

**UNIVERSIDAD DON BOSCO  
DESARROLLO DE SOFTWARE PARA  
MOVILES DSM941**



**MANUAL TECNICO FASE 3 DEL PROYECTO  
Categoría GRUPAL**

Grupo 01T

**INTEGRANTES:**

Apellidos	Nombres	Carnet
Cabezas Vaquero	Gerardo Antonio	CV152055
Pineda González	Gabriela María	PG120866
Escalante Guardad	Jonathan Baltazar	EG161132
Argueta Rivas	William Alexis	AR160955

**DOCENTE:  
Ing. Alexander Alberto Sigüenza Campos**

**FECHA DE PRESENTACIÓN:  
10/12/2023**

## Índice

<b>Introducción.....</b>	3
<b>App Veterinaria Santa Barbara.....</b>	3
<b>Requisitos de Hardware y Software.....</b>	3
<b>Instalación y Pantalla Principal.....</b>	4
<b>Tecnologías utilizadas.....</b>	5
<b>Pantallas y Código fuente del desarrollo.....</b>	6

## **Introducción.**

Este manual se realizó con el fin de detallar las partes técnicas del sistema, requerimientos y otros datos técnicos para entender la lógica de la aplicación ‘Sistema de Citas Médicas Veterinaria Santa Barbara’.

## **App Veterinaria Santa Barbara.**

Esta app se desarrolló con la finalidad de llevar un control más óptimo de todo el control de citas y datos de las mascotas, así como el control de usuarios y doctores.

La aplicación cuenta con las siguientes características:

- Un menú principal de todas las opciones
- Registros de datos de usuarios
- Registros de datos de doctores
- Registros de datos de doctores
- Registros de citas
- Mantenimiento de datos de los usuarios, doctores y mascotas
- Mantenimiento de todos los datos de las citas programadas
- Ver historial de citas

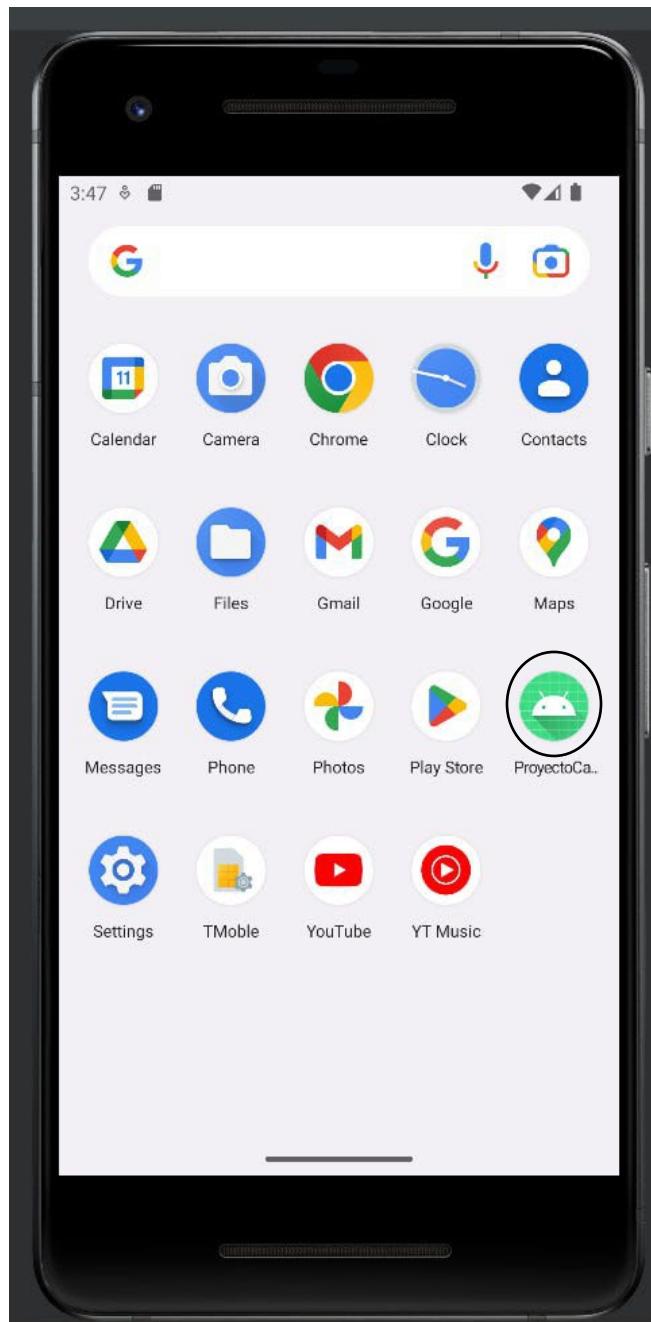
## **Requisitos de Hardware y Software.**

