Actividad Principal (MainActivity)

La MainActivity es la actividad principal de la aplicación. Nos proporciona una UI con varios botones que permiten al usuario realizar acciones específicas.

Botones y Acciones

- **Llamada:** Al hacer clic en este botón, se inicia una actividad de llamada.
- **Alarma:** Muestra un mensaje de tostada y configura una alarma para sonar en 2 minutos.
- **Navegador:** Abre una página web predefinida en el navegador.
- **Música:** Abre la aplicación de Spotify para reproducir una musica.
- **Cartas:** Navega a una actividad que probablemente esté relacionada con cartas.
- **Chistes:** Navega a una actividad que nos dice 10 chistes.

```
class MainActivity : AppCompatActivity(), OnClickListener {

    private lateinit var botonLlamada: ImageButton
    private lateinit var botonAlarma: ImageButton
    private lateinit var botonNavegador: ImageButton
    private lateinit var botonMusica: ImageButton
    private lateinit var botonCartas : ImageButton
    private lateinit var botonChistes : ImageButton

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        botonLlamada = findViewById(R.id.botonLlamada)
        botonAlarma = findViewById(R.id.botonAlarma)
        botonNavegador = findViewById(R.id.botonNavegador)
        botonMusica = findViewById(R.id.botonMusica)
        botonCartas = findViewById(R.id.botonCartas)
        botonChistes = findViewById(R.id.botonChistes)

        initEvent()
    }

    private fun initEvent() {
        botonLlamada.setOnClickListener {
            val intent = Intent(this, CallActivity::class.java)
            startActivity(intent)
        }

        botonAlarma.setOnClickListener {
            Toast.makeText(this@MainActivity, "Pulsado boton Alarma",
Toast.LENGTH_SHORT).show()
            crearAlarma()
        }

        botonNavegador.setOnClickListener {
            Log.d("MainActivity", "Clic en el botón de URL")
```

```
            val url = "https://www.google.es"
            openWebPage(url)
        }
        botonMusica.setOnClickListener {
            abrirSpotify("spotify:track:6nK2pIKFcRc5frrZKHgsiT");
        }
        botonCartas.setOnClickListener {
            val intent = Intent(this, CartasActivity::class.java)
            startActivity(intent)
        }
        botonChistes.setOnClickListener {
            val intent = Intent(this, ChistesActivity::class.java)
            startActivity(intent)
        }

    }
    private fun abrirSpotify(url: String) {
        val spotifyDeepLink = Uri.parse(url)

        //
        val uri = Uri.parse(spotifyDeepLink.toString())
        val intent = Intent(Intent.ACTION_VIEW, uri)
        if (intent.resolveActivity(packageManager) != null) {
            startActivity(intent)
        } else {
            Log.d("MainActivity", "Spotify no está instalado en el dispositivo.")
        }
    }
    private fun crearAlarma() {
        val alarma = Calendar.getInstance()
        alarma.add(Calendar.MINUTE, 2) // Add 2 minutes to the current time

        val intent = Intent(AlarmClock.ACTION_SET_ALARM)
        intent.putExtra(AlarmClock.EXTRA_MESSAGE, "Alarma en 2 minutos")
        intent.putExtra(AlarmClock.EXTRA_HOUR, alarma.get(Calendar.HOUR_OF_DAY))
        intent.putExtra(AlarmClock.EXTRA_MINUTES, alarma.get(Calendar.MINUTE))

        if (intent.resolveActivity(packageManager) != null) {
            startActivity(intent)
        } else {
            Toast.makeText(this, "No se puede crear la alarma",
Toast.LENGTH_SHORT).show()
        }
    }


    private fun openWebPage(url: String) {
        val webpage = Uri.parse(url)
        val intent = Intent(Intent.ACTION_VIEW, webpage)
        if (intent.resolveActivity(packageManager) != null) {
            startActivity(intent)
        }
    }
```

```
    override fun onClick(dialog: DialogInterface?, which: Int) {
        TODO("Not yet implemented")
    }
}
```

Actividad Login (LoginActivity)

La clase LoginActivity representa la pantalla de inicio de sesión de la aplicación.

