Actividad 3.9. Light & Dark

Básicamente, un tema oscuro muestra principalmente superficies oscuras con menor brillo y contrasta con un tema luminoso dominado por fondos blancos y superficies brillantes.

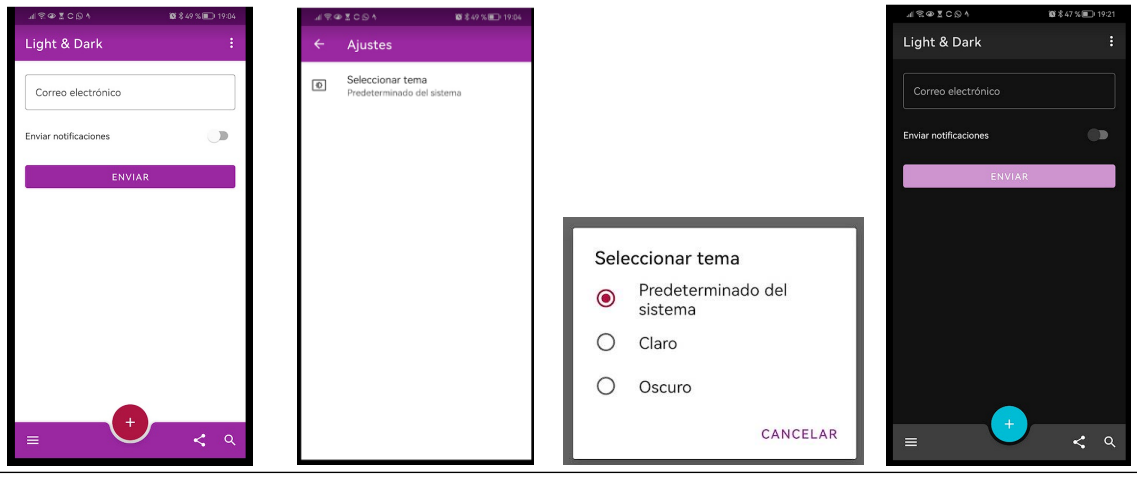
El usuario puede tener la opción de seleccionar un tema claro u oscuro, o que la aplicación tome por defecto el tema que tenga seleccionado el usuario en el sistema operativo.

Por ejemplo, Spotify en 2014, rediseñó sus aplicaciones para darles un estilo común basado en una estética oscura que pervive hasta hoy. Incluso se convirtió en un elemento distintivo de la marca. La noticia «¿Por qué Spotify se volvió negro?», refiere los motivos del cambio. Uno de ellos: «Un juego de colores más oscuros acentúa las carátulas, las fotografías de los artistas y los botones de navegación más importantes, como el de reproducción.»

Obviando discusiones sobre si es mejor uno u otro estilo, nosotros ahora vamos a crear una pequeña App donde trabajaremos y diseñaremos ambos estilos y donde el usuario podrá cambiar de uno a otro según prefiera. Crea un proyecto y llámale “Light&Dark”:

1. Diseño de los 2 layout de la App y la PreferenceScreen/Fragmento

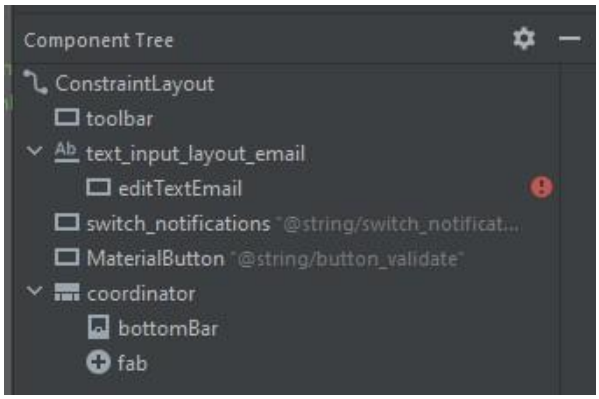
La aplicación constará de una layout principal, uno segundo layout para acceder al menú de seleccionar tema, y una pantalla de preferencias creada como “PreferenceScreen”:



