CÓDIGO FUENTE COMPLETO

1. build.gradle.kts (Module: app)

kotlin

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
    id("kotlin-parcelize")
}

android {
    namespace = "com.cortez.examen2parcial"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.cortez.examen2parcial"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
```

```
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
    }
}

dependencies {
    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")
    implementation("androidx.activity:activity-compose:1.8.2")
    implementation(platform("androidx.compose:compose-bom:2023.10.01"))
    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.ui:ui-graphics")
```

```
    implementation("androidx.compose.ui:ui-tooling-preview")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.navigation:navigation-compose:2.7.5")

    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
    androidTestImplementation(platform("androidx.compose:compose-bom:2023.10.01"))
    androidTestImplementation("androidx.compose.ui:ui-test-junit4")
    debugImplementation("androidx.compose.ui:ui-tooling")
    debugImplementation("androidx.compose.ui:ui-test-manifest")
}
```

2. MainActivity.kt

kotlin

```
// MainActivity.kt
package com.cortez.examen2parcial

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.ui.Modifier
```

```
import com.cortez.examen2parcial.navigation.ExamenNavigation
import com.cortez.examen2parcial.ui.theme.ExamenAppTheme

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

3. data/UserInfo.kt

kotlin

```
// data/UserInfo.kt
package com.cortez.examen2parcial.data

data class UserInfo(
  var nombre: String = "",
```

```
    var apellidoPaterno: String = "",
    var apellidoMaterno: String = "",
    var dia: String = "",
    var mes: String = "",
    var anio: String = "",
    var sexo: String = "Masculino"
)
```

4. data/Pregunta.kt

kotlin

```
// data/Pregunta.kt
package com.cortez.examen2parcial.data

data class Pregunta(
    val pregunta: String,
    val opciones: List<String>,
    val respuestaCorrecta: Int
)
```

5. data/ExamenData.kt

kotlin

```
// data/ExamenData.kt
package com.cortez.examen2parcial.data

object ExamenData {
    val preguntas = listOf(
```

```
Pregunta(
    "¿Cuánto es 2 + 2?",
    listOf("3", "4", "5", "6"),
    1
),
Pregunta(
    "¿Cuál es el color del sol?",
    listOf("Azul", "Amarillo", "Verde", "Rojo"),
    1
),
Pregunta(
    "¿Cuántos días tiene una semana?",
    listOf("5", "6", "7", "8"),
    2
),
Pregunta(
    "¿Cuál es la capital de México?",
    listOf("Guadalajara", "Ciudad de México", "Monterrey", "Cancún"),
    1
),
Pregunta(
    "¿Cuánto es 5 x 2?",
    listOf("8", "10", "12", "15"),
    1
),
Pregunta(
```

```
            "¿En qué estación del año hace más calor?",
            listOf("Primavera", "Verano", "Otoño", "Invierno"),
            1
        )
    )
}
```

6. utils/ZodiacalCalculator.kt

kotlin

```
// utils/ZodiacalCalculator.kt
package com.cortez.examen2parcial.utils

object ZodiacalCalculator {

    fun calcularSignoZodiacalChino(anio: Int): String {
        val signos = arrayOf(
            "Mono", "Gallo", "Perro", "Cerdo", "Rata", "Buey",
            "Tigre", "Conejo", "Dragón", "Serpiente", "Caballo", "Cabra"
        )
        return signos[anio % 12]
    }

    fun calcularEdad(anio: Int): Int {
        val anioActual = java.util.Calendar.getInstance().get(java.util.Calendar.YEAR)
        return anioActual - anio
    }
```

```
    fun getImagenSigno(signo: String): String {
        return when (signo.lowercase()) {
            "rata" -> "🐭"
            "buey" -> "🐂"
            "tigre" -> "🐅"
            "conejo" -> "🐇"
            "dragón" -> "🐉"
            "serpiente" -> "🐍"
            "caballo" -> "🐎"
            "cabra" -> "🐐"
            "mono" -> "🐒"
            "gallo" -> "🐓"
            "perro" -> "🐕"
            "cerdo" -> "🐖"
            else -> "🐎"
        }
    }
}
```

