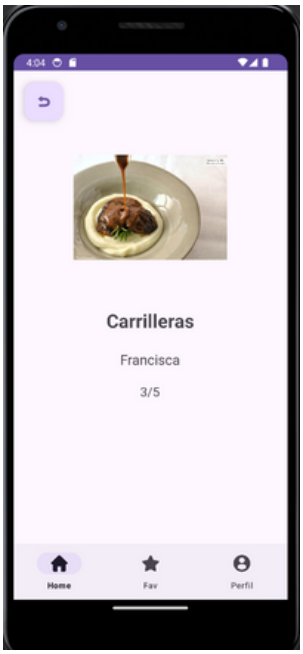
4. Al pulsar sobre un elemento de una lista se debe navegar a su fragmento de detalle en el que se obtienen los datos específicos del elemento marcado. (1)

Hemos creado un fragmento en el cual almacenará los detalles del elemento que pulsemos.



Con ayuda de un bundle, guardaremos la información de los detalles y lo cargaremos en el fragmento creado con anterioridad.

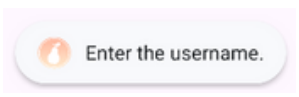
```
class recetasViewHolder(view: View, private val navController: NavController) : RecyclerView.ViewHolder(view) {

    val binding = RecetasLayoutBinding.bind(view)

    fun render(datosRecetasModel: datosRecetas) {
        binding.nombreRecetaTextView.text = datosRecetasModel.nombre
        binding.autorRecetaTextView.text = datosRecetasModel.autor
        binding.calificacionRecetaTextView.text = datosRecetasModel.dificultad
        Glide.with(binding.imagenReceta.context).load(datosRecetasModel.urlFoto).into(binding.imagenReceta)

        itemView.setOnClickListener { it: View!
            val bundle = Bundle().apply { this: Bundle
                putString("nombre", datosRecetasModel.nombre)
                putString("autor", datosRecetasModel.autor)
                putString("dificultad", datosRecetasModel.dificultad)
                putString("urlFoto", datosRecetasModel.urlFoto)
            }
            navController.navigate(R.id.action_inicioFragment_to_contenidoCeldaFragment, bundle)
        }

        binding.favButton.setOnClickListener { it: View!
            Toast.makeText(itemView.context, text: "Se ha añadido a favoritos!", Toast.LENGTH_SHORT).show()
        }
    }
}
```

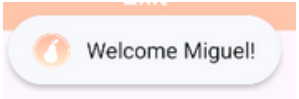


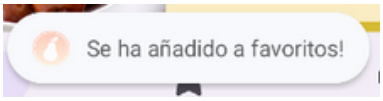
5. Incluir mensajes emergentes de información en los casos que consideres necesarios. Se deben incluir un mínimo de dos de estos elementos. Se pueden usar “Toast” o “Snackbar”. (0,5)

He incluido tres elementos Toast. Uno por si el usuario no ha introducido ningún nick, otro de bienvenida y por último cuando agregas una receta a favoritos.

```
binding.aceptarButton.setOnClickListener{ it: View!

    var usuario = binding.nickText.text.toString()

    val toastError = "Introduce el nombre del usuario."
    val toast = "Bienvenido"

    if(usuario.isNotEmpty()){
        svm.usuario = usuario
        findNavController().navigate(R.id.action_loginFragment_to_menuFragment)
        Toast.makeText(requireContext(), text: toast + " " + usuario + "!", Toast.LENGTH_SHORT).show()
    }
    else{
        Toast.makeText(requireContext(),toastError, Toast.LENGTH_SHORT).show()
    }
}
```








```
binding.favButton.setOnClickListener { it: View!
    Toast.makeText(itemView.context, text: "Se ha añadido a favoritos!", Toast.LENGTH_SHORT).show()
}
```

