DatosUsuario.kt

```
package com.example.examencardiel


object DatosUsuario {
    var nombre = ""
    var apellidos = ""
    var dia = 1
    var mes = 1
    var anio = 2000
    var sexo = ""
    var calificacion = 0
}
```

MainActivity.kt

```
package com.example.examencardiel

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.runtime.*
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.google.firebase.FirebaseApp

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Inicializa Firebase con el contexto de la aplicación
        FirebaseApp.initializeApp(this)
        setContent {
            // Crea un controlador de navegación para manejar las
pantallas
            val navController = rememberNavController()
            // Configura el sistema de navegación entre pantallas
            NavHost(navController = navController, startDestination =
"formulario") {
                composable("formulario") {
                    PantallaFormulario(navController)
                }
                composable("examen") {
                    PantallaExamen(navController)
                }
                composable("resultado") {
                    PantallaResultado(navController)
                }
            }
        }
    }
}
```

PantallaExamen.kt

```
package com.example.examencardiel

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController

// Clase que representa una pregunta con su texto, opciones y la
respuesta correcta (por índice)
data class Pregunta(val texto: String, val opciones: List<String>, val
respuestaCorrecta: Int)

// Composable principal para mostrar el examen
@Composable
fun PantallaExamen(navController: NavController) {

    // Lista de preguntas con sus opciones y respuestas correctas
    val preguntas = listOf(
        Pregunta("¿Cuánto es 2 + 2?", listOf("8", "6", "4", "3"), 2),
        Pregunta("Capital de México", listOf("Monterrey", "CDMX",
"Puebla", "Cancún"), 1),
        Pregunta("Días de una semana", listOf("10", "6", "5", "7"), 3),
        Pregunta("Color de rojo + azul", listOf("Verde", "Morado",
"Naranja", "Café"), 1),
        Pregunta("Planeta más cercano al Sol", listOf("Marte", "Venus",
"Tierra", "Mercurio"), 3),
        Pregunta("¿Cuántas patas tiene una araña?", listOf("6", "8", "4",
"10"), 1)
    )

    // Lista mutable que almacena las respuestas del usuario (-1
significa sin contestar)
    val respuestasUsuario = remember { mutableStateListOf(-1, -1, -1, -1,
-1, -1) }

    // Estado para permitir hacer scroll en la columna
    val scrollState = rememberScrollState()

    // Contenedor principal con scroll vertical y padding
    Column(
        modifier = Modifier
            .padding(16.dp)
            .verticalScroll(scrollState)
    ) {
        // Se muestra cada pregunta con sus opciones
        preguntas.forEachIndexed { i, pregunta ->
            // Título de la pregunta
```

```
            Text("${i + 1}. ${pregunta.texto}", style =
MaterialTheme.typography.bodyLarge)

            // Opciones con RadioButton
            pregunta.opciones.forEachIndexed { j, opcion ->
                Row {
                    RadioButton(
                        selected = respuestasUsuario[i] == j, // Marca si
está seleccionada
                        onClick = { respuestasUsuario[i] = j } // Cambia
la selección
                    )
                    Text(opcion)
                }
            }

            // Espacio entre preguntas
            Spacer(Modifier.height(8.dp))
        }

        // Botón para terminar el examen
        Button(onClick = {
            val puntosPorPregunta = 10.0 / preguntas.size // Ponderación
de cada pregunta
            var calificacion = 0.0

            // Verifica cuántas respuestas son correctas
            respuestasUsuario.forEachIndexed { i, r ->
                if (r == preguntas[i].respuestaCorrecta) {
                    calificacion += puntosPorPregunta
                }
            }
            // Asegura que la calificación esté entre 0 y 10
            calificacion = calificacion.coerceIn(0.0, 10.0)
            // Guarda la calificación en una variable global
            DatosUsuario.calificacion = calificacion.toInt()
            navController.navigate("resultado")
        }) {
            Text("Terminar")
        }
    }
}
```

PantallaFormulario.kt

```
package com.example.examencardiel


import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.*
import androidx.compose.ui.unit.dp
```

```
import androidx.navigation.NavController
// Composable que representa el formulario de captura de datos personales
@Composable
fun PantallaFormulario(navController: NavController) {

    // Estados para almacenar lo que el usuario escribe en el formulario
