Pruebas y estándares de calidad

Dra. Mª del Puerto Paule Ruiz

Dpto. Informática

Conceptos previos

- Desarrollo iterativo
- Desarrollo de módulos. Módulo es una tarea específica que hace el usuario para completar la app.

Tipos de pruebas

- Pruebas unitarias (Pruebas de nivel inferior)
- Pruebas de integración (Pruebas de nivel intermedio)
- Pruebas de la interfaz de usuario (Pruebas de nivel superior)

Pruebas en Android

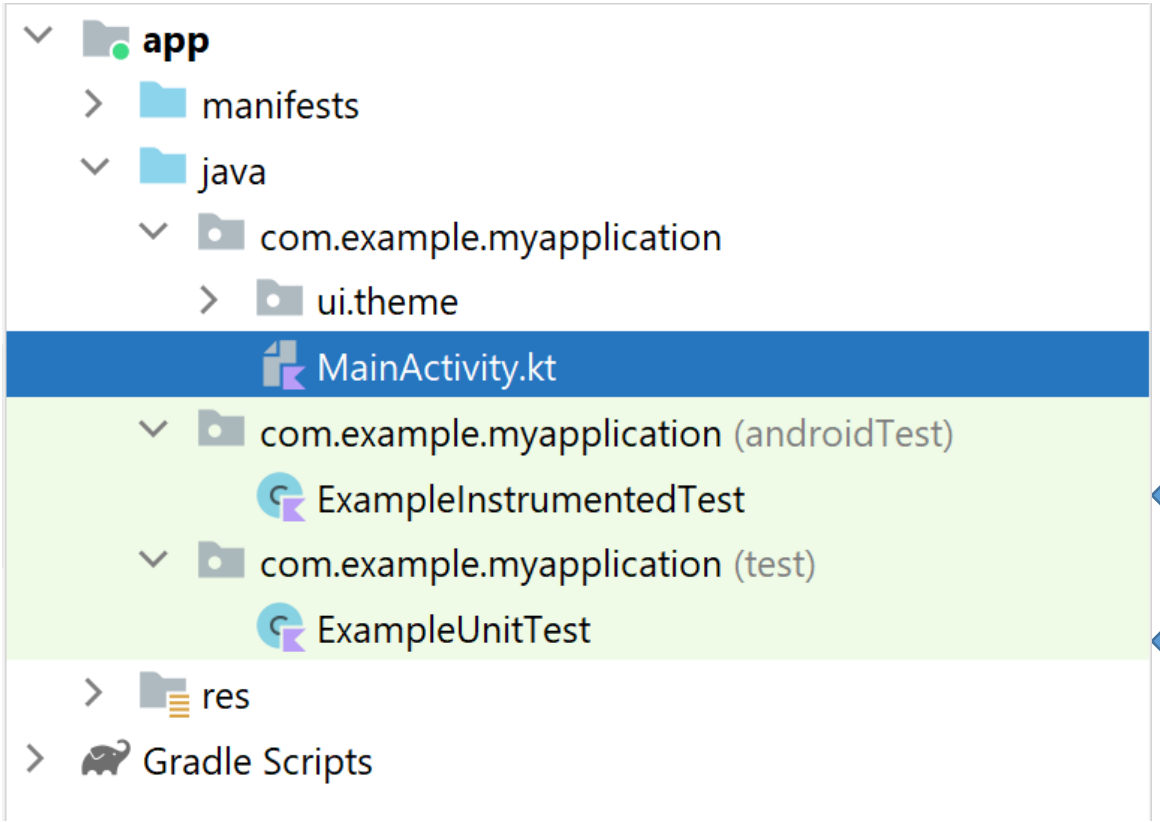
- Se aplican los mismos tipos de pruebas que otros sistemas
- Podemos dividir las pruebas por la infraestructura necesaria para llevarlas a cabo
  - Sólo máquina virtual de Java, si no probamos interfaz o características específicas de Android
  - Sistema android
- https://developer.android.com/training/testing/fundamentals

Configuración del entorno de pruebas

- Dos directorios
  - androidTest: pruebas que se ejecutan en dispositivos reales o virtuales
  - Test: Pruebas unitarias

  - Regla general: Mejor probar en el dispositivo físico. Si no es posible, entonces se puede usar el emulador cuando:
    - Operaciones largas, procesamiento de archivos grandes
    - Acciones no herméticas, conexión a un puerto abierto
    - Configuraciones difíciles de crear