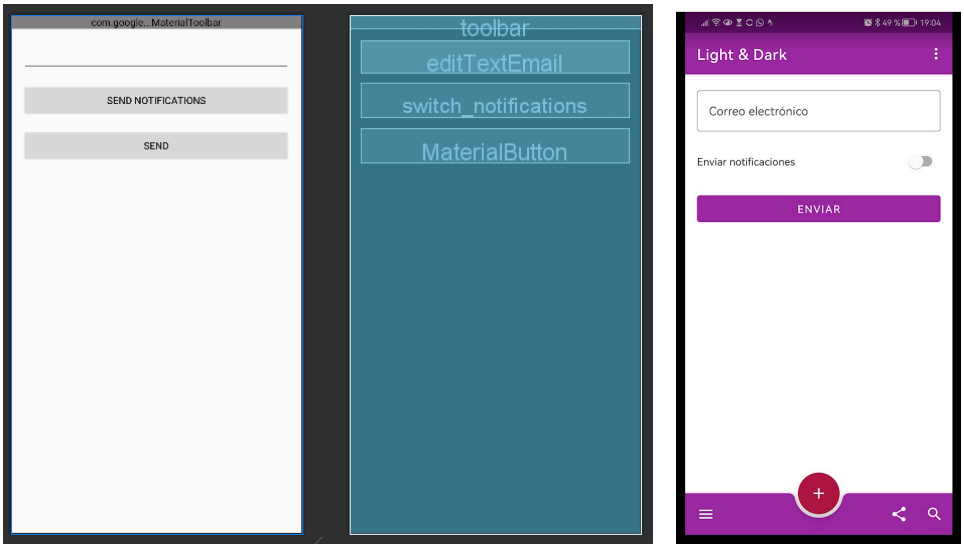
1.1. Layout/Ventana principal “activity_main”

La “activity_main”, será un **ConstraintLayout** que contiene estos componentes:

- Una **MaterialToolbar** que actúa de barra de la aplicación (action bar o app bar). Incluye un menú desplegable.
- Un campo de entrada **TextInputLayout** para escribir una dirección de correo electrónico.
- Un botón que verifica el formato del texto introducido en el **TextInputLayout.**
- Un **conmutador de opción (Switch).**
- Una barra inferior de tipo **BottomAppBar** con un botón flotante (FAB o Floating Action Button) incrustado.



Con el resultado de:



Vete insertando los elementos según estos parámetros principales (tienes que ir completando tú el resto (“Android:layout_xxxx”, “app:layout_constraintxxxxxx”, “android:hint”…). Los **iconos** descárgalos de esta actividad del Moodle (zip) y déjalos en la carpeta correcta:

```
<com.google.android.material.appbar.MaterialToolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/text_input_layout_email"
```

```
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/editTextEmail"
```

```
<com.google.android.material.switchmaterial.SwitchMaterial
    android:id="@+id/switch_notifications"
```

Además, se han introducido dos “onClick en los botones:

```
<com.google.android.material.button.MaterialButton
    android:onClick="validate"
    android:text="Send"
```

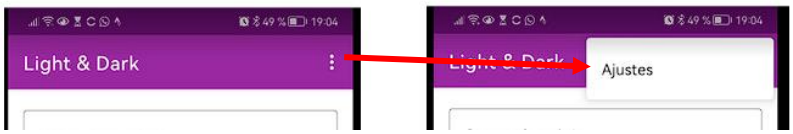
```
<com.google.android.material.button.MaterialButton
    android:onClick="validate"
    android:text="Send"
```

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/coordinator"
```

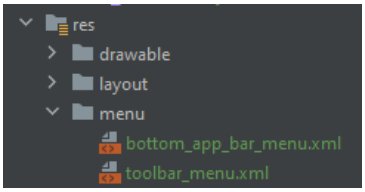
```
<com.google.android.material.bottomappbar.BottomAppBar
    android:id="@+id/bottomBar"
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab"
```

- Ahora también vamos a crear la ventana que sale al pulsar los “3 puntos” de la barra de menú:



Para ello crea una carpeta dentro de “res” y llámala “toolbar_menu”:




```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:title="Settings"
        app:showAsAction="never" />

</menu>
```

SettingsActivity

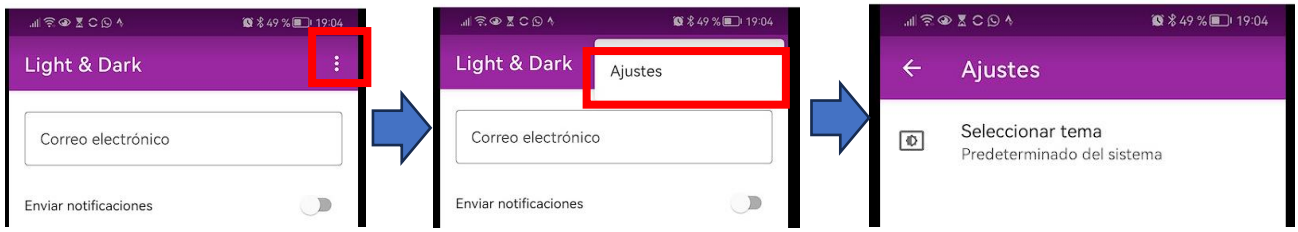
- Y lo mismo para los botones que aparecen en la parte de debajo de la ventana principal:

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/bottom_app_bar_menu_share"
        android:title="Share"
        android:icon="@drawable/ic_share"
        app:showAsAction="always"/>

    <item
        android:id="@+id/bottom_app_bar_menu_search"
        android:title="Search"
        android:icon="@drawable/ic_search"
        app:showAsAction="always"/>
</menu>
```

bottom_app_bar_menu

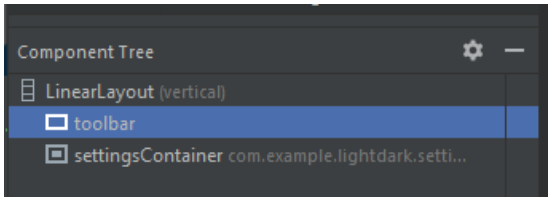
1.2. Layout/Ventana de "Settings". "activity_settings"

Desde la layout principal, a través de un botón iremos a esta segunda layout donde aparecerá el menú para seleccionar el tema que queremos elegir:



Activity_settings.xml

Será un "LinearLayout" con una "**MaterialToolbar**" y un "**FragmentConntainerView**", ambos en el **grupo Containers** de la ventana diseño "**Palette**":



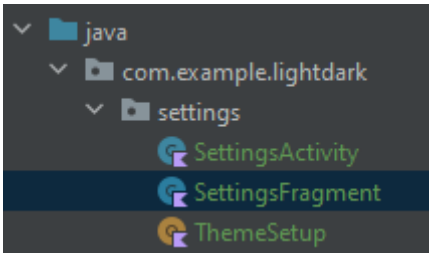
La vista **FragmentContainerView** es una vista personalizada que deriva de FrameLayout y se utiliza como contenedor recomendado para fragmentos. A diferencia de otros ViewGroups (FrameLaout, LinearLoyout, ...), FragmentContainerView solo acepta vistas de fragmentos.

```
<com.google.android.material.appbar.MaterialToolbar
    android:id="@+id/toolbar"
    navigationIcon="?attr/homeAsUpIndicator"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize" />
```

Y el contenedor del fragmento:

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/settingsContainer"
    android:tag="settings_fragment_tab"
    android:name="com.example.lightdark.settings.SettingsFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

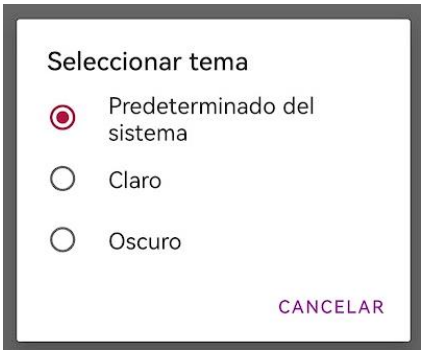
Donde vemos que el fragmento será "com,example.lightdark.sttings.SettingsFragment", el cual tenemos que crear. Para ello, crearemos una carpeta dentro de la parte "lógica/kotlin" del proyecto y la llamaremos "settings", dentro de ella, crearemos varios archivos, entre ellos, el que nos interesa ahora, "**SettingsFragment**":