7. navigation/ExamenNavigation.kt

kotlin

// navigation/ExamenNavigation.kt

package com.cortez.examen2parcial.navigation

```
import androidx.compose.runtime.*
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.cortez.examen2parcial.data.UserInfo
import com.cortez.examen2parcial.screens.ExamenScreen
import com.cortez.examen2parcial.screens.FormularioScreen
import com.cortez.examen2parcial.screens.ResultadosScreen

@Composable
fun ExamenNavigation() {
    val navController = rememberNavController()
    var userInfo by remember { mutableStateOf(UserInfo()) }
    var respuestasUsuario by remember { mutableStateOf(listOf<Int>()) }

    NavHost(
        navController = navController,
        startDestination = "formulario"
    ) {
        composable("formulario") {
            FormularioScreen(
                navController = navController,
                userInfo = userInfo,
                onUserInfoChange = { newUserInfo ->
                    userInfo = newUserInfo
                }
```

```
            )
        }

        composable("examen") {
            ExamenScreen(
                navController = navController,
                onRespuestasComplete = { respuestas ->
                    respuestasUsuario = respuestas
                }
            )
        }

        composable("resultados") {
            ResultadosScreen(
                userInfo = userInfo,
                respuestasUsuario = respuestasUsuario,
                navController = navController
            )
        }
    }
}
```

