EXAMEN SEGUNDO PARCIAL – ZODIACO CHINO

Materia: Desarrollo para dispositivos inteligentes

Grupo: IDGS 904

Matricula: 22001661

Nombre Alumno: Armando Daniel Rodríguez Fajardo

Lugar: León, Gto.

Fecha: 04/07/2025

Institución: Universidad Tecnología de León

Carrera: Ingeniería en Desarrollo y Gestión de Software

Índice

MainActivity.kt

```
package com.lifethech.zoodiacochino

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import androidx.navigation.compose.rememberNavController
import com.lifethech.zoodiacochino.screens.navegacion
import com.lifethech.zoodiacochino.screens.screenPantallaDatos
import com.lifethech.zoodiacochino.ui.theme.ZoodiacochinoTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //enableEdgeToEdge()
        setContent {
```

```
            ZoodiacochinoTheme {
                // Crea un fondo con el color del tema
                Surface(color = MaterialTheme.colorScheme.background) {
                    // Controlador de navegación para cambiar entre
pantallas
                    val navegacionC = rememberNavController()
                    // Llama a la función de la navegación
                    navegacion(navegacionC)
                }
            }
        }
    }
}
```

cambioPantallas.kt

```
package com.lifethech.zoodiacochino.screens

import androidx.compose.runtime.Composable
import androidx.navigation.compose.NavHost
import androidx.navigation.NavHostController
import androidx.navigation.compose.composable


@Composable
fun navegacion(navegacionC: NavHostController){
    // Define la navegación entre pantallas usando el NavHost
    NavHost(navegacionC, startDestination = "datos") {
        composable("datos") { screenPantallaDatos(navegacionC) }
        composable("examen") { screenExamen(navegacionC) }
        composable("repaso") { screenRepaso(navegacionC) }
        composable("resultadoFinal") { screenResultadoFinal() }

    }
}
```

screenPantallaDatos.kt

```
package com.lifethech.zoodiacochino.screens

import android.content.Context
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
```

```
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.selection.selectableGroup
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.RadioButton
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.semantics.Role
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController

private val filename = "datos_personales.txt"

@Composable
fun screenPantallaDatos(navegacionC: NavController) {
    val context = LocalContext.current
    // Variables para guardar los datos ingresados
    var nombre by remember { mutableStateOf("") }
    var apePaterno by remember { mutableStateOf("") }
    var apeMaterno by remember { mutableStateOf("") }
    var dia by remember { mutableStateOf("") }
    var mes by remember { mutableStateOf("") }
    var anio by remember { mutableStateOf("") }
    var sexoSeleccionado by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.padding(16.dp).fillMaxWidth(),
        verticalArrangement = Arrangement.spacedBy(10.dp)
    ) {
```

```
        Text("Datos Personales", fontSize = 20.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally))

        // Campos de texto para capturar nombre y apellidos
        TextField(
            value = nombre,
            onValueChange = { nombre = it },
            label = { Text("Nombre") },
            modifier = Modifier.fillMaxWidth()
        )

        TextField(
            value = apePaterno,
            onValueChange = { apePaterno = it },
            label = { Text("Apellido Paterno") },
            modifier = Modifier.fillMaxWidth()
        )

        TextField(
            value = apeMaterno,
            onValueChange = { apeMaterno = it },
            label = { Text("Apellido Materno") },
            modifier = Modifier.fillMaxWidth()
        )

        Text("Fecha de nacimiento", fontSize = 16.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally))

        Row(horizontalArrangement = Arrangement.spacedBy(8.dp)) {
            // Campos para la Fecha de nacimiento
            TextField(
                value = dia,
                onValueChange = { dia = it },
                label = { Text("Día") },
                keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
                modifier = Modifier.weight(1f)
            )
            TextField(
                value = mes,
                onValueChange = { mes = it },
                label = { Text("Mes") },
                keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
```

```
                modifier = Modifier.weight(1f)
            )
            TextField(
                value = anio,
                onValueChange = { anio = it },
                label = { Text("Año") },
                keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
                modifier = Modifier.weight(1f)
            )
        }
        // Selección de sexo con radio buttons
        Text("Sexo", fontSize = 16.sp)

        val opcionesSexo = listOf("Masculino", "Femenino")

        Column(Modifier.selectableGroup()) {
            opcionesSexo.forEach { opcion ->
                Row(
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(56.dp)
                        .selectable(
                            selected = (opcion == sexoSeleccionado),
                            onClick = { sexoSeleccionado = opcion },
                            role = Role.RadioButton
                        )
                        .padding(horizontal = 8.dp),
                    verticalAlignment = Alignment.CenterVertically
                ) {
                    RadioButton(
                        selected = (opcion == sexoSeleccionado),
                        onClick = null
                    )
                    Text(
                        text = opcion,
                        style = MaterialTheme.typography.bodyLarge,
                        modifier = Modifier.padding(start = 8.dp)
                    )
                }
            }
        }

        Row(
```

```
            horizontalArrangement = Arrangement.SpaceBetween,
            modifier = Modifier.fillMaxWidth()
        ) {
            // Botones para limpiar
            Button(onClick = {
                nombre = ""
                apePaterno = ""
                apeMaterno = ""
                dia = ""
                mes = ""
                anio = ""
                sexoSeleccionado = ""
            }) {
                Text("Limpiar")
            }
            //Boton para guardar y navegar
            Button(onClick = {
                //Guarda los datos en un archivo
                guardarDatos(context, nombre, apePaterno, apeMaterno,
dia, mes, anio, sexoSeleccionado)
                // Navega a la siguiente pantalla
                navegacionC.navigate("examen")
            }) {
                Text("Siguiente")
            }
        }
    }

}

// Función para guardar los datos en un archivo
fun guardarDatos(context: Context, nombre: String, apePaterno: String,
                 apeMaterno: String, dia: String, mes: String, anio:
String,
                 sexo: String) {
    // Formato del texto a guardar
    val text = "nombre:$nombre\n" +
            "apePaterno:$apePaterno\n" +
            "apeMaterno:$apeMaterno\n" +
            "dia:$dia\n" +
            "mes:$mes\n" +
            "anio:$anio\n" +
            "sexo:$sexo\n"
```

```
    try {
        context.openFileOutput(filename, Context.MODE_PRIVATE).use {
            it.write(text.toByteArray())
        }
    }catch (e: Exception) {
        e.printStackTrace()
    }
}
```