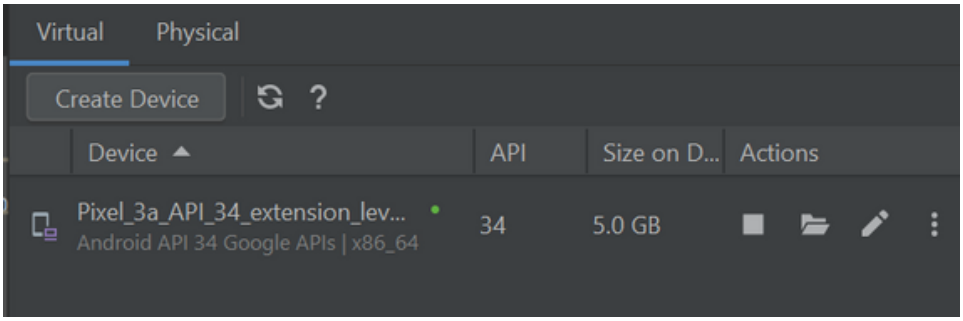
6. Realizar pruebas de uso en el emulador, indicando si encuentra fallos y sus soluciones. (0,5).

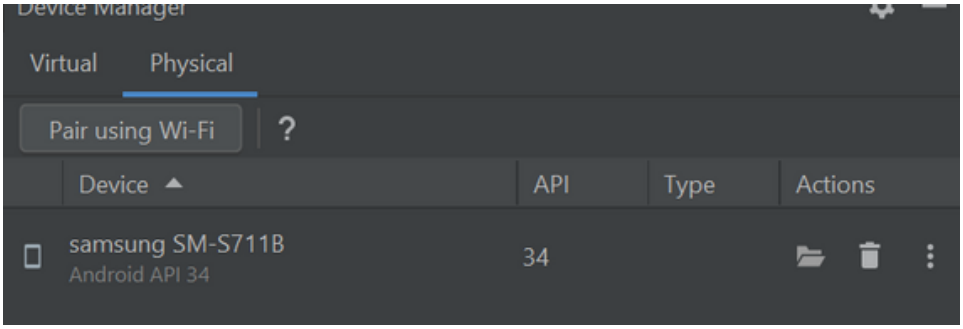
En el directorio raíz del proyecto incluyo un vídeo del funcionamiento.

```
app
2024-06-12 06:13:08: Launching app on 'Pixel_3a_API_34_extension_level_7_x86_64.
$ adb shell am start -n "com.makulky.socialcooking/com.makulky.socialcooking.SplashActivity" -a android.intent.action.MAIN -c android.intent.category.LAUNCHER --spla

Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.makulky.socialcooking/.SplashActivity }

Open logcat panel for emulator Pixel_3a_API_34_extension_level_7_x86_64
Connected to process 1486 on device 'Pixel_3a_API_34_extension_level_7_x86_64 [emulator-5554]'.
```

7. Interfaces de usuario. (1).

Estos son los dispositivos que he usado para probar la aplicación.