8. screens/FormularioScreen.kt

kotlin

```
// screens/FormularioScreen.kt
package com.cortez.examen2parcial.screens
```

```
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Clear
import androidx.compose.material.icons.filled.Person
import androidx.compose.material.icons.filled.PlayArrow
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.cortez.examen2parcial.data.UserInfo
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun FormularioScreen(
    navController: NavController,
    userInfo: UserInfo,
    onUserInfoChange: (UserInfo) -> Unit
) {
    // Manejo el estado local de todos los campos del formulario
    var nombre by remember { mutableStateOf(userInfo.nombre) }
    var apellidoPaterno by remember { mutableStateOf(userInfo.apellidoPaterno) }
    var apellidoMaterno by remember { mutableStateOf(userInfo.apellidoMaterno) }
    var dia by remember { mutableStateOf(userInfo.dia) }
    var mes by remember { mutableStateOf(userInfo.mes) }
    var anio by remember { mutableStateOf(userInfo.anio) }
    var sexo by remember { mutableStateOf(userInfo.sexo) }

    // Valido que todos los campos estén llenos antes de permitir continuar
    val todosLosCamposLlenos = nombre.isNotEmpty() &&
                               apellidoPaterno.isNotEmpty() &&
                               apellidoMaterno.isNotEmpty() &&
                               dia.isNotEmpty() &&
                               mes.isNotEmpty() &&
                               anio.isNotEmpty()

    // Diseño con colores formales y profesionales
    Box(
```

```
    modifier = Modifier
        .fillMaxSize()
        .background(Color(0xFFF5F7FA)) // Gris claro profesional
) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .verticalScroll(rememberScrollState())
            .padding(24.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Spacer(modifier = Modifier.height(32.dp))

        // Header corporativo y formal
        Card(
            modifier = Modifier.fillMaxWidth(),
            elevation = CardDefaults.cardElevation(defaultElevation = 4.dp),
            colors = CardDefaults.cardColors(
                containerColor = Color.White
            ),
            shape = RoundedCornerShape(8.dp)
        ) {
            Column(
                modifier = Modifier.padding(32.dp),
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
```

```
// Icono corporativo
Icon(
    imageVector = Icons.Default.Person,
    contentDescription = "Formulario",
    modifier = Modifier.size(40.dp),
    tint = Color(0xFF1F2937)
)

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

Text(
    text = "FORMULARIO DE DATOS PERSONALES",
    fontSize = 20.sp,
    fontWeight = FontWeight.Bold,
    color = Color(0xFF1F2937),
    textAlign = TextAlign.Center,
    letterSpacing = 0.5.sp
)

Text(
    text = "Complete la información solicitada",
    fontSize = 14.sp,
    color = Color(0xFF6B7280),
    fontWeight = FontWeight.Normal,
    modifier = Modifier.padding(top = 8.dp)
)
```

```
    }
}

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

// Formulario principal con diseño corporativo
Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
) {
    Column(
        modifier = Modifier.padding(32.dp)
    ) {
        Text(
            text = "Información Personal",
            fontSize = 16.sp,
            fontWeight = FontWeight.SemiBold,
            color = Color(0xFF374151),
            modifier = Modifier.padding(bottom = 24.dp)
        )

        // Campos de texto con diseño formal
```

```
OutlinedTextField(
    value = nombre,
    onValueChange = { nombre = it },
    label = { Text("Nombre *", color = Color(0xFF6B7280)) },
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 6.dp),
    shape = RoundedCornerShape(6.dp),
    colors = OutlinedTextFieldDefaults.colors(
        focusedBorderColor = Color(0xFF374151),
        focusedLabelColor = Color(0xFF374151),
        unfocusedBorderColor = Color(0xFFD1D5DB)
    )
)

OutlinedTextField(
    value = apellidoPaterno,
    onValueChange = { apellidoPaterno = it },
    label = { Text("Apellido Paterno *", color = Color(0xFF6B7280)) },
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 6.dp),
    shape = RoundedCornerShape(6.dp),
    colors = OutlinedTextFieldDefaults.colors(
        focusedBorderColor = Color(0xFF374151),
        focusedLabelColor = Color(0xFF374151),
```

```
        unfocusedBorderColor = Color(0xFFD1D5DB)
    )
)

OutlinedTextField(
    value = apellidoMaterno,
    onValueChange = { apellidoMaterno = it },
    label = { Text("Apellido Materno *", color = Color(0xFF6B7280)) },
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 6.dp),
    shape = RoundedCornerShape(6.dp),
    colors = OutlinedTextFieldDefaults.colors(
        focusedBorderColor = Color(0xFF374151),
        focusedLabelColor = Color(0xFF374151),
        unfocusedBorderColor = Color(0xFFD1D5DB)
    )
)

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

// Sección para fecha de nacimiento
Text(
    text = "Fecha de Nacimiento",
    fontSize = 14.sp,
    fontWeight = FontWeight.SemiBold,
```

```
            color = Color(0xFF374151),
            modifier = Modifier.padding(bottom = 12.dp)
        )

        // Organizo los campos de fecha en una fila
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.spacedBy(12.dp)
        ) {
            // Campo día con validación numérica
            OutlinedTextField(
                value = dia,
                onValueChange = { if (it.length <= 2 && it.all { char -> char.isDigit() }) dia =
it },
                label = { Text("Día", color = Color(0xFF6B7280)) },
                keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
                modifier = Modifier.weight(1f),
                shape = RoundedCornerShape(6.dp),
                colors = OutlinedTextFieldDefaults.colors(
                    focusedBorderColor = Color(0xFF374151),
                    focusedLabelColor = Color(0xFF374151),
                    unfocusedBorderColor = Color(0xFFD1D5DB)
                )
            )

            // Campo mes con validación numérica
```

```
            OutlinedTextField(
              value = mes,
              onValueChange = { if (it.length <= 2 && it.all { char -> char.isDigit() }) mes
= it },
              label = { Text("Mes", color = Color(0xFF6B7280)) },
              keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
              modifier = Modifier.weight(1f),
              shape = RoundedCornerShape(6.dp),
              colors = OutlinedTextFieldDefaults.colors(
                focusedBorderColor = Color(0xFF374151),
