# UNIVERSIDAD TECNOLÓGICA DE LEÓN

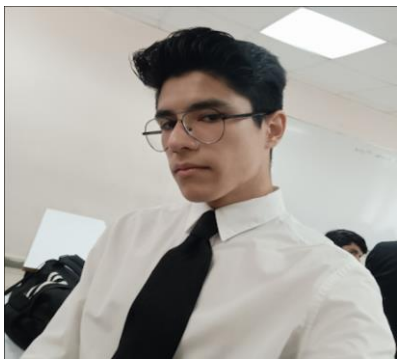
**INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE**

Desarrollo para Dispositivos Inteligentes

Examen parcial 2

presenta:

Gutierrez Ascencio Luis Angel Tadeo



IDGS901

Fecha: 02/07/2025

```
package org.utl.examenparcialdos

import android.widget.Toast
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import org.utl.examenparcialdos.Pregunta
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ExamenScreen(navController: NavController, documentId: String) {
    val questions = remember {
        listOf(
            Pregunta("¿Cual es 7*7?", listOf("77", "48", "49", "17"), 2),
            Pregunta("¿Capital de Francia?", listOf("Tlaxcala", "Madrid", "Paris", "Francia"), 2),
            Pregunta("¿Quien le gana a Goku?", listOf("Nadie", "Vegeta", "Halo Verde",
"Linux"), 0),
            Pregunta("¿Numero de continentes?", listOf("8", "5", "6", "7"), 3),
            Pregunta("¿Instrumento musical de cuerda?", listOf("Trompeta", "Violin", "Flauta",
"Bateria"), 1),
            Pregunta("¿Animal mas raro segun la ciencia?", listOf("Perezoso", "Ajolote",
"ornitorrinco", "Armadillo"), 2)
        )
    }

    val selectedOptions = remember { mutableStateMapOf<Int, Int?>() }
    val context = LocalContext.current
    val scrollState = rememberScrollState()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
```

```
        .verticalScroll(scrollState),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
  ) {
    Text(text = "Examen de Conocimientos", style =
MaterialTheme.typography.headlineMedium)
    Spacer(modifier = Modifier.height(24.dp))

    questions.forEachIndexed { qIndex, question ->
      Text(text = "${qIndex + 1}.- ${question.questionText}", style =
MaterialTheme.typography.titleMedium)
      Spacer(modifier = Modifier.height(8.dp))
      question.options.forEachIndexed { oIndex, option ->
        Row(
          verticalAlignment = Alignment.CenterVertically,
          modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp)
        ) {
          RadioButton(
            selected = selectedOptions[qIndex] == oIndex,
            onClick = { selectedOptions[qIndex] = oIndex }
          )
          Text(text = option)
        }
      }
      Spacer(modifier = Modifier.height(16.dp))
    }

    Spacer(modifier = Modifier.height(24.dp))

    Button(onClick = {
      var score = 0
      var allAnswered = true
      for (i in questions.indices) {
        if (selectedOptions[i] == null) {
          allAnswered = false
          break
        }
        if (selectedOptions[i] == questions[i].correctAnswerIndex) {
```

```
                score++
            }
        }

        if (allAnswered) {
            navController.navigate("quiz_result/$score/$documentId")
        } else {
            Toast.makeText(context, "Responde todas las preguntas >:v .",
Toast.LENGTH_SHORT).show()
        }
    }) {
        Text("Terminar Examen")
    }
  }
}
```

```
package org.utl.examenparcialdos

import android.util.Log
import com.google.firebase.firestore.FirebaseFirestore
import org.utl.examenparcialdos.Usuario

class FirestoreManager {

    private val db = FirebaseFirestore.getInstance()
    private val usersCollection = db.collection("Usuarios")

    fun saveUserData(userData: Usuario, onSuccess: (String) -> Unit, onFailure: (Exception)
-> Unit) {
        usersCollection.add(userData)
            .addOnSuccessListener { documentReference ->
                Log.d("FirestoreManager", "DocumentSnapshot added with ID:
${documentReference.id}")
                onSuccess(documentReference.id)
            }
            .addOnFailureListener { e ->
                Log.w("FirestoreManager", "Error adding document", e)
                onFailure(e)
            }
    }

    fun updateUserData(documentId: String, updates: Map<String, Any>, onSuccess: () ->
Unit, onFailure: (Exception) -> Unit) {
        usersCollection.document(documentId)
            .update(updates)
            .addOnSuccessListener {
                Log.d("FirestoreManager", "DocumentSnapshot successfully updated!")
                onSuccess()
            }
            .addOnFailureListener { e ->
                Log.w("FirestoreManager", "Error updating document", e)
                onFailure(e)
            }
    }
}
```

```
package org.utl.examenparcialdos

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import org.utl.examenparcialdos.ui.theme.ExamenParcialDosTheme
