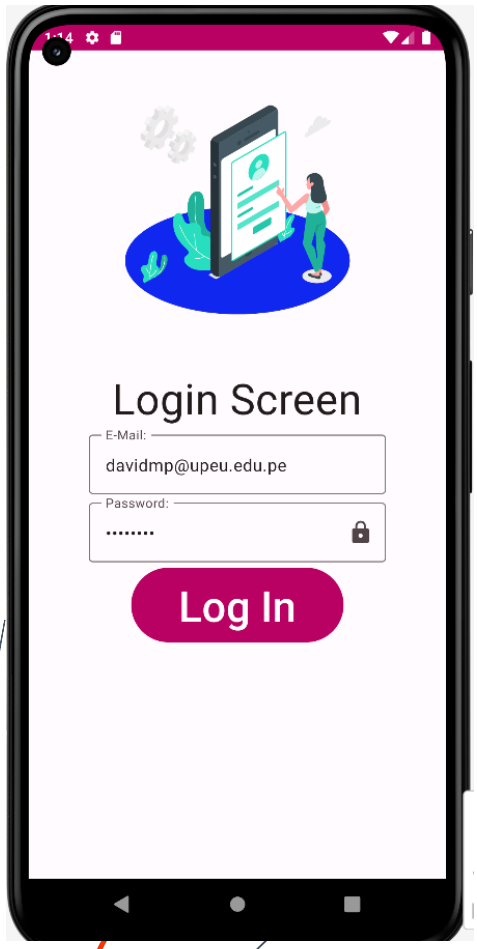
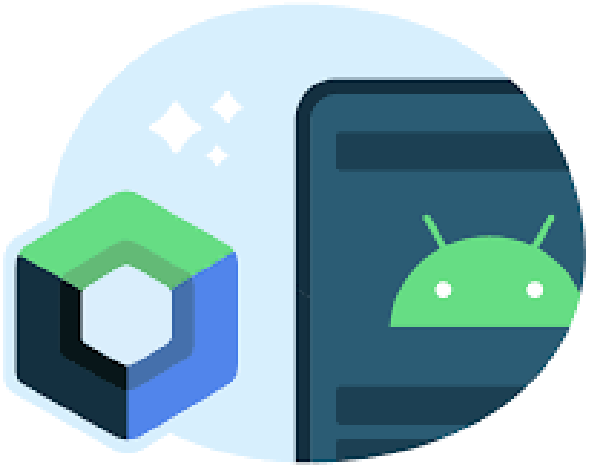
11-4-2024

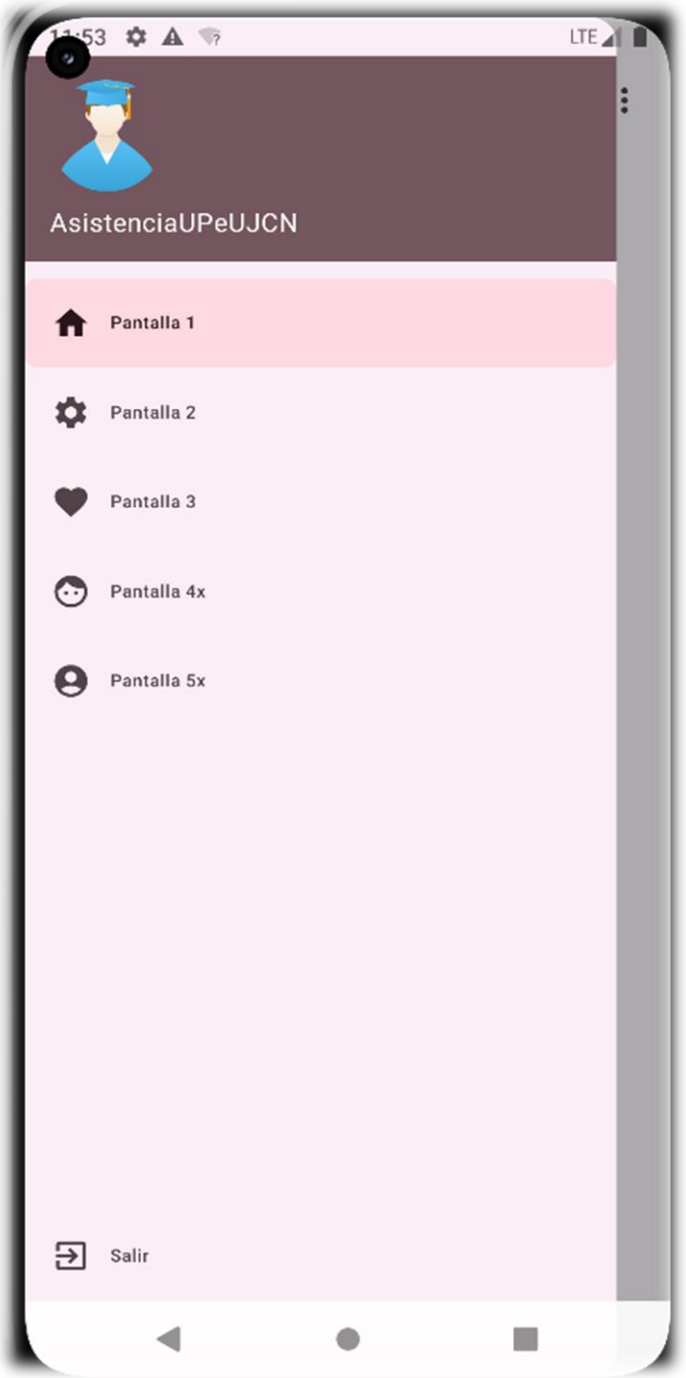
JetPack Compose





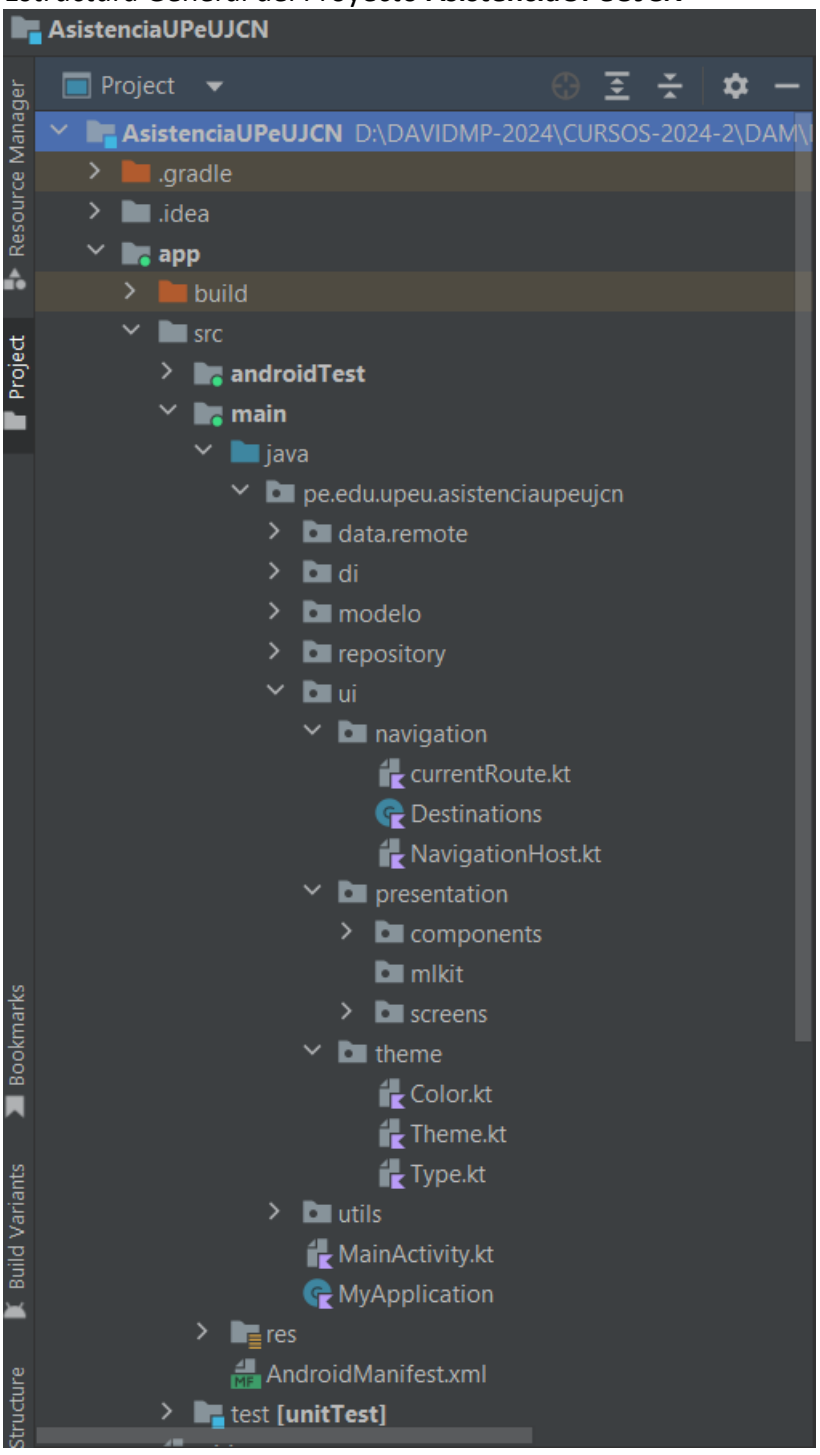
David Mamani Pari
SYSCENTERLIFE@

Configurando Proyecto



1. Estructura del Proyecto

Estructura General del Proyecto **AsistenciaUPeUJCN**



1.1. Copiar Recursos en res

Copiar el contenido de la carpeta Recursos (res) a cada carpeta correspondiente de res

1.2. Configurar gradle en el proyecto

En gradle.properties agregar:

```
android.enableJetifier=true
```

En build.gradle.kts del proyecto agregar:

```
buildscript{
    dependencies{
        classpath("com.google.dagger:hilt-android-gradle-plugin:2.52") //cambiado 2.52 old 2.47 old 2.45
    }
}
```

Dentro de plugins:

```
id("com.android.library") version "8.1.4" apply false
```

Debajo de plugins:

```
tasks.register("clean", Delete::class) {
    delete(rootProject.buildDir)
}
```

1.3. Configurar gradle en el módulo app del proyecto

Agregar dentro de plugins:

```
id("kotlin-kapt") //Agregado
id("dagger.hilt.android.plugin") //Agregado
```

Agregar las Dependencias:

```
//Navegacion
val nav_version = "2.7.7"
implementation("androidx.navigation:navigation-compose:$nav_version")
//Agregados  Dagger - Hilt
implementation ("com.google.dagger:hilt-android:2.52") //old 2.47
kapt ("com.google.dagger:hilt-compiler:2.52") //old 2.47
//Agregado Dagger - Hilt Compose
implementation("androidx.hilt:hilt-navigation-compose:1.2.0") //old 1.0.0
implementation("com.valentinilk.shimmer:compose-shimmer:1.3.1") //old 1.0.5
implementation ("io.coil-kt:coil-compose:2.7.0") //old 2.4.0
//Agregado LiveData compose
```

```
//implementation ("androidx.compose.ui:ui-tooling")
implementation ("androidx.compose.foundation:foundation")
implementation ("androidx.compose.runtime:runtime-livedata")
//Formularios
implementation ("com.github.k0shk0sh:compose-easyforms:0.2.0")
// Retrofit
implementation ("com.squareup.retrofit2:retrofit:2.9.0")
implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
implementation ("com.squareup.retrofit2:converter-moshi:2.9.0")
//App Compact para detectar modo dia noche
val appcompat_version = "1.7.0" //old 1.6.1
implementation("androidx.appcompat:appcompat:$appcompat_version")//Agrega
do recien
```

1.4. Configurar Android Manifest

Agregar Permisos:

```
<uses-permission android:name="android.permission.INTERNET"
/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

En la etiqueta application considerar similar contenido:

```
<application
    android:name=".MyApplication"
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AsistenciaUPeUJCR"
    tools:targetApi="31"
    android:usesCleartextTraffic="true"
    >
    <activity
        android:name=".MainActivity"
        android:exported="true"
        android:label="@string/app_name"
        android:theme="@style/Theme.AsistenciaUPeUJCR">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER"
/>
        </intent-filter>
    </activity>
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</application>
```

1.5. Cambiar contenido del paquete theme

En Color.kt agregar colores:
Tener en cuenta esta página:
Material Design
Lo generado de allí colocar solo los colores:

```
val md_theme_light_primary = Color(0xFF326B00)
val md_theme_light_onSecondary = Color(0xFFFFFFFF)
val md_theme_light_scrim = Color(0xFF000000)
val purple_theme_light_primary = Color(0xFF6750A4)
val purple_theme_light_onPrimary = Color(0xFFFFFFFF)
val purple_theme_light_primaryContainer = Color(0xFFEADDFF)
val purple_theme_light_onPrimaryContainer = Color(0xFF21005D)
val purple_theme_light_secondary = Color(0xFF625B71)
val purple_theme_light_onSecondary = Color(0xFFFFFFFF)
val purple_theme_light_secondaryContainer = Color(0xFFE8DEF8)
val purple_theme_light_onSecondaryContainer =
Color(0xFF1D192B)
val purple_theme_light_tertiary = Color(0xFF7D5260)
val purple_theme_light_onTertiary = Color(0xFFFFFFFF)
val purple_theme_light_tertiaryContainer = Color(0xFFFFD8E4)
val purple_theme_light_onTertiaryContainer =
Color(0xFF31111D)

val purple_theme_light_error = Color(0xFFB3261E)
val purple_theme_light_onError = Color(0xFFFFFFFF)
val purple_theme_light_errorContainer = Color(0xFFF9DEDC)
val purple_theme_light_onErrorContainer = Color(0xFF410E0B)
val purple_theme_light_outline = Color(0xFF79747E)
val purple_theme_light_background = Color(0xFFFFFBFE)
val purple_theme_light_onBackground = Color(0xFF1C1B1F)
val purple_theme_light_surface = Color(0xFFFFFBFE)
val purple_theme_light_onSurface = Color(0xFF1C1B1F)
val purple_theme_light_surfaceVariant = Color(0xFFE7E0EC)
val purple_theme_light_onSurfaceVariant = Color(0xFF49454F)
val purple_theme_light_inverseSurface = Color(0xFF313033)
val purple_theme_light_inverseOnSurface = Color(0xFFF4EFF4)
val purple_theme_light_inversePrimary = Color(0xFFD0BCFF)
val purple_theme_light_shadow = Color(0xFF000000)
val purple_theme_light_surfaceTint = Color(0xFF6750A4)
val purple_theme_light_outlineVariant = Color(0xFFCAC4D0)
val purple_theme_light_scrim = Color(0xFF000000)
val purple_theme_dark_primary = Color(0xFFD0BCFF)
val purple_theme_dark_onPrimary = Color(0xFF381E72)
val purple_theme_dark_primaryContainer = Color(0xFF4F378B)
```

```
val purple_theme_dark_onPrimaryContainer = Color(0xFFEADDFF)
val purple_theme_dark_secondary = Color(0xFFCCC2DC)
val purple_theme_dark_onSecondary = Color(0xFF332D41)
val purple_theme_dark_secondaryContainer = Color(0xFF4A4458)
val purple_theme_dark_onSecondaryContainer =
Color(0xFFE8DEF8)
val purple_theme_dark_tertiary = Color(0xFFEFB8C8)
val purple_theme_dark_onTertiary = Color(0xFF492532)
val purple_theme_dark_tertiaryContainer = Color(0xFF633B48)
val purple_theme_dark_onTertiaryContainer = Color(0xFFFFD8E4)
val purple_theme_dark_error = Color(0xFFF2B8B5)
val purple_theme_dark_onError = Color(0xFF601410)
val purple_theme_dark_errorContainer = Color(0xFF8C1D18)
val purple_theme_dark_onErrorContainer = Color(0xFFF9DEDC)
val purple_theme_dark_outline = Color(0xFF938F99)
val purple_theme_dark_background = Color(0xFF1C1B1F)
val purple_theme_dark_onBackground = Color(0xFFE6E1E5)
val purple_theme_dark_surface = Color(0xFF1C1B1F)
val purple_theme_dark_onSurface = Color(0xFFE6E1E5)
val purple_theme_dark_surfaceVariant = Color(0xFF49454F)
val purple_theme_dark_onSurfaceVariant = Color(0xFFCAC4D0)
val purple_theme_dark_inverseSurface = Color(0xFFE6E1E5)
val purple_theme_dark_inverseOnSurface = Color(0xFF313033)
val purple_theme_dark_inversePrimary = Color(0xFF6750A4)
val purple_theme_dark_shadow = Color(0xFF000000)
val purple_theme_dark_surfaceTint = Color(0xFFD0BCFF)
val purple_theme_dark_outlineVariant = Color(0xFF49454F)
val purple_theme_dark_scrim = Color(0xFF000000)
val seed = Color(0xFF6750A4)
val red_theme_light_primary = Color(0xFFB90063)
val red_theme_light_onPrimary = Color(0xFFFFFFFF)
val red_theme_light_primaryContainer = Color(0xFFFFD9E2)
val red_theme_light_onPrimaryContainer = Color(0xFF3E001D)
val red_theme_light_secondary = Color(0xFF74565F)

val red_theme_light_onSecondary = Color(0xFFFFFFFF)
val red_theme_light_secondaryContainer = Color(0xFFFFD9E2)
val red_theme_light_onSecondaryContainer = Color(0xFF2B151C)
val red_theme_light_tertiary = Color(0xFFA82F63)
val red_theme_light_onTertiary = Color(0xFFFFFFFF)
val red_theme_light_tertiaryContainer = Color(0xFFFFD9E2)
val red_theme_light_onTertiaryContainer = Color(0xFF3E001E)
val red_theme_light_error = Color(0xFFBA1A1A)
val red_theme_light_errorContainer = Color(0xFFFFDAD6)
val red_theme_light_onError = Color(0xFFFFFFFF)
val red_theme_light_onErrorContainer = Color(0xFF410002)
val red_theme_light_background = Color(0xFFFFFBFF)
```

```
val red_theme_light_onBackground = Color(0xFF201A1B)
val red_theme_light_surface = Color(0xFFFFFBFF)
val red_theme_light_onSurface = Color(0xFF201A1B)
val red_theme_light_surfaceVariant = Color(0xFFF2DDE1)
val red_theme_light_onSurfaceVariant = Color(0xFF514347)
val red_theme_light_outline = Color(0xFF837377)
val red_theme_light_inverseOnSurface = Color(0xFFFAEEEF)
val red_theme_light_inverseSurface = Color(0xFF352F30)
val red_theme_light_inversePrimary = Color(0xFFFFB1C8)
val red_theme_light_shadow = Color(0xFF000000)
val red_theme_light_surfaceTint = Color(0xFFB90063)
val red_theme_light_outlineVariant = Color(0xFFD5C2C6)
val red_theme_light_scrim = Color(0xFF000000)
val red_theme_dark_primary = Color(0xFFFFB1C8)
val red_theme_dark_onPrimary = Color(0xFF650033)
val red_theme_dark_primaryContainer = Color(0xFF8E004A)
val red_theme_dark_onPrimaryContainer = Color(0xFFFFD9E2)
val red_theme_dark_secondary = Color(0xFFE3BDC6)
val red_theme_dark_onSecondary = Color(0xFF422931)
val red_theme_dark_secondaryContainer = Color(0xFF5A3F47)
val red_theme_dark_onSecondaryContainer = Color(0xFFFFD9E2)
val red_theme_dark_tertiary = Color(0xFFFFB0C9)
val red_theme_dark_onTertiary = Color(0xFF650034)
val red_theme_dark_tertiaryContainer = Color(0xFF88134B)
val red_theme_dark_onTertiaryContainer = Color(0xFFFFD9E2)
val red_theme_dark_error = Color(0xFFFFB4AB)
val red_theme_dark_errorContainer = Color(0xFF93000A)
val red_theme_dark_onError = Color(0xFF690005)
val red_theme_dark_onErrorContainer = Color(0xFFFFDAD6)
val red_theme_dark_background = Color(0xFF201A1B)
val red_theme_dark_onBackground = Color(0xFFEBE0E1)
val red_theme_dark_surface = Color(0xFF201A1B)
val red_theme_dark_onSurface = Color(0xFFEBE0E1)
val red_theme_dark_surfaceVariant = Color(0xFF514347)
val red_theme_dark_onSurfaceVariant = Color(0xFFD5C2C6)
val red_theme_dark_outline = Color(0xFF9E8C90)
val red_theme_dark_inverseOnSurface = Color(0xFF201A1B)
val red_theme_dark_inverseSurface = Color(0xFFEBE0E1)
val red_theme_dark_inversePrimary = Color(0xFFB90063)
val red_theme_dark_shadow = Color(0xFF000000)
val red_theme_dark_surfaceTint = Color(0xFFFFB1C8)
val red_theme_dark_outlineVariant = Color(0xFF514347)
val red_theme_dark_scrim = Color(0xFF000000)

val green_theme_light_primary = Color(0xFF3A6A00)
val green_theme_light_onPrimary = Color(0xFFFFFFFF)
val green_theme_light_primaryContainer = Color(0xFFB4F575)
```

```
val green_theme_light_onPrimaryContainer = Color(0xFF0E2000)
val green_theme_light_secondary = Color(0xFF57624A)
val green_theme_light_onSecondary = Color(0xFFFFFFFF)
val green_theme_light_secondaryContainer = Color(0xFFDBE7C9)
val green_theme_light_onSecondaryContainer =
Color(0xFF151E0C)
val green_theme_light_tertiary = Color(0xFF386664)
val green_theme_light_onTertiary = Color(0xFFFFFFFF)
val green_theme_light_tertiaryContainer = Color(0xFFBBECE9)
val green_theme_light_onTertiaryContainer = Color(0xFF00201F)
val green_theme_light_error = Color(0xFFBA1A1A)
val green_theme_light_errorContainer = Color(0xFFFFDAD6)
val green_theme_light_onError = Color(0xFFFFFFFF)
val green_theme_light_onErrorContainer = Color(0xFF410002)
val green_theme_light_background = Color(0xFFFDFCF5)
val green_theme_light_onBackground = Color(0xFF1B1C18)
val green_theme_light_surface = Color(0xFFFDFCF5)
val green_theme_light_onSurface = Color(0xFF1B1C18)
val green_theme_light_surfaceVariant = Color(0xFFE0E4D5)
val green_theme_light_onSurfaceVariant = Color(0xFF44483E)
val green_theme_light_outline = Color(0xFF74796C)
val green_theme_light_inverseOnSurface = Color(0xFFF2F1EA)
val green_theme_light_inverseSurface = Color(0xFF2F312C)
val green_theme_light_inversePrimary = Color(0xFF99D85C)
val green_theme_light_shadow = Color(0xFF000000)
val green_theme_light_surfaceTint = Color(0xFF3A6A00)
val green_theme_light_outlineVariant = Color(0xFFC4C8BA)
val green_theme_light_scrim = Color(0xFF000000)
val green_theme_dark_primary = Color(0xFF99D85C)
val green_theme_dark_onPrimary = Color(0xFF1B3700)
val green_theme_dark_primaryContainer = Color(0xFF2A5000)
val green_theme_dark_onPrimaryContainer = Color(0xFFB4F575)
val green_theme_dark_secondary = Color(0xFFBFCBAE)
val green_theme_dark_onSecondary = Color(0xFF29341F)
val green_theme_dark_secondaryContainer = Color(0xFF3F4A34)
val green_theme_dark_onSecondaryContainer = Color(0xFFDBE7C9)
val green_theme_dark_tertiary = Color(0xFFA0CFCD)
val green_theme_dark_onTertiary = Color(0xFF003736)
val green_theme_dark_tertiaryContainer = Color(0xFF1E4E4C)
val green_theme_dark_onTertiaryContainer = Color(0xFFBBECE9)
val green_theme_dark_error = Color(0xFFFFB4AB)
val green_theme_dark_errorContainer = Color(0xFF93000A)
val green_theme_dark_onError = Color(0xFF690005)
val green_theme_dark_onErrorContainer = Color(0xFFFFDAD6)
val green_theme_dark_background = Color(0xFF1B1C18)
val green_theme_dark_onBackground = Color(0xFFE3E3DC)
val green_theme_dark_surface = Color(0xFF1B1C18)
```

```
val green_theme_dark_onSurface = Color(0xFFE3E3DC)
val green_theme_dark_surfaceVariant = Color(0xFF44483E)
val green_theme_dark_onSurfaceVariant = Color(0xFFC4C8BA)
val green_theme_dark_outline = Color(0xFF8E9285)
val green_theme_dark_inverseOnSurface = Color(0xFF1B1C18)
val green_theme_dark_inverseSurface = Color(0xFFE3E3DC)
val green_theme_dark_inversePrimary = Color(0xFF3A6A00)
val green_theme_dark_shadow = Color(0xFF000000)
val green_theme_dark_surfaceTint = Color(0xFF99D85C)
val green_theme_dark_outlineVariant = Color(0xFF44483E)
val green_theme_dark_scrim = Color(0xFF000000)
```

En Theme.kt cambiar o adaptar contenido:

Agregar el siguiente contenido:

```
enum class ThemeType{RED, PURPLE, GREEN}

public val LightPurpleColors = lightColorScheme(
    primary = purple_theme_light_primary,
    onPrimary = purple_theme_light_onPrimary,
    primaryContainer = purple_theme_light_primaryContainer,
    onPrimaryContainer =
purple_theme_light_onPrimaryContainer,
    secondary = purple_theme_light_secondary,
    onSecondary = purple_theme_light_onSecondary,
    secondaryContainer =
purple_theme_light_secondaryContainer,
    onSecondaryContainer =
purple_theme_light_onSecondaryContainer,
    tertiary = purple_theme_light_tertiary,
    onTertiary = purple_theme_light_onTertiary,
    tertiaryContainer = purple_theme_light_tertiaryContainer,
    onTertiaryContainer =
purple_theme_light_onTertiaryContainer,
    error = purple_theme_light_error,
    onError = purple_theme_light_onError,
    errorContainer = purple_theme_light_errorContainer,
    onErrorContainer = purple_theme_light_onErrorContainer,
    outline = purple_theme_light_outline,
    background = purple_theme_light_background,
    onBackground = purple_theme_light_onBackground,
    surface = purple_theme_light_surface,
    onSurface = purple_theme_light_onSurface,
    surfaceVariant = purple_theme_light_surfaceVariant,
    onSurfaceVariant = purple_theme_light_onSurfaceVariant,
    inverseSurface = purple_theme_light_inverseSurface,
```

```
    inverseOnSurface = purple_theme_light_inverseOnSurface,
    inversePrimary = purple_theme_light_inversePrimary,
    surfaceTint = purple_theme_light_surfaceTint,
    outlineVariant = purple_theme_light_outlineVariant,
    scrim = purple_theme_light_scrim,
)
public val DarkPurpleColors = darkColorScheme(
    primary = purple_theme_dark_primary,
    onPrimary = purple_theme_dark_onPrimary,
    primaryContainer = purple_theme_dark_primaryContainer,
    onPrimaryContainer =
purple_theme_dark_onPrimaryContainer,
    secondary = purple_theme_dark_secondary,
    onSecondary = purple_theme_dark_onSecondary,
    secondaryContainer =
purple_theme_dark_secondaryContainer,
    onSecondaryContainer =
purple_theme_dark_onSecondaryContainer,
    tertiary = purple_theme_dark_tertiary,
    onTertiary = purple_theme_dark_onTertiary,
    tertiaryContainer = purple_theme_dark_tertiaryContainer,
    onTertiaryContainer =
purple_theme_dark_onTertiaryContainer,
    error = purple_theme_dark_error,
    onError = purple_theme_dark_onError,
    errorContainer = purple_theme_dark_errorContainer,
    onErrorContainer = purple_theme_dark_onErrorContainer,
    outline = purple_theme_dark_outline,
    background = purple_theme_dark_background,
    onBackground = purple_theme_dark_onBackground,
    surface = purple_theme_dark_surface,
    onSurface = purple_theme_dark_onSurface,
    surfaceVariant = purple_theme_dark_surfaceVariant,
    onSurfaceVariant = purple_theme_dark_onSurfaceVariant,
    inverseSurface = purple_theme_dark_inverseSurface,
    inverseOnSurface = purple_theme_dark_inverseOnSurface,
    inversePrimary = purple_theme_dark_inversePrimary,
    surfaceTint = purple_theme_dark_surfaceTint,
    outlineVariant = purple_theme_dark_outlineVariant,
    scrim = purple_theme_dark_scrim,
)

public val LightRedColors = lightColorScheme(
    primary = red_theme_light_primary,
    onPrimary = red_theme_light_onPrimary,
    primaryContainer = red_theme_light_primaryContainer,
    onPrimaryContainer = red_theme_light_onPrimaryContainer,
```

```
    secondary = red_theme_light_secondary,
    onSecondary = red_theme_light_onSecondary,
    secondaryContainer = red_theme_light_secondaryContainer,
    onSecondaryContainer =
red_theme_light_onSecondaryContainer,
    tertiary = red_theme_light_tertiary,
    onTertiary = red_theme_light_onTertiary,
    tertiaryContainer = red_theme_light_tertiaryContainer,
    onTertiaryContainer =
red_theme_light_onTertiaryContainer,
    error = red_theme_light_error,
    errorContainer = red_theme_light_errorContainer,
    onError = red_theme_light_onError,
    onErrorContainer = red_theme_light_onErrorContainer,
    background = red_theme_light_background,
    onBackground = red_theme_light_onBackground,
    surface = red_theme_light_surface,
    onSurface = red_theme_light_onSurface,
    surfaceVariant = red_theme_light_surfaceVariant,
    onSurfaceVariant = red_theme_light_onSurfaceVariant,
    outline = red_theme_light_outline,
    inverseOnSurface = red_theme_light_inverseOnSurface,
    inverseSurface = red_theme_light_inverseSurface,
    inversePrimary = red_theme_light_inversePrimary,
    surfaceTint = red_theme_light_surfaceTint,
    outlineVariant = red_theme_light_outlineVariant,
    scrim = red_theme_light_scrim,
)
public val DarkRedColors = darkColorScheme(
    primary = red_theme_dark_primary,
    onPrimary = red_theme_dark_onPrimary,
    primaryContainer =
red_theme_dark_primaryContainer,onPrimaryContainer =
red_theme_dark_onPrimaryContainer,
    secondary = red_theme_dark_secondary,
    onSecondary = red_theme_dark_onSecondary,
    secondaryContainer = red_theme_dark_secondaryContainer,
    onSecondaryContainer =
red_theme_dark_onSecondaryContainer,
    tertiary = red_theme_dark_tertiary,
    onTertiary = red_theme_dark_onTertiary,
    tertiaryContainer = red_theme_dark_tertiaryContainer,
    onTertiaryContainer = red_theme_dark_onTertiaryContainer,
    error = red_theme_dark_error,
    errorContainer = red_theme_dark_errorContainer,
    onError = red_theme_dark_onError,
    onErrorContainer = red_theme_dark_onErrorContainer,
```

```
    background = red_theme_dark_background,
    onBackground = red_theme_dark_onBackground,
    surface = red_theme_dark_surface,
    onSurface = red_theme_dark_onSurface,
    surfaceVariant = red_theme_dark_surfaceVariant,
    onSurfaceVariant = red_theme_dark_onSurfaceVariant,
    outline = red_theme_dark_outline,
    inverseOnSurface = red_theme_dark_inverseOnSurface,
    inverseSurface = red_theme_dark_inverseSurface,
    inversePrimary = red_theme_dark_inversePrimary,
    surfaceTint = red_theme_dark_surfaceTint,
    outlineVariant = red_theme_dark_outlineVariant,
    scrim = red_theme_dark_scrim,
)
public val LightGreenColors = lightColorScheme(
    primary = green_theme_light_primary,
    onPrimary = green_theme_light_onPrimary,
    primaryContainer = green_theme_light_primaryContainer,
    onPrimaryContainer =
green_theme_light_onPrimaryContainer,
    secondary = green_theme_light_secondary,
    onSecondary = green_theme_light_onSecondary,
    secondaryContainer =
green_theme_light_secondaryContainer,
    onSecondaryContainer =
green_theme_light_onSecondaryContainer,
    tertiary = green_theme_light_tertiary,
    onTertiary = green_theme_light_onTertiary,
    tertiaryContainer = green_theme_light_tertiaryContainer,
    onTertiaryContainer =
green_theme_light_onTertiaryContainer,
    error = green_theme_light_error,
    errorContainer = green_theme_light_errorContainer,
    onError = green_theme_light_onError,
    onErrorContainer = green_theme_light_onErrorContainer,
    background = green_theme_light_background,
    onBackground = green_theme_light_onBackground,
    surface = green_theme_light_surface,
    onSurface = green_theme_light_onSurface,
    surfaceVariant = green_theme_light_surfaceVariant,
    onSurfaceVariant = green_theme_light_onSurfaceVariant,
    outline = green_theme_light_outline,
    inverseOnSurface = green_theme_light_inverseOnSurface,
    inverseSurface = green_theme_light_inverseSurface,
    inversePrimary = green_theme_light_inversePrimary,
    surfaceTint = green_theme_light_surfaceTint,
    outlineVariant = green_theme_light_outlineVariant,
```

```
    scrim = green_theme_light_scrim,
)

public val DarkGreenColors = darkColorScheme(
    primary = green_theme_dark_primary,
    onPrimary = green_theme_dark_onPrimary,
    primaryContainer = green_theme_dark_primaryContainer,
    onPrimaryContainer = green_theme_dark_onPrimaryContainer,
    secondary = green_theme_dark_secondary,
    onSecondary = green_theme_dark_onSecondary,
    secondaryContainer = green_theme_dark_secondaryContainer,
    onSecondaryContainer =
green_theme_dark_onSecondaryContainer,
    tertiary = green_theme_dark_tertiary,
    onTertiary = green_theme_dark_onTertiary,
    tertiaryContainer = green_theme_dark_tertiaryContainer,
    onTertiaryContainer =
green_theme_dark_onTertiaryContainer,
    error = green_theme_dark_error,
    errorContainer = green_theme_dark_errorContainer,
    onError = green_theme_dark_onError,
    onErrorContainer = green_theme_dark_onErrorContainer,
    background = green_theme_dark_background,
    onBackground = green_theme_dark_onBackground,
    surface = green_theme_dark_surface,
    onSurface = green_theme_dark_onSurface,
    surfaceVariant = green_theme_dark_surfaceVariant,
    onSurfaceVariant = green_theme_dark_onSurfaceVariant,
    outline = green_theme_dark_outline,
    inverseOnSurface = green_theme_dark_inverseOnSurface,
    inverseSurface = green_theme_dark_inverseSurface,
    inversePrimary = green_theme_dark_inversePrimary,
    surfaceTint = green_theme_dark_surfaceTint,
    outlineVariant = green_theme_dark_outlineVariant,
    scrim = green_theme_dark_scrim,
)
```

El siguiente paso es adaptar el composable:

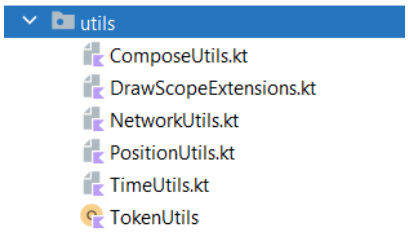
```
@Composable
fun AsistenciaUPeUJCNTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    colorScheme: ColorScheme,
    content: @Composable () -> Unit
) {
    /*val colorScheme = when {
```

```
        dynamicColor && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context)
else dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }*/

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

1.6. Copiar archivos de Recursos a paquete utils



Corregir las importaciones o paquetes de ubicación en los archivos copiados

1.7. Crear el archivo MyApplication para trabajar con Dagger Hilt

Crear el archivo MyApplication a la altura de MainActivity:

```
@ExperimentalCoroutinesApi
@HiltAndroidApp
class MyApplication : Application(){
    override fun onCreate() {
        super.onCreate()
        val mode=if (isNight()){
            AppCompatDelegate.MODE_NIGHT_YES
        }else{
            AppCompatDelegate.MODE_NIGHT_NO
        }
        AppCompatDelegate.setDefaultNightMode(mode)
    }
}
```

1.8. Trabajar en el paquete modelo

Crear ComboModel:

```
data class ComboModel(val code:String, val name:String):
PickerValue(){
    override fun searchFilter(query: String): Boolean {
        return this.name.startsWith(query)
    }
}
abstract class PickerValue{
    abstract fun searchFilter(query:String):Boolean
}
```

Crear Usuario:

```
data class Usuario(
    val nombres: String,
    val apellidos: String,
    val correo: String,
    val password: String,
    val token: String,
    val dni: String,
    val perfilPrin: String,
    val estado: String,
    val offlinex: String,
)


data class UsuarioDto(
    var correo: String,
    var password: String,
)

data class UsuarioResp(
    val id: Long,
    val nombres: String,
    val apellidos: String,
    val correo: String,
    val token: String,
    val dni: String,
    val perfilPrin: String,
    val estado: String,
```

```
    val offlinex: String,
)
```

1.9. Trabajar en data/remote

Crear archivo RestUsuario:

```
interface RestUsuario {
    @POST("/asis/login")
    suspend fun login(@Body user:UsuarioDto):
Response<UsuarioResp>
}
```

1.10. Trabajar en el paquete Repository

Crear archivo UsuarioRepository:

```
interface UsuarioRepository {
    suspend fun loginUsuario(user:UsuarioDto):
Response<UsuarioResp>
}
class UsuarioRepositoryImp @Inject constructor(private val
restUsuario:

RestUsuario):UsuarioRepository{
    override suspend fun loginUsuario(user:UsuarioDto):
            Response<UsuarioResp>{
        return restUsuario.login(user)
    }
}
```

1.11. Trabajar Inyección de dependencias di

Crear archivo DataSourceModule:

```
@Module
@InstallIn(SingletonComponent::class)
class DataSourceModule {
    var retrofit: Retrofit?=null
    @Singleton
    @Provides
```

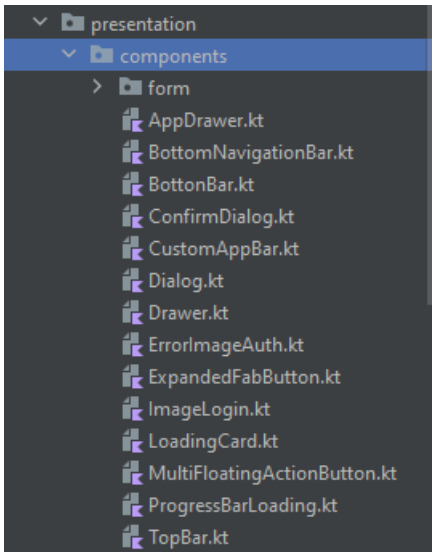
```
    @Named("BaseUrl")
    fun provideBaseUrl()=TokenUtils.API_URL
    @Singleton
    @Provides
    fun provideRetrofit(@Named("BaseUrl") baseUrl:String):
Retrofit {
        val okHttpClient= OkHttpClient.Builder()
            .connectTimeout(1, TimeUnit.MINUTES)
            .readTimeout(30, TimeUnit.SECONDS)
            .writeTimeout(15, TimeUnit.SECONDS)
            .build()
        if (retrofit==null){
            retrofit= Retrofit.Builder()

.addConverterFactory(GsonConverterFactory.create())
                .client(okHttpClient)
                .baseUrl(baseUrl).build()
        }
        return retrofit!!
    }
    @Singleton
    @Provides
    fun restUsuario(retrofit: Retrofit):RestUsuario{
        return retrofit.create(RestUsuario::class.java)
    }
}
```

Crear archivo RepositoryModule:

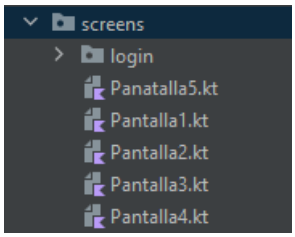
```
@Module
@InstallIn(SingletonComponent::class)
abstract class RepositoryModule {
    @Binds
    @Singleton
    abstract fun
userRepository(userRepos:UsuarioRepositoryImp):UsuarioReposit
ory
}
```

1.1. Copiar archivos de Recursos a paquete components



Corregir las importaciones o paquetes de ubicación en los archivos copiados

1.2. Copiar archivos de Recursos a paquete screens



Corregir las importaciones o paquetes de ubicación en los archivos copiados

1.3. Implementar en navigation para la navegación entre distintas pantallas.

Crear archivo currentRoute:

```
@Composable
fun currentRoute(navController: NavHostController): String? {
    val navBackStackEntry by
navController.currentBackStackEntryAsState()
    return navBackStackEntry?.destination?.route
}
```

Crear archivo **Destinations**:

```
sealed class Destinations(
    val route: String,
    val title: String,
    val icon: ImageVector
```

```
) {
    object Login:Destinations("login", "Login",
Icons.Filled.Settings)
    object Pantalla1 : Destinations( "pantalla1", "Pantalla
1", Icons.Filled.Home )
    object Pantalla2 :
Destinations("pantalla2/?newText={newText}", "Pantalla 2",
Icons.Filled.Settings) {
        fun createRoute(newText: String) =
"pantalla2/?newText=$newText"
    }
    object Pantalla3 : Destinations("pantalla3", "Pantalla
3", Icons.Filled.Favorite)
    object Pantalla4 : Destinations("pantalla4", "Pantalla
4x", Icons.Filled.Face )

    object Pantalla5 : Destinations("pantalla5", "Pantalla
5x", Icons.Filled.AccountCircle )

}
```

Crear el archivo NavigationHost:

```
@Composable
fun NavigationHost(
    navController: NavHostController,
    darkMode: MutableState<Boolean>,
    modif:PaddingValues
) {

    NavHost(
        navController = navController, startDestination =
Destinations.Login.route
    ) {
        composable(Destinations.Login.route){
            LoginScreen(navigateToHome = {

navController.navigate(Destinations.Pantalla1.route)})
        }
        composable(Destinations.Pantalla1.route) {
            Pantalla1(
                navegarPantalla2 = { newText -
>navController.navigate(Destinations.Pantalla2.createRoute(ne
wText))
                }
            )
        }
```

```
        composable( Destinations.Pantalla2.route,
            arguments = listOf(navArgument("newText") {
                defaultValue = "Pantalla 2"
            })
        ) { navBackStackEntry ->
            var newText =

navBackStackEntry.arguments?.getString("newText")
            requireNotNull(newText)
            Pantalla2(newText, darkMode)
        }
        composable(Destinations.Pantalla3.route) {
Pantalla3() }
        composable(Destinations.Pantalla4.route) {
Pantalla4() }

        composable(Destinations.Pantalla5.route) {
Pantalla5() }

    }
}
```

Finalmente adecuamos el archivo MainActivity

Paso 1: Agregar por encima del nombre de la clase la siguiente anotación:

```
@AndroidEntryPoint
```

Paso 2: Dentro del archivo agregar el siguiente compose:

```
@Composable
fun MainScreen(
    navController: NavHostController,
    darkMode: MutableState<Boolean>,
    themeType: MutableState<ThemeType>
) {
    val drawerState = rememberDrawerState(initialValue =
DrawerValue.Closed)
    val scope = rememberCoroutineScope()
    val openDialog = remember { mutableStateOf(false) }
    val navigationItems = listOf(
        Destinations.Pantalla1,
        Destinations.Pantalla2,
        Destinations.Pantalla3,
```

```
        Destinations.Pantalla4,
        Destinations.Pantalla5,
        //Destinations.ActividadUI,
        //Destinations.MaterialesxUI,
        //Destinations.PantallaQR
    )
    val navigationItems2 = listOf(
        Destinations.Pantalla1,
        Destinations.Pantalla2,
        Destinations.Pantalla3,
    )
    val currentNavBackStackEntry by
navController.currentBackStackEntryAsState()
    val currentRoute =
currentNavBackStackEntry?.destination?.route ?:
    Destinations.Pantalla1.route
    val list = currentRoute.split("/", "?")
    ModalNavigationDrawer(
        drawerContent = {
            AppDrawer(route = list[0], scope = scope,
scaffoldState =
            drawerState,
                navController = navController, items =
navigationItems)
        },
        drawerState = drawerState) {
        val snackbarHostState = remember {
SnackbarHostState() }
        val snackbarMessage = "Succeed!"
        val showSnackbar = remember { mutableStateOf(false) }
        val context = LocalContext.current
        val fabItems = listOf(
            FabItem(
                Icons.Filled.ShoppingCart,
                "Shopping Cart"
            ) {
                val toast = Toast.makeText(context, "Hola
Mundo",
                    Toast.LENGTH_LONG) // in Activity
                toast.view!!.getBackground().setColorFilter(
                    Color.CYAN,
                    PorterDuff.Mode.SRC_IN)
                toast.show()
            },
            FabItem(
                Icons.Filled.Favorite,
                "Favorite"
```

```
            ) { /*TODO*/ }
        )
        Scaffold(
            topBar = { CustomTopAppBar(
                list[0],
                darkMode = darkMode,
                themeType = themeType,
                navController = navController,
                scope = scope,
                scaffoldState = drawerState,
                openDialog={openDialog.value=true}
            )
            }, modifier = Modifier,
            /*floatingActionButton = {
            MultiFloatingActionButton(
            navController=navController,
            fabIcon = Icons.Filled.Add,
            items = fabItems,
            showLabels = true
            )
            },
            floatingActionButtonPosition = FabPosition.End,
            bottomBar = { BottomAppBar {
            BottomNavigationBar(navigationItems2,
navController =
            navController)
            }}*/
        ) {
            NavigationHost(navController, darkMode, modif= it
)
        }
    }
    Dialog(showDialog = openDialog.value, dismissDialog = {
        openDialog.value = false })
}
```

Paso 3: Adecuar contenido dentro de Setcontent que se ubica dentro de la función oonCreate:

```
val themeType= remember{ mutableStateOf(ThemeType.RED) }
val darkThemex= isNight()
val darkTheme = remember { mutableStateOf(darkThemex) }
val colorScheme=when(themeType.value){
    ThemeType.PURPLE->{if (darkTheme.value) DarkPurpleColors
else
```

```
            LightPurpleColors}
        ThemeType.RED->{if (darkTheme.value) DarkRedColors else
            LightRedColors}
        ThemeType.GREEN->{if (darkTheme.value) DarkGreenColors
else
            LightGreenColors}
        else->{LightRedColors}
    }
    TokenUtils.CONTEXTO_APPX=this@MainActivity
    AsistenciaUPeUJCNTheme(colorScheme=colorScheme) {
        Surface(
            modifier = Modifier.fillMaxSize(),
            color = MaterialTheme.colorScheme.background
        ) {
            //Greeting("Android")
            val navController= rememberNavController()
            MainScreen(navController, darkMode = darkTheme,
                themeType=themeType)
        }
        /*Scaffold(modifier = Modifier.fillMaxSize()) {
    innerPadding ->
            Greeting(
                name = "Android",
                modifier = Modifier.padding(innerPadding)
            )
        }*/
    }
```