                focusedLabelColor = Color(0xFF374151),
                unfocusedBorderColor = Color(0xFFD1D5DB)
              )
            )

            // Campo año con validación numérica
            OutlinedTextField(
              value = anio,
              onValueChange = { if (it.length <= 4 && it.all { char -> char.isDigit() }) anio
= it },
              label = { Text("Año", color = Color(0xFF6B7280)) },
              keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
              modifier = Modifier.weight(1f),
              shape = RoundedCornerShape(6.dp),
              colors = OutlinedTextFieldDefaults.colors(
```

```
            focusedBorderColor = Color(0xFF374151),
            focusedLabelColor = Color(0xFF374151),
            unfocusedBorderColor = Color(0xFFD1D5DB)
        )
    )
}

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

// Selección de sexo con diseño formal
Text(
    text = "Sexo",
    fontSize = 14.sp,
    fontWeight = FontWeight.SemiBold,
    color = Color(0xFF374151),
    modifier = Modifier.padding(bottom = 12.dp)
)

// Radio buttons con diseño corporativo
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.spacedBy(16.dp)
) {
    // Opción Masculino
    Card(
        modifier = Modifier
```

```
            .weight(1f)
            .selectable(
                selected = sexo == "Masculino",
                onClick = { sexo = "Masculino" }
            ),
        colors = CardDefaults.cardColors(
            containerColor = if (sexo == "Masculino")
                Color(0xFF374151).copy(alpha = 0.05f)
            else
                Color(0xFFFAFAFA)
        ),
        border = if (sexo == "Masculino")
            BorderStroke(2.dp, Color(0xFF374151))
        else
            BorderStroke(1.dp, Color(0xFFE5E7EB)),
        shape = RoundedCornerShape(6.dp),
        elevation = CardDefaults.cardElevation(defaultElevation = 0.dp)
    ) {
        Row(
            modifier = Modifier.padding(16.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            RadioButton(
                selected = sexo == "Masculino",
                onClick = { sexo = "Masculino" },
                colors = RadioButtonDefaults.colors(
```

```
            selectedColor = Color(0xFF374151),
            unselectedColor = Color(0xFF9CA3AF)
          )
        )
        Spacer(modifier = Modifier.width(8.dp))
        Text(
          text = "Masculino",
          fontWeight = FontWeight.Medium,
          color = Color(0xFF374151),
          fontSize = 14.sp
        )
      }
    }

    // Opción Femenino
    Card(
      modifier = Modifier
        .weight(1f)
        .selectable(
          selected = sexo == "Femenino",
          onClick = { sexo = "Femenino" }
        ),
      colors = CardDefaults.cardColors(
        containerColor = if (sexo == "Femenino")
          Color(0xFF374151).copy(alpha = 0.05f)
        else
```

```
            Color(0xFFFAFAFA)
        ),
        border = if (sexo == "Femenino")
            BorderStroke(2.dp, Color(0xFF374151))
        else
            BorderStroke(1.dp, Color(0xFFE5E7EB)),
        shape = RoundedCornerShape(6.dp),
        elevation = CardDefaults.cardElevation(defaultElevation = 0.dp)
    ) {
        Row(
            modifier = Modifier.padding(16.dp),
            verticalAlignment = Alignment.CenterVertically
        ) {
            RadioButton(
                selected = sexo == "Femenino",
                onClick = { sexo = "Femenino" },
                colors = RadioButtonDefaults.colors(
                    selectedColor = Color(0xFF374151),
                    unselectedColor = Color(0xFF9CA3AF)
                )
            )
            Spacer(modifier = Modifier.width(8.dp))
            Text(
                text = "Femenino",
                fontWeight = FontWeight.Medium,
                color = Color(0xFF374151),
```

```
                fontSize = 14.sp
            )
        }
    }
  }
}

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

// Botones de acción con diseño corporativo
Row(
  modifier = Modifier.fillMaxWidth(),
  horizontalArrangement = Arrangement.spacedBy(16.dp)
) {
  // Botón limpiar - resetea todos los campos
  OutlinedButton(
    onClick = {
      // Limpio todos los campos del formulario
      nombre = ""
      apellidoPaterno = ""
      apellidoMaterno = ""
      dia = ""
      mes = ""
      anio = ""
      sexo = "Masculino"
```

```
    },
    modifier = Modifier
        .weight(1f)
        .height(48.dp),
    shape = RoundedCornerShape(6.dp),
    border = BorderStroke(1.dp, Color(0xFF6B7280))
) {
    Icon(
        imageVector = Icons.Default.Clear,
        contentDescription = "Limpiar",
        modifier = Modifier.size(18.dp),
        tint = Color(0xFF6B7280)
    )
    Spacer(modifier = Modifier.width(8.dp))
    Text(
        text = "Limpiar",
        fontWeight = FontWeight.Medium,
        fontSize = 14.sp,
        color = Color(0xFF6B7280)
    )
}

// Botón siguiente - solo se habilita si todos los campos están llenos
Button(
    onClick = {
        // Creo el objeto UserInfo con toda la información y navego al examen
```

```
                val newUserInfo = UserInfo(
                    nombre = nombre,
                    apellidoPaterno = apellidoPaterno,
                    apellidoMaterno = apellidoMaterno,
                    dia = dia,
                    mes = mes,
                    anio = anio,
                    sexo = sexo
                )
                onUserInfoChange(newUserInfo)
                navController.navigate("examen")
            },
            enabled = todosLosCamposLlenos, // Solo habilito si todos los campos
están completos
            modifier = Modifier
                .weight(1f)
                .height(48.dp),
            shape = RoundedCornerShape(6.dp),
            colors = ButtonDefaults.buttonColors(
                containerColor = Color(0xFF1F2937),
                contentColor = Color.White,
                disabledContainerColor = Color(0xFFD1D5DB),
                disabledContentColor = Color(0xFF9CA3AF)
            )
        ) {
            Text(
```

```
                text = "Continuar",
                fontWeight = FontWeight.Medium,
                fontSize = 14.sp
            )
            Spacer(modifier = Modifier.width(8.dp))
            Icon(
                imageVector = Icons.Default.PlayArrow,
                contentDescription = "Siguiente",
                modifier = Modifier.size(18.dp)
            )
          }
        }

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

9. screens/ExamenScreen.kt

kotlin