screenExamen.kt

```
package com.lifethech.zoodiacochino.screens

import android.content.Context
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.selection.selectableGroup
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.Button
import androidx.compose.material3.RadioButton
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.semantics.Role
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import androidx.navigation.compose.rememberNavController

//Clase para las preguntas del examen
data class Pregunta(
```

```
    val texto: String,
    val opciones: List<String>,
    val correcta: Int
)

// Lista de preguntas del examen
val preguntas = listOf(
    Pregunta("¿Cuál es la suma de 2 + 2?", listOf("3", "4", "2", "5"),
1),
    Pregunta("¿Cuál es la resta de 10 - 4?", listOf("6", "5", "7", "4"),
0),
    Pregunta("¿Cuánto es 3 x 3?", listOf("6", "9", "8", "7"), 1),
    Pregunta("¿Cuánto es 8 / 2?", listOf("4", "2", "6", "3"), 0),
    Pregunta("¿Cuál es la raíz cuadrada de 49?", listOf("6", "8", "7",
"9"), 2),
    Pregunta("¿Cuánto es 5 + 7?", listOf("11", "12", "13", "10"), 1)
)

@Composable
fun screenExamen(navegacionC: NavController) {
    val context = LocalContext.current
    // variable para almacenar las respuestas seleccionadas
    var respuestasSeleccionadas by remember {
mutableStateOf(List(preguntas.size) { -1 }) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
        .verticalScroll(rememberScrollState()),
        verticalArrangement = Arrangement.spacedBy(20.dp)
    ) {
        Text(
            "Examen",
            fontSize = 24.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally)
        )
        //Iteración sobre las preguntas para mostrarlas
        preguntas.forEachIndexed { index, pregunta ->
            Column {
                Text(text = "${index + 1}. ${pregunta.texto}", fontSize =
18.sp)
                Column(Modifier.selectableGroup()) {
                    pregunta.opciones.forEachIndexed { opcionIndex,
```

```
opcion ->
                        Row(
                            Modifier
                                .fillMaxWidth()
                                .selectable(
                                    selected =
respuestasSeleccionadas[index] == opcionIndex,
                                    onClick = {
                                        respuestasSeleccionadas =
respuestasSeleccionadas.toMutableList().also {
                                            it[index] = opcionIndex
                                        }
                                    },
                                    role = Role.RadioButton
                                )
                                .padding(vertical = 4.dp),
                            verticalAlignment =
Alignment.CenterVertically
                        ) {
                            RadioButton(
                                selected = respuestasSeleccionadas[index]
== opcionIndex,
                                onClick = null
                            )
                            Text(text = "${'a' + opcionIndex}) $opcion",
modifier = Modifier.padding(start = 8.dp))
                        }
                    }
                }
            }
        }

        Button(
            onClick = {
                //Guardar las respuestas
                guardarRespuestasYResultado(context,
respuestasSeleccionadas)
                // Navegar a la pantalla de repaso
                navegacionC.navigate("repaso")
            },
            modifier = Modifier.align(Alignment.CenterHorizontally)
        ) {
            Text("Entregar examen")
        }
```

```
    }
}

// Función para guardar las respuestas y los aciertos en archivos
fun guardarRespuestasYResultado(context: Context, respuestas: List<Int>)
{
    var texto = ""
    var correctas = 0

    // Iteracion sobre las respuestas para crear el texto a guardar y
contar las correctas
    for (i in respuestas.indices) {
        texto += "pregunta_${i + 1}:${respuestas[i]}\n"
        if (respuestas[i] == preguntas[i].correcta) {
            correctas++
        }
    }

    try {
        // Guardar las respuestas en un archivo
        context.openFileOutput("respuestas.txt",
Context.MODE_PRIVATE).use {
            it.write(texto.toByteArray())
        }

        // Leer los datos personales del archivo y agregar los aciertos
        val datos =
context.openFileInput("datos_personales.txt").bufferedReader().useLines {
lines ->
            lines.joinToString("\n")
        }
        // Agregar los aciertos al final de los datos
        val nuevosDatos = datos + "\naciertos:$correctas\n"
        // Guardar los nuevos datos en el archivo
        context.openFileOutput("datos_personales.txt",
Context.MODE_PRIVATE).use {
            it.write(nuevosDatos.toByteArray())
        }

    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

screenResultadosExamen.kt