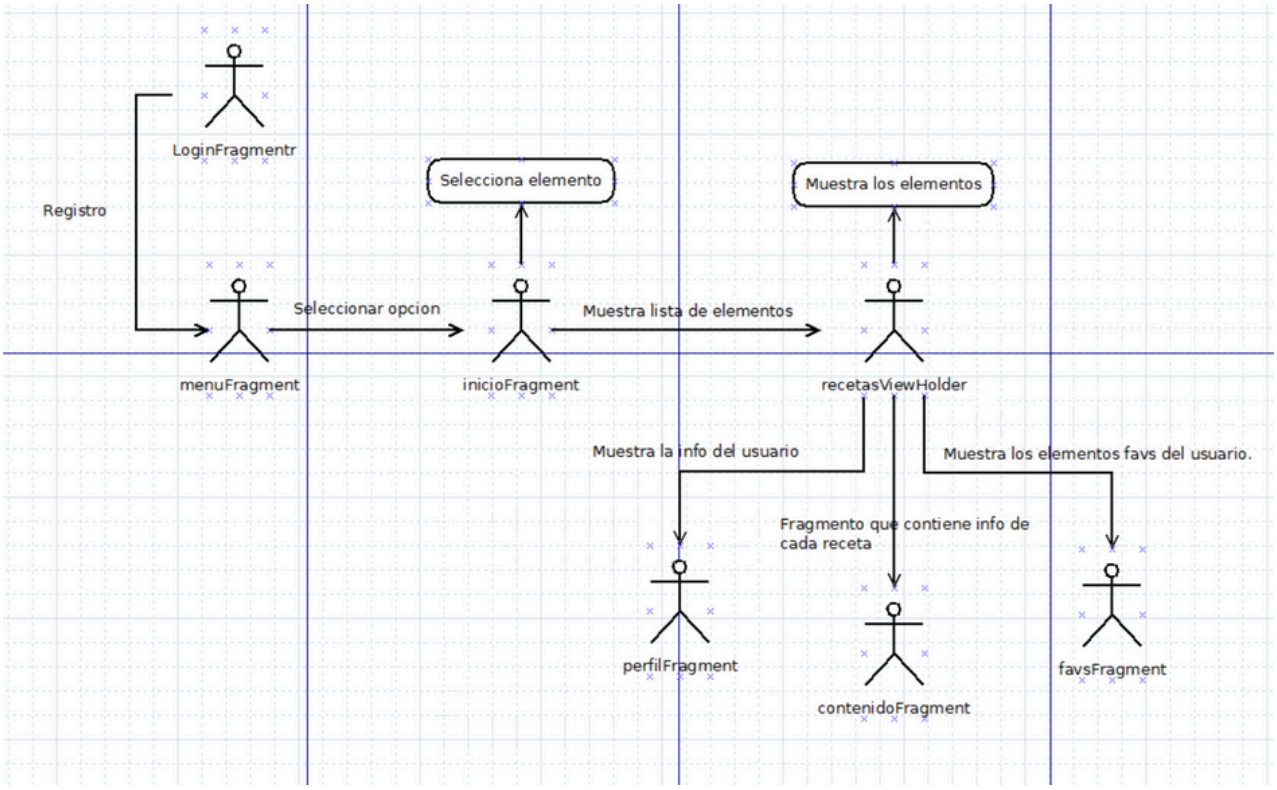




8. Diagrama de clases de la aplicación. (1).

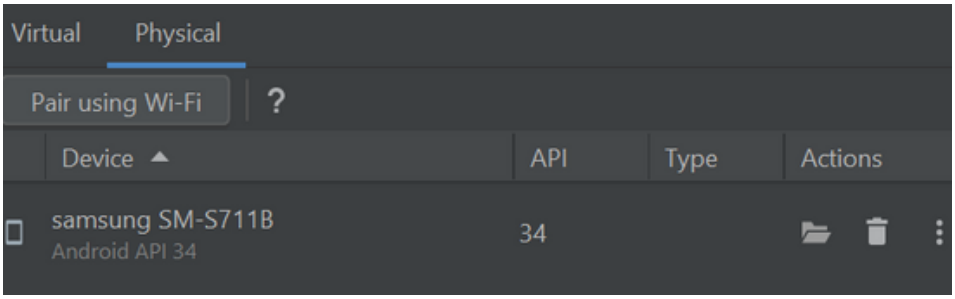
El diagrama de clase se encuentra dentro del directorio raíz del proyecto.

9. Diagrama de casos de usos (1).



10. Configuración de un dispositivo móvil para instalar la aplicación creada. (1).

Tenemos que activar las opciones de desarrollador, una vez activada, activaremos la depuración USB, conectaremos nuestro móvil al PC, aceptaremos y daremos permiso a la huella digital y una vez hecho esto, Android Studio detectará y vinculará nuestro móvil para poder ejecutar la aplicación desde él, instalándola previamente. La prueba de funcionamiento lo he hecho desde mi propio dispositivo móvil.



11. Pruebas de uso de la aplicación en el móvil, mostrando su comportamiento en ambos formatos: vertical y apaisado. (1).

En el directorio raíz del proyecto se encuentra el vídeo.