Los requisitos mínimos del dispositivo donde se instalará la aplicación deben ser los siguientes:

- Sistema Operativo Android 5 o superior.
- Memoria RAM mínima de 2GB y capacidad de almacenamiento de 50mb.

## Instalación y Pantalla Principal

Primeramente, generamos el archivo .apk y procedemos a ejecutarlo e instalarlo en el dispositivo.



Y así será la vista principal de la aplicación.



## Tecnologías utilizadas.

Se utilizo el IDE Android Studio empleando el lenguaje KOTLIN que es un lenguaje de programación estático de código abierto, este permite programación funcional y orientada a objetos, su sintaxis tiene similitud a otros lenguajes de programación como lo son C#, Java y Scala.

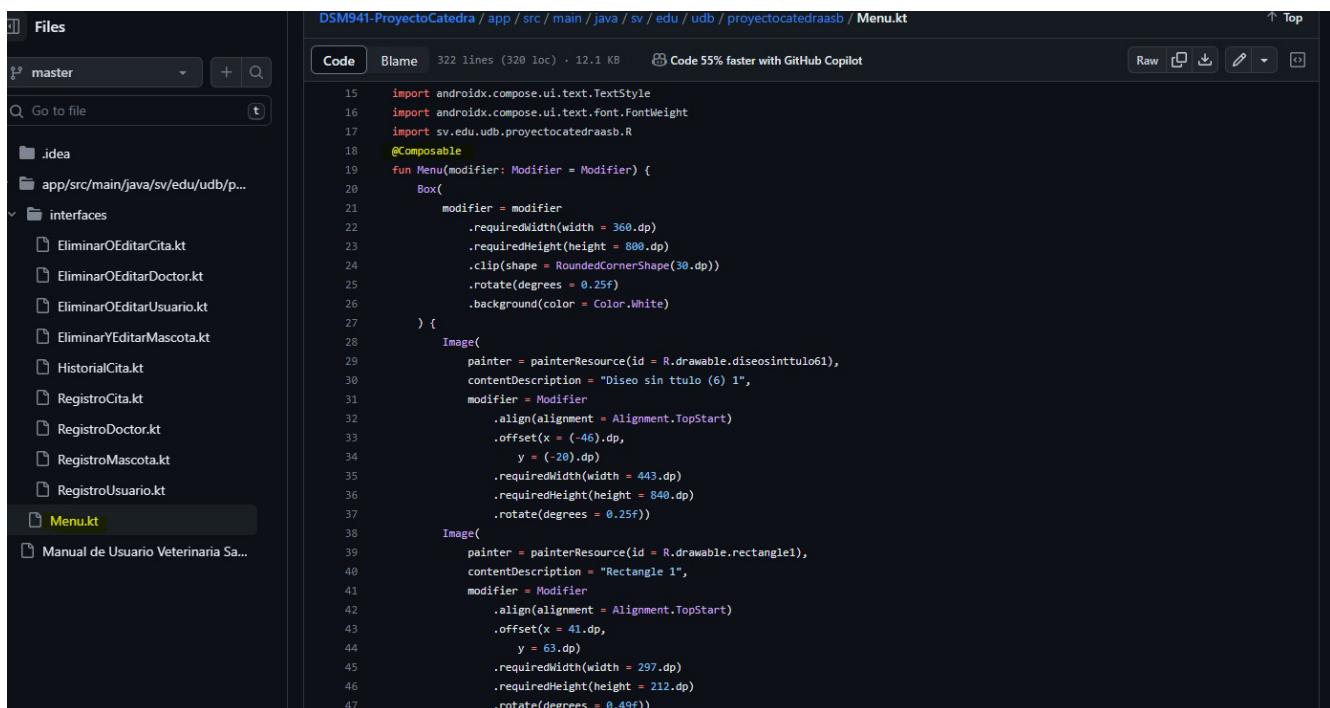
Se empleo el uso de una biblioteca llamada COMPOSE que pertenece a Android Studio el cual permite a los desarrolladores crear interfaces de usuario de manera más sencilla y eficiente en comparación con el enfoque tradicional basado en XML y Java.

Cada composable es una función que describe como se debe dibujar un elemento de la interfaz de usuario en respuesta a cambios en el estado o datos. Estas funciones componibles se pueden generar de manera jerárquica para construir

interfaces de usuarios un poco más complejas.

Siguiendo de manera más detallada y explicara un poco el código se mostraran las siguientes pantallas con el código fuente de cada parte desarrollada de la App.

## Pantallas y Código fuente del desarrollo.



The screenshot shows a GitHub repository interface for the project 'DSM941-ProyectoCatedra'. The repository path is 'app / src / main / java / sv / edu / udb / proyectocatedraasb / Menu.kt'. The code editor displays the 'Menu.kt' file, which contains Kotlin code for a Composable menu. The code uses the AndroidX Compose library to define a 'Box' with specific styling (width: 360.dp, height: 800.dp, rounded corners, rotation, white background) and two 'Image' components with painter resources and alignment offsets. The code is annotated with line numbers from 15 to 47.

```
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18 @Composable
19 fun Menu(modifier: Modifier = Modifier) {
20     Box(
21         modifier = modifier
22             .requiredWidth(width = 360.dp)
23             .requiredHeight(height = 800.dp)
24             .clip(shape = RoundedCornerShape(30.dp))
25             .rotate(degrees = 0.25f)
26             .background(color = Color.White)
27     ) {
28         Image(
29             painter = painterResource(id = R.drawable.diseosintitulo61),
30             contentDescription = "Diseño sin título (6) 1",
31             modifier = Modifier
32                 .align(alignment = Alignment.TopStart)
33                 .offset(x = (-46).dp,
34                         y = (-28).dp)
35                 .requiredWidth(width = 443.dp)
36                 .requiredHeight(height = 840.dp)
37                 .rotate(degrees = 0.25f))
38         Image(
39             painter = painterResource(id = R.drawable.rectangle1),
40             contentDescription = "Rectangle 1",
41             modifier = Modifier
42                 .align(alignment = Alignment.TopStart)
43                 .offset(x = 41.dp,
44                         y = 63.dp)
45                 .requiredWidth(width = 297.dp)
46                 .requiredHeight(height = 212.dp)
47                 .rotate(degrees = 0.49f))
    }
}
```

### Menú.

Código fuente donde se detalla la sintaxis utilizada para el desarrollo del menú principal, así como se dibuja cada elemento con 'x' tamaño y una posición, todo esto empleando la librería compose.

DSM941-ProyectoCatedra / app / src / main / java / sv / edu / udb / proyectocatedraasb / interfaces / RegistroUsuario.kt

**Code** Blame 244 lines (241 loc) · 8.57 KB Code 55% faster with GitHub Copilot

```

18
19     @Composable
20     fun RegistroUsuario(modifier: Modifier = Modifier) {
21         Box(
22             modifier = modifier
23                 .requiredWidth(width = 360.dp)
24                 .requiredHeight(height = 800.dp)
25                 .clip(shape = RoundedCornerShape(30.dp))
26                 .rotate(degrees = 0.25f)
27                 .background(color = Color.White)
28         ) {
29             Image(
30                 painter = painterResource(id = R.drawable.diseosintitulo61),
31                 contentDescription = "Diseo sin tulo (6) 1",
32                 modifier = Modifier
33                     .align(alignment = Alignment.TopStart)
34                     .offset(x = (-46).dp,
35                             y = (-28).dp)
36                     .requiredWidth(width = 443.dp)
37                     .requiredHeight(height = 840.dp)
38                     .rotate(degrees = 0.25f))
39             Image(
40                 painter = painterResource(id = R.drawable.rectangle1),
41                 contentDescription = "Rectangle 1",
42                 modifier = Modifier
43                     .align(alignment = Alignment.TopStart)
44                     .offset(x = 41.dp,
45                             y = 63.dp)
46                     .requiredWidth(width = 297.dp)
47                     .requiredHeight(height = 212.dp)
48                     .rotate(degrees = 0.49f))
49             Image(
50                 painter = painterResource(id = R.drawable.vector),
51                 contentDescription = "Vector",
52                 modifier = Modifier

```

DSM941-ProyectoCatedra / app / src / main / java / sv / edu / udb / proyectocatedraasb / interfaces / RegistroDoctor.kt

**Code** Blame 243 lines (241 loc) · 8.55 KB Code 55% faster with GitHub Copilot

```

14     import androidx.compose.material3.Text
15     import androidx.compose.ui.text.TextStyle
16     import androidx.compose.ui.text.font.FontWeight
17     import sv.edu.udb.proyectocatedraasb.R
18     @Composable
19     fun RegistroDoctor(modifier: Modifier = Modifier) {
20         Box(
21             modifier = modifier
22                 .requiredWidth(width = 360.dp)
23                 .requiredHeight(height = 800.dp)
24                 .clip(shape = RoundedCornerShape(30.dp))
25                 .rotate(degrees = 0.25f)
26                 .background(color = Color.White)
27         ) {
28             Image(
29                 painter = painterResource(id = R.drawable.diseosintitulo61),
30                 contentDescription = "Diseo sin tulo (6) 1",
31                 modifier = Modifier
32                     .align(alignment = Alignment.TopStart)
33                     .offset(x = (-46).dp,
34                             y = (-28).dp)
35                     .requiredWidth(width = 443.dp)
36                     .requiredHeight(height = 840.dp)
37                     .rotate(degrees = 0.25f))
38             Image(
39                 painter = painterResource(id = R.drawable.rectangle1),
40                 contentDescription = "Rectangle 1",
41                 modifier = Modifier
42                     .align(alignment = Alignment.TopStart)
43                     .offset(x = 41.dp,
44                             y = 63.dp)
45                     .requiredWidth(width = 297.dp)
46                     .requiredHeight(height = 212.dp)
47                     .rotate(degrees = 0.49f))
48             Image(

```

The screenshot shows the GitHub interface for the RegistroMascota.kt file. The file contains 252 lines of code, 8.87 KB in size, and is 55% faster than the previous version. The code uses Composable functions to draw a box with rounded corners, containing two images: one for 'Diseño sin título (6) 1' and another for 'Rectangle 1'. The code specifies dimensions like width=360.dp, height=800.dp, and rotation angles.

```

14 import androidx.compose.material3.Text
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.font.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18 @Composable
19 fun RegistroMascota(modifier: Modifier = Modifier) {
20     Box(
21         modifier = modifier
22             .requiredWidth(width = 360.dp)
23             .requiredHeight(height = 800.dp)
24             .clipShape = RoundedCornerShape(30.dp))
25             .rotate(degrees = 0.25f)
26             .background(color = Color.White)
27     ) {
28         Image(
29             painter = painterResource(id = R.drawable.diseosintitulo61),
30             contentDescription = "Diseño sin título (6) 1",
31             modifier = Modifier
32                 .align(alignment = Alignment.TopStart)
33                 .offset(x = (-46).dp,
34                         y = (-28).dp)
35                 .requiredWidth(width = 443.dp)
36                 .requiredHeight(height = 840.dp)
37                 .rotate(degrees = 0.25f))
38         Image(
39             painter = painterResource(id = R.drawable.rectangle1),
40             contentDescription = "Rectangle 1",
41             modifier = Modifier
42                 .align(alignment = Alignment.TopStart)
43                 .offset(x = 41.dp,
44                         y = 63.dp)
45                 .requiredWidth(width = 297.dp)
46                 .requiredHeight(height = 212.dp)
47                 .rotate(degrees = 0.49f))
48         Image(

