# Case Test Android

Este documento contiene los casos de pruebas realizados en Android

## **Enunciado:**

* Desarrollar un app con Android studio que posea dos pantallas
  + La primer acivity debe poseer 2 botones y un textbox
    - Cuando se presione el botón 1 se debe publicar mensaje en el topico /casa/luz del bróker MQTT con URL:test.mosquitto.org
    - Se debe realizar la suscripción al topico /casa/cmd y que cada vez que se reciba mensaje de ese topico se muestre en el textbox
    - Cuando se presione el botón 2 se debe abrir la segunda Activity
  + La segunda actvity deberá cambiar su color de fondo,cada vez que se detecte un shake(agitación del smarptohne)

## **PROMPTS UTILIZADOS:**

* **LENGUAJE DE PROGRAMACION**:
  + Utilizar Java
* **PLATAFORMA:**
  + Android
* **FUNCIONALIDAD A IMPLEMENTAR:**
  + Desarrollar un app con Android studio que posea dos pantallas
* **REQUISITOS ESPECIFICOS:**
  + La primer acivity debe poseer 2 botones y un textbox
    - Cuando se presione el botón 1 se debe publicar mensaje en el topico /casa/luz del bróker MQTT con URL:test.mosquitto.org
    - Se debe realizar la suscripción al topico /casa/cmd y que cada vez que se reciba mensaje de ese topico se muestre en el textbox
    - Cuando se presione el botón 2 se debe abrir la segunda Activity
  + La segunda actvity deberá cambiar su color de fondo,cada vez que se detecte un shake(agitación del smarptohne)
* **RESTRICCIONES:**
  + Que la segunda pantalla vaya cambiando su fondo entre los colores rojo,negro,azul,verde y amarillo

## RESULTADOS OBTENIDOS CON CURSOR

1. Se ejecuto el prompt, indicándole que se base en los documentación con @docs de
   1. <https://gitlab.com/so-unlam/Material-SOA/-/tree/master/Ejemplos%20Android/Arduino-Android%20Mqtt>
   2. <https://developer.android.com>
2. Al utilizar el prompt adecuado se genero el prompt de una. Mientras que en ejecuciones previas generaba problemas de incompatibilidad en la biblioteca mqtt
3. No es muy generado código en kotlin, dado que varias incompatiblidades.

## RESULTADOS OBTENIDOS CON WINDSURF en Kotlin

1. Se ejecuto el prompt, indicándole que se base en los documentación con @web de
   1. <https://gitlab.com/so-unlam/Material-SOA/-/tree/master/Ejemplos%20Android/Arduino-Android%20Mqtt>
   2. <https://developer.android.com>
2. Al utilizar el prompt adecuado se genero el prompt de una.Se tuvo que realizar algunas refinamientos en el prompt, pero finalmente se pudo generar el programa correctamente
3. Se utilizo el lenguaje kotlin usando el modelo SWE-1

Si bien Windfurf es más lento que cursor para generar el código, es mucho mas estable que cursor. No necesita tanto refinamiento.

## RESULTADOS OBTENIDOS CON WINDSURF en Java

1. Se ejecuto el prompt, indicándole que se base en los documentación con @web de
   1. <https://gitlab.com/so-unlam/Material-SOA/-/tree/master/Ejemplos%20Android/Arduino-Android%20Mqtt>
   2. <https://developer.android.com>
2. Al utilizar el prompt adecuado se genero el prompt de una.Se tuvo que realizar algunas refinamientos en el prompt, pero finalmente se pudo generar el programa correctamente
3. Se utilizo el lenguaje Java usando el modelo GPT4.1

Si bien Windfurf es más lento que cursor para generar el código, es mucho mas estable que cursor. No necesita tanto refinamiento.

## RESULTADOS OBTENIDOS CON GITUHBCOPILOT

1. Se utilizo el plugin de githubcopilot de AndroidStudio
2. Se utilizaron los modelos de GPT4.1 y de CloudSonnet 3.5
3. Se ejecuto el prompt indicado
4. Github Copilot no genera el proyecto automáticamente, pero si te genera el código de los archivos. Luego el usuario debe ir copiándolo a mano en el proyecto de Android Studio.
5. Si bien genera el código del proyecto es necesario hacer sucesivas refinamientos, debido a que el código generado genera muchos problemas de dependencias y problemas durante la compilación.
6. Por este motiva se demora más tiempo que con Cursor y Windsurf en generar el proyecto funcional.

## RESULTADOS OBTENIDOS CON WORKIK AI

1. Se utilizo el plugin de Workik de Vscode, ya que desde su sitio web no funciona bien
2. Se utilizaron los modelos de GPT4.1
3. Se creo el proyecto en java, pero también se puede crear en kotlin
4. Se ejecuto el prompt indicado
5. Workik no genera el proyecto automáticamente, pero si te genera el código de los archivos. Luego el usuario debe ir copiándolo a mano en el proyecto de Android Studio.
6. Genero bien el código de una. Casi sin problemas de dependencias
7. El código generado funciona bien.