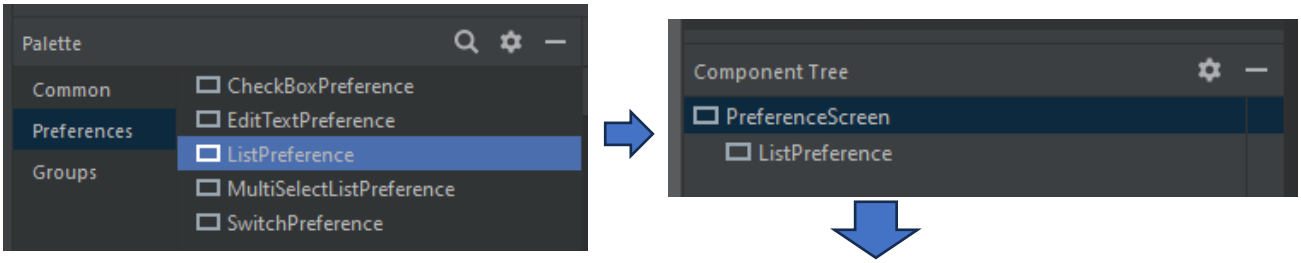
Con el **código kotlin** para habilitarlo:

```kotlin
package com.example.lightdark.settings

import ...


new *
class SettingsFragment : PreferenceFragmentCompat() {
    new *
    override fun onCreatePreferences(savedInstanceState: Bundle?, rootKey: String?) {
        setPreferencesFromResource(R.xml.menu_settings, rootKey)

        findPreference<Preference>(getString(R.string.settings_theme_key))?.setOnPreferenceChangeListener { preference, newValue ->
            if (newValue is String) {
                ThemeSetup.applyTheme(newValue, requireContext())
                true ^setOnPreferenceChangeListener
            } else {
                false ^setOnPreferenceChangeListener
            }
        }
    }
}
```

Como ves, aquí llama al archivo xml "menú_settings", que es donde se encontrará el contenido del fragmento, que en este caso será esta ventana:



Esta lista de selección, denominada **"ListPreference**", la cual la tienes que crear en un archivo xml del tipo "**XML Resource File**" que tienes que crear dentro la carpeta **res\values\xml,** al que le tines que llamar "menu_settings". Abriendo el archivo, fíjate que el tipo de ventana es una "**PreferenceScreen**" y dentro de ella el contenido que tienes que introducir es una "**ListPreference**" (presenta una lista de opciones en un cuadro de diálogo. De todas ellas, el usuario elegirá una, si así lo desea):



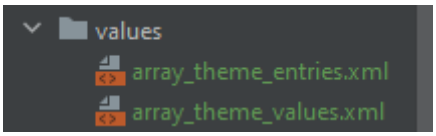

```
<androidx.preference.ListPreference
    android:key="theme"
    android:title="Choose theme"
    android:icon="@drawable/baseline_settings_brightness_24"
    android:dialogTitle="Choose theme"
    app:useSimpleSummaryProvider="true"
    android:entries="@array/settings_theme_entries"
    android:entryValues="@array/settings_theme_values"
    android:defaultValue="DEFAULT"/>
```

Solo visualizamos este aspecto:

Pero en el Código, hacemos ya mención de 2 campos importantes, el nombre que el usuario verá en la pantalla y el valor :

a. android:entries="@array/settings_theme_entries" → Será el "texto" de la lista.
b. android:entryValues="@array/settings_theme_values" → Serán los "valores" que elegimos.

Por lo tanto tendremos que crear ahora esos 2 archivos dentro la carpeta **"res/values"**



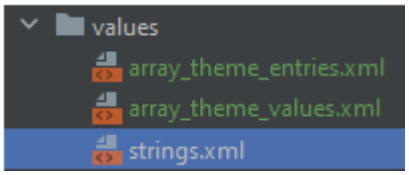
array_theme_entries.xml (fíjate que el name es el que buscamos de la "ListPreference", "entries)

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="settings_theme_entries">
        <item>Predeterminado del sistema</item>
        <item>Claro</item>
        <item>Oscuro</item>
    </string-array>
</resources>
```

array_theme_values.xml (fíjate que el name es el que buscamos de la "ListPreference", entryValues)

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="settings_theme_values">
        <item>@string/settings_theme_value_default</item>
        <item>@string/settings_theme_value_light</item>
        <item>@string/settings_theme_value_dark</item>
    </string-array>
</resources>
```

Estos valores que elegirá el usuario, los vamos a declarar como valores como constantes recuperables tanto en XML como en el código en el archivo **strings.xml:**



donde tendrías que colocar el siguiente código:

```xml
<string name="settings_theme_value_default" translatable="false">DEFAULT</string>
<string name="settings_theme_value_light" translatable="false">LIGHT</string>
<string name="settings_theme_value_dark" translatable="false">DARK</string>
```

Como el proyecto lleva más contenido en ese archivo y para no alargarlo, descarga el archivo "strings.xml" de Moodle y cárgalo en tu proyecto (te vendrá bien para sacar ideas e funcionamiento de la app).

La lógica para la ejecución de este fragmento se realiza a través del archivo "**SettingsFragment.kt**" que has creado antes.

Dimensiones App → dimens.xml

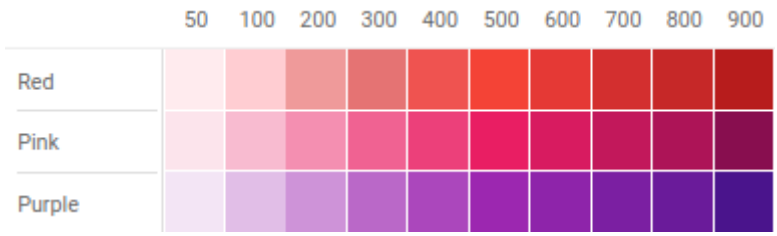
Para facilitar la gestión, la coherencia en la apariencia de la interfaz, y ajustar fácilmente estas dimensiones desde un solo archivo, vamos a crear el archivo "res\values\dimens.xml":

```xml
<resources>
    <!-- Márgenes de pantalla predeterminados -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

2. Tema claro y oscuro de la app

2.1. Colores

Como sabemos, los colores se definen en el fichero **/res/values/colors.xml,** en este caso vamos a utilizar 9 colores para toda la app, y los obtenemos de las paletas:



El valor que va de 50 a 900, define la intensidad del color, y utilizaremos ese criterio para el **nombre** del color que vayamos a utilizar:

```xml
<resources>
    <color name="purple_500" type="color">#9a28a1</color>
    <color name="purple_900" type="color">#6c0f74</color>
    <color name="purple_300" type="color">#cd94d0</color>

    <color name="red_500" type="color">#af1541</color>
    <color name="red_300" type="color">#c75b7a</color>

    <color name="white_50">#ffffff</color>

    <color name="black_900">#000000</color>

    <color name="grey_900">#252525</color>

    <color name="cyan_500">#00bcd4</color>

</resources>
```

colors.xml

Como vimos en actividades anteriores, puedes seleccionar los colores base en herramientas gratuitas como Figma, Material Design o Adobe color.

2.2. Temas

Como podremos seleccionar **3 temas** ("Predeterminado del sistema", "Claro" y "Oscuro"), lo que vamos a hacer es utilizar 1 tema para los 2 primeros (**res\values\themes.xml**) y un segundo tema para el 3º (**res\values-night\themes.xml**).

De **manera genérica**, la lista de temas de componentes de materiales que se podrían utilizar para obtener los últimos estilos de componentes y atributos a nivel de tema en "Material Design 2" serían:

- `Theme.MaterialComponents`
- `Theme.MaterialComponents.NoActionBar`
- `Theme.MaterialComponents.Light`
- `Theme.MaterialComponents.Light.NoActionBar`
- `Theme.MaterialComponents.Light.DarkActionBar`
- `Theme.MaterialComponents.DayNight`
- `Theme.MaterialComponents.DayNight.NoActionBar`
- `Theme.MaterialComponents.DayNight.DarkActionBar`

https://m2.material.io/develop/android/docs/getting-started

A) Establecemos los colores para el tema Claro/Oscuro "Predeterminado del sistema".
En este caso se va a utilizar "Theme.MaterialComponents.Light.NoActionBar".

El nombre del estilo (style name) define un tema llamado AppTheme.LightDark. El resto del fichero sobrescribe algunas de los numerosos ítems (colores, estilos de componentes, etcétera) de Theme.MaterialComponents.Light.NoActionBar. Dado que el tema padre ya provee un diseño visual completo, sobrescribe solo aquellos ítems que quieras personalizar

```
<!-- Tema base para el Claro-Oscuro (Predeterminado por el sistema) -->

<style name="Base.AppTheme.LightDark" parent="Theme.MaterialComponents.DayNight.NoActionBar">
    <item name="android:forceDarkAllowed" tools:targetApi="q">false</item>
    <item name="colorOnPrimary">@color/white_50</item>
    <item name="colorOnSecondary">@color/white_50</item>
    <item name="colorError">@color/red_300</item>
    <item name="bottomAppBarStyle">@style/Widget.MaterialComponents.BottomAppBar.PrimarySurface</item>

    <!-- toolbar -->

    <item name="toolbarStyle">@style/Widget.MaterialComponents.Toolbar.PrimarySurface</item>
    <item name="actionOverflowButtonStyle">@style/ToolbarStyle.Overflow</item>
    <item name="toolbarNavigationButtonStyle">@style/Toolbar.Button.Navigation.Tinted</item>

    <!-- menus -->

    <item name="popupTheme">@style/Theme.MaterialComponents.DayNight</item>
</style>

<style name="ToolbarStyle.Overflow" parent="Widget.AppCompat.ActionButton.Overflow">
    <item name="android:tint">@color/white_50</item>
</style>

<style name="Toolbar.Button.Navigation.Tinted" parent="Widget.AppCompat.Toolbar.Button.Navigation">
    <item name="tint">@color/white_50</item>
</style>
```

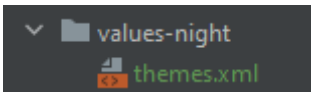
res\values\themes.xml (2)

B) Para el tema claro, vamos a crear un estilo personalizado para solo establecer los colores primarios y secundarios que vamos a necesitar:

```
<!-- Tema claro de la App)-->

<style name="AppTheme.LightDark" parent="Base.AppTheme.LightDark">
    <item name="colorPrimary">@color/purple_500</item>
    <item name="colorPrimaryVariant">@color/purple_900</item>
    <item name="colorSecondary">@color/red_500</item>
    <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
</style>
```

res\values\themes.xml (1)

C) Ahora hay que crear el tema para el modo oscuro en el archivo **res\values-night\themes.xml:**



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <!-- App theme (dark version)-->
    <style name="AppTheme.LightDark" parent="Base.AppTheme.LightDark">
        <item name="colorPrimary">@color/purple_300</item>
        <item name="colorPrimaryVariant">@color/purple_500</item>
        <item name="colorSecondary">@color/cyan_500</item>
        <item name="colorSurface">@color/grey_900</item>
        <item name="android:statusBarColor">@color/black_900</item>
    </style>

</resources>
```

res\values-night\themes.xml

3. AndroidManifest.xml

En este caso, en este archivo tendremos que tener en cuenta las 2 activities que tenemos creadas:

```xml
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

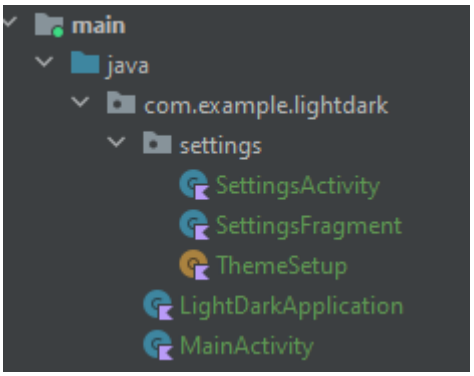
MainActivity

```xml
<activity
    android:name=".settings.SettingsActivity"
    android:label="Settings">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

SettingsActivity

4. Lógica (archivos kotlin del proyecto)

Los archivos que necesitamos para que la app funcione se compone de los siguientes archivos:



Para tenerlo organizado, los 3 archivos que necesitamos y que hacen referencia al **cambio de tema claro y oscuro,** están dentro la carpeta "**settings**"

A) SettingsFragment ya lo has creado antes.
B) SettingsActivity

```kotlin
new *
class SettingsActivity : AppCompatActivity() {
    new *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_settings)
        setSupportActionBar(findViewById(R.id.toolbar))
        supportActionBar!!.setDisplayHomeAsUpEnabled(true)
    }

    new *
    companion object {
        new *
        fun start(context: Context) {
            val intent = Intent(context, SettingsActivity::class.java)
            context.startActivity(intent)
        }
    }
}
```

C) ThemeSetup

```
new *
object ThemeSetup {
    new *
    fun applyTheme(mode: String?, context: Context) {
        if ("DARK" == mode) {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES)
        } else if ("LIGHT" == mode) {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO)
        } else {
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_FOLLOW_SYSTEM)
        }
    }

    new *
    fun applyTheme(context: Context) {
        val defaultSharedPreferences = PreferenceManager.getDefaultSharedPreferences(context)
        val value = defaultSharedPreferences.getString(
            "theme",
            "DEFAULT"
        )
        applyTheme(value, context)
    }
}
```

D) LightDarkApplication

```
package com.example.lightdark

import ...

new *
class LightDarkApplication : Application() {

    new *
    override fun onCreate() {
        super.onCreate()
        ThemeSetup.applyTheme(this)
    }
}
```

E) MainActivity:

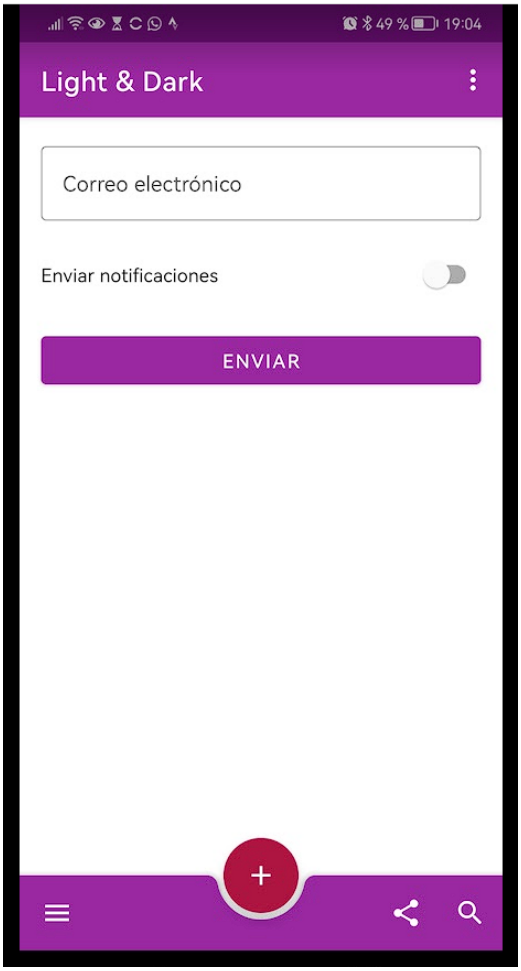
```
1  package com.example.lightdark
2
3  import android.content.Context
4  import android.os.Bundle
5  import android.text.TextUtils
6  import android.view.Menu
7  import android.view.MenuItem
8  import android.view.View
9  import android.view.inputmethod.InputMethodManager
10 import android.widget.EditText
11 import androidx.appcompat.app.AppCompatActivity
12 import androidx.appcompat.widget.Toolbar
13 import com.example.lightdark.settings.SettingsActivity
14 import com.google.android.material.textfield.TextInputEditText
15 import com.google.android.material.textfield.TextInputLayout
```

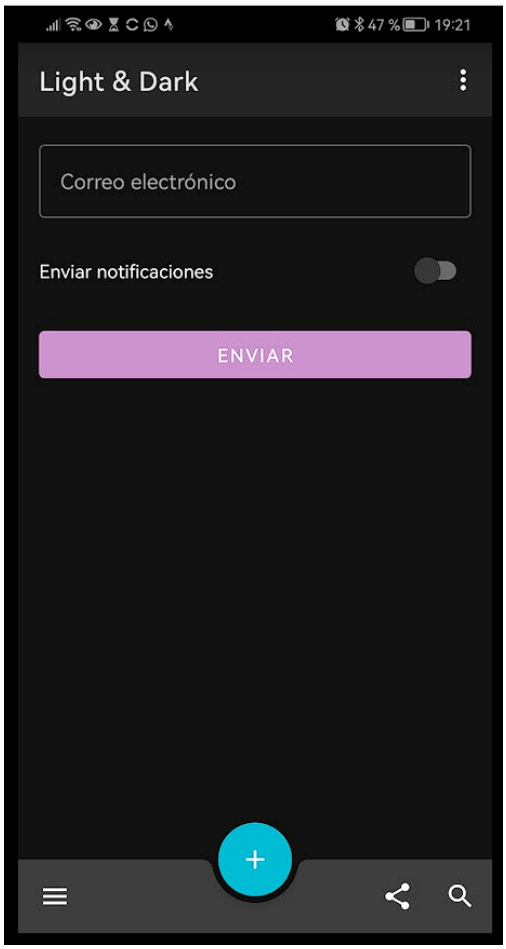
```
new *
17 class MainActivity : AppCompatActivity() {
18
19     private lateinit var editTextEmail: TextInputEditText
20     private lateinit var textInputEmail: TextInputLayout
21
       new *
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         setContentView(R.layout.activity_main)
25
26         val toolbar: Toolbar = findViewById(R.id.toolbar)
27         setSupportActionBar(toolbar)
28
29         editTextEmail = findViewById(R.id.editTextEmail)
30         textInputEmail = findViewById(R.id.text_input_layout_email)
31     }
32
       new *
33     override fun onCreateOptionsMenu(menu: Menu): Boolean {
34         menuInflater.inflate(R.menu.toolbar_menu, menu)
35         return super.onCreateOptionsMenu(menu)
36     }
```

```
    new *
33  override fun onCreateOptionsMenu(menu: Menu): Boolean {
34      menuInflater.inflate(R.menu.toolbar_menu, menu)
35      return super.onCreateOptionsMenu(menu)
36  }
37
    new *
38  override fun onOptionsItemSelected(item: MenuItem): Boolean {
39      SettingsActivity.start(this)
40      return true
41  }
42
    new *
43  fun validate(view: View) {
44      if (TextUtils.isEmpty(editTextEmail.text)) {
45          textInputEmail.error = "This field is mandatory"
46          textInputEmail.isErrorEnabled = true
47      } else {
48          textInputEmail.error = null
49          textInputEmail.isErrorEnabled = false
50      }
51
52      clearFocus()
53  }
```

```
    new *
55  private fun clearFocus() {
56      val view = currentFocus
57      if (view is EditText) {
58          val inputMethodManager =
59              getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
60          inputMethodManager.hideSoftInputFromWindow(view.windowToken,  flags: 0)
61          view.clearFocus()
62      }
63  }
64 }
65
```

Una vez completado todo, compila el proyecto para ver errores y prueba a visualizarlo en un dispositivo virtual, el resultado debería ser algo similar a esto:





Crea el archivo apk, el proyecto comprimido en zip y súbelos a Moodle.