Para validar las credenciales, se obtienen los valores ya registrados en InputUser y InputPass. Luego, se verifica si coinciden con las constantes de usuario y contraseña predefinidas (MYUSER y MYPASS). Si las credenciales son correctas, se inicia la actividad principal (MainActivity) y se pasan las credenciales como extras en el intent. De lo contrario, se muestra un mensaje de error.

```
class LoginActivity : AppCompatActivity() {

    private val MYUSER = "user"
    private val MYPASS = "1234"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.login_activity)

        val usernameEditText: EditText = findViewById(R.id.usernameEditText)
        val passwordEditText: EditText = findViewById(R.id.passwordEditText)
        val validarButton: Button = findViewById(R.id.validarButton)

        validarButton.setOnClickListener {
            val inputUser = usernameEditText.text.toString()
            val inputPass = passwordEditText.text.toString()

            if (inputUser == MYUSER && inputPass == MYPASS) {
                val intent = Intent(this@LoginActivity, MainActivity::class.java)
                intent.putExtra("USERNAME", inputUser)
                intent.putExtra("PASSWORD", inputPass)
                startActivity(intent)
            } else {
                Toast.makeText(this@LoginActivity, "Credenciales incorrectas",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Actividad Cartas (CartasActivity)

La clase `CartasActivity` en la aplicación gestiona el juego de cartas. Utiliza un temporizador para simular el barajeo de cartas y calcular la suma de sus valores.

Lógica del Juego

El método game() inicia la lógica del juego llamando a sheduleRun(), que utiliza un temporizador para simular el lanzamiento de cartas.

Lanzamiento de Cartas

throwDadoInTime() simula el lanzamiento de cartas y actualiza las imágenes en intervalos específicos.

Método selectView

El método selectView se encarga de seleccionar la imagen de una carta y asignarla a las cartas.

Resultado del Juego

viewResult() calcula la suma de los valores de las cartas y actualiza la interfaz.

```
class CartasActivity : AppCompatActivity() {
    private lateinit var bindingMain: ActivityCartasBinding
    private var sum: Int = 0
    private lateinit var handler: Handler
    private lateinit var cardValues: IntArray

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        bindingMain = ActivityCartasBinding.inflate(layoutInflater)
        setContentView(bindingMain.root)
        handler = Handler(Looper.getMainLooper())
        initEvent()
        bindingMain.btnBack.setOnClickListener {
            finish()
        }
    }

    private fun initEvent() {
        bindingMain.txtResultado.visibility = View.INVISIBLE
        bindingMain.btnBarajear.setOnClickListener {
            bindingMain.txtResultado.visibility = View.VISIBLE
            sum = 0 // Reinicia la suma al hacer clic en el botón
            game()
        }
    }

    private fun game() {
        sheduleRun()
    }

    private fun sheduleRun() {
        val schedulerExecutor = Executors.newSingleThreadScheduledExecutor()
        val msc = 1000

        for (i in 1..5) {
```

```
            schedulerExecutor.schedule({
                handler.post {
                    throwDadoInTime()
                }
            }, msc * i.toLong(), TimeUnit.MILLISECONDS)
        }

        schedulerExecutor.schedule({
            handler.post {
                viewResult()
            }
        }, msc * 7.toLong(), TimeUnit.MILLISECONDS)

        schedulerExecutor.shutdown()
    }

    private fun throwDadoInTime() {
        val numDados = IntArray(3) { Random.nextInt(1, 6) }
        cardValues = numDados.clone()

        val imagViews: Array<ImageView> = arrayOf(
            bindingMain.imagviewCard1,
            bindingMain.imagviewCard2,
            bindingMain.imagviewCard3
        )

        for (i in 0 until 3) {
            handler.post {
                selectView(imagViews[i], numDados[i])
            }
        }
    }

    private fun selectView(imgV: ImageView, v: Int) {
        when (v) {
            1 -> imgV.setImageResource(R.drawable.carta1)
            2 -> imgV.setImageResource(R.drawable.carta2)
            3 -> imgV.setImageResource(R.drawable.carta3)
            4 -> imgV.setImageResource(R.drawable.carta4)
            5 -> imgV.setImageResource(R.drawable.carta5)
            6 -> imgV.setImageResource(R.drawable.carta6)
        }
    }