```
// screens/ExamenScreen.kt
package com.cortez.examen2parcial.screens

import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
```

```
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material.icons.filled.ArrowForward
import androidx.compose.material.icons.filled.CheckCircle
import androidx.compose.material.icons.filled.Edit
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.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.cortez.examen2parcial.data.ExamenData

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ExamenScreen(
    navController: NavController,
    onRespuestasComplete: (List<Int>) -> Unit
) {
```

```
val preguntas = ExamenData.preguntas
var preguntaActual by remember { mutableStateOf(0) }

// Uso mutableStateMapOf para manejar las respuestas de forma más eficiente
val respuestasSeleccionadas = remember {
    mutableStateMapOf<Int, Int>().apply {
        for (i in 0 until preguntas.size) {
            this[i] = -1 // -1 significa que no ha seleccionado respuesta
        }
    }
}

// Obtengo la respuesta actual para facilitar el acceso
val respuestaActual = respuestasSeleccionadas[preguntaActual] ?: -1

// Diseño formal con colores corporativos
Box(
    modifier = Modifier
        .fillMaxSize()
        .background(Color(0xFFF5F7FA)) // Fondo gris claro profesional
) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(20.dp)
    ) {
```

```
// Header corporativo con información del progreso
Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
) {
    Column(
        modifier = Modifier.padding(24.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        // Icono formal
        Icon(
            imageVector = Icons.Default.Edit,
            contentDescription = "Examen",
            modifier = Modifier.size(32.dp),
            tint = Color(0xFF1F2937)
        )

        Spacer(modifier = Modifier.height(12.dp))

        Text(
            text = "EVALUACIÓN",
            fontSize = 18.sp,
```

```
    fontWeight = FontWeight.Bold,
    color = Color(0xFF1F2937),
    letterSpacing = 0.5.sp
)

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

// Muestro el progreso actual
Text(
    text = "Pregunta ${preguntaActual + 1} de ${preguntas.size}",
    fontSize = 14.sp,
    color = Color(0xFF6B7280),
    fontWeight = FontWeight.Medium
)

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

// Barra de progreso sobria y profesional
Column {
    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        Text(
            text = "Progreso",
            fontSize = 12.sp,
```

```
        color = Color(0xFF6B7280)
    )
    Text(
        text = "${((preguntaActual + 1) * 100 / preguntas.size)}%",
        fontSize = 12.sp,
        color = Color(0xFF6B7280),
        fontWeight = FontWeight.Medium
    )
}

Spacer(modifier = Modifier.height(6.dp))

Box(
    modifier = Modifier
        .fillMaxWidth()
        .height(6.dp)
        .clip(RoundedCornerShape(3.dp))
        .background(Color(0xFFE5E7EB))
) {
    // Barra de progreso con color corporativo
    Box(
        modifier = Modifier
            .fillMaxWidth((preguntaActual + 1).toFloat() / preguntas.size)
            .fillMaxHeight()
            .clip(RoundedCornerShape(3.dp))
            .background(Color(0xFF374151))
```

```
            )
          }
        }
      }
}

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

// Card principal para la pregunta actual
Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
) {
    Column(
        modifier = Modifier.padding(28.dp)
    ) {
        // Cabecera de la pregunta
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
```

```
Text(
    text = "Pregunta ${preguntaActual + 1}",
    fontSize = 12.sp,
    fontWeight = FontWeight.SemiBold,
    color = Color(0xFF6B7280),
    modifier = Modifier
        .background(
            Color(0xFFF3F4F6),
            RoundedCornerShape(4.dp)
        )
        .padding(horizontal = 8.dp, vertical = 4.dp)
)

// Indicador si ya respondió esta pregunta
if (respuestaActual != -1) {
    Row(
        verticalAlignment = Alignment.CenterVertically
    ) {
        Icon(
            imageVector = Icons.Default.CheckCircle,
            contentDescription = "Respondida",
            tint = Color(0xFF059669),
            modifier = Modifier.size(16.dp)
        )
        Spacer(modifier = Modifier.width(4.dp))
        Text(
```

```
            text = "Respondida",
            fontSize = 12.sp,
            color = Color(0xFF059669),
            fontWeight = FontWeight.Medium
        )
      }
    }
}

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

// Texto de la pregunta con tipografía formal
Text(
    text = preguntas[preguntaActual].pregunta,
    fontSize = 18.sp,
    fontWeight = FontWeight.SemiBold,
    color = Color(0xFF1F2937),
    lineHeight = 24.sp,
    modifier = Modifier.padding(bottom = 24.dp)
)

// Opciones de respuesta con diseño corporativo
preguntas[preguntaActual].opciones.forEachIndexed { index, opcion ->
    val isSelected = respuestaActual == index

    Card(
```

```
    modifier = Modifier
        .fillMaxWidth()
        .padding(vertical = 4.dp)
        .clickable {
            // Actualizo la respuesta seleccionada
            respuestasSeleccionadas[preguntaActual] = index
        },
    elevation = CardDefaults.cardElevation(
        defaultElevation = if (isSelected) 1.dp else 0.dp
    ),
    colors = CardDefaults.cardColors(
        containerColor = if (isSelected)
            Color(0xFF1F2937)
        else
            Color(0xFFFAFAFA)
    ),
    shape = RoundedCornerShape(6.dp),
    border = BorderStroke(
        width = 1.dp,
        color = if (isSelected)
            Color(0xFF374151)
        else
            Color(0xFFE5E7EB)
    )
) {
    Row(
```

