



UNIVERSIDAD TECNOLÓGICA DE LEÓN

INGENIERIA EN DESARROLLO Y GESTION DE SOFTWARE

Desarrollo para dispositivos inteligentes

Examen Parcial 2

presenta:

Perez Isaguirre Jose de Jesus



IDGS901

Fecha: 02/Jul/2025

Tabla de contenido

PANTALLAS DE LA APLICACIÓN	3
MAINACTIVITY	5
MAINSCREEN.....	6
NAVEGACION APP.....	11
PREGUNTAS	12
EXAMEN SCREEN.....	12
RESULTADOS SCREEN	17
RESUMEN SCREEN.....	21

PANTALLAS DE LA APLICACIÓN

Datos Personales

Nombre

Abigail

Apaterno

Mayorga

Amaterno

Rios

Fecha de nacimiento

Día

09

Mes

06

Año

2004

Sexo

☐ Masculino

☒ Femenino

Limpiar

Siguiente

Examen

¿Cuál es la suma de $2 + 2$?

☐ a) 8

☐ b) 6

☒ c) 4

☐ d) 3

¿Cuál es la resta de $10 - 3$?

☐ a) 8

☒ b) 7

☐ c) 5

☐ d) 9

¿Cuál es la multiplicación de 3×3 ?

☐ a) 7

☐ b) 1

☒ c) 9

☐ d) 4

¿Cuál es la división de $20 \div 5$?

☐ a) 2

☐ b) 3

☐ c) 5

☒ d) 4

¿Cuántos días tiene una semana?

☐ a) 5

☐ b) 6

☒ c) 7

☐ d) 8

¿Cual es la raiz cuadrada de 4?

☒ a) 2

☐ b) 4

☐ c) 3

☐ d) 8

Ver Resultados

Resultados

¿Cuál es la suma de $2 + 2$?

- ☐ a) 8 ☐ b) 6 ☒ c) 4 ☐ d) 3

¿Cuál es la resta de $10 - 3$?

- ☐ a) 8 ☒ b) 7 ☐ c) 5 ☐ d) 9

¿Cuál es la multiplicación de 3×3 ?

- ☐ a) 7 ☐ b) 1 ☒ c) 9 ☐ d) 4

¿Cuál es la división de $20 \div 5$?

- ☐ a) 2 ☐ b) 3 ☐ c) 5 ☒ d) 4

¿Cuántos días tiene una semana?

- ☐ a) 5 ☐ b) 6 ☒ c) 7 ☐ d) 8

¿Cuál es la raíz cuadrada de 4?

- ☒ a) 2 ☐ b) 4 ☐ c) 3 ☐ d) 8

[Ver información](#)

Hola Abigail Mayorga Rios

Tienes 21 años y tu signo zodiacal es Mono



Calificación: 10 / 10

[Terminar](#)

MAINACTIVITY

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            ExamenParcial2Theme {  
                val navController = rememberNavController()  
                AppNavigation(navController)  
            }  
        }  
    }  
}
```

@Composable

```
fun Greeting(name: String, modifier: Modifier = Modifier) {  
    Text(  
        text = "Hello $name!",  
        modifier = modifier  
    )  
}
```

@Preview(showBackground = true)

@Composable

```
fun GreetingPreview() {
```

```

ExamenParcial2Theme {
    Greeting("Android")
}
}

```

MAINSCREEN

```

@Composable
fun MainScreen(navController: NavController) {
    val context = LocalContext.current

    var nombre by remember { mutableStateOf("") }
    var apaterno by remember { mutableStateOf("") }
    var amaterno by remember { mutableStateOf("") }
    var dia by remember { mutableStateOf("") }
    var mes by remember { mutableStateOf("") }
    var anio by remember { mutableStateOf("") }
    var sexo by remember { mutableStateOf("") }

    Column(
        modifier = Modifier
            .padding(16.dp)
            .fillMaxSize(),
        verticalArrangement = Arrangement.Top
    ) {
        Text("Datos Personales", fontSize = 20.sp)
    }
}

```

```
OutlinedTextField(  
    value = nombre,  
    onChange = { nombre = it },  
    label = { Text("Nombre") }  
)  
  
OutlinedTextField(  
    value = apaterno,  
    onChange = { apaterno = it },  
    label = { Text("Apaterno") }  
)  
  
OutlinedTextField(  
    value = amaterno,  
    onChange = { amaterno = it },  
    label = { Text("Amaterno") }  
)
```

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

```
Text("Fecha de nacimiento")
```

```
Row {  
    OutlinedTextField(  
        value = dia,  
        onChange = { dia = it },  
        label = { Text("Día") },
```

```

        keyboardOptions      =      KeyboardOptions(keyboardType      =
KeyboardType.Number),

        modifier = Modifier

            .weight(1f)

            .padding(end = 4.dp)
    )

    OutlinedTextField(

        value = mes,

        onChange = { mes = it },

        label = { Text("Mes") },

        keyboardOptions      =      KeyboardOptions(keyboardType      =
KeyboardType.Number),

        modifier = Modifier

            .weight(1f)

            .padding(horizontal = 4.dp)
    )

    OutlinedTextField(

        value = anio,

        onChange = { anio = it },

        label = { Text("Año") },

        keyboardOptions      =      KeyboardOptions(keyboardType      =
KeyboardType.Number),

        modifier = Modifier

            .weight(1f)

            .padding(start = 4.dp)

```



```
)  
}
```

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

```
Text("Sexo")
```

```
Row {
```

```
    RadioButton(  
        selected = sexo == "Masculino",  
        onClick = { sexo = "Masculino" }  
    )
```

```
        selected = sexo == "Masculino",
```

```
        onClick = { sexo = "Masculino" }  
    )
```

```
Text("Masculino", modifier = Modifier.padding(end = 8.dp))
```

```
RadioButton(  
        selected = sexo == "Femenino",  
        onClick = { sexo = "Femenino" }  
    )
```

```
        selected = sexo == "Femenino",
```

```
        onClick = { sexo = "Femenino" }  
    )
```

```
Text("Femenino")
```

```
}
```

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

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

```
    horizontalArrangement = Arrangement.SpaceBetween,
```

```
    modifier = Modifier.fillMaxWidth()  
){
```

```
    Button(onClick = {
```

```
nombre = ""  
apaterno = ""  
amaterno = ""  
dia = ""  
mes = ""  
anio = ""  
sexo = ""  
}) {  
    Text("Limpiar")  
}
```

```
Button(onClick = {  
    val datos = ""  
        Nombre: $nombre  
        Apaterno: $apaterno  
        Amaterno: $amaterno  
        Fecha de nacimiento: $dia/$mes/$anio  
        Sexo: $sexo  
    """.trimIndent()  
  
    CoroutineScope(Dispatchers.IO).launch {  
        guardarEnArchivo(context, datos)  
    }  
})
```

```

        navController.navigate("examen")
    }) {
        Text("Siguiente")
    }
}
}
}
}
}

```

```

fun guardarEnArchivo(context: Context, contenido: String) {
    val file = File(context.filesDir, "informacion.txt")
    file.writeText(contenido)
}

```

NAVEGACION APP

@Composable

```

fun AppNavigation(navController: NavHostController) {
    NavHost(
        navController = navController,
        startDestination = "main"
    ) {
        composable("main") { MainScreen(navController) }
        composable("examen") { ExamenScreen(navController) }
        composable("resultados") { ResultadosScreen(navController) }
        composable("resumen") { ResumenScreen() }
    }
}

```

```
}  
}
```

PREGUNTAS

```
package org.utl.examenparcial2
```

```
data class Preguntas(  
    val enunciado: String,  
    val opciones: List<String>,  
    val respuestaCorrecta: Int  
)
```

EXAMEN SCREEN

```
val preguntas = listOf(  
    Preguntas(  
        "¿Cuál es la suma de 2 + 2?",  
        listOf("a) 8", "b) 6", "c) 4", "d) 3"),  
        2  
    ),  
    Preguntas(  
        "¿Cuál es la resta de 10 - 3?",  
        listOf("a) 8", "b) 7", "c) 5", "d) 9"),  
        1  
    )
```

),

Preguntas(

"¿Cuál es la multiplicación de 3×3 ?",

listOf("a) 7", "b) 1", "c) 9", "d) 4"),

2

),

Preguntas(

"¿Cuál es la división de $20 \div 5$?",

listOf("a) 2", "b) 3", "c) 5", "d) 4"),

3

),

Preguntas(

"¿Cuántos días tiene una semana?",

listOf("a) 5", "b) 6", "c) 7", "d) 8"),

2

),

Preguntas(

"¿Cual es la raiz cuadrada de 4?",

listOf("a) 2", "b) 4", "c) 3", "d) 8"),

0

)

)

@Composable

```

fun ExamenScreen(navController: NavController) {

    val respuestasUsuario = remember {

        mutableStateListOf<Int?>().apply {

            repeat(preguntas.size) { add(null) }

        }

    }
}

```

```

Column(

    modifier = Modifier

        .fillMaxSize()

        .padding(16.dp)

) {

    Text(

        "Examen",

        fontSize = 24.sp,

        modifier = Modifier.align(Alignment.CenterHorizontally)

    )
}

```

```

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

```

```

LazyColumn(

    modifier = Modifier

        .weight(1f)

        .fillMaxWidth(),

```

```

        verticalArrangement = Arrangement.spacedBy(12.dp)
    ){
        itemsIndexed(preguntas) { index, pregunta ->
            Card(
                modifier = Modifier.fillMaxWidth(),
                elevation = CardDefaults.cardElevation(4.dp)
            ){
                Column(modifier = Modifier.padding(12.dp)) {
                    Text(pregunta.enunciado, fontSize = 18.sp)
                    Spacer(modifier = Modifier.height(8.dp))

                    Row(
                        horizontalArrangement = Arrangement.spacedBy(12.dp),
                        verticalAlignment = Alignment.CenterVertically
                    ){
                        pregunta.opciones.forEachIndexed { i, opcion ->
                            Row(
                                verticalAlignment = Alignment.CenterVertically
                            ){
                                RadioButton(
                                    selected = respuestasUsuario[index] == i,
                                    onClick = { respuestasUsuario[index] = i }
                                )
                                Text(opcion, fontSize = 14.sp)
                            }
                        }
                    }
                }
            }
        }
    }
}

```

$$\left\{ \begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array} \right\}$$

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

```
Button(
    onClick = {
        navController.currentBackStackEntry
            ?.savedStateHandle
            ?.set("respuestasUsuario", ArrayList(respuestasUsuario))
        navController.navigate("resultados")
    },
    modifier = Modifier
        .align(Alignment.CenterHorizontally)
        .padding(bottom = 8.dp)
) {
    Text("Ver Resultados")
}
```



```
}
```

RESULTADOS SCREEN

```
@Composable
```

```
fun ResultadosScreen(navController: NavController) {
```

```
    val respuestasUsuario = navController.previousBackStackEntry
```

```
        ?.savedStateHandle
```

```
        ?.get<List<Int?>>("respuestasUsuario") ?: emptyList()
```

```
    val respuestasCorrectas = preguntas.zip(respuestasUsuario).count { (pregunta,  
    respuestaUsuario) ->
```

```
        pregunta.respuestaCorrecta == respuestaUsuario
```

```
}
```

```
val calificacion = (respuestasCorrectas * 10) / preguntas.size
```

```
val context = LocalContext.current
```

```
Column(
```

```
    modifier = Modifier
```

```
        .fillMaxSize()
```

```
        .padding(16.dp)
```

```
) {
```

```
    Text(
```

```
    "Resultados",  
    fontSize = 24.sp,  
    modifier = Modifier.align(Alignment.CenterHorizontally)  
  )
```

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

```
LazyColumn(  
    modifier = Modifier  
        .weight(1f)  
        .fillMaxWidth(),  
    verticalArrangement = Arrangement.spacedBy(12.dp)  
) {
```

```
    itemsIndexed(preguntas) { index, pregunta ->
```

```
        Card(  
            modifier = Modifier.fillMaxWidth(),  
            elevation = CardDefaults.cardElevation(4.dp)  
        ) {
```

```
            Column(modifier = Modifier.padding(12.dp)) {
```

```
                Text(pregunta.enunciado, fontSize = 18.sp)
```

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

```
                Row(  
                    horizontalArrangement = Arrangement.spacedBy(12.dp),
```

```

        verticalAlignment = Alignment.CenterVertically
    ){
        pregunta.opciones.forEachIndexed { i, opcion ->
            val esCorrecta = i == pregunta.respuestaCorrecta
            val respuestaUsuario = respuestasUsuario.getOrNull(index)
            val esSeleccionado = i == respuestaUsuario

            val backgroundColor = when {
                esSeleccionado && esCorrecta -> Color(0xFFB9FBC0)
                esSeleccionado && !esCorrecta -> Color(0xFFFFFADAD)
                else -> Color.Transparent
            }

            Row(
                verticalAlignment = Alignment.CenterVertically,
                modifier = Modifier
                    .background(backgroundColor,
RoundedCornerShape(4.dp))
            ){
                RadioButton(
                    selected = esSeleccionado,
                    onClick = {},
                    enabled = false
                )
                Text(opcion, fontSize = 14.sp)
            }
        }
    }
}

```

```

        }
    }
}
}
}
}
}
}
}

```

```

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

```

```

Button(
    onClick = {
        CoroutineScope(Dispatchers.IO).launch {
            guardarCalificacion(context, calificacion)
        }
        navController.navigate("resumen")
    },
    modifier = Modifier
        .align(Alignment.CenterHorizontally)
        .padding(bottom = 8.dp)
) {
    Text("Ver información")
}
}

```

```
}
```

```
fun guardarCalificacion(context: Context, calificacion: Int) {  
    val file = File(context.filesDir, "calificacion.txt")  
    file.writeText(calificacion.toString())  
}
```

RESUMEN SCREEN

@Composable

```
fun ResumenScreen() {  
    val context = LocalContext.current  
  
    var nombreCompleto by remember { mutableStateOf("") }  
    var edad by remember { mutableStateOf(0) }  
    var signoZodiacal by remember { mutableStateOf("") }  
    var calificacion by remember { mutableStateOf(0) }  
    var imagenSigno by remember { mutableStateOf(R.drawable.default) }  
  
    LaunchedEffect(Unit) {  
        withContext(Dispatchers.IO) {  
            val datosPersonales = leerInformacionArchivo(context)  
            val calif = leerCalificacionArchivo(context)  
            val nombre = datosPersonales["Nombre"] ?: ""  
            val apaterno = datosPersonales["Apaterno"] ?: ""
```

```
val amaterno = datosPersonales["Amaterno"] ?: ""  
val fechaNacimiento = datosPersonales["Fecha"] ?: ""  
val anioNacimiento = fechaNacimiento.split("/").getOrNull(2)?.toIntOrNull()  
?: 2000
```

```
val edadCalculada = calcularEdad(anioNacimiento)  
val signo = obtenerSignoZodiacalChino(anioNacimiento)  
val imagen = obtenerImagenDelSigno(signo)
```

```
withContext(Dispatchers.Main) {  
    nombreCompleto = "$nombre $apaterno $amaterno"  
    edad = edadCalculada  
    signoZodiacal = signo  
    calificacion = calif  
    imagenSigno = imagen  
}  
}  
}
```

```
Card(  
    modifier = Modifier  
        .fillMaxSize()  
        .padding(16.dp),  
    elevation = CardDefaults.cardElevation(8.dp)  
) {
```

```

Column(
    modifier = Modifier
        .fillMaxSize()
        .padding(16.dp),
    verticalArrangement = Arrangement.spacedBy(20.dp)
) {
    Text(text = "Hola $nombreCompleto", fontSize = 22.sp)

    Text(text = "Tienes $edad años y tu signo zodiacal es $signoZodiacal",
fontSize = 18.sp)

    Image(
        painter = painterResource(id = imagenSigno),
        contentDescription = "Signo Zodiacal Chino",
        modifier = Modifier
            .size(150.dp)
            .align(Alignment.CenterHorizontally)
    )

    Text(
        text = "Calificación: $calificacion / 10",
        fontSize = 24.sp,
        modifier = Modifier.align(Alignment.CenterHorizontally)
    )

    Button(
        onClick = {
            borrarArchivos(context)
        },

```

```

        modifier = Modifier.align(Alignment.CenterHorizontally)
    ) {
        Text("Terminar")
    }
}
}
}

```

```

suspend fun leerInformacionArchivo(context: Context): Map<String, String> {
    val file = File(context.filesDir, "informacion.txt")
    if (!file.exists()) return emptyMap()

    val contenido = file.readText()
    val lines = contenido.lines()
    val datos = mutableMapOf<String, String>()

    lines.forEach { line ->
        when {
            line.startsWith("Nombre:") -> datos["Nombre"] =
line.removePrefix("Nombre:").trim()
            line.startsWith("Apaterno:") -> datos["Apaterno"] =
line.removePrefix("Apaterno:").trim()
            line.startsWith("Amaterno:") -> datos["Amaterno"] =
line.removePrefix("Amaterno:").trim()

```



```

        line.startsWith("Fecha de nacimiento:") -> datos["Fecha"] =
line.removePrefix("Fecha de nacimiento:").trim()

        line.startsWith("Sexo:") -> datos["Sexo"] = line.removePrefix("Sexo:").trim()

    }

}

return datos

}

```

```

suspend fun leerCalificacionArchivo(context: Context): Int {

    val file = File(context.filesDir, "calificacion.txt")

    if (!file.exists()) return 0

    return file.readText().trim().toIntOrNull() ?: 0

}

```

```

fun calcularEdad(anioNacimiento: Int): Int {

    val anioActual = Calendar.getInstance().get(Calendar.YEAR)

    return anioActual - anioNacimiento

}

```

```

fun obtenerSignoZodiacalChino(anioNacimiento: Int): String {

    val signos = listOf(

        "Rata", "Buey", "Tigre", "Conejo", "Dragón", "Serpiente",

        "Caballo", "Cabra", "Mono", "Gallo", "Perro", "Cerdo"

    )

```

```
val index = (anioNacimiento - 1900) % 12  
return signos[index]  
}
```

```
fun obtenerImagenDelSigno(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.default  
    }  
}
```

```
fun borrarArchivos(context: Context) {  
    val infoFile = File(context.filesDir, "informacion.txt")
```

```
val califFile = File(context.filesDir, "calificacion.txt")
```

```
if (infoFile.exists()) infoFile.writeText("")
```

```
if (califFile.exists()) califFile.writeText("")
```

```
}
```