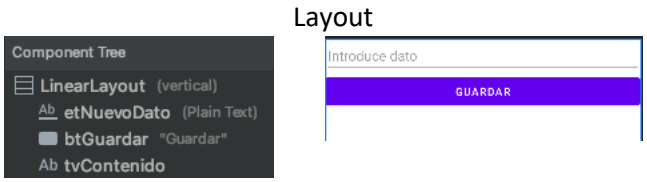
Práctica 1
Almacenamiento interno en aplicaciones móviles

Sigue los pasos que se describen para crear una aplicación móvil para Android empleando Kotlin que almacene un fichero en memoria interna y sube una memoria en la que incluyas los pantallazos y la respuesta a las preguntas:

1. Crea un proyecto nuevo a partir de un Empty Activity llamado AlmacenInterno (el dispositivo para el que desarrollamos es un Pixel 2 API 25 con minSdk 21 y targetSdk 32)
2. Incluye en el activity_main un Linear Layout simple como el que se propone (el diseño no es relevante en este ejemplo) y modifica los id's, textos y el hint del Plain Text para que coincidan con la imagen



3. Incluye el enlace entre vista y controlador que permita referirnos a los id's visuales. Es decir:
   a. Incluye en build.gradle (Module)

```
buildFeatures{
    viewBinding true
}
```

   b. Incluye en MainActivity.kt la variable binding y modifica setContentView de la siguiente manera:

```
class MainActivity : AppCompatActivity() {

    lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        //setContentView(R.layout.activity_main)
        setContentView(binding.root)

    }
}
```

   A partir de aquí podremos referirnos por ejemplo al botón con id btGuardar como binding.btGuardar

4. Incluye dentro de onCreate la respuesta al evento del botón Guardar con el siguiente código:

```
binding.btGuardar.setOnClickListener{
    guardar(binding.etNuevoDato.text.toString())
    binding.tvContenido.text = cargar()
}
```

5. Incluye el código de las funciones guardar y cargar que almacenan y muestran un fichero llamado datosInternos.txt en el que se incorpora el contenido que escribimos en la aplicación:

```
//Guardar recibe texto y lo almacena en la memoria interna
fun guardar(texto: String) {
    try {
        val stream: FileOutputStream = openFileOutput("datosInternos.txt",
            Context.MODE_APPEND)
        stream.use{ it.write(("$texto \n").toByteArray()) }
    } catch (e: Exception) {
        Toast.makeText(
            this,
            "No ha sido posible guardar",
            Toast.LENGTH_SHORT
        ).show()
    }
}

//Cargar devuelve lo almacenado en el fichero
fun cargar() :String{
    var texto = "" //Preparamos cadena vacía por si no conseguimos leer
    try {
        val fichero = BufferedReader(InputStreamReader(
            openFileInput("datosInternos.txt")))
        texto = fichero.use(BufferedReader::readText)
    }
    catch (e: Exception){
        Toast.makeText(
            this,
            "No ha sido posible leer el fichero ${e.toString()}",
            Toast.LENGTH_SHORT
        ).show()
    }
    return texto
}
```

6. Comprueba el funcionamiento de la aplicación y localiza dentro del Device File Explorer de Android Studio el fichero que hemos almacenado.
    a. Realiza un pantallazo del árbol de ficheros
    b. ¿En qué caso crees que es conveniente emplear este tipo de almacenamiento? ¿Qué alternativas tenemos? ¿Qué ventajas e inconvenientes tiene desde el punto de vista de la seguridad?
    c. ¿Cómo podría ser más seguro el almacenamiento interno?
    d. Especifica en AndroidManifest.xml que esta aplicación sólo puede instalarse en almacenamiento interno empleando para ello la opción:

```
installLocation="internalOnly"
```

7. Modifica las funciones guardar y cargar para que el almacenamiento se realice ahora en la memoria caché, de la siguiente manera:

```
//Guardar recibe texto y lo almacena en la memoria interna
fun guardar(texto: String) {
    try {
        val fichero = File(applicationContext.cacheDir, "ficheroCache.tmp")
        fichero.appendText("$texto \n")
    } catch (e: Exception) {
        Toast.makeText(
            this,
            "No ha sido posible guardar",
            Toast.LENGTH_SHORT
        ).show()
```

```
    }
}

//Cargar devuelve lo almacenado en el fichero
fun cargar() :String{
    var texto = "" //Preparamos cadena vacía por si no conseguimos leer
    try {

        val fichero = File(applicationContext.cacheDir, "ficheroCache.tmp")
        texto = FileInputStream(fichero).bufferedReader().use{
            it.readText()
        }
    catch (e: Exception){
        Toast.makeText(
            this,
            "No ha sido posible leer el fichero ${e.toString()}",
            Toast.LENGTH_SHORT
        ).show()
    }
    return texto
}
```

8. Localiza el fichero creado en el árbol de directorios mediante Device File Explorer:
    a. Realiza un pantallazo
    b. ¿Cuándo tiene sentido emplear la memoria caché?
    c. ¿Qué problemas podemos tener con este tipo de memoria?

9. Añade la funcionalidad necesaria a la aplicación para borrar la memoria caché mediante un botón. Para ello:
    a. Incorpora el elemento visual btBorrarCache
    b. Incorpora el código siguiente en MainActivity

```
binding.btBorrarCache.setOnClickListener{
    val fichero = File(applicationContext.cacheDir, "ficheroCache.tmp")
    if (fichero.exists()){
        fichero.delete()
        Toast.makeText(
            this,
            "La caché ha sido borrada",
            Toast.LENGTH_SHORT
        ).show()
    }
    Toast.makeText(
        this,
        "No existe ningún fichero en la caché",
        Toast.LENGTH_SHORT
    ).show()
}
```

    c. Comprueba que el fichero se borra

10. Modifica la aplicación para que el borrado de la caché se produzca automáticamente cuando se cierra la aplicación y comprueba su uso. Incorpora el código siguiente en MainActivity:

```
override fun onStop() {
    applicationContext.cacheDir.deleteRecursively()
    super.onStop()
}
```