Pruebas en la estructura de un proyecto



Test de instrumentación

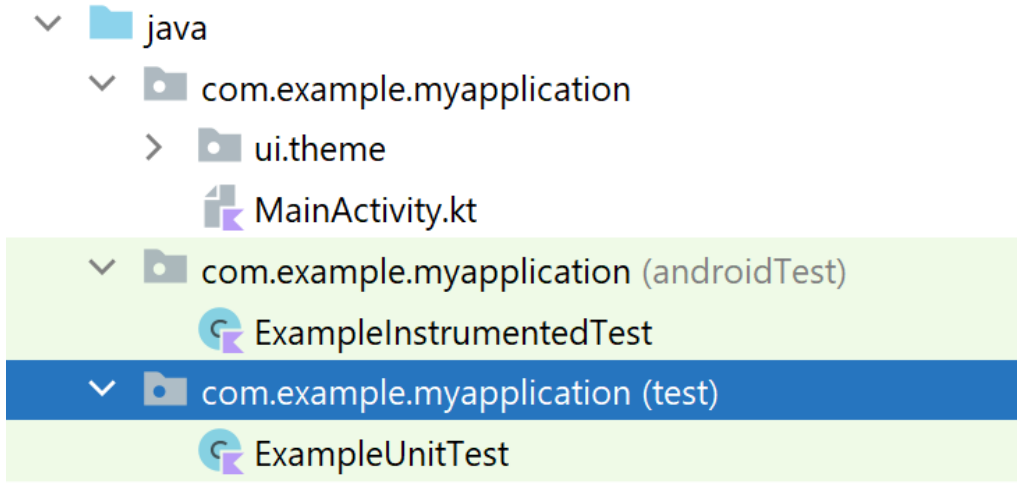
Test pruebas unitarias y de integración

Frameworks para hacer las pruebas

- JUnit , destinado a pruebas unitarias
- Mockito, Permite que las depedencias sean mockeadas, en vez de reales
- Espresso, destinado a pruebas de la aplicación incluyendo la interfaz. Su objetivo es evaluar las interacciones que hace el usuario con las vistas, así como el flujo de navegación en la UI. Adicionalmente, puede evaluar el contenido de las vistas.
- Robolectric, que no vamos a ver aquí

Pruebas unitarias

- Se denominan pruebas unitarias locales por el testImplementation:
- Anadir en el build.gradle del módulo de la app en dependencies




```
dependencies {
  // Required -- JUnit 4 framework
  testImplementation "junit:junit:4.13.2"
  // Optional -- Robolectric environment
  testImplementation "androidx.test:core:$androidXTestVersion"
  // Optional -- Mockito ramework
  testImplementation "org.mockito:mockito-core:$mockitoVersion"
  // Optional -- mockito-kotlin
  testImplementation "org.mockito.kotlin:mockito-
kotlin:$mockitoKotlinVersion"
  // Optional -- Mockk framework
  testImplementation "io.mockk:mockk:$mockkVersion"
}
```

Mockito

- Framework que permite simular las interacciones de una clase a probar con otras.
- Si quiero probar un método que depende de otro método, Mockito simulará una comunicación externa entre ambos.
- Añadir al gradle, en dependences:

```
// Optional -- Mockito framework
testImplementation "org.mockito:mockito-core:$mockitoVersion"
// Optional -- mockito-kotlin
testImplementation "org.mockito.kotlin:mockito-kotlin:$mockitoKotlinVersion"
```

Pruebas de integración

- Probamos varias entidades con sus relaciones
- Se realizarían después de las unitarias
- Aquí podemos mockear las llamadas a una API externa

Pruebas instrumentadas

- Comprueban el comportamiento de la interfaz gráfica
- Validan los flujos de ejecución de la aplicación
- Herramienta para hacer la prueba Espresso (https://developer.android.com/training/testing/espresso)
- Espresso realiza pruebas tanto en dispositivos físicos como virtuales y además en la nube como Firebase Test Lab.

Empezar con Espresso

- Las pruebas de Espresso consisten en dos componentes fundamentales: las interacciones y las aserciones de la IU en los View.
- Las interacciones incluyen las acciones que hace una persona cuando usa la app
- Aserciones verifican la existencia o el contenido de elementos visuales en la pantalla.
- Ejemplo, en la app de Login:
  - Interacciones: escribir en los campos de correo y password, además de hacer click en el botón
  - Aserciones: verificar la existencia del botón y de contenido en el correo y la password.
- IMPORTANTE: Desactivar las animaciones en el móvil para evitar resultados indeseados

Ejemplo- App que permita el login

- Partir de cero. No usar la plantilla.
- La idea es la siguiente:
  - Si el usuario es “maria” y la passwd es “123”, entonces lanza otra activity con un mensaje de “La prueba ha funcionado”, sino saca un mensaje “Ese usuario no existe”
  - Hacedlo bien y mal. Vamos a probar ambos casos de pruebas.

Grabadora de Espresso

- Grabadora de Espresso es una herramienta de ayuda para realizar las pruebas cuando no tienes ganas de crear pruebas a mano o bien no tienes tiempo a aprender el framework.
- Funciona muy bien cuando sólo tratamos de ver la interacción con la app y el contenido de las vistas.
- Otras situaciones más complejas, es mejor hacer las pruebas a mano.

Primeros pasos

- Si no hay un dispositivo disponible, lanza aquel donde quieras hacer las pruebas.
- En el menú de Run→ Record Espresso Test
- La grabadora hará una compilación

del proyecto, y la app se instala y se inicia en