```
package com.lifethech.zoodiacochino.screens

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController

@Composable
fun screenRepaso(navegacionC: NavController) {
    val context = LocalContext.current
    var mapaRespuestas: Map<String, String> = emptyMap()

    try {
        // Leer el archivo de respuestas guardado
        val contenido =
context.openFileInput("respuestas.txt").bufferedReader().useLines {
it.toList() }
        mapaRespuestas = contenido.associate {
            val partes = it.split(":")
            partes[0] to partes[1]
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }

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

```
            .verticalScroll(rememberScrollState()),
        verticalArrangement = Arrangement.spacedBy(10.dp)
    ) {
        Text("Revisión del Examen", fontSize = 20.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally))
        // Mostrar las preguntas y respuestas
        preguntas.forEachIndexed { idx, pregunta ->
            val seleccion = mapaRespuestas["pregunta_${idx +
1}"]?.toIntOrNull()
            val correcta = pregunta.correcta
            // Verificar si la respuesta seleccionada es correcta
            val esCorrecta = seleccion == correcta
            // Le da color a la respuesta según si es correcta o
incorrecta
            val color = if (esCorrecta) Color(0xFF2E7D32) else Color.Red
            // Texto que muestra si la respuesta es correcta o incorrecta
            val textoResultado = if (esCorrecta) {
                "Correcto, tu repuesta fue:
${pregunta.opciones[correcta]}"
            } else {
                val respuestaUsuario = seleccion?.let {
pregunta.opciones.getOrNull(it) } ?: "Sin responder"
                "Incorrecto\nTu respuesta fue: $respuestaUsuario\nOpcion
Correcta: ${pregunta.opciones[correcta]}"
            }

            Column {
                // Mostrar el número de la pregunta y el texto de la
pregunta
                Text("${idx + 1}. ${pregunta.texto}")
                Text(textoResultado, modifier = Modifier.padding(start =
8.dp),
                    color = color)
            }
        }
        // navegar a la pantalla de resultados finales
        Button(onClick = { navegacionC.navigate("resultadoFinal") },
modifier = Modifier.align(Alignment.CenterHorizontally)) {
            Text("Finalizar")
        }
    }
}
```

screenResultadosFinales.kt

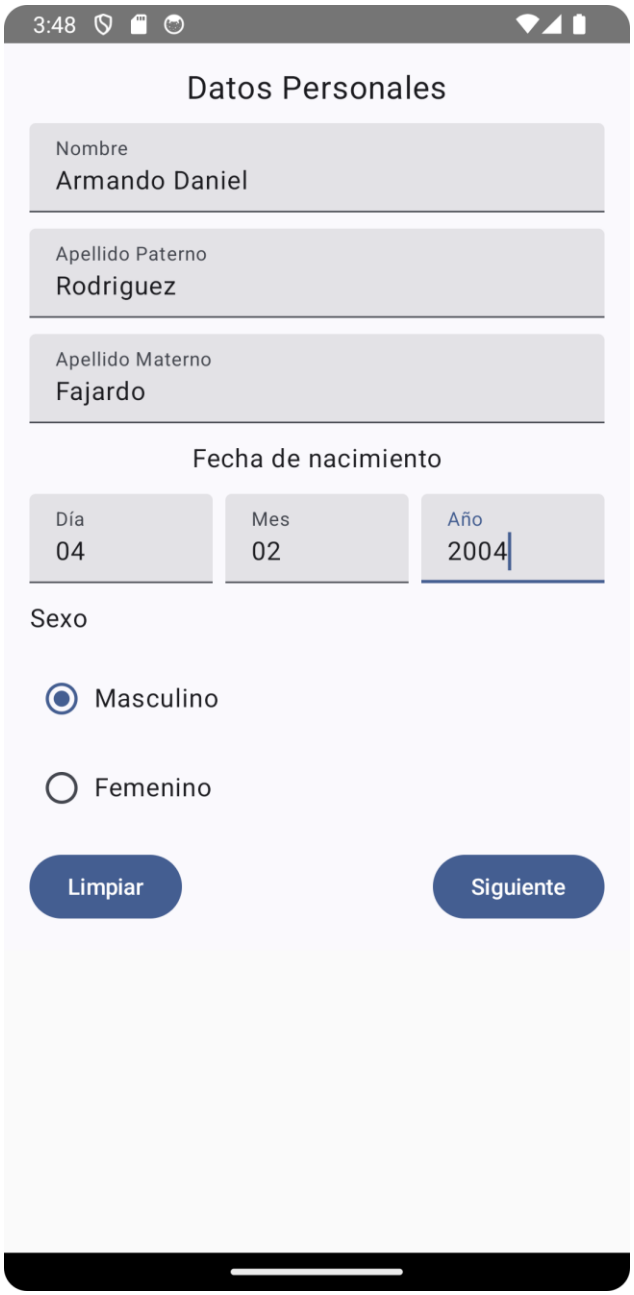
```
package com.lifethech.zoodiacochino.screens

import android.content.Context
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
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.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import java.io.FileNotFoundException
import java.util.Calendar


@Composable
fun screenResultadoFinal() {
    val context = LocalContext.current
    val datos = leerDatos(context)
    // Verifica si los datos fueron leídos correctamente evitando un Null
    if (datos != null) {
        // Obtiene los datos
        val nombreCompleto = "${datos["nombre"]} ${datos["apePaterno"]}
${datos["apeMaterno"]}"
        val dia = datos["dia"]?.toIntOrNull() ?: 1
        val mes = datos["mes"]?.toIntOrNull() ?: 1
        val anio = datos["anio"]?.toIntOrNull() ?: 2000
        val aciertos = datos["aciertos"]?.toIntOrNull() ?: 0

        val edad = calcularEdad(dia, mes, anio)
        val signo = obtenerSignoZodiacoChino(anio)
        val calificacion = (aciertos * 10) / 6.0
```

```
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.spacedBy(12.dp)
        ) {
            // Mostrar el mensaje de bienvenida con el nombre completo
            Text(text = "¡Hola, $nombreCompleto!", fontSize = 24.sp,
fontWeight = FontWeight.Bold,
                modifier = Modifier.align(Alignment.CenterHorizontally))
            //Muestra la edad
            Text(text = "Tienes $edad años")
            // Muestra el signo zodiacal chino
            Text(text = "Tu signo zodiacal chino es:", fontSize = 18.sp)

            // Cargar imagen del signo
            val imageId = context.resources.getIdentifier(
                signo.lowercase(), "drawable", context.packageName
            )
            if (imageId != 0) {
                Image(
                    painter = painterResource(id = imageId),
                    contentDescription = signo,
                    modifier = Modifier
                        .height(150.dp)
                        .padding(8.dp)
                )
            }

            Text(text = signo.capitalize(), fontSize = 20.sp)

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

            // Mostrar los aciertos y la calificación
            Text(
                text = "Tuviste $aciertos aciertos",
                fontSize = 18.sp,
                fontWeight = FontWeight.SemiBold
            )
            Text(
                text = "Tu calificación es: ${String.format("%.1f",
calificacion)}",
                fontSize = 26.sp,
```

