DECISIONES DISEÑO INTERFAZ API

ADRIÁN ARROYO GÓMEZ

15 DE DICIEMBRE DE 2024

I.E.S RAFAEL ALBERTI

ÍNDICE

Introducción 2
Decisiones Claves Tomadas 3
  Colores 3
  Diseño principal 4
Opciones Adicionales 6
  Opciones menú hamburguesa 6
  Modo oscuro / modo claro 9
  Toast 11
  Navegación entre pantallas 12

Introducción

Este diseño está enfocado en la creación de una App con una API.

La idea principal es una aplicación que te permite hacer un seguimiento a tiempo real de tu ubicación, y registrar cada ruta que hagas para poder consultarla luego.

Para ello, he desarrollado esta interfaz gráfica a modo boceto, para con el tiempo seguir agrandándola, implementando nuevas utilidades y funcionalidades.

Decisiones Claves Tomadas

Colores

En cuanto a los colores, me he enfocado en intentar usar en la mayoría de las ocasiones, el **colorScheme** de **ui.theme.Theme.kt.** Esto me permite tener un control sobre los colores por defecto que se van a usar a lo largo de la App, ya que cualquier color de cualquier componente por defecto se puede cambiar desde aquí, permitiendo además implementar un modo oscuro de una manera muy sencilla que explicaré más adelante.

En algún lugares, he usado colores definidos en el archivo de **ui.theme.Color.kt** directamente, sin pasar por el intermediario de **colorScheme**.

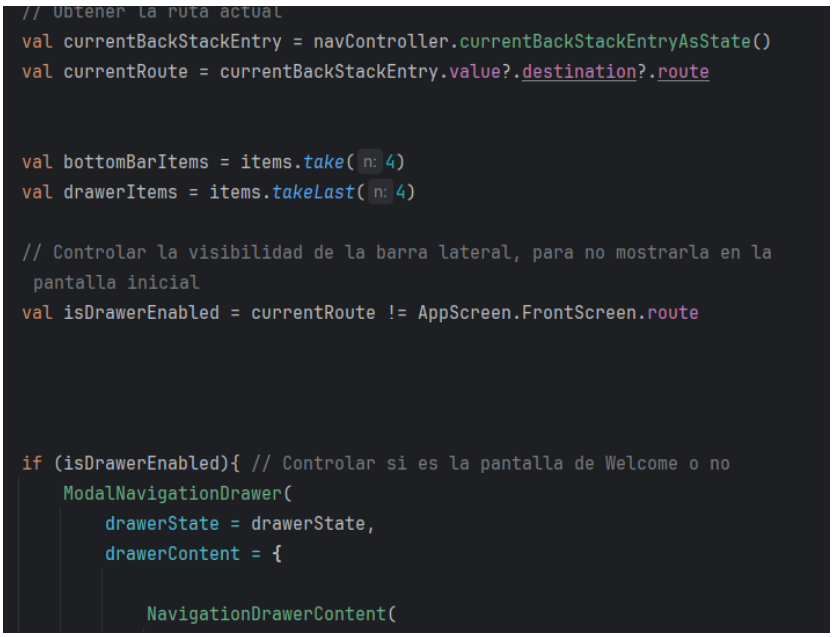
Diseño principal

En cuanto al diseño, he decidido crear tanto una barra superior como una inferior. Dentro de la superior, tengo un título de la App a modo de prueba, el icono para volver hacia detrás, y el menú hamburguesa, en el que se encuentras varias opciones para hace navegación entre pantallas.

En la barra inferior, he decido poner los iconos para la navegación principal, ya que son los que están mas a mano y cerca de donde sujetamos el dispositivo. Además, ya estamos acostumbrados a convivir y usar este tipo de aplicaciones que hacen el menú de esta forma, por lo que nos resulta intuitivo y solo con verlo sabemos lo que hace y cómo funciona.

He decidido que el fondo de estas dos barras será negro con las letras en blanco siempre, aunque cambie a modo claro, porque así sigue manteniendo un estilo característico, y no cambia tan bruscamente toda la App.

Estas dos barras, solo son visible una vez el usuario entra en la aplicación desde la portada. Esto lo he realizado usando un **Scaffold**, y comprobando cual es la ruta actual, para mostrar o no este **Scaffold**.



