

# Examen Segundo Parcial

## Desarrollo de Dispositivos Inteligentes

Nombre: Angel Eduardo Puga Barrientos Fecha: 04/07/2025

Grupo: IDGS904

Para los siguientes problemas crearas una App utilizando Jetpack Compose, uso de clases para crear el formulario implementa la utilización de archivos de texto o utiliza Firebase cualquiera de las dos opciones son válidas.

1.-Crear 4 ventanas con el siguiente contenido y funcionalidad

2.- Crearas la siguiente ventana con los componentes que se muestran en la siguiente imagen.

- Llena todos los campos con la información que se te pide, en la fecha de nacimiento se te sugiere que manejes números para cada campo ejemplo Día=22, Mes=02, Año=1990
- El botón limpiar nos permite limpiar los componentes de la ventana.
- El botón siguiente nos enviara la ventana dos
- En el apartado de sexo elegirás tu sexo con radio botones.



3.- crear una ventana donde colocaras un examen de 6 preguntas cada una de las preguntas tendrá 4 posibles respuestas el usuario podrá seleccionar con radio botones la respuesta correcta.

¿Cuál es la suma de 2 + 2?			
a) 8	b) 6	c) 4	d) 3

Al terminar el examen dará clic a un botón terminar y mostrará una ventana con los resultados

3.- Deberás desplegar la información como se muestra en la siguiente imagen, se calcula la edad de la persona y además se identifica que signo es se imprimirá la imagen de si signo zodiacal chino



## Contenido

AuthScreen.kt .....	3
ExamenScreen.kt .....	5
ExamenResults .....	8
FirebaseRepository.kt .....	8
MainActivity .....	13
Navigation.kt .....	14
UserViewModel.kt .....	19
Capturas de la Aplicacion .....	20
Firebase .....	24

## AuthScreen.kt

```
package com.example.examensegundoparcial

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.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import kotlinx.coroutines.launch
import com.example.examensegundoparcial.navigation.Screens
@Composable
fun AuthScreen(
    navController: NavHostController,
    repository: FirebaseRepository
){
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var isLoading by remember { mutableStateOf(false) }
    var errorMessage by remember { mutableStateOf<String?>(null) }

    val coroutineScope = rememberCoroutineScope()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(32.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ){
        Text("Iniciar Sesión", style = MaterialTheme.typography.headlineMedium)

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

        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )

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

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Contraseña") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )
    }
}
```

```

        )

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

if (isLoading) {
    CircularProgressIndicator()
} else {
    Button(
        onClick = {
            if (email.isBlank() || password.isBlank()) {
                errorMessage = "Por favor ingresa email y contraseña"
                return@Button
            }

            isLoading = true
            errorMessage = null

            coroutineScope.launch {
                try {
                    repository.signInWithEmailAndPassword(email, password)
                    navController.navigate(Screens.Formulario.name) {
                        // Opciones de navegación
                        popUpTo(Screens.Auth.name) { inclusive = true }
                    }
                } catch (e: Exception) {
                    errorMessage = "Error al iniciar sesión: ${e.message}"
                } finally {
                    isLoading = false
                }
            }
        },
        modifier = Modifier.fillMaxWidth()
    ) {
        Text("Iniciar Sesión")
    }
}

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

TextButton(
    onClick = {
        if (email.isBlank()) {
            errorMessage = "Por favor ingresa un email"
            return@TextButton
        }

        isLoading = true
        errorMessage = null

        coroutineScope.launch {
            try {
                repository.createUserWithEmailAndPassword(email, password)
                navController.navigate(Screens.Formulario.name) {

```

```
// Opciones de navegación
popUpTo(Screens.Auth.name) { inclusive = true
}
} catch (e: Exception) {
    errorMessage = "Error al registrar: ${e.message}"
} finally {
    isLoading = false
}
}
)
{
    Text("¿No tienes cuenta? Regístrate")
}

errorMessage?.let {
    Spacer(modifier = Modifier.height(8.dp))
    Text(
        text = it,
        color = MaterialTheme.colorScheme.error
    )
}
}
```