```
    modifier = Modifier
        .fillMaxWidth()
        .padding(16.dp),
    verticalAlignment = Alignment.CenterVertically
) {
    // Indicador visual tipo radio button formal
    Box(
        modifier = Modifier
            .size(20.dp)
            .clip(RoundedCornerShape(10.dp))
            .background(
                if (isSelected)
                    Color.White
                else
                    Color(0xFFE5E7EB)
            ),
        contentAlignment = Alignment.Center
    ) {
        if (isSelected) {
            Text(
                text = "✓",
                fontSize = 12.sp,
                fontWeight = FontWeight.Bold,
                color = Color(0xFF1F2937)
            )
        } else {
```

```
                    // Letra de la opción cuando no está seleccionada
                    Text(
                        text = ('A' + index).toString(),
                        fontSize = 10.sp,
                        fontWeight = FontWeight.Bold,
                        color = Color(0xFF6B7280)
                    )
                }
            }

            Spacer(modifier = Modifier.width(12.dp))

            // Texto de la opción
            Text(
                text = opcion,
                fontSize = 14.sp,
                fontWeight = FontWeight.Medium,
                color = if (isSelected) Color.White else Color(0xFF374151),
                modifier = Modifier.weight(1f),
                lineHeight = 20.sp
            )
        }
    }
  }
}
```

```
Spacer(modifier = Modifier.weight(1f))

// Navegación inferior con botones corporativos
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Botón Anterior - solo visible si no es la primera pregunta
    if (preguntaActual > 0) {
        OutlinedButton(
            onClick = {
                preguntaActual--
            },
            border = BorderStroke(1.dp, Color(0xFF6B7280)),
            shape = RoundedCornerShape(6.dp),
            modifier = Modifier
                .height(44.dp)
                .width(120.dp)
        ) {
            Icon(
                imageVector = Icons.Default.ArrowBack,
                contentDescription = "Anterior",
                modifier = Modifier.size(16.dp),
                tint = Color(0xFF6B7280)
            )
```

```
                Spacer(modifier = Modifier.width(6.dp))
                Text(
                    text = "Anterior",
                    fontSize = 12.sp,
                    fontWeight = FontWeight.Medium,
                    color = Color(0xFF6B7280)
                )
            }
        } else {
            // Spacer invisible para mantener el layout
            Spacer(modifier = Modifier.width(120.dp))
        }

        // Botón Siguiente/Terminar
        if (preguntaActual < preguntas.size - 1) {
            Button(
                onClick = {
                    preguntaActual++
                },
                enabled = respuestaActual != -1, // Solo habilito si seleccionó una
respuesta
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color(0xFF1F2937),
                    contentColor = Color.White,
                    disabledContainerColor = Color(0xFFD1D5DB),
                    disabledContentColor = Color(0xFF9CA3AF)
```

```
        ),
        shape = RoundedCornerShape(6.dp),
        modifier = Modifier
            .height(44.dp)
            .width(120.dp)
    ) {
        Text(
            text = "Siguiente",
            fontSize = 12.sp,
            fontWeight = FontWeight.Medium
        )
        Spacer(modifier = Modifier.width(6.dp))
        Icon(
            imageVector = Icons.Default.ArrowForward,
            contentDescription = "Siguiente",
            modifier = Modifier.size(16.dp)
        )
    }
} else {
    // Botón terminar examen
    Button(
        onClick = {
            // Convierto el mapa a lista y navego a resultados
            val respuestasFinal = mutableListOf<Int>()
            for (i in 0 until preguntas.size) {
                respuestasFinal.add(respuestasSeleccionadas[i] ?: -1)
```

```
                }
                onRespuestasComplete(respuestasFinal)
                navController.navigate("resultados")
            },
            enabled = respuestasSeleccionadas.values.all { it != -1 }, // Solo si respondió todas
            colors = ButtonDefaults.buttonColors(
                containerColor = Color(0xFF059669),
                contentColor = Color.White,
                disabledContainerColor = Color(0xFFD1D5DB),
                disabledContentColor = Color(0xFF9CA3AF)
            ),
            shape = RoundedCornerShape(6.dp),
            modifier = Modifier
                .height(44.dp)
                .width(120.dp)
        ) {
            Icon(
                imageVector = Icons.Default.CheckCircle,
                contentDescription = "Terminar",
                modifier = Modifier.size(16.dp)
            )
            Spacer(modifier = Modifier.width(6.dp))
            Text(
                text = "Finalizar",
                fontSize = 12.sp,
```

```
                    fontWeight = FontWeight.Medium
                )
            }
        }
    }

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

10. screens/ResultadosScreen.kt

kotlin

```
// screens/ResultadosScreen.kt
package com.cortez.examen2parcial.screens

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Home
import androidx.compose.material.icons.filled.Star
import androidx.compose.material.icons.filled.Person
import androidx.compose.material3.*
```

```
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.cortez.examen2parcial.data.ExamenData
import com.cortez.examen2parcial.data.UserInfo
import com.cortez.examen2parcial.utils.ZodiacalCalculator

@Composable
fun ResultadosScreen(
    userInfo: UserInfo,
    respuestasUsuario: List<Int>,
    navController: NavController
) {
    // Calculo la calificación comparando respuestas del usuario vs respuestas correctas
    val respuestasCorrectas = ExamenData.preguntas.map { it.respuestaCorrecta }
    val calificacion = respuestasUsuario.zip(respuestasCorrectas).count { it.first == it.second }

    // Uso mis funciones utilitarias para calcular edad y signo zodiacal
    val edad = ZodiacalCalculator.calcularEdad(userInfo.anio.toIntOrNull() ?: 2000)
```

```
    val signoZodiacal =
ZodiacalCalculator.calcularSignoZodiacalChino(userInfo.anio.toIntOrNull() ?: 2000)
    val imagenSigno = ZodiacalCalculator.getImagenSigno(signoZodiacal)
    val porcentaje = (calificacion * 100) / ExamenData.preguntas.size

    // Determino el estado según la calificación para feedback profesional
    val estadoEvaluacion = when {
        calificacion >= 5 -> "Excelente"
        calificacion >= 3 -> "Satisfactorio"
        else -> "Necesita Mejorar"
    }

    // Diseño formal con fondo corporativo
    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(Color(0xFFF5F7FA)) // Fondo gris claro profesional
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .verticalScroll(rememberScrollState())
                .padding(20.dp),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Spacer(modifier = Modifier.height(20.dp))
```

```
// Header corporativo
Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
) {
    Column(
        modifier = Modifier.padding(32.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Icon(
            imageVector = Icons.Default.Star,
            contentDescription = "Resultados",
            modifier = Modifier.size(40.dp),
            tint = Color(0xFF1F2937)
        )

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

        Text(
            text = "EVALUACIÓN COMPLETADA",
            fontSize = 20.sp,
```

```
            fontWeight = FontWeight.Bold,
            color = Color(0xFF1F2937),
            textAlign = TextAlign.Center,
            letterSpacing = 0.5.sp
        )

        Text(
            text = "Reporte de resultados",
            fontSize = 14.sp,
            color = Color(0xFF6B7280),
            fontWeight = FontWeight.Normal,
            modifier = Modifier.padding(top = 8.dp)
        )
    }
}

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

// Información personal con diseño formal
Card(
    modifier = Modifier.fillMaxWidth(),
    elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
    colors = CardDefaults.cardColors(
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
```

```
        ) {
            Column(
                modifier = Modifier.padding(24.dp)
            ) {
                Row(
                    verticalAlignment = Alignment.CenterVertically,
                    modifier = Modifier.padding(bottom = 16.dp)
                ) {
                    Icon(
                        imageVector = Icons.Default.Person,
                        contentDescription = "Participante",
                        modifier = Modifier.size(20.dp),
                        tint = Color(0xFF6B7280)
                    )
                    Spacer(modifier = Modifier.width(8.dp))
                    Text(
                        text = "Información del Participante",
                        fontSize = 14.sp,
                        fontWeight = FontWeight.SemiBold,
                        color = Color(0xFF374151)
                    )
                }

                Text(
                    text = "${userInfo.nombre} ${userInfo.apellidoPaterno}
${userInfo.apellidoMaterno}",
```

```
    fontSize = 18.sp,
    fontWeight = FontWeight.Bold,
    color = Color(0xFF1F2937),
    modifier = Modifier.padding(bottom = 12.dp)
)

// Información adicional organizada en tabla formal
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Columna izquierda
    Column {
        Text(
            text = "Edad:",
            fontSize = 12.sp,
            color = Color(0xFF6B7280),
            fontWeight = FontWeight.Medium
        )
        Text(
            text = "$edad años",
            fontSize = 16.sp,
            fontWeight = FontWeight.SemiBold,
            color = Color(0xFF374151)
        )
    }
```

```
// Columna central
Column {
    Text(
        text = "Sexo:",
        fontSize = 12.sp,
        color = Color(0xFF6B7280),
        fontWeight = FontWeight.Medium
    )
    Text(
        text = userInfo.sexo,
        fontSize = 16.sp,
        fontWeight = FontWeight.SemiBold,
        color = Color(0xFF374151)
    )
}

// Columna derecha
Column {
    Text(
        text = "Signo Zodiacal:",
        fontSize = 12.sp,
        color = Color(0xFF6B7280),
        fontWeight = FontWeight.Medium
    )
    Row(
```

```
            verticalAlignment = Alignment.CenterVertically
        ) {
            Text(
                text = imagenSigno,
                fontSize = 16.sp
            )
            Spacer(modifier = Modifier.width(4.dp))
            Text(
                text = signoZodiacal,
                fontSize = 16.sp,
                fontWeight = FontWeight.SemiBold,
                color = Color(0xFF374151)
            )
        }
      }
    }
  }
}

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

// Resultados de la evaluación con diseño corporativo
Card(
  modifier = Modifier.fillMaxWidth(),
  elevation = CardDefaults.cardElevation(defaultElevation = 2.dp),
  colors = CardDefaults.cardColors(
```

```
        containerColor = Color.White
    ),
    shape = RoundedCornerShape(8.dp)
) {
    Column(
        modifier = Modifier.padding(32.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(
            text = "RESULTADOS DE LA EVALUACIÓN",
            fontSize = 14.sp,
            fontWeight = FontWeight.Bold,
            color = Color(0xFF374151),
            letterSpacing = 1.sp,
            modifier = Modifier.padding(bottom = 16.dp)
        )

        // Calificación principal con diseño formal
        Row(
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center,
            modifier = Modifier.padding(bottom = 12.dp)
        ) {
            Text(
                text = "$calificacion",
                fontSize = 48.sp,
```

```
        fontWeight = FontWeight.Bold,
        color = Color(0xFF1F2937)
    )

    Text(
        text = " / ${ExamenData.preguntas.size}",
        fontSize = 24.sp,
        fontWeight = FontWeight.Medium,
        color = Color(0xFF6B7280),
        modifier = Modifier.padding(start = 4.dp)
    )
}

Text(
    text = "Respuestas Correctas",
    fontSize = 14.sp,
    color = Color(0xFF6B7280),
    fontWeight = FontWeight.Medium,
    modifier = Modifier.padding(bottom = 12.dp)
)

// Detalles adicionales en formato corporativo
Card(
    modifier = Modifier.fillMaxWidth(),
    colors = CardDefaults.cardColors(
        containerColor = Color(0xFFF9FAFB)
```

```
    ),
    shape = RoundedCornerShape(6.dp),
    elevation = CardDefaults.cardElevation(defaultElevation = 0.dp)
) {
    Column(
        modifier = Modifier.padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Text(
                text = "Porcentaje:",
                fontSize = 14.sp,
                color = Color(0xFF6B7280),
                fontWeight = FontWeight.Medium
            )
            Text(
                text = "$porcentaje%",
                fontSize = 14.sp,
                fontWeight = FontWeight.Bold,
                color = Color(0xFF374151)
            )
        }
```

