Universidad Central del Ecuador
nforme de Diseño App Básica

- **Hurtado Alexis**
- **Quinatoa John**
- **Rodríguez Gabriela**
- **Vergara Dario**

Nombre de la App: GACS Wheel

Llamamos a la aplicación GACS Wheel porque usamos las iniciales de los nombres que cada integrante del equipo prefiere y elegimos "Wheel" porque está relacionado con autos y movilidad, que es justamente el enfoque de la app. El nombre busca ser sencillo, fácil de recordar y conectado con el servicio que ofrecemos: la renta de autos.

Logo:

1. Introducción

El presente informe tiene como objetivo documentar el modelo de navegación implementado en la aplicación móvil “GACS Wheel”, desarrollada utilizando Jetpack Compose y el patrón de arquitectura MVVM (Model-View-ViewModel). La aplicación permite a los usuarios autenticarse, registrarse, acceder a un panel principal y gestionar vehículos a través de una interfaz gráfica moderna y adaptable.

2. Tipo de Navegación Utilizada

La aplicación hace uso del sistema de “navegación declarativa de Jetpack Compose”, a través de la biblioteca oficial ‘androidx.navigation:navigation-compose’. Este enfoque permite gestionar de forma explícita y modular los distintos destinos de la aplicación (pantallas), manteniendo un flujo de navegación claro, escalable y controlado desde un único punto.

3. Componentes Principales

3.1 ‘NavController’

Es el componente encargado de orquestar el flujo de navegación. Se declara dentro del ‘setContent’ de ‘MainActivity.kt’ mediante el hook ‘rememberNavController()’ y se pasa como argumento a las pantallas que requieren navegar entre vistas.

3.2 ‘NavHost’

Define el gráfico de navegación de la aplicación. Se encarga de asociar cada ruta con el composable correspondiente. Se implementa dentro de un 'Scaffold', el cual estructura el diseño de la interfaz de usuario principal.

3.3 Rutas ('Routes')

Cada pantalla está identificada mediante un string único que actúa como ruta. Estas rutas se definen de forma directa en el 'NavHost'.

4. Definición del Gráfico de Navegación

En 'MainActivity.kt' se define el gráfico de navegación con los siguientes destinos:

kotlin
kotlin
CopiarEditar

```
NavHost(
   navController = navController,
   startDestination = "login"
) {
   composable("login") {
      LoginScreen(navController)
   }
   composable("registro") {
      RegistroScreen(navController)
   }
   composable("inicio") {
      InicioScreen(navController)
   }
   composable("admin") {
      AdminVehiculoActivity(navController)
   }
}
```

Este gráfico establece como pantalla inicial el login y permite navegar hacia el registro, pantalla principal y panel administrativo.

5. Lógica de Navegación

La navegación entre pantallas se ejecuta utilizando el método 'navigate()' del 'NavController'. Por ejemplo:

kotlin

```
navController.navigate("inicio")
```

Para limpiar el back stack o regresar a una pantalla específica, se pueden utilizar configuraciones como:

kotlin

```
navController.navigate("admin") {
   popUpTo("agregar") { inclusive = true }
}
```

La navegación también puede condicionarse mediante validaciones de formulario, como en la pantalla de agregar vehículo, donde se verifica que los campos no estén vacíos antes de permitir el avance.

6. Validación y Control de Estados

En pantallas como 'AgregarVehiculoScreen', se valida que todos los campos requeridos estén correctamente completados antes de ejecutar acciones lógicas:

kotlin
```
if (placa.isBlank() || marca.isBlank() || anioInt == null || color.isBlank() || costoDouble == null) {
   Toast.makeText(context, "Completa todos los campos correctamente", Toast.LENGTH_SHORT).show()
} else {
   // Inserción del vehículo
}
```

Este enfoque evita la navegación errónea y mejora la estabilidad de la aplicación.

7. Estructura del Proyecto

El proyecto está organizado en los siguientes paquetes:

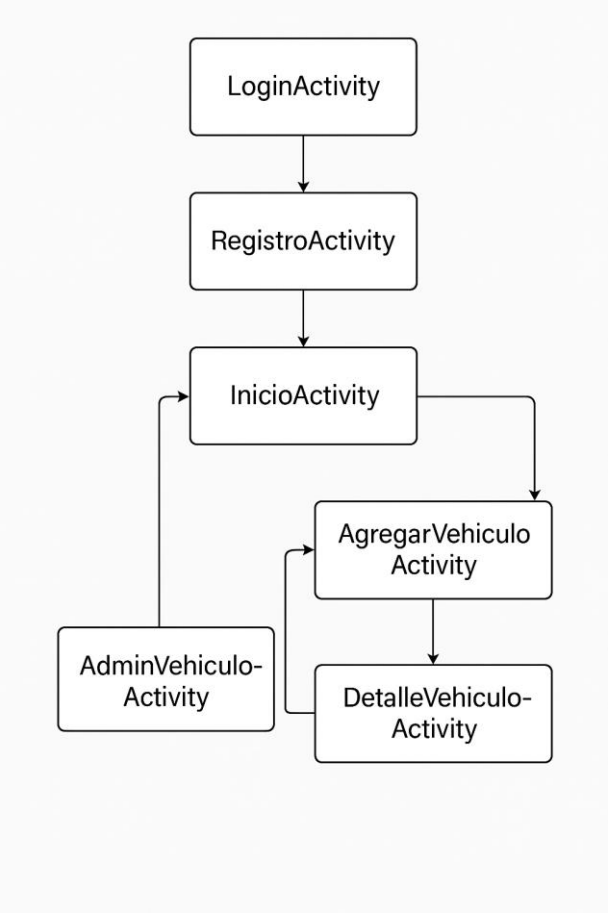
- 'View/': Contiene las interfaces gráficas ('LoginScreen', 'RegistroScreen', 'InicioScreen', 'AdminVehiculoActivity', etc.).
- 'Controller/': Incluye la lógica de manipulación de datos, como inserciones a base de datos.
- 'Model/': Contiene las definiciones de datos, como la clase 'Vehiculo'.

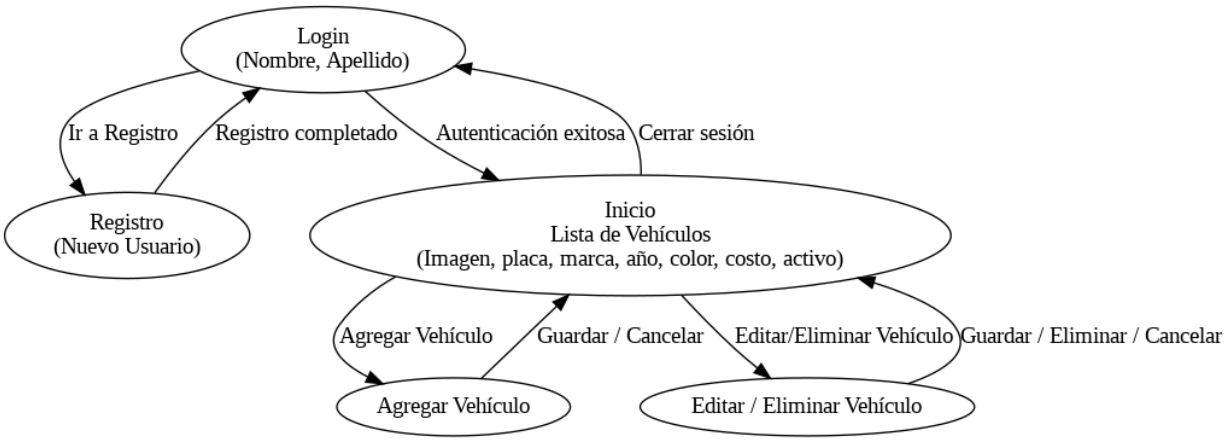
8. Conclusiones

El modelo de navegación implementado en GACS Wheel sigue las buenas prácticas de Jetpack Compose, favoreciendo una estructura clara, mantenible y extensible. La utilización de 'NavController' y 'NavHost' permite definir un flujo de pantallas controlado, mientras que la validación previa a la navegación garantiza la integridad de los datos y una experiencia de usuario consistente.

Se recomienda mantener el gráfico de navegación centralizado y considerar el uso de rutas con argumentos en futuras expansiones del sistema, para permitir el paso de datos entre pantallas de manera segura y flexible.

8. Diagrama de navegación de Activity


LoginActivity
RegistroActivity
InicioActivity
AgregarVehiculo Activity
AdminVehiculo-Activity
DetalleVehiculo-Activity


Login
(Nombre, Apellido)
Ir a Registro
Registro completado
Autenticación exitosa
Cerrar sesión
Registro
(Nuevo Usuario)
Inicio
Lista de Vehículos
(Imagen, placa, marca, año, color, costo, activo)
Agregar Vehículo
Guardar / Cancelar
Editar/Eliminar Vehículo
Guardar / Eliminar / Cancelar
Agregar Vehículo
Editar / Eliminar Vehículo