# ExamenScreen.kt

```
package com.example.examensegundoparcial

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import com.example.examensegundoparcial.navigation.Screens
import kotlinx.coroutines.launch
@Composable
fun ExamenScreen(
    navController: NavHostController,
    repository: FirebaseRepository
) {
    val preguntas = listOf(
        Pregunta("¿Cuál es la suma de 2 + 2?", listOf("8", "6", "4", "3"), 2),
        Pregunta("¿Qué protocolo se utiliza para asignar direcciones IP automáticamente?", listOf("DNS", "TCP", "DHCP", "FTP"), 2),
        Pregunta("¿Cuál es el protocolo utilizado para acceder a páginas web?", listOf("HTTP", "IP", "SSH", "SMTP"), 0),
        Pregunta("¿Qué significa la sigla 'IP?'", listOf("Internet Provider", "Internet Protocol", "Internal Port", "External Port"), 2)
    )
}
```

```

    "Intranet Protocol"), 1),
    Pregunta("¿Qué protocolo se utiliza para enviar correos electrónicos?", listOf("FTP", "SMTP", "HTTP",
    "SNMP"), 1),
    Pregunta("¿Qué capa del modelo OSI se encarga del enrutamiento?", listOf("Física", "Red",
    "Transporte", "Aplicación"), 1),
    Pregunta("¿Cuál de los siguientes es un protocolo seguro de transferencia de archivos?", listOf("FTP", "TFTP", "SFTP", "POP3"), 2),
    Pregunta("¿Qué tecnología permite conectarse de forma inalámbrica a una red local?", listOf("Bluetooth", "Wi-Fi", "USB", "Ethernet"), 1),
    Pregunta("¿Cuál es la dirección IP de loopback?", listOf("192.168.1.1", "10.0.0.1", "127.0.0.1",
    "0.0.0.0"), 2),
    Pregunta("¿Qué dispositivo conecta redes diferentes y dirige el tráfico?", listOf("Switch", "Hub",
    "Router", "Bridge"), 2),
    Pregunta("¿Qué puerto usa HTTPS por defecto?", listOf("80", "22", "443", "8080"), 2),
    Pregunta("¿Qué protocolo se usa para acceder de forma remota a una terminal segura?", listOf("Telnet", "SSH", "RDP", "FTP"), 1),
    Pregunta("¿Qué protocolo resuelve nombres de dominio en direcciones IP?", listOf("ARP", "DNS",
    "IP", "DHCP"), 1),
    Pregunta("¿Qué protocolo se utiliza para monitorear dispositivos de red?", listOf("SMTP", "ICMP",
    "SNMP", "ARP"), 2),
    Pregunta("¿Qué tipo de dirección IP cambia cada vez que te conectas a la red?", listOf("Estática",
    "Pública", "Privada", "Dinámica"), 3),
    Pregunta("¿Cuál es la máscara de subred para una red clase C típica?", listOf("255.255.255.0",
    "255.0.0.0", "255.255.0.0", "255.255.255.255"), 0),
    Pregunta("¿Qué capa del modelo OSI corresponde al cableado físico?", listOf("Enlace de datos",
    "Red", "Física", "Transporte"), 2),
    Pregunta("¿Qué dispositivo de red trabaja en la capa de enlace de datos?", listOf("Router", "Switch",
    "Modem", "Firewall"), 1),
    Pregunta("¿Qué puerto se usa para SSH?", listOf("23", "80", "443", "22"), 3),
    Pregunta("¿Qué tipo de red conecta dispositivos dentro de un edificio o zona local?", listOf("WAN",
    "MAN", "LAN", "PAN"), 2)
)

```

```

var respuestas by remember { mutableStateOf(List(preguntas.size) { -1 }) }
var showAlert by remember { mutableStateOf(false) }
val coroutineScope = rememberCoroutineScope()

Column(
    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)
        .verticalScroll(rememberScrollState()),
    verticalArrangement = Arrangement.spacedBy(16.dp)
) {
    Text("Examen de Redes", style = MaterialTheme.typography.headlineMedium)

    preguntas.forEachIndexed { index, pregunta ->
        Column(modifier = Modifier.padding(vertical = 8.dp)) {
            Text(
                text = "${index + 1}. ${pregunta.texto}",
                style = MaterialTheme.typography.bodyLarge
            )
        }
    }
}