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

        Row(
          modifier = Modifier.fillMaxWidth(),
          horizontalArrangement = Arrangement.SpaceBetween
        ) {
          Text(
            text = "Estado:",
            fontSize = 14.sp,
            color = Color(0xFF6B7280),
            fontWeight = FontWeight.Medium
          )
          Text(
            text = estadoEvaluacion,
            fontSize = 14.sp,
            fontWeight = FontWeight.Bold,
            color = when {
              calificacion >= 5 -> Color(0xFF059669) // Verde corporativo
              calificacion >= 3 -> Color(0xFFD97706) // Naranja corporativo
              else -> Color(0xFFDC2626)          // Rojo corporativo
            }
          )
        }
      }
    }
  }
```

```
}

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

// Botón para reiniciar con diseño corporativo
Button(
  onClick = {
    // Navego al inicio y limpio el stack de navegación
    navController.navigate("formulario") {
      popUpTo("formulario") { inclusive = true }
    }
  },
  colors = ButtonDefaults.buttonColors(
    containerColor = Color(0xFF1F2937),
    contentColor = Color.White
  ),
  shape = RoundedCornerShape(6.dp),
  modifier = Modifier
    .fillMaxWidth()
    .height(48.dp)
) {
  Icon(
    imageVector = Icons.Default.Home,
    contentDescription = "Inicio",
    modifier = Modifier.size(20.dp)
  )
```

```
            Spacer(modifier = Modifier.width(12.dp))

            Text(
                text = "Nueva Evaluación",
                fontSize = 14.sp,
                fontWeight = FontWeight.Medium
            )
        }

        Spacer(modifier = Modifier.height(40.dp))
    }
  }
}
```

11. AndroidManifest.xml

xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
```

```
    android:supportsRtl="true"
    android:theme="@android:style/Theme.Material.DayNight.DarkActionBar"
    tools:targetApi="31">
    <activity
      android:name="com.cortez.examen2parcial.MainActivity"
      android:exported="true"
      android:theme="@android:style/Theme.Material.DayNight.DarkActionBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```