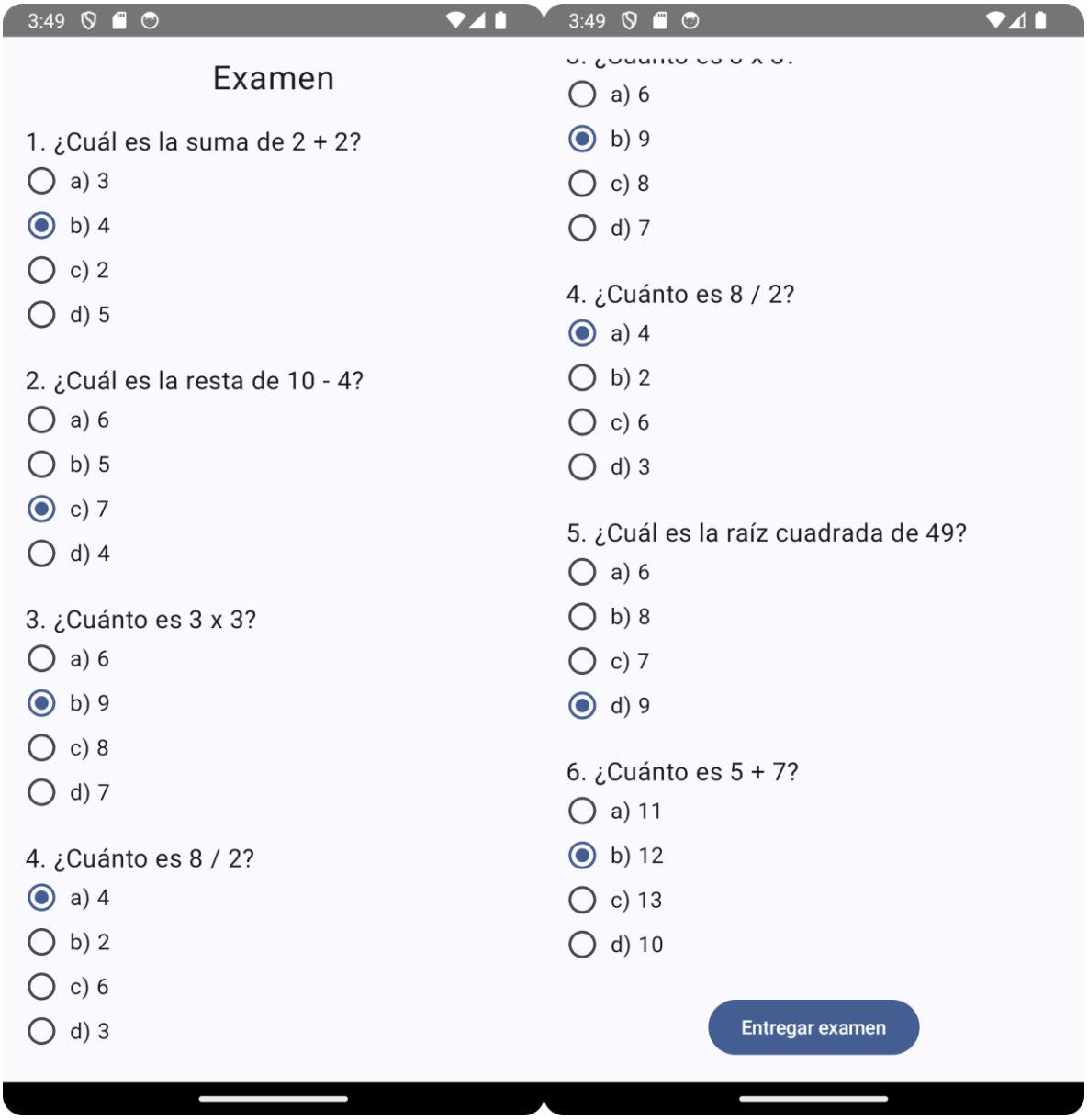
```
                fontWeight = FontWeight.Bold
            )
        }
    } else {
        Text("No se pudo leer la información.")
    }
}

// Función para leer los datos del archivo
fun leerDatos(context: Context): Map<String, String>? {
    return try {
        // Abre el archivo y lee su contenido
        val contenido = context.openFileInput("datos_personales.txt")
            .bufferedReader().useLines { it.joinToString("\n") }
        // Divde el contenido en líneas y crea un mapa
        val lineas = contenido.split("\n")
        val mapa = mutableMapOf<String, String>()
        // Itera sobre las líneas y separa las claves y valores
        for (linea in lineas) {
            val partes = linea.split(":")
            if (partes.size == 2) {
                mapa[partes[0].trim()] = partes[1].trim()
            }
        }
        mapa
    } catch (e: FileNotFoundException) {
        null
    } catch (e: Exception) {
        e.printStackTrace()
        null
    }
}

// Función para calcular la edad con la fecha de nacimiento
fun calcularEdad(dia: Int, mes: Int, anio: Int): Int {

    val hoy = Calendar.getInstance()
    val nacimiento = Calendar.getInstance()
    nacimiento.set(anio, mes - 1, dia)
    // Calcula la edad restando el año de nacimiento al año actual
    var edad = hoy.get(Calendar.YEAR) - nacimiento.get(Calendar.YEAR)
    // Verifica si el cumpleaños ya paso este año
    if (hoy.get(Calendar.DAY_OF_YEAR) <
nacimiento.get(Calendar.DAY_OF_YEAR)) {
```

```
        edad--
    }
    return edad
}
// Función para obtener el signo zodiacal chino según el año de
nacimiento
fun obtenerSignoZodiacoChino(anio: Int): String {
    // Lista de signos zodiacales chinos
    val signos = listOf(
        "rata", "buey", "tigre", "conejo", "dragon", "serpiente",
        "caballo", "cabra", "mono", "gallo", "perro", "cerdo"
    )
    // Obtiene el indice del signo zodiacal chino basado en el año
    val index = (anio - 1900) % 12
    return signos.getOrElse(index) { "desconocido" }
}
```

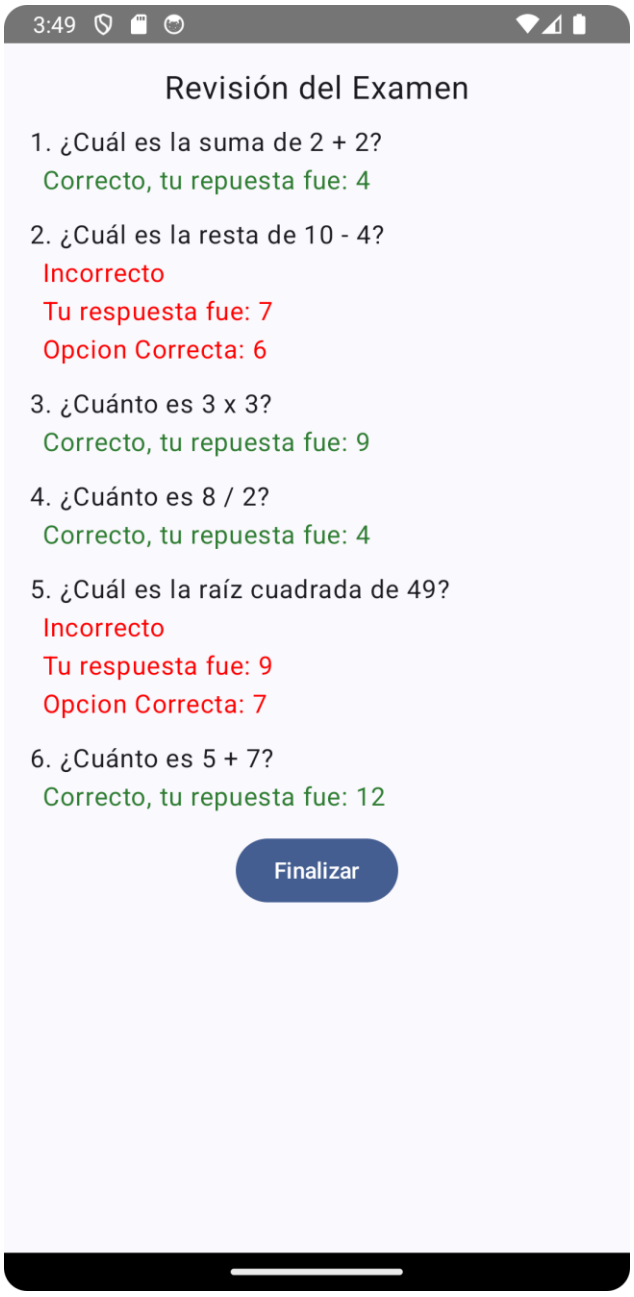
Pagina primera sección – Datos Personales



Pagina segunda sección – Examen



Pagina tercera sección – Resultados Examen



Pagina cuarta sección – Resultados Finales – signo zodiacal