```

The screenshot shows the GitHub interface for the RegistroCita.kt file. The file contains 244 lines of code, 8.57 KB in size, and is 55% faster than the previous version. The code is identical to the RegistroMascota.kt file, using Composable functions to draw a box with rounded corners, containing two images: one for 'Diseño sin título (6) 1' and another for 'Rectangle 1'. The code specifies dimensions like width=360.dp, height=800.dp, and rotation angles.

```

14 import androidx.compose.material3.Text
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.font.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18
19 @Composable
20 fun RegistroCita(modifier: Modifier = Modifier) {
21     Box(
22         modifier = modifier
23             .requiredWidth(width = 360.dp)
24             .requiredHeight(height = 800.dp)
25             .clipShape = RoundedCornerShape(30.dp))
26             .rotate(degrees = 0.25f)
27             .background(color = Color.White)
28     ) {
29         Image(
30             painter = painterResource(id = R.drawable.diseosintitulo61),
31             contentDescription = "Diseño sin título (6) 1",
32             modifier = Modifier
33                 .align(alignment = Alignment.TopStart)
34                 .offset(x = (-46).dp,
35                         y = (-28).dp)
36                 .requiredWidth(width = 443.dp)
37                 .requiredHeight(height = 840.dp)
38                 .rotate(degrees = 0.25f))
39         Image(
40             painter = painterResource(id = R.drawable.rectangle1),
41             contentDescription = "Rectangle 1",
42             modifier = Modifier
43                 .align(alignment = Alignment.TopStart)
44                 .offset(x = 41.dp,
45                         y = 63.dp)
46                 .requiredWidth(width = 297.dp)
47                 .requiredHeight(height = 212.dp)
48                 .rotate(degrees = 0.49f))

```

Estas pantallas llevan el código fuente para los menús de creación de usuarios, doctores, mascotas y citas, se muestra de igual forma como se crean y dibujan cada elemento en la pantalla, tamaño de la caja de texto y posiciones de los demás elementos.

The screenshot shows the Android Studio interface with the code editor open to the file `EliminarOEditarUsuario.kt`. The code is a Kotlin function that defines a Composable function named `EliminarOEditarUsuario`. This function creates a `Box` with specific styling (width, height, rounded corners, rotation, background color) and contains two `Image` components. Each `Image` has its own set of properties like alignment, offset, required width, required height, and rotation. The code is annotated with `@Composable` and uses `Modifier` to apply styles.

```
Code 55% faster with GitHub Copilot
```

```
14 import androidx.compose.material3.Text
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.font.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18
19 @Composable
20 fun EliminarOEditarUsuario(modifier: Modifier = Modifier) {
21     Box(
22         modifier = modifier
23             .requiredWidth(width = 360.dp)
24             .requiredHeight(height = 800.dp)
25             .clip(shape = RoundedCornerShape(30.dp))
26             .rotate(degrees = 0.25f)
27             .background(color = Color.White)
28     ) {
29         Image(
30             pointer = painterResource(id = R.drawable.diseosintitulo6),
31             contentDescription = "Diseño sin título (6) 1",
32             modifier = Modifier
33                 .align(alignment = Alignment.TopStart)
34                 .offset(x = (-46).dp,
35                         y = (-20).dp)
36                 .requiredWidth(width = 443.dp)
37                 .requiredHeight(height = 840.dp)
38                 .rotate(degrees = 0.25f))
39         Image(
40             pointer = painterResource(id = R.drawable.rectangle1),
41             contentDescription = "Rectangle 1",
42             modifier = Modifier
43                 .align(alignment = Alignment.TopStart)
44                 .offset(x = 41.dp,
45                         y = 63.dp)
46                 .requiredWidth(width = 297.dp)
47                 .requiredHeight(height = 212.dp))
```

The screenshot shows the Android Studio interface with the following details:

- Left Sidebar (Files):** Shows the project structure. The file `EliminarYEditarMascota.kt` is highlighted in yellow.
- Top Bar:** Displays the project name `DSM941-ProyectoCatedra`, the module `app`, the source set `src/main/java`, the package `sv.edu.udb.proyectocatedraasb`, and the file `interfaces/EliminarYEditarMascota.kt`. It also includes links for "Blame", "Code 55% faster with GitHub Copilot", and navigation icons.
- Code Editor:** The content of the `EliminarYEditarMascota.kt` file is displayed. The code uses Jetpack Compose to define two `Image` components with specific styles and positions.

```
Code Blame 268 lines (265 loc) · 9.43 KB Code 55% faster with GitHub Copilot

14 import androidx.compose.material3.Text
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.font.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18
19 @Composable
20 fun EliminarYEditarMascota(modifier: Modifier = Modifier) {
21     Box(
22         modifier = modifier
23             .requiredWidth(width = 360.dp)
24             .requiredHeight(height = 800.dp)
25             .clip(shape = RoundedCornerShape(30.dp))
26             .rotate(degrees = 0.25f)
27             .background(color = Color.White)
28     ) {
29         Image(
30             painter = painterResource(id = R.drawable.diseosintitulo61),
31             contentDescription = "Diseo sin tulo (6) 1",
32             modifier = Modifier
33                 .align(alignment = Alignment.TopStart)
34                 .offset(x = (-46).dp,
35                         y = (-20).dp)
36                 .requiredWidth(width = 443.dp)
37                 .requiredHeight(height = 840.dp)
38                 .rotate(degrees = 0.25f))
39         Image(
40             painter = painterResource(id = R.drawable.rectangle1),
41             contentDescription = "Rectangle 1",
42             modifier = Modifier
43                 .align(alignment = Alignment.TopStart)
44                 .offset(x = 41.dp,
45                         y = 63.dp)
46                 .requiredWidth(width = 297.dp)
47                 .requiredHeight(height = 212.dp)
48                 .rotate(degrees = 0.49f))
49     }
50 }
```

Aquí se muestra la parte de la edición o eliminación de datos de los usuarios, doctores mascotas y citas, siempre siguiendo ese orden en cada pantalla, se carga los datos en las cajas de texto y ya así permite editar la información o eliminarla.

```

14 import androidx.compose.material3.Text
15 import androidx.compose.ui.text.TextStyle
16 import androidx.compose.ui.text.font.FontWeight
17 import sv.edu.udb.proyectocatedraasb.R
18
19 @Composable
20 fun HistorialCita(modifier: Modifier = Modifier) {
21     Box(
22         modifier = modifier
23             .requiredWidth(width = 360.dp)
24             .requiredHeight(height = 800.dp)
25             .clipShape = RoundedCornerShape(30.dp)
26             .rotate(degrees = 0.25f)
27             .background(color = Color.White)
28     ) {
29         Image(
30             painter = painterResource(id = R.drawable.diseosintitulo61),
31             contentDescription = "Diseo sin tulo (6) 1",
32             modifier = Modifier
33                 .align(alignment = Alignment.TopStart)
34                 .offset(x = (-46).dp,
35                         y = (-20).dp)
36                 .requiredWidth(width = 443.dp)
37                 .requiredHeight(height = 840.dp)
38                 .rotate(degrees = 0.25f)
39         Image(
40             painter = painterResource(id = R.drawable.rectangle1),
41             contentDescription = "Rectangle 1",
42             modifier = Modifier
43                 .align(alignment = Alignment.TopStart)
44                 .offset(x = 41.dp,
45                         y = 63.dp)
46                 .requiredWidth(width = 297.dp)
47                 .requiredHeight(height = 212.dp)
48                 .rotate(degrees = 0.49f))

```

Y por último el código de la pantalla del historial donde carga toda la información registrada de cada cita y depende la información que cargue así se arma y dibuja los elementos en la pantalla del dispositivo.

```

49 Row(
50     modifier = Modifier
51         .fillMaxWidth()
52         .height(140.dp)
53         .background(Color(0xFFFF7BF0C))
54         .clickable { navController.navigate(route: "RegistroMascota") },
55 ) { this:RowScope
56     Column(
57         modifier = Modifier.weight(1f),
58         verticalArrangement = Arrangement.Center,
59         horizontalAlignment = Alignment.CenterHorizontally
60     ) { this:ColumnScope
61         Image(
62             painter = painterResource(id = R.drawable.pacientes),
63             contentDescription = null,
64             modifier = Modifier.size(80.dp)
65                 .padding(all = 8.dp),
66             contentScale = ContentScale.Fit
67         )
68         Spacer(modifier = Modifier.height(8.dp))
69         Text(
70             text = "Pacientes",
71             color = Color(0xFF492000),
72         )