```

```

pregunta.opciones.forEachIndexed { optionIndex, option ->
    Row(
        Modifier
            .fillMaxWidth()
            .selectable(
                selected = (respuestas[index] == optionIndex),
                onClick = {
                    val newRespuestas = respuestas.toMutableList()
                    newRespuestas[index] = optionIndex
                    respuestas = newRespuestas
                }
            )
            .padding(8.dp),
        verticalAlignment = Alignment.CenterVertically
    ) {
        RadioButton(
            selected = (respuestas[index] == optionIndex),
            onClick = {
                val newRespuestas = respuestas.toMutableList()
                newRespuestas[index] = optionIndex
                respuestas = newRespuestas
            }
        )
        Text(text = option, modifier = Modifier.padding(start = 8.dp))
    }
}
Divider(modifier = Modifier.padding(vertical = 8.dp))
}

Button(
    onClick = {
        if (respuestas.all { it != -1 }) {
            val calificacion = calcularCalificacion(preguntas, respuestas)

            coroutineScope.launch {
                repository.saveExamResults(
                    userId = repository.getCurrentUserId() ?: "",
                    calificacion = calificacion,
                    respuestas = respuestas
                )
            }

            navController.navigate("${Screens.Resultados.name}/${calificacion}") {
                popUpTo(Screens.Examen.name) { inclusive = true }
            }
        } else {
            showAlert = true
        }
    },
    modifier = Modifier.fillMaxWidth()
) {
    Text("Terminar Examen")
}

```

```

        }

    }

    if (showAlert) {
        AlertDialog(
            onDismissRequest = { showAlert = false },
            title = { Text("Examen incompleto") },
            text = { Text("Por favor responde todas las preguntas antes de terminar.") },
            confirmButton = {
                Button(onClick = { showAlert = false }) {
                    Text("OK")
                }
            }
        )
    }
}

fun calcularCalificacion(preguntas: List<Pregunta>, respuestas: List<Int>): Int {
    val correctas = preguntas.indices.count { respuestas[it] == preguntas[it].respuestaCorrecta }
    return (correctas * 100) / preguntas.size
}

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

```

## ExamenResults

```

package com.example.examensegundoparcial
data class ExamResults(
    val calificacion: Int = 0,
    val respuestas: List<Int> = emptyList(),
    val fecha: String = ""
)

```

## FirebaseRepository.kt

```

package com.example.examensegundoparcial

import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebaseio.firebaseio.FirebaseFirestore
import com.google.firebaseio.firebaseio.ktx.firebaseio
import com.google.firebaseio.ktx.Firebase
import kotlinx.coroutines.tasks.await
import java.text.SimpleDateFormat
import java.util.Date

```

```

import java.util.Locale

class FirebaseRepository {
    private val auth: FirebaseAuth = Firebase.auth
    private val db: FirebaseFirestore = Firebase.firestore

    // Autenticación
    suspend fun signInWithEmailAndPassword(email: String, password: String) {
        auth.signInWithEmailAndPassword(email, password).await()
    }

    suspend fun createUserWithEmailAndPassword(email: String, password: String) {
        auth.createUserWithEmailAndPassword(email, password).await()
    }

    fun getCurrentUserId(): String? = auth.currentUser?.uid

    fun signOut() {
        auth.signOut()
    }

    // Firestore - Datos del usuario
    suspend fun saveUserData(userId: String, userData: UserData) {
        db.collection("users").document(userId).set(userData).await()
    }

    suspend fun getUserData(userId: String): UserData? {
        return db.collection("users").document(userId).get().await().toObject(UserData::class.java)
    }

    suspend fun saveExamResults(
        userId: String,
        calificacion: Int,
        respuestas: List<Int>
    ) {
        val results = mapOf(
            "calificacion" to calificacion,
            "respuestas" to respuestas,
            "fecha" to SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(Date())
        )
        db.collection("exam_results").document(userId).set(results).await()
    }

    suspend fun getExamResults(userId: String): FirebaseExamResults? {
        return db.collection("exam_results").document(userId).get().await().toObject(
            FirebaseExamResults::class.java
        )
    }
}

data class FirebaseUserData(
    val nombre: String = "",
    val apellidoPaterno: String = "",
    val apellidoMaterno: String = ""
)

```

```

    val diaNacimiento: Int = 0,
    val mesNacimiento: Int = 0,
    val anioNacimiento: Int = 0,
    val sexo: String = ""
)
data class FirebaseExamResults(
    val calificacion: Int = 0,
    val respuestas: List<Int> = emptyList(),
    val fecha: String = ""
)

```

## FormularioScreen.kt

```

package com.example.examensegundoparcial

import android.annotation.SuppressLint
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import kotlinx.coroutines.launch

@Suppress("UnrememberedMutableState")
@Composable
fun FormularioScreen(
    navController: NavHostController,
    repository: FirebaseRepository,
    userViewModel: UserViewModel
){
    var nombre by remember { mutableStateOf("") }
    var apellidoPaterno by remember { mutableStateOf("") }
    var apellidoMaterno by remember { mutableStateOf("") }
    var dia by remember { mutableStateOf("") }
    var mes by remember { mutableStateOf("") }
    var anio by remember { mutableStateOf("") }

    val sexoOptions = listOf("Masculino", "Femenino", "Otro")
    var selectedSexo by remember { mutableStateOf(sexoOptions[0]) }

    val coroutineScope = rememberCoroutineScope()
    val userId = repository.getCurrentUser()
    var errorMessage by remember { mutableStateOf<String?>(null) }

```

```

Column(
    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)
        .verticalScroll(rememberScrollState()),
    verticalArrangement = Arrangement.spacedBy(16.dp)
){
    Text("Formulario de Datos Personales", style = MaterialTheme.typography.headlineMedium)

    OutlinedTextField(
        value = nombre,
        onValueChange = { nombre = it },
        label = { Text("Nombre(s)") },
        modifier = Modifier.fillMaxWidth()
    )

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

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

    Text("Fecha de Nacimiento", style = MaterialTheme.typography.labelLarge)
    Row(
        modifier = Modifier.fillMaxWidth(),
        horizontalArrangement = Arrangement.spacedBy(8.dp)
    ){
        OutlinedTextField(
            value = dia,
            onValueChange = { if (it.length <= 2) dia = it },
            label = { Text("Día") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
            modifier = Modifier.weight(1f)
        )

        OutlinedTextField(
            value = mes,
            onValueChange = { if (it.length <= 2) mes = it },
            label = { Text("Mes") },
            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
            modifier = Modifier.weight(1f)
        )

        OutlinedTextField(

```

```

        value = anio,
        onValueChange = { if (it.length <= 4) anio = it },
        label = { Text("Año") },
        keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
        modifier = Modifier.weight(1f)
    )
}

Text("Sexo", style = MaterialTheme.typography.labelLarge)
Column {
    sexoOptions.forEach { option ->
        Row(
            Modifier
                .fillMaxWidth()
                .selectable(
                    selected = (option == selectedSexo),
                    onClick = { selectedSexo = option }
                )
                .padding(8.dp),
            verticalAlignment = Alignment.CenterVertically
        ){
            RadioButton(
                selected = (option == selectedSexo),
                onClick = { selectedSexo = option }
            )
            Text(
                text = option,
                modifier = Modifier.padding(start = 8.dp)
            )
        }
    }
}

Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.spacedBy(16.dp)
){
    Button(
        onClick = {
            nombre = ""
            apellidoPaterno = ""
            apellidoMaterno = ""
            dia = ""
            mes = ""
            anio = ""
            selectedSexo = sexoOptions[0]
            errorMessage = null
        },
        modifier = Modifier.weight(1f)
    ){
        Text("Limpiar")
    }
}

```

```
        Button(
            onClick = {
                if (nombre.isBlank() || apellidoPaterno.isBlank() || dia.isBlank() || mes.isBlank() || anio.isBlank()) {
                    errorMessage = "Por favor llena todos los campos requeridos"
                    return@Button
                }

                val userData = UserData(
                    nombre = nombre,
                    apellidoPaterno = apellidoPaterno,
                    apellidoMaterno = apellidoMaterno,
                    diaNacimiento = dia.toIntOrNull() ?: 0,
                    mesNacimiento = mes.toIntOrNull() ?: 0,
                    anioNacimiento = anio.toIntOrNull() ?: 0,
                    sexo = selectedSexo
                )

                userViewModel.setUserData(userData)
                coroutineScope.launch {
                    if (userId != null) {
                        repository.saveUserData(userId, userData)
                        navController.navigate("Examen") {
                            popUpTo("Formulario") { inclusive = true }
                        }
                    } else {
                        errorMessage = "Usuario no autenticado"
                    }
                }
            },
            modifier = Modifier.weight(1f)
        ) {
            Text("Siguiente")
        }
    }

    errorMessage?.let { err ->
        Spacer(modifier = Modifier.height(8.dp))
        Text(
            text = err,
            color = MaterialTheme.colorScheme.error
        )
    }
}
```

# MainActivity

```
package com.example.examensegundoparcial
```

```
import ExamenTheme
```

```

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.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.ui.Modifier
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.compose.rememberNavController
import com.example.examensegundoparcial.navigation.AppNavigation
import com.google.firebase.Firebase
import com.google.firebase.initializeApp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Firebase.initializeApp(this)
        enableEdgeToEdge()
        setContent {
            ExamenTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    val navController = rememberNavController()
                    val repository = FirebaseRepository()
                    val userViewModel: UserViewModel = viewModel()
                    AppNavigation(navController = navController, repository = repository, userViewModel =
userViewModel)
                }
            }
        }
    }
}

```

## Navigation.kt

```

package com.example.examensegundoparcial.navigation

import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable

```

```
import com.example.examensegundoparcial.AuthScreen
import com.example.examensegundoparcial.ExamenScreen
import com.example.examensegundoparcial.FirebaseRepository
import com.example.examensegundoparcial.FormularioScreen
import com.example.examensegundoparcial.ResultadosScreen
import com.example.examensegundoparcial.UserViewModel

@Composable
fun AppNavigation(
    navController: NavHostController,
    repository: FirebaseRepository,
    userViewModel: UserViewModel
) {
    val isLoggedIn = repository.getCurrentUserId() != null
    val userViewModel: UserViewModel = viewModel()
    val userData = userViewModel.userData.collectAsState().value // Obtener datos
    actuales
    NavHost(
        navController = navController,
        startDestination = if (isLoggedIn) Screens.Formulario.name else
        Screens.Auth.name
    ) {
        composable(Screens.Auth.name) {
            AuthScreen(
                navController = navController,
                repository = repository
            )
        }
        composable(Screens.Formulario.name) {
            FormularioScreen(
                navController = navController,
                repository = repository,
                userViewModel = userViewModel
            )
        }
        composable(Screens.Examen.name) {
```

```

        ExamenScreen(
            navController = navController,
            repository = repository
        )
    }

    composable("${Screens.Resultados.name}/{calificacion}") { backStackEntry ->
        val calificacion = backStackEntry.arguments?.getString("calificacion")?.toInt() ?: 0
        val userData = userViewModel.userData.collectAsState().value // ✅ obtener datos
        ResultadosScreen(
            navController = navController,
            userData = userData,
            repository = repository,
            calificacion = calificacion
        )
    }
}

enum class Screens {
    Auth,
    Formulario,
    Examen,
    Resultados
}

```

## ResultadosScreen.kt

```

package com.example.examensegundoparcial
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp

```

```
import androidx.navigation.NavHostController
import java.util.*

@Composable
fun ResultadosScreen(
    calificacion: Int,
    repository: FirebaseRepository,
    navController: NavHostController,
    userData: UserData,
) {
    val edad = calcularEdad(userData.diaNacimiento, userData.mesNacimiento,
    userData.anioNacimiento)
    val signoZodiacal = determinarSignoZodiacalChino(userData.anioNacimiento)

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "Hola ${userData.nombre} ${userData.apellidoPaterno}
${userData.apellidoMaterno}",
            style = MaterialTheme.typography.headlineMedium
        )

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

        Text(
            text = "Tienes $edad años y tu signo zodiacal chino es:",
            style = MaterialTheme.typography.bodyLarge
        )

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

        Image(
```

```
painter = painterResource(id = getSignoZodiacalImage(signoZodiacal)),
contentDescription = signoZodiacal,
modifier = Modifier.size(100.dp)
)

Text(
    text = signoZodiacal,
    style = MaterialTheme.typography.headlineSmall
)

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

Text(
    text = "Calificación: $calificacion",
    style = MaterialTheme.typography.headlineMedium
)
Button(
    onClick = {
        navController.navigate("Formulario") {
            popUpTo("Resultados") { inclusive = true } // Limpia la pila hasta
        "Resultados"
        }
    },
    modifier = Modifier.padding(top = 24.dp)
){
    Text("Volver al Formulario")
}
}

// Cálculo de edad considerando si ya pasó el cumpleaños en este año
fun calcularEdad(dia: Int, mes: Int, anio: Int): Int {
    val hoy = Calendar.getInstance()
    val cumple = Calendar.getInstance().apply {
        set(Calendar.YEAR, hoy.get(Calendar.YEAR))
        set(Calendar.MONTH, mes - 1)
        set(Calendar.DAY_OF_MONTH, dia)
```

```

    }

    var edad = hoy.get(Calendar.YEAR) - anio
    if (hoy.before(cumple)) edad--
    return edad
}

// Signo zodiacal chino según el año de nacimiento
fun determinarSignoZodiacalChino(anio: Int): String {
    val signos = listOf(
        "Mono", "Gallo", "Perro", "Cerdo", "Rata", "Buey",
        "Tigre", "Conejo", "Dragón", "Serpiente", "Caballo", "Cabra"
    )
    return signos[anio % 12]
}

// Obtiene el recurso de imagen según el signo zodiacal chino
fun getSignoZodiacalImage(signo: String): Int {
    return when (signo) {
        "Rata" -> R.drawable.rata
        "Buey" -> R.drawable.buey
        "Tigre" -> R.drawable.tigre
        "Conejo" -> R.drawable.conejo
        "Dragón" -> R.drawable.dragon
        "Serpiente" -> R.drawable.serpiente
        "Caballo" -> R.drawable.caballo
        "Cabra" -> R.drawable.cabra
        "Mono" -> R.drawable.mono
        "Gallo" -> R.drawable.gallo
        "Perro" -> R.drawable.perro
        "Cerdo" -> R.drawable.cerdo
        else -> R.drawable.unknown // Por si no coincide
    }
}