    var nombre by remember { mutableStateOf("") }
    var apellidos by remember { mutableStateOf("") }
    var dia by remember { mutableStateOf("") }
    var mes by remember { mutableStateOf("") }
    var anio by remember { mutableStateOf("") }
    var sexo by remember { mutableStateOf("Masculino") }

    // Estructura principal con espacio alrededor
    Column(modifier = Modifier.padding(16.dp)) {

        // Título del formulario
        Text("Formulario", style =
MaterialTheme.typography.headlineMedium)

        // Campos de texto para capturar datos del usuario
        OutlinedTextField(value = nombre, onValueChange = { nombre = it
}, label = { Text("Nombre") })
        OutlinedTextField(value = apellidos, onValueChange = { apellidos
= it }, label = { Text("Apellidos") })
        OutlinedTextField(value = dia, onValueChange = { dia = it },
label = { Text("Día de nacimiento") })
        OutlinedTextField(value = mes, onValueChange = { mes = it },
label = { Text("Mes") })
        OutlinedTextField(value = anio, onValueChange = { anio = it },
label = { Text("Año") })

        // Selección de sexo usando botones de radio
        Row {
            RadioButton(selected = sexo == "Masculino", onClick = { sexo
= "Masculino" })
            Text("Masculino")
            RadioButton(selected = sexo == "Femenino", onClick = { sexo =
"Femenino" })
            Text("Femenino")
        }

        // Botones para limpiar o continuar el formulario
        Row(Modifier.padding(top = 16.dp)) {

            // Botón para limpiar todos los campos
            Button(onClick = {
                nombre = ""
                apellidos = ""
                dia = ""
                mes = ""
                anio = ""
            }) { Text("Limpiar") }
```

```
            Spacer(Modifier.width(16.dp))

            // Botón para guardar los datos y pasar a la pantalla de
examen
            Button(onClick = {
                // Guarda los datos en una clase compartida
                DatosUsuario.nombre = nombre
                DatosUsuario.apellidos = apellidos
                DatosUsuario.dia = dia.toIntOrNull() ?: 1
                DatosUsuario.mes = mes.toIntOrNull() ?: 1
                DatosUsuario.anio = anio.toIntOrNull() ?: 2000
                DatosUsuario.sexo = sexo

                // Navega a la pantalla del examen
                navController.navigate("examen")
            }) { Text("Siguiente") }
        }
    }
}
```

PantallaResultado.kt

```
package com.example.examencardiel

import android.content.Context
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.google.firebase.firestore.FirebaseFirestore
import java.util.Calendar

@Composable
fun PantallaResultado(navController: NavController) {
    // Obtiene el año actual del sistema
    val currentYear = Calendar.getInstance().get(Calendar.YEAR)

    // Calcula la edad del usuario en base a su año de nacimiento
    val edad = currentYear - DatosUsuario.anio

    // Determina el signo zodiacal chino en función de su fecha de
nacimiento
```

```
    val signo = calcularSignoZodiacoChinoPorFecha(
        DatosUsuario.anio, DatosUsuario.mes, DatosUsuario.dia
    )

    // Obtiene el contexto actual de la aplicación (necesario para
guardar archivos, recursos, etc.)
    val context = LocalContext.current

    // Efecto que se lanza una vez cuando se carga la pantalla
    LaunchedEffect(true) {
        // Construye un texto con los datos del usuario
        val contenido = buildString {
            append("Nombre: ${DatosUsuario.nombre}
${DatosUsuario.apellidos}\n")
            append("Edad: $edad\n")
            append("Signo: $signo\n")
            append("Calificación: ${DatosUsuario.calificacion}\n")
        }

        // Guarda los datos en un archivo local llamado
"datos_usuario.txt"
        context.openFileOutput("datos_usuario.txt",
Context.MODE_PRIVATE).use {
            it.write(contenido.toByteArray())
        }

        // Prepara el acceso a Firebase Firestore
        val db = FirebaseFirestore.getInstance()

        // Crea un mapa con los datos del usuario
        val datos = hashMapOf(
            "nombre" to DatosUsuario.nombre,
            "apellidos" to DatosUsuario.apellidos,
            "edad" to edad,
            "signo" to signo,
            "calificacion" to DatosUsuario.calificacion
        )

        // Intenta guardar el documento en la colección "usuarios" en
Firebase
        db.collection("usuarios")
            .add(datos)
            .addOnSuccessListener { /* Guardado exitoso */ }
            .addOnFailureListener { /* Error al guardar */ }
    }

    // Contenedor principal de la pantalla
    Box(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        contentAlignment = Alignment.Center
    ) {
        // Tarjeta que contiene los resultados
```

```
        Card(
            shape = RoundedCornerShape(16.dp),
            colors = CardDefaults.cardColors(containerColor =
Color(0xFFEFF2F5)),
            modifier = Modifier
                .fillMaxWidth()
                .padding(8.dp),
            elevation = CardDefaults.cardElevation(defaultElevation =
6.dp)
        ) {
            // Columna que organiza los elementos verticalmente
            Column(
                modifier = Modifier
                    .padding(24.dp)
                    .fillMaxWidth(),
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
                // Título principal
                Text(
                    text = "🎉 Resultado Final 🎉",
                    style = MaterialTheme.typography.headlineMedium,
                    fontWeight = FontWeight.Bold,
                    color = Color(0xFF2C3E50)
                )

                Spacer(modifier = Modifier.height(20.dp))

                // Saludo con nombre completo
                Text(
                    text = "Hola ${DatosUsuario.nombre}
${DatosUsuario.apellidos}",
                    fontSize = 20.sp,
                    fontWeight = FontWeight.SemiBold
                )

                Spacer(modifier = Modifier.height(8.dp))

                // Muestra la edad calculada
                Text(
                    text = "Tienes $edad años",
                    fontSize = 18.sp,
                    color = Color(0xFF34495E)
                )

                // Muestra el signo zodiacal
                Text(
                    text = "Tu signo zodiacal chino es:",
                    fontSize = 18.sp,
                    color = Color(0xFF34495E)
                )

                // Texto del signo
                Text(
                    text = signo,
```

```
                    fontSize = 20.sp,
                    fontWeight = FontWeight.Bold,
                    color = Color(0xFF2980B9)
                )

                Spacer(modifier = Modifier.height(16.dp))

                // Carga una imagen desde drawable con el nombre del
signo (si existe)
                val imagenId = context.resources.getIdentifier(
                    signo.lowercase(), "drawable", context.packageName
                )

                if (imagenId != 0) {
                    Image(
                        painter = painterResource(id = imagenId),
                        contentDescription = "Imagen del signo $signo",
                        modifier = Modifier
                            .size(140.dp)
                            .clip(RoundedCornerShape(12.dp))
                            .background(Color.White)
                    )
                }

                Spacer(modifier = Modifier.height(20.dp))

                // Muestra la calificación obtenida
                Text(
                    text = "Calificación obtenida:
${DatosUsuario.calificacion}/10",
                    fontSize = 18.sp,
                    fontWeight = FontWeight.Medium,
                    color = Color(0xFF27AE60)
                )
            }
        }
    }
}
fun calcularSignoZodiacoChinoPorFecha(anio: Int, mes: Int, dia: Int):
String {
    // Mapa con las fechas del Año Nuevo Chino por año (formato: año ->
(mes, día))
    val fechasAnoNuevo = mapOf(
        2000 to Pair(2, 5), 2001 to Pair(1, 24), 2002 to Pair(2, 12),
2003 to Pair(2, 1), 2004 to Pair(1, 22),
        2005 to Pair(2, 9), 2006 to Pair(1, 29), 2007 to Pair(2, 18),
2008 to Pair(2, 7), 2009 to Pair(1, 26),
        2010 to Pair(2, 14), 2011 to Pair(2, 3), 2012 to Pair(1, 23),
2013 to Pair(2, 10), 2014 to Pair(1, 31),
        2015 to Pair(2, 19), 2016 to Pair(2, 8), 2017 to Pair(1, 28),
2018 to Pair(2, 16), 2019 to Pair(2, 5),
        2020 to Pair(1, 25), 2021 to Pair(2, 12), 2022 to Pair(2, 1),
2023 to Pair(1, 22), 2024 to Pair(2, 10)
    )
```

```
    // Lista de los 12 signos del zodiaco chino, en orden cíclico
    val signos = listOf(
        "Mono", "Gallo", "Perro", "Cerdo", "Rata", "Buey",
        "Tigre", "Conejo", "Dragon", "Serpiente", "Caballo", "Cabra"
    )

    // Obtiene el mes y día del Año Nuevo Chino correspondiente al año
dado
    val (mesAnoNuevo, diaAnoNuevo) = fechasAnoNuevo[anio] ?: Pair(2, 4)
    // Verifica si la persona nació antes del Año Nuevo Chino
    val esAntesAnoNuevo = (mes < mesAnoNuevo) || (mes == mesAnoNuevo &&
dia < diaAnoNuevo)
    // Si nació antes, se considera el signo del año anterior
    val anioSigno = if (esAntesAnoNuevo) anio - 1 else anio
    // Devuelve el signo según el módulo 12 del año ajustado
    return signos[anioSigno % 12]
}
```