```

Para detallar un poco más lo del menú cabe mencionar que esta creado con filas y columnas componibles donde se adecua su comportamiento.

```
@Composable
fun InicioDeSesion(modifier: Modifier = Modifier, navController: NavHostController, loginViewModel: LoginViewModel = viewModel()) {
    var usuario by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }

    Column(
        modifier = modifier.fillMaxWidth(),
        horizontalAlignment = Alignment.CenterHorizontally
            /*.requiredWidth(width = 360.dp)
            .requiredHeight(height = 800.dp)
            .clip(shape = RoundedCornerShape(30.dp))
            .rotate(degrees = 0.25f)
            .background(color = Color.White)*/
    ) { this@ColumnScope
        Image(
            painter = painterResource(id = R.drawable.Logo),
            contentDescription = "Logo",
            modifier = Modifier
                .align(alignment = Alignment.CenterHorizontally)
                .offset(x = 0.dp,
                    y = 0.dp)
                .requiredWidth(width = 200.dp)
        )
    }
}
```

Las variables están declaradas de manera global y no de sistema, mas que todo para el manejo más óptimo de la información.

```
1 package sv.edu.udb.proyectocatedraasb
2
3 import ...
4
5 class MainActivity : ComponentActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContent {
9             ProjetoCatedraAppTheme {
10                 // Create a NavController
11                 val navController = rememberNavController()
12
13                 NavHost(
14                     navController = navController,
15                     startDestination = "menu"
16                 ) { this: NavGraphBuilder
17                     composable(route: "login") { this: AnimatedContentScope<NavBackStackEntry>
18                         InicioDeSesion(navController = navController)
19                     }
20                     composable(route: "splash") { this: AnimatedContentScope<NavBackStackEntry>
21                         SplashScreen(navController = navController)
22                     }
23                     composable(route: "registroUsuario") { this: AnimatedContentScope<NavBackStackEntry>
24                         RegistroUsuario(navController = navController)
25                     }
26                 }
27             }
28         }
29     }
30 }
```

Tenemos una función MainActivity donde se define las variables de entorno para cada una de las funciones, de esta forma se facilita la navegación y se define de manera más clara que se va a cargar primero al ejecutar la aplicación.

```

55     private fun performLoginRequest(username: String, password: String): String {
56         val url = URL("http://localhost:8080/Login.php")
57         val connection = url.openConnection() as HttpURLConnection
58         connection.requestMethod = "POST"
59         connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")
60         connection.doOutput = true
61
62         val postData = "username=$username&password=$password"
63         val writer = OutputStreamWriter(connection.outputStream)
64         writer.write(postData)
65         writer.flush()
66
67         val response = StringBuilder()
68         try {
69             val reader = BufferedReader(InputStreamReader(connection.inputStream))
70             var line: String?
71             while (reader.readLine().also { line = it } != null) {
72                 response.append(line)
73             }
74         } finally {
75             connection.disconnect()
76         }
77         return response.toString()
78     }

```

Función donde se arma el mensaje SOAP y realiza la petición al WebService para validar la información.

```

class LoginViewModel : ViewModel() {
    private lateinit var navController: NavController;

    fun performLogin(username: String, password: String) {
        CoroutineScope(Dispatchers.IO).launch { thisCoroutineScope
            val response = performLoginRequest(username, password)
            // Handle the response here (e.g., validation, navigation)
            val jsonResponse = JSONObject(response)
            val loginSuccess = jsonResponse.getBoolean("login_success")
            if (loginSuccess) {
                navController.navigate("menu");
            } else {
                navController.navigate("login");
            }
        }
    }
}

```

Función para obtener la respuesta del login y se define que navegación continuara.

```

44 // Check the request method and perform actions accordingly
45 if ($_SERVER['REQUEST_METHOD'] === 'GET') {
46     if (isset($_GET['action']) && $_GET['action'] === 'getAnimals') {
47         header('Content-Type: application/json');
48         $animals = getAnimals();
49         echo json_encode($animals);
50     } else {
51         http_response_code(404);
52         echo json_encode(array("message" => "Invalid request."));
53     }
54 } elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
55
56     if (isset($_GET['action'])) {
57         switch ($_GET['action']) {
58             case 'insertAnimal':
59                 // Validate and sanitize input before using in the query
60                 $name = $_POST['nombre'] ?? '';
61                 $species = $_POST['tipo'] ?? '';
62                 $ownerID = $_POST['IDDuenio'] ?? 0;
63                 $medicalHistory = $_POST['HistoriaMMedica'] ?? '';

```

WebService donde se reciben las peticiones.