    private fun viewResult() {
        sum += cardValues.sum()
        bindingMain.txtResultado.text = sum.toString()
        println("Cartas: ${cardValues.joinToString()} Suma: $sum")
    }
}
```

Actividad Chistes (ChistesActivity)

La clase ChistesActivity en la aplicación gestiona la presentación de chistes, utilizando TextToSpeech para la síntesis de voz y una lógica especial para manejar toques simples y dobles.

Método TextToSpeech

La configuración del sistema TextToSpeech se realiza en configureTextToSpeech. Se inicializa y se establece el idioma predeterminado del dispositivo.

Método initHander

Se inicia un hilo que simula una carga inicial y presenta tres botones después de cierto tiempo.

Método initEvent

Se manejan toques simples y dobles, ejecutando diferentes acciones según la lógica definida en initEvent.

Método speakMeDescription

La síntesis de voz se realiza en speakMeDescription, donde se utiliza el sistema TextToSpeech para hablar una cadena proporcionada.

Metodo onDestroy

En onDestroy, se detiene y cierra el sistema TextToSpeech para liberar recursos cuando la actividad se muere.

```
class ChistesActivity : AppCompatActivity() {

    private lateinit var binding: ActivityChistesBinding
    private lateinit var textToSpeech: TextToSpeech
    private val TOUCH_MAX_TIME = 500
    private var touchLastTime: Long = 0
    private lateinit var handler: Handler
    val MYTAG = "LOGCAT"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityChistesBinding.inflate(layoutInflater)
        setContentView(binding.root)
        configureTextToSpeech()
        initHander()
        initEvent()

        // Configura el listener para el botón de retroceso
        binding.btnBack.setOnClickListener {
            finish() // Cierra la actividad actual y retrocede a la anterior
        }
    }

    private fun initHander() {
        handler = Handler(Looper.getMainLooper())
        binding.progressBar.visibility = View.VISIBLE
        binding.btnExample.visibility = View.GONE
```

```
        Thread {
            Thread.sleep(3000)
            handler.post {
                binding.progressBar.visibility = View.GONE
                val description = getString(R.string.describe).toString()
                Thread.sleep(4000)
                Log.i(MYTAG, "Se ejecuta correctamente el hilo")
                binding.btnExample.visibility = View.VISIBLE
                speakMeDescription(description)
            }
        }.start()
    }

    private fun configureTextToSpeech() {
        textToSpeech = TextToSpeech(applicationContext,
TextToSpeech.OnInitListener {
            if (it != TextToSpeech.ERROR) {
                textToSpeech.language = Locale.getDefault()
                Log.i(MYTAG, "Sin problemas en la configuración TextToSpeech")
            } else {
                Log.i(MYTAG, "Error en la configuración TextToS 6ymn  /h")
            }
        })
    }

    private fun initEvent() {
        binding.btnExample.setOnClickListener {
            val currentTime = System.currentTimeMillis()
            if (currentTime - touchLastTime < TOUCH_MAX_TIME) {
                // Doble toque
                executorDoubleTouch()
            } else {
                // Toque simple
                speakRandomJoke()
            }

            touchLastTime = currentTime
        }
    }

    private fun speakRandomJoke() {
        val randomJoke = ChistesManager.chiste.random()
        speakMeDescription(randomJoke)
    }

    private fun executorDoubleTouch() {
        val chiste = resources.getString(R.string.chiste)
        speakMeDescription(chiste)
    }

    private fun speakMeDescription(s: String) {
        textToSpeech.speak(s, TextToSpeech.QUEUE_FLUSH, null, null)
    }
```

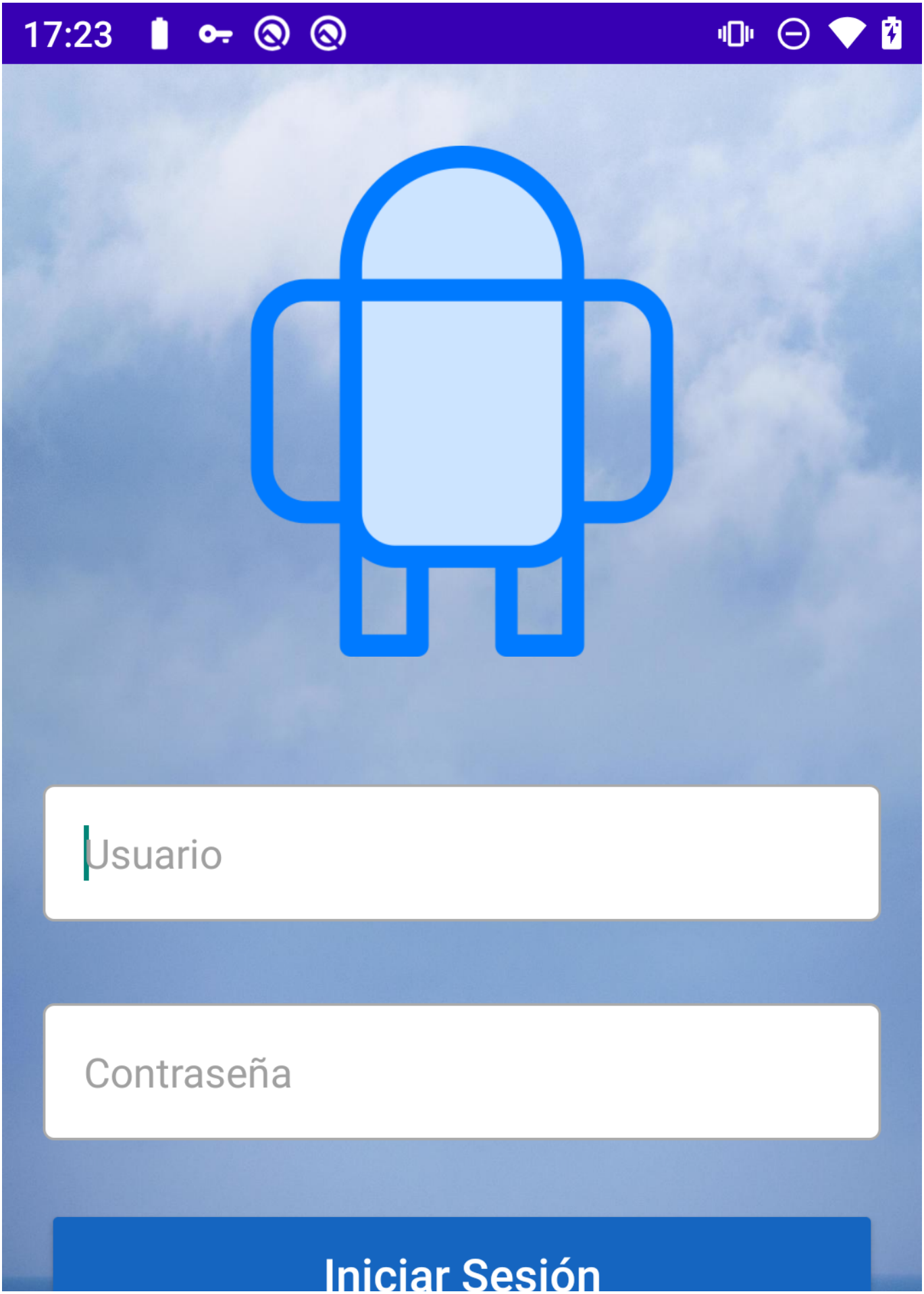
```
    override fun onDestroy() {
        if (::textToSpeech.isInitialized) {
            textToSpeech.stop()
            textToSpeech.shutdown()
        }
        super.onDestroy()
    }
}
```

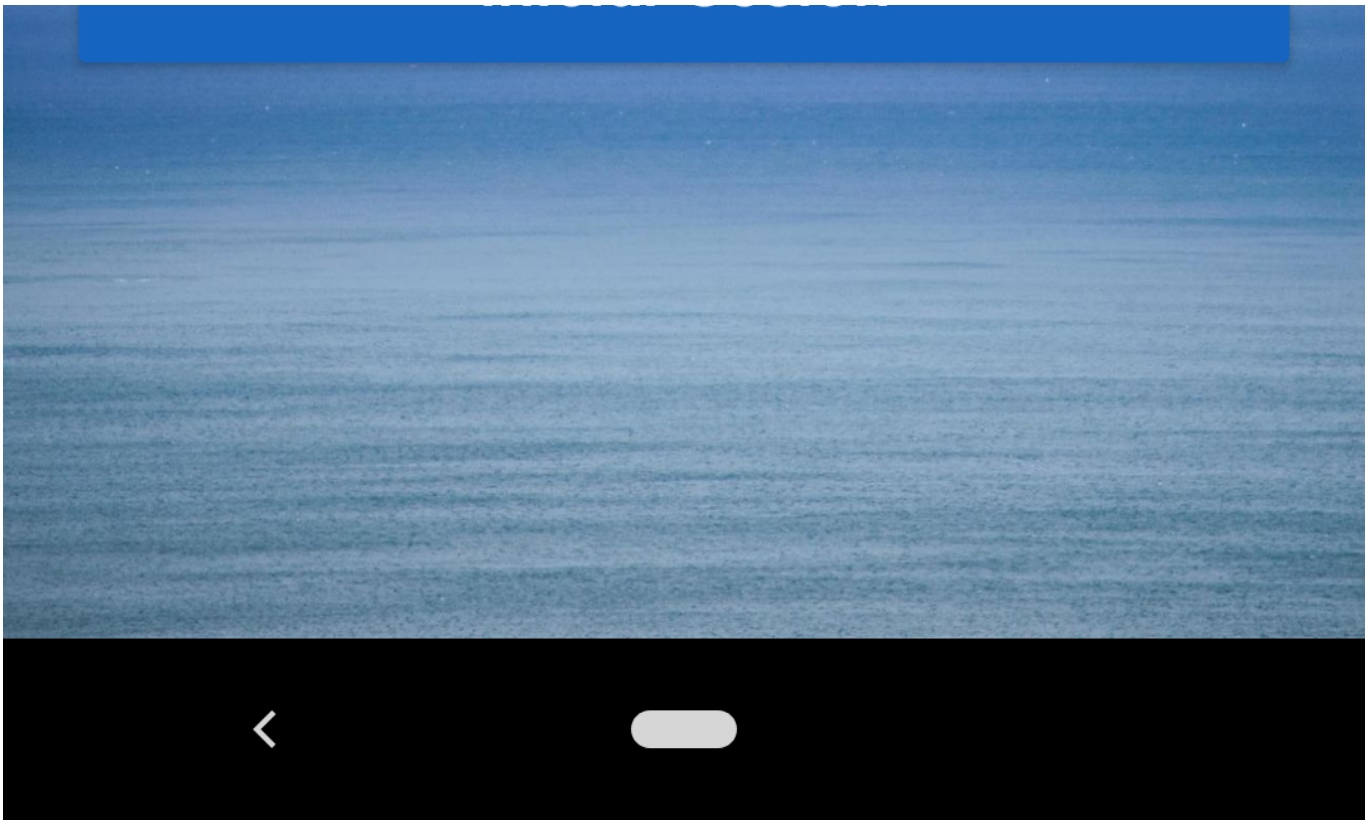
ChistesManager

La clase ChistesManager es un objeto singleton que proporciona una lista de chistes. Cada chiste es representado como una cadena de texto. Hay tres categorias de chistes.

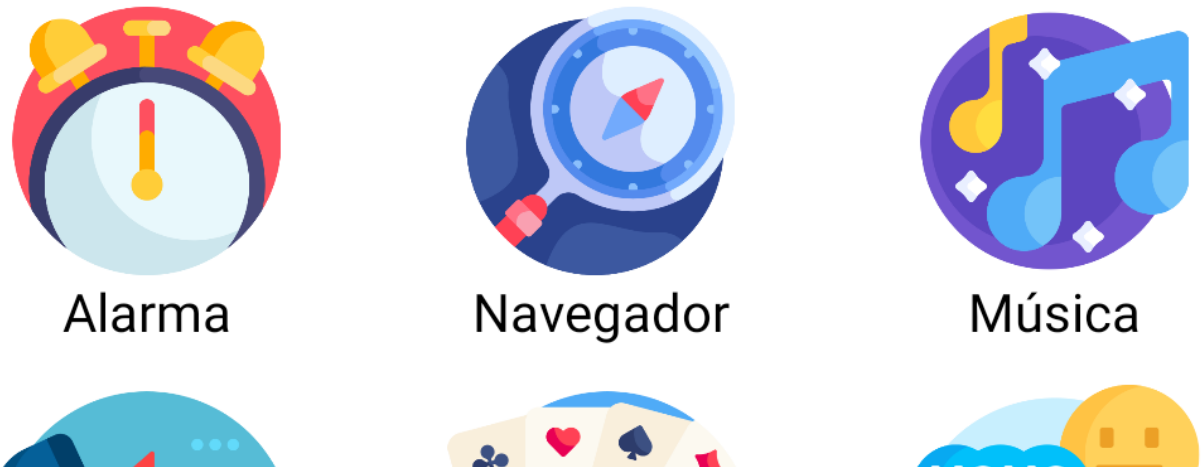
```
object ChistesManager {

    val chistesAnimales = listOf(
        "¿Qué hace una abeja en el gimnasio? ¡Zum-ba!",
        "¿Cuál es el animal más antiguo? La cebra, ¡porque está en blanco y
negro!",
        "¿Cómo llama un pez a otro pez? ¡Hola pez!",
        "¿Por qué los pájaros no usan Facebook? Porque ya tienen Twitter.",
        "¿Qué hace una rata cuando suena el despertador? ¡Ratata!",
        "¿Cuál es el colmo de los astronautas? Tomar la leche directo de la Vía
Láctea."
    )

    val chistesCiencia = listOf(
        "¿Cuál es el colmo de un electricista? No encontrar su corriente de
trabajo.",
        "Un neutrino entra a un bar. El camarero le dice: 'Lo siento, aquí no
servimos partículas que se mueven más rápido que la luz'. El neutrino responde:
'¡Pero si soy ligero!'",
        "¿Qué es una hipotenusa con acento? Una hipotenusá.",
        "¿Por qué los programadores prefieren el invierno? Porque la nieve es
blanca y el texto en blanco es invisible.",
        "¿Cómo organizan una fiesta los biólogos moleculares? Con mucha celula-
rización.",
        "¿Cuál es el animal más peligroso del mundo? La bacteria, porque es la
única que es resistente a los antibióticos."
    )

    val chistesDeportes = listOf(
        "¿Cuál es el deporte favorito de las ardillas? El esquías.",
        "¿Cuál es el colmo de un jardinero? Tener malas hierbas.",
        "¿Por qué el libro de matemáticas está estresado? Porque tiene demasiados
problemas.",
        "¿Qué hace una abeja en el gimnasio? ¡Zum-ba!",
        "¿Cómo se llama el campeón de buceo japonés? Tokofondo.",
        "¿Cuál es el colmo de un panadero? No encontrar su panadero."
```

```
    )
}
```


17:23
Usuario
Contraseña
Iniciar Sesión



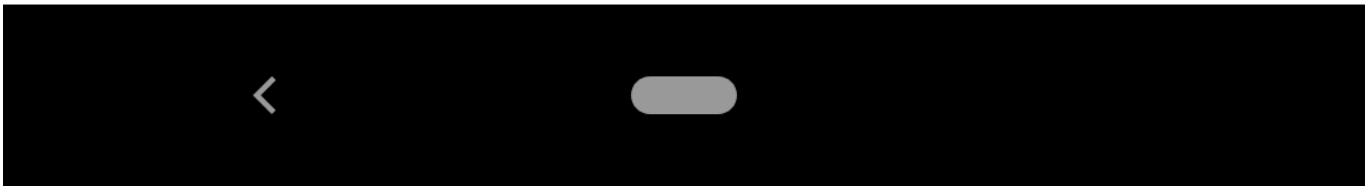

17:23
APLICACIÓN MANUEL
PABLO MOLINA


Alarma
Navegador
Música

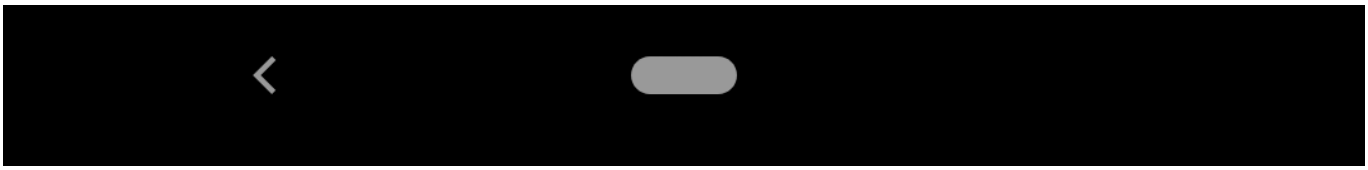
Llamada

Cartas

Chistes



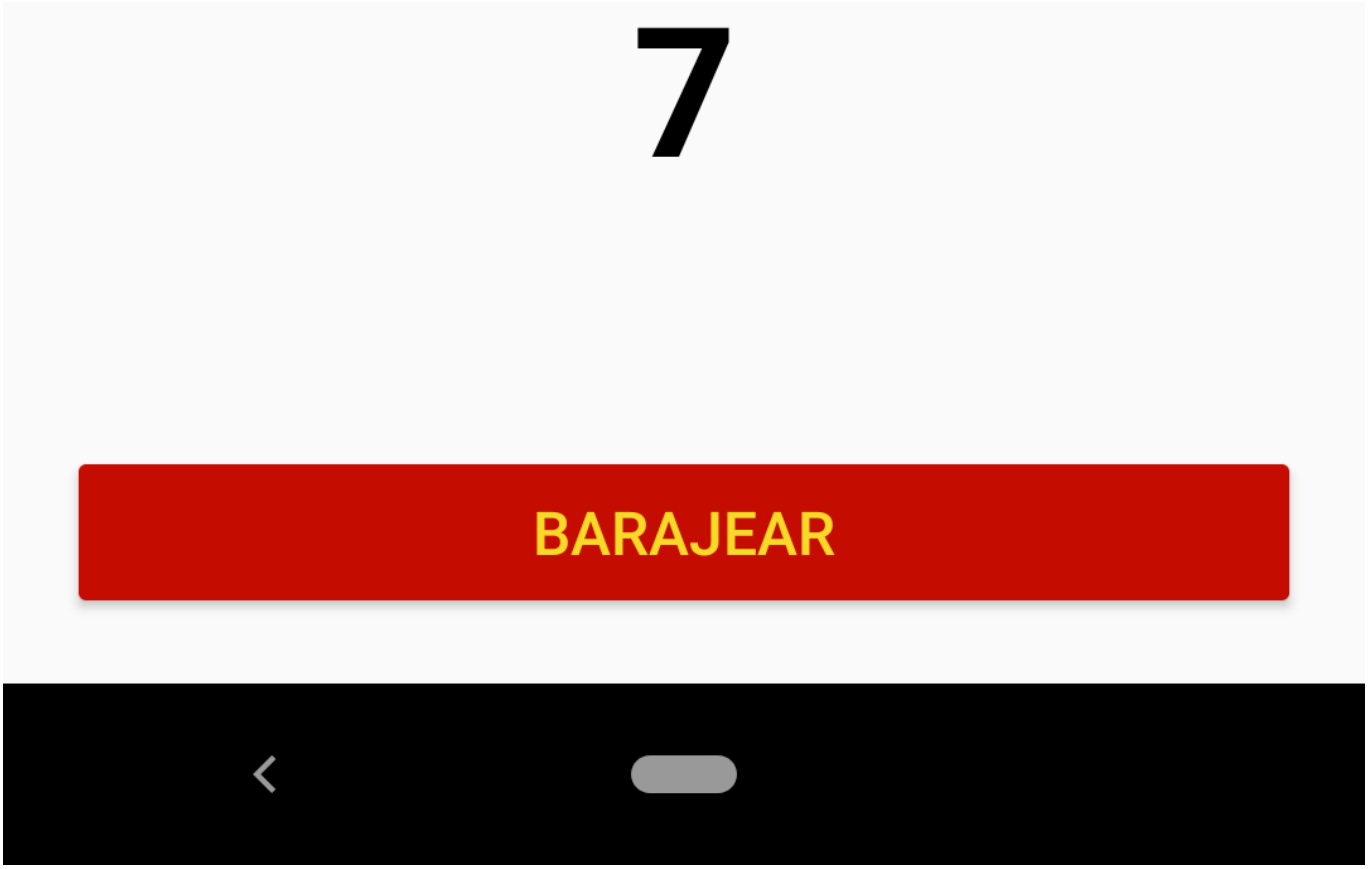
BARAJEAR




17:24


UNO


3
3
3
2
2
2
2
2
2


7
BARAJEAR


18:17
Selecciona una categoría


18:17
Selecciona una categoría

Ciencia

Animales

Deportes