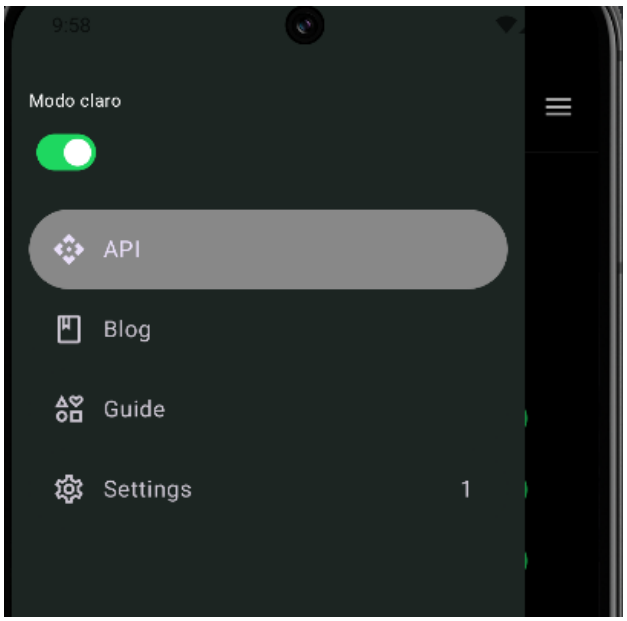
```
// Obtener la ruta actual
val currentBackStackEntry = navController.currentBackStackEntryAsState()
val currentRoute = currentBackStackEntry.value?.destination?.route


val bottomBarItems = items.take( n: 4)
val drawerItems = items.takeLast( n: 4)

// Controlar la visibilidad de la barra lateral, para no mostrarla en la
 pantalla inicial
val isDrawerEnabled = currentRoute != AppScreen.FrontScreen.route




if (isDrawerEnabled){ // Controlar si es la pantalla de Welcome o no
    ModalNavigationDrawer(
        drawerState = drawerState,
        drawerContent = {

            NavigationDrawerContent(
```

Como se puede ver, solo se llama a la función encargada de mostrar las barras, cuando la ruta actual no es **FrontScreen**, que es la portada de la App.

Opciones Adicionales

Opciones menú hamburguesa

He creado el menú hamburguesa con sus opciones dentro, las cuales al clicar te lleva a otras pantallas:

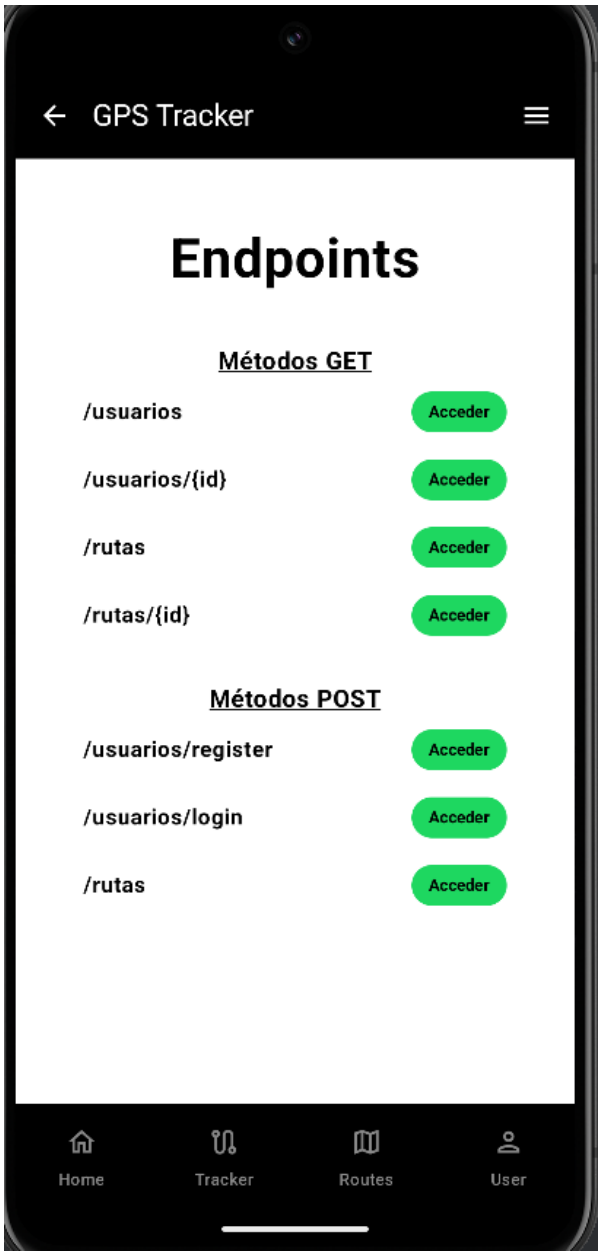


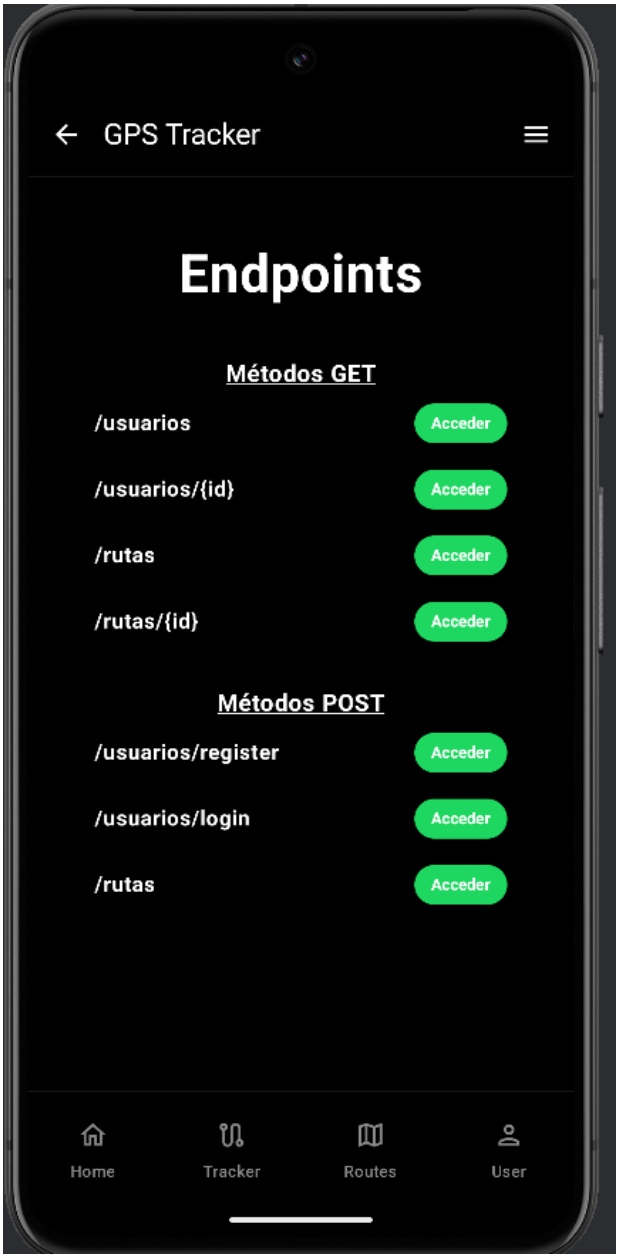
Esto lo hago mediante la iteración de objetos de tipo **NavigationItem**, para tener el código modularizado y poder poner todas las que quiera sin repetir código:

```
data class NavigationItem(
    val title: String,
    val selectedIcon: ImageVector,
    val unselectedIcon: ImageVector,
    var badgeCount: Int? = null,
    val route: String
)
```

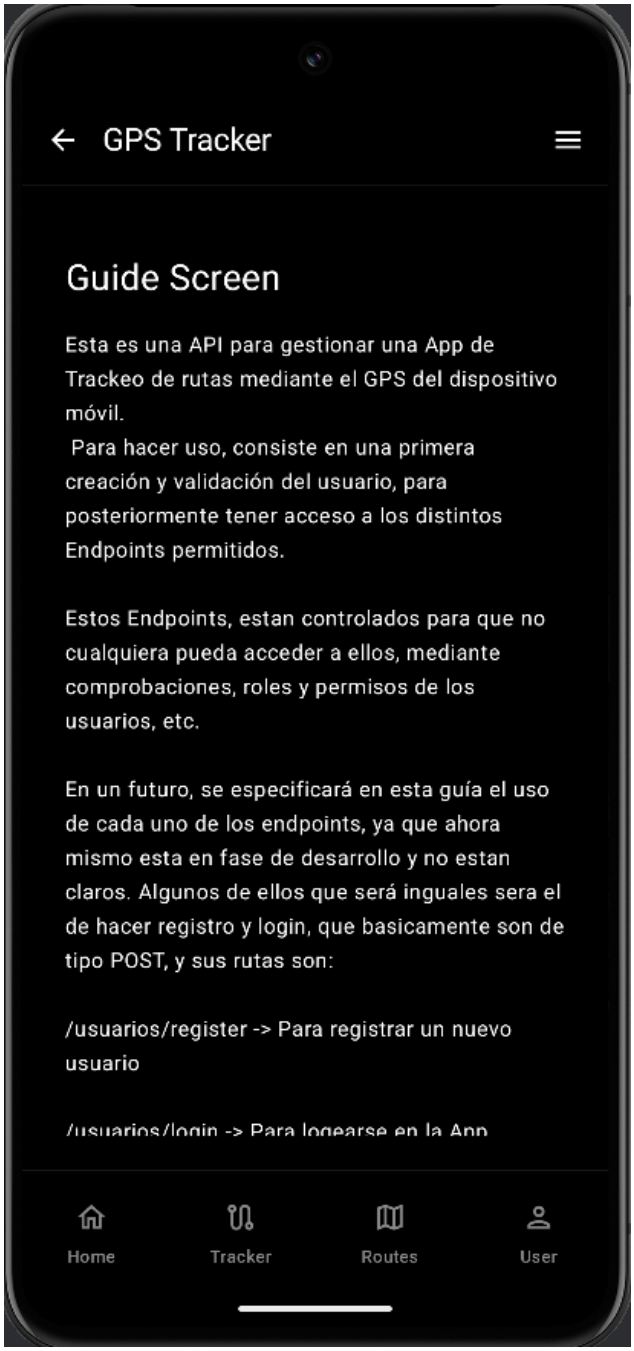
Algunas de las opciones que tienen contenido, son las de la API y la de la Guide.

API Screen:





Guide Screen:



Modo oscuro / modo claro

He creado un switch para cambiar de modo oscuro a modo claro. Esto también implica que haya usado en los componentes que han de cambiar de color el **MatherialTheme.colorScheme**, ya que te permite usar un color u otro en función de si se cambia a modo claro o no.

He usado también una función para cambiar el color de tipo **hex** a **Color**, para facilitar las cosas a la hora de cambiar de tema:

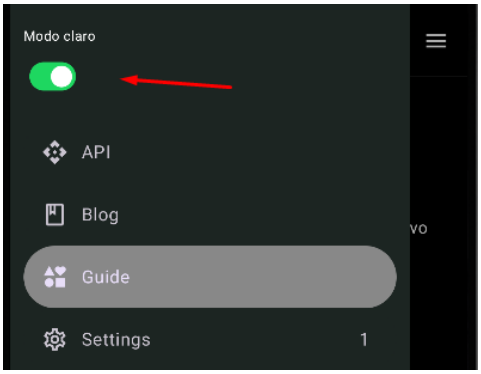
```
/**
 * Extra añadido. Funcion para convertir colores HEX a Color
 */

fun hexToColor(hex: String): Color {
    val color = if (hex.startsWith( prefix: "#")) {
        hex.substring( startIndex: 1)
    } else {
        hex
    }

    val colorValue = if (color.length == 6) "FF$color" else color // Prevenir
     colores RGBA

    // Hex -> Int -> Color
    return Color(android.graphics.Color.parseColor( colorString: "#$colorValue"))
}
```

Luego en el **ui.theme**.**Theme** establezco los colores para cada tema:

```
private val DarkColorScheme = darkColorScheme(
    primary = green,
    onPrimary = white,
    background = black,
)

private val LightColorScheme = lightColorScheme(
    primary = green,
    onPrimary = black,
    background = white
```

Y de esta forma decide automáticamente si el móvil está en modo oscuro te pone el modo oscuro por defecto, además de poder hacerlo en el switch que hay en el menú.

```kotlin
@Composable
fun InterfazProyAPITheme(
    darkTheme: Boolean = isSystemInDarkTheme(), // Pone por defecto el tema como lo
     tengamos cofigurado en el movil
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) DarkColorScheme else LightColorScheme

    MaterialTheme(
        colorScheme = colors,
        typography = Typography,
        content = content
    )
```



Para implementar esto, también necesito crear un estado que se le pasa desde el mismo **MainActivity.kt:**

```kotlin
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            var isDarkMode by rememberSaveable { mutableStateOf( value: true) }
            val navController = rememberNavController()

            InterfazProyAPITheme(darkTheme = isDarkMode) {

                // Se obtiene el innerPadding y lo pasamos al Modifier
                Scaffold(
                    content = { innerPadding -> // Este es el innerPadding de
                     Scaffold

                        MyAppWithDrawer(
                            navController = navController,
                            isDarkMode = isDarkMode,
                            onDarkModeSwitchChange = { isDarkMode = it },
                            modifier = Modifier.padding(innerPadding)
                        )
                    }
                )
            }
        }
    }
}
```

Toast

He implementado una pequeña función para poder mostrar mensajes personalizados al clicar algo por ejemplo usando un **Toast:**

```
fun customToast(context: Context, msg: String){
    Toast.makeText(context, msg, Toast.LENGTH_SHORT).show()
}
```

Le paso el mensaje personalizado cada vez que lo llamo, y el contexto.

Navegación entre pantallas

También he implementado la navegación entre pantallas, tanto en las opciones del menú hamburguesa como en las opciones de la barra inferior.

Cuando alguna de estas opciones se clica, se marca para que el usuario lo sepa: