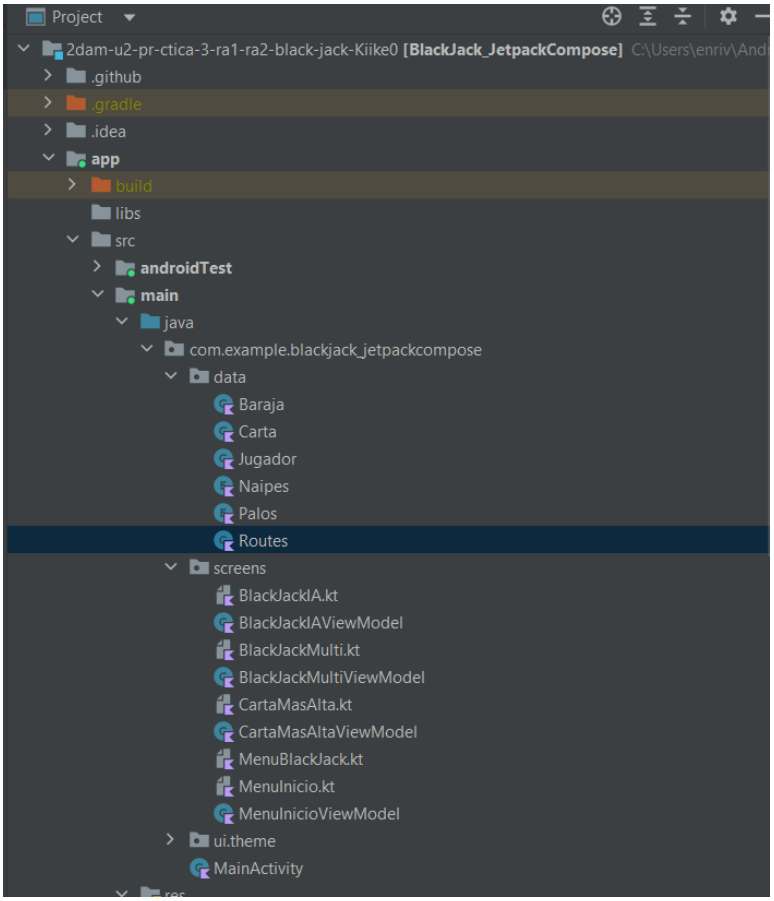
Enrique Villegas Reyes

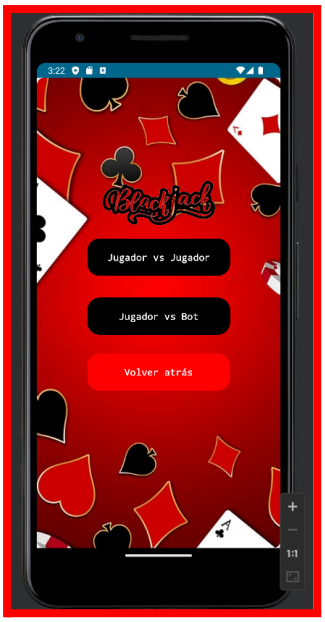
# Estructura del proyecto BlackJack


El proyecto está dividido en varias clases con una orientación a objetos que ayudan al funcionamiento del programa. La clase Baraja, la data class Carta, Jugador, Naipes, Palos y la clase Routes. También hemos añadido paquetes a la práctica. En el otro paquete que no hemos mencionado, “screens”, están los ViewModels donde damos comportamiento al programa utilizando métodos que se usarán dentro del mismo ViewModel o desde el Jetpack Compose, que en este caso es una clase de archivos kotlin donde hemos creado todos los métodos composable de la práctica.




En la clase baraja hemos creado un companion object donde metemos los métodos que se utilizan en las respectivas clases de ViewModel de cada juego. Así no tenemos que crear los objetos en dichas clases. La utilizamos en varias ocasiones:

- Creación de la baraja
- Barajar las cartas
- Para visualizar la puntuación de las cartas
- Para visualizar las imágenes
- Para pedir carta




Cuando abrimos la aplicación nos encontramos el menú principal, donde elegimos entre las opciones si vamos al juego de la carta más alta o elegimos el BlackJack. Esta última, nos llevará al menú BlackJack donde decidir si jugar contra otro jugador o contra la máquina. También hemos implementado un botón de salir por si queremos cerrar la aplicación en el menú de inicio y un botón para volver atrás en el menú del BlackJack.

- La estructura de las rutas es importante para navegar entre las pantallas de dicha aplicación. En la práctica está todo lo que se utiliza explicado con comentarios.

```
/**
* Clase sellada que representa las diferentes pantallas de la
aplicación.
*
* @property route El identificador de la ruta
*/
sealed class Routes(val route: String) {
    /**
     * Objeto que representa la pantalla del Menu Inicial.
     */
    object GameScreen : Routes("GameScreen")

    /**
     * Objeto que representa la pantalla del juego del
BlackJack.
     */
    object MenuBlackJackScreen : Routes("menuBlackJackScreen")

    /**
     * Objeto que representa la pantalla del juego de la Carta
más alta.
     */
    object CartaMasAltaScreen : Routes("cartaMasAltaScreen")
```

```
    /**
     * Objeto que representa la pantalla del Multijugador.
     */
    object MultiScreen : Routes("multiScreen")

    /**
     * Objeto que representa la pantalla de Jugador vs IA.
     */
    object BotScreen : Routes("botScreen")
}
```

Hemos añadido una ventana de configuración con un Dialog para que los jugadores puedan elegir sus nicks en el juego. Es muy intuitiva, ya que las imágenes te muestran a qué jugador pertenece y no es necesario una explicación con detalles en la pantalla para que ocupe espació e información innecesaria.



Se finaliza el juego cuando los dos jugadores se plantan o ambos han sobrepasado el límite de 21. En este caso se dispara una ventana de diálogo en la que podemos ver las puntuaciones que han conseguido cada jugador o reiniciar el juego reseteando todas sus características.

- Podemos acceder a la puntuación del ganador con el método getGanador() de la viewModel correspondiente. Se ha añadido además, que si llega a 21 la puntuación de uno de los jugadores justo le sale un mensaje de que ha conseguido un blackjack. Este diálogo funciona de tal forma que si ha conseguido el blackjack la cadena es más grande, por ello el tamaño de letra sea más pequeño.

```
fun getGanador(): String {
    try {
        return when {
            _jugador1.value!!.puntos == 21 &&
_jugador2.value!!.puntos != 21 -> {
                // Caso: Si Jugador 1 ha conseguido 21: Blackjack
                _conseguidoBlackJack.value = true
                _puntuacionJ1.value = "${_jugador1.value!!.puntos}"
                _puntuacionJ2.value = "${_jugador2.value!!.puntos}"
                "¡Ha ganado ${_jugador1.value!!.nick} con un
Blackjack!"
            }
            _jugador2.value!!.puntos == 21 &&
_jugador1.value!!.puntos != 21 -> {
                // Caso: Si Jugador 2 ha conseguido 21: Blackjack
                _conseguidoBlackJack.value = true
                _puntuacionJ1.value = "${_jugador1.value!!.puntos}"
                _puntuacionJ2.value = "${_jugador2.value!!.puntos}"
                "¡Ha ganado ${_jugador2.value!!.nick} con un
Blackjack!"
            }
            _jugador1.value!!.puntos > 21 &&
_jugador2.value!!.puntos <= 21 -> {
                // Caso: Jugador 1 se ha pasado de 21
                _puntuacionJ1.value = "${_jugador1.value!!.puntos}"
                _puntuacionJ2.value = "${_jugador2.value!!.puntos}"
                "¡Ha ganado ${_jugador2.value!!.nick}!"
            }
            _jugador1.value!!.puntos <= 21 &&
_jugador2.value!!.puntos > 21 -> {
                // Caso: Jugador 2 se ha pasado de 21
                _puntuacionJ1.value = "${_jugador1.value!!.puntos}"
                _puntuacionJ2.value = "${_jugador2.value!!.puntos}"
                "¡Ha ganado ${_jugador1.value!!.nick}!"
            }
            _jugador1.value!!.puntos < 21 &&
_jugador2.value!!.puntos < 21 -> {
                // Caso: Si ambos jugadores se han plantado antes
de 21
                when {
```

```
                    _jugador1.value!!.puntos >
_jugador2.value!!.puntos -> {
                        _puntuacionJ1.value =
"${_jugador1.value!!.puntos}"
                        _puntuacionJ2.value =
"${_jugador2.value!!.puntos}"
                        "¡Ha ganado ${_jugador1.value!!.nick}!"
                    }
                    _jugador2.value!!.puntos >
_jugador1.value!!.puntos -> {
                        _puntuacionJ1.value =
"${_jugador1.value!!.puntos}"
                        _puntuacionJ2.value =
"${_jugador2.value!!.puntos}"
                        "¡Ha ganado ${_jugador2.value!!.nick}!"
                    }
                    else -> {
                        _puntuacionJ1.value =
"${_jugador1.value!!.puntos}"
                        _puntuacionJ2.value =
"${_jugador2.value!!.puntos}"
                        "¡Empate!"
                    }
                }
            }
            else -> {
                _puntuacionJ1.value = "${_jugador1.value!!.puntos}"
                _puntuacionJ2.value = "${_jugador2.value!!.puntos}"
                "¡Empate!"
            }
        }
    } catch (e: Exception) {
        return "Error al determinar al ganador"
    }
}
```

- Otra cosa que tiene la aplicación es que cuando el turno cambia se tapan las cartas boca abajo del jugador que no juega, y además desaparece su nick y su puntuación para que no lo vea el otro. En el caso de la IA no aparecen sus cartas, pero pueden verse una vez vemos las puntuaciones al terminar el juego.


Una vez finalizado el juego podemos ver las puntuaciones si pulsamos el botón puntuaciones para ver las que ha conseguido cada jugador. Esto se consigue con un AlertDialog.

*Datos a tener en cuenta*: La IA funciona, pero no correctamente, cuando pido carta la IA sigue pidiendo carta, no se planta por una posible solución para ella, sigue pidiendo hasta que me planto, que en ese caso pide una carta más y se planta, pero eso pasa siempre. Le he puesto condicionales pero no me funcionan, no me observa bien el estado de las puntuaciones de la máquina y por contra sigue pidiendo. No sé por qué. Lo voy a dejar así, ya que la IA era opcional, pero por lo menos para que vea que lo he intentado. No me ha dado tiempo de seguir investigando.