```

## UserViewModel.kt

```
package com.example.examensegundoparcial

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class UserViewModel : ViewModel() {
    private val _userData = MutableStateFlow(UserData())
    val userData: StateFlow<UserData> = _userData

    fun setUserData(user: UserData) {
        _userData.value = user
    }
}

data class UserData(
    val nombre: String = "",
    val apellidoPaterno: String = "",
    val apellidoMaterno: String = "",
    val diaNacimiento: Int = 0,
    val mesNacimiento: Int = 0,
    val anioNacimiento: Int = 0,
    val sexo: String = ""
)
```

## Capturas de la Aplicacion

11:40

3.48 kB/s

11:41

0.00 kB/s

8

6

4

3

Hola Angel Puga Barrientos

2. ¿Qué protocolo se utiliza para asignar direcciones IP automáticamente?

Tienes 20 años y tu signo zodiacal chino es:



Mono

Calificación: 30

[Volver al Formulario](#)

3. ¿Cuál es el protocolo utilizado para acceder a páginas web?

HTTP

IP

SSH

11:40 0.15 kb/s 🔍 ⚡ ⌂ 🔋

# Examen de Redes

1. ¿Cuál es la suma de  $2 + 2$ ?

8

6

4

3

---

19. ¿Qué puerto se usa para SSH?

23

80

443

22

---

20. ¿Qué tipo de red conecta dispositivos dentro de un edificio o zona local?

WAN

MAN

LAN

PAN

---

2. ¿Qué protocolo se utiliza para asignar direcciones IP automáticamente?

DNS

TCP

DHCP

FTP

---

3. ¿Cuál es el protocolo utilizado para acceder a páginas web?

HTTP

IP

[Terminar Examen](#)

11:40 G ...

0.00 kB/s



## Formulario de Datos Personales

Nombre(s) —

Angel

Apellido Paterno —

Puga

Apellido Materno —

Barrientos

Fecha de Nacimiento

Día —

2

Mes —

8

Año —

2004

Sexo



Masculino



Femenino



Otro

Limpiar

Siguiente

# Firebase

ExamenSegundoParcial ▾ Cloud Firestore

Vista del panel Compilador de consultas

Ubicación de la base de datos: nam5

The screenshot shows the Firebase Cloud Firestore interface. The left sidebar lists collections: exam\_results and users. Under users, there is a document with the ID FsXW9lsKpCQl5cD2Yjh38pMoj012. The document details are as follows:

- anioNacimiento: 2004
- apellidoMaterno: "Barrientos"
- apellidoPaterno: "Puga"
- diaNacimiento: 2
- mesNacimiento: 8
- nombre: "Angel"
- sexo: "Masculino"

(cadena)

ExamenSegundoParcial ▾ Cloud Firestore

Vista del panel Compilador de consultas

Ubicación de la base de datos: nam5

The screenshot shows the Firebase Cloud Firestore interface. The left sidebar lists collections: exam\_results and users. Under exam\_results, there is a document with the ID FsXW9lsKpCQl5cD2Yjh38pMoj012. The document details are as follows:

- calificacion: 30
- fecha: "04/07/2025"

responses

- 0 2
- 1 1
- 2 1
- 3 1
- 4 2
- 5 1
- 6 0
- 7 2
- 8 0
- 9 1