import org.utl.examenparcialdos.ExamenScreen
import org.utl.examenparcialdos.PersonalDataFormScreen
import org.utl.examenparcialdos.QuizResultadoScreen
import org.utl.examenparcialdos.ZodiacoResuladotScreen

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ExamenParcialDosTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    AppNavigation()
                }
            }
        }
    }
}

@Composable
fun AppNavigation() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "personal_data_form") {
        composable("personal_data_form") {
```

```
            PersonalDataFormScreen(navController = navController)
        }

composable("zodiac_result/{name}/{paternal}/{maternal}/{day}/{month}/{year}/{documentId}") { backStackEntry ->
            val name = backStackEntry.arguments?.getString("name") ?: ""
            val paternal = backStackEntry.arguments?.getString("paternal") ?: ""
            val maternal = backStackEntry.arguments?.getString("maternal") ?: ""
            val day = backStackEntry.arguments?.getString("day")?.toIntOrNull() ?: 1
            val month = backStackEntry.arguments?.getString("month")?.toIntOrNull() ?: 1
            val year = backStackEntry.arguments?.getString("year")?.toIntOrNull() ?: 2000
            val documentId = backStackEntry.arguments?.getString("documentId") ?: ""

            ZodiacoResuladotScreen(
                navController = navController,
                name = name,
                paternal = paternal,
                maternal = maternal,
                day = day,
                month = month,
                year = year,
                documentId = documentId
            )
        }
        composable("exam/{documentId}") { backStackEntry ->
            val documentId = backStackEntry.arguments?.getString("documentId") ?: ""
            ExamenScreen(navController = navController, documentId = documentId)
        }
        composable("quiz_result/{score}/{documentId}") { backStackEntry ->
            val score = backStackEntry.arguments?.getString("score")?.toIntOrNull() ?: 0
            val documentId = backStackEntry.arguments?.getString("documentId") ?: ""
            QuizResultadoScreen(navController = navController, score = score, documentId = documentId)
        }
    }
}
```

```
package org.utl.examenparcialdos

import android.widget.Toast
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import org.utl.examenparcialdos.Usuario
import org.utl.examenparcialdos.FirestoreManager
import com.google.firebase.Timestamp

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun PersonalDataFormScreen(navController: NavController) {
    var name by remember { mutableStateOf("") }
    var paternal by remember { mutableStateOf("") }
    var maternal by remember { mutableStateOf("") }
    var day by remember { mutableStateOf("") }
    var month by remember { mutableStateOf("") }
    var year by remember { mutableStateOf("") }
    var selectedSex by remember { mutableStateOf("") }
    val context = LocalContext.current
    val firestoreManager = remember { FirestoreManager() }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(text = "Datos Personales", style = MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(24.dp))
```

```
        OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            label = { Text("Nombre") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = paternal,
            onValueChange = { paternal = it },
            label = { Text("Apaterno") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        OutlinedTextField(
            value = maternal,
            onValueChange = { maternal = it },
            label = { Text("Amaterno") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))

        Text(text = "Fecha de nacimiento", style = MaterialTheme.typography.titleMedium)
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceAround,
            verticalAlignment = Alignment.CenterVertically
        ) {
            OutlinedTextField(
                value = day,
                onValueChange = { if (it.length <= 2) day = it },
                label = { Text("Dia") },
                keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
                modifier = Modifier.weight(1f)
            )
            Spacer(modifier = Modifier.width(8.dp))
            OutlinedTextField(
                value = month,
                onValueChange = { if (it.length <= 2) month = it },
```

```
            label = { Text("Mes") },
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
            modifier = Modifier.weight(1f)
        )
        Spacer(modifier = Modifier.width(8.dp))
        OutlinedTextField(
            value = year,
            onValueChange = { if (it.length <= 4) year = it },
            label = { Text("Año") },
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
            modifier = Modifier.weight(1f)
        )
    }
    Spacer(modifier = Modifier.height(16.dp))

    Text(text = "SEXO", style = MaterialTheme.typography.titleMedium)
    Row(
        modifier = Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.CenterVertically
    ) {
        RadioButton(
            selected = selectedSex == "Masculino",
            onClick = { selectedSex = "Masculino" }
        )
        Text("Masculino")
        Spacer(modifier = Modifier.width(16.dp))
        RadioButton(
            selected = selectedSex == "Femenino",
            onClick = { selectedSex = "Femenino" }
        )
        Text("Femenino")
    }
    Spacer(modifier = Modifier.height(24.dp))

    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.SpaceAround
    ) {
```

```
        Button(onClick = {
            name = ""
            paternal = ""
            maternal = ""
            day = ""
            month = ""
            year = ""
            selectedSex = ""
        }) {
            Text("Limpiar")
        }
        Button(onClick = {
            if (name.isNotBlank() && paternal.isNotBlank() && maternal.isNotBlank() &&
                day.isNotBlank() && month.isNotBlank() && year.isNotBlank() &&
                selectedSex.isNotBlank()
            ) {
                val userData = Usuario(
                    nombre = name,
                    apaterno = paternal,
                    amaterno = maternal,
                    dia = day.toInt(),
                    mes = month.toInt(),
                    anio = year.toInt(),
                    sexo = selectedSex,
                    puntuacion = 0.0,
                    horoscopo = "",
                    timestamp = Timestamp.now()
                )

                firestoreManager.saveUserData(
                    userData = userData,
                    onSuccess = { documentId ->
                        Toast.makeText(context, "Datos guardados",
Toast.LENGTH_SHORT).show()

navController.navigate("zodiac_result/$name/$paternal/$maternal/$day/$month/$year/$doc
umentId")
                    },
                    onFailure = { e ->
                        Toast.makeText(context, "Error al guardar: ${e.message}",
```

```
Toast.LENGTH_LONG).show()
                    }
                )

            } else {
                Toast.makeText(context, "Complete todos los campos",
Toast.LENGTH_SHORT).show()
            }
        }) {
            Text("Siguiente")
        }
    }
}
```

```
package org.utl.examenparcialdos

data class Pregunta (
    val questionText: String,
    val options: List<String>,
    val correctAnswerIndex: Int
)
```

```
package org.utl.examenparcialdos

import com.google.firebase.Timestamp

data class Usuario (
    val nombre: String = "",
    val apaterno: String = "",
    val amaterno: String = "",
    val dia: Int = 0,
    val mes: Int = 0,
    val anio: Int = 0,
    val sexo: String = "",
    var puntuacion: Double = 0.0,
    var horoscopo: String = "",
    val timestamp: Timestamp = Timestamp.now()
)
```

```
package org.utl.examenparcialdos

import android.widget.Toast
import androidx.compose.foundation.layout.*
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import org.utl.examenparcialdos.FirestoreManager

@Composable
fun QuizResultadoScreen(navController: NavController, score: Int, documentId: String) {
    val context = LocalContext.current
    val firestoreManager = remember { FirestoreManager() }

    val totalQuestions = 6
    val calificacionFinal = (score.toDouble() / totalQuestions) * 10

    LaunchedEffect(key1 = documentId, key2 = calificacionFinal) {
        if (documentId.isNotBlank()) {
            val updates = mapOf("puntuacion" to calificacionFinal)
            firestoreManager.updateUserData(
                documentId = documentId,
                updates = updates,
                onSuccess = {
                    Toast.makeText(context, "Calificacion final guardada",
Toast.LENGTH_SHORT).show()
                },
                onFailure = { e ->
                    Toast.makeText(context, "Error al guardar calificacion final: ${e.message}",
Toast.LENGTH_LONG).show()
                }
            )
```

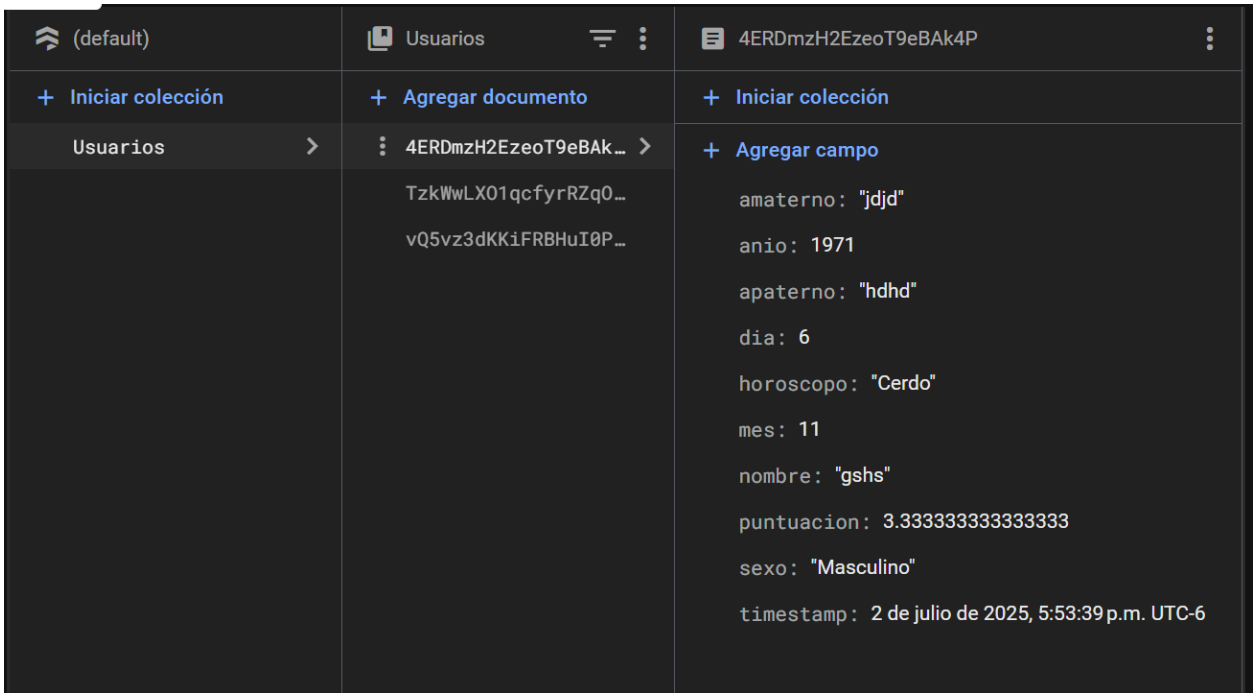
```
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(text = "Resultados del Cuestionario", style =
MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(24.dp))

        Text(text = "Respuestas bien: $score de $totalQuestions", style =
MaterialTheme.typography.titleLarge)
        Spacer(modifier = Modifier.height(8.dp))

        val calificacionFormateada = String.format("%.2f", calificacionFinal)
        Text(text = "Calificacion: $calificacionFormateada", style =
MaterialTheme.typography.titleLarge)
        Spacer(modifier = Modifier.height(32.dp))

        Button(onClick = {
            navController.navigate("personal_data_form") {
                popUpTo("personal_data_form") {
                    inclusive = true
                }
            }
        }) {
            Text("Hacer Otro Registro")
        }
    }
}
```

```
package org.utl.examenparcialdos

import java.util.Calendar


object ZodiacoCalculadora {
    fun calculateAge(day: Int, month: Int, year: Int): Int {
        val dob = Calendar.getInstance().apply {
            set(year, month - 1, day) // El mes es de índice 0
        }
        val today = Calendar.getInstance()

        var age = today.get(Calendar.YEAR) - dob.get(Calendar.YEAR)
        if (today.get(Calendar.DAY_OF_YEAR) < dob.get(Calendar.DAY_OF_YEAR)) {
            age--
        }
        return age
    }

    fun getChineseZodiac(year: Int): Pair<String, String> {
        val animals = arrayOf(
            "Rata", "Buey", "Tigre", "Conejo", "Dragón", "Serpiente",
            "Caballo", "Cabra", "Mono", "Gallo", "Perro", "Cerdo"
        )
        val imageResourceNames = arrayOf(
            "rata", "buey", "tigre", "conejo", "dragon", "serpiente",
            "caballo", "cabra", "mono", "gallo", "perro", "cerdo"
        )
        val index = (year - 1900) % 12
        return Pair(animals.getOrNull(index) ?: "", imageResourceNames.getOrNull(index) ?:
"")
    }
}
```

```
package org.utl.examenparcialdos

import android.widget.Toast
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import org.utl.examenparcialdos.ZodiacoCalculadora
import org.utl.examenparcialdos.R
import org.utl.examenparcialdos.FirestoreManager

@Composable
fun ZodiacoResuladotScreen(
    navController: NavController,
    name: String,
    paternal: String,
    maternal: String,
    day: Int,
    month: Int,
    year: Int,
    documentId: String
) {
    val fullName = "$name $paternal $maternal"
    val age = ZodiacoCalculadora.calculateAge(day, month, year)
    val (zodiacAnimal, zodiacImageName) = ZodiacoCalculadora.getChineseZodiac(year)
    val context = LocalContext.current
    val firestoreManager = remember { FirestoreManager() }

    val imageResourceId =
androidx.compose.ui.platform.LocalContext.current.resources.getIdentifier(
```

```
        zodiacImageName,
        "drawable",
        androidx.compose.ui.platform.LocalContext.current.packageName
    )

    LaunchedEffect(key1 = documentId, key2 = zodiacAnimal) {
        if (documentId.isNotBlank() && zodiacAnimal.isNotBlank()) {
            val updates = mapOf("horoscopo" to zodiacAnimal)
            firestoreManager.updateUserData(
                documentId = documentId,
                updates = updates,
                onSuccess = {
                    Toast.makeText(context, "Horoscopo guardado",
Toast.LENGTH_SHORT).show()
                },
                onFailure = { e ->
                    Toast.makeText(context, "Error al guardar horoscopo: ${e.message}",
Toast.LENGTH_LONG).show()
                }
            )
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(text = "Hola $fullName", style = MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Tienes $age años y tu signo zodiacal", style =
MaterialTheme.typography.titleLarge)
        Spacer(modifier = Modifier.height(16.dp))

        if (imageResourceId != 0) {
            Image(
                painter = painterResource(id = imageResourceId),
                contentDescription = zodiacAnimal,
```

```
            modifier = Modifier.size(120.dp)
        )
    } else {
        Text(text = "Imagen del signo zodiacal no encontrada", style =
MaterialTheme.typography.bodyMedium)
    }

    Spacer(modifier = Modifier.height(8.dp))
    Text(text = "Es $zodiacAnimal", style = MaterialTheme.typography.titleLarge)

    Spacer(modifier = Modifier.height(32.dp))

    Button(onClick = {
        navController.navigate("exam/$documentId")
    }) {
        Text("Hacer Encuesta")
    }
  }
}
```


(default)
Usuarios
4ERDmzH2EzeoT9eBAk4P
+ Iniciar colección
+ Agregar documento
+ Iniciar colección
Usuarios
4ERDmzH2EzeoT9eBAk…
TzkWwLXO1qcfyrRZqO…
vQ5vz3dKKiFRBHuI0P…
+ Agregar campo
amaterno: "jdjd"
anio: 1971
apaterno: "hdhd"
dia: 6
horoscopo: "Cerdo"
mes: 11
nombre: "gshs"
puntuacion: 3.333333333333333
sexo: "Masculino"
timestamp: 2 de julio de 2025, 5:53:39 p.m. UTC-6


5:58
93
Datos Personales
Nombre
Apaterno
Amaterno
Fecha de nacimiento
Dia
Mes
Año
SEXO
Masculino
Femenino
Limpiar
Siguiente

5:58 TEMU 93

# Hola sjs hdhd hdhd

## Tienes 25 años y tu signo zodiacal

Es Dragón

Hacer Encuesta

Horoscopo guardado


5:58
93
Resultados del Cuestionario
Respuestas bien: 3 de 6
Calificacion: 5.00
Hacer Otro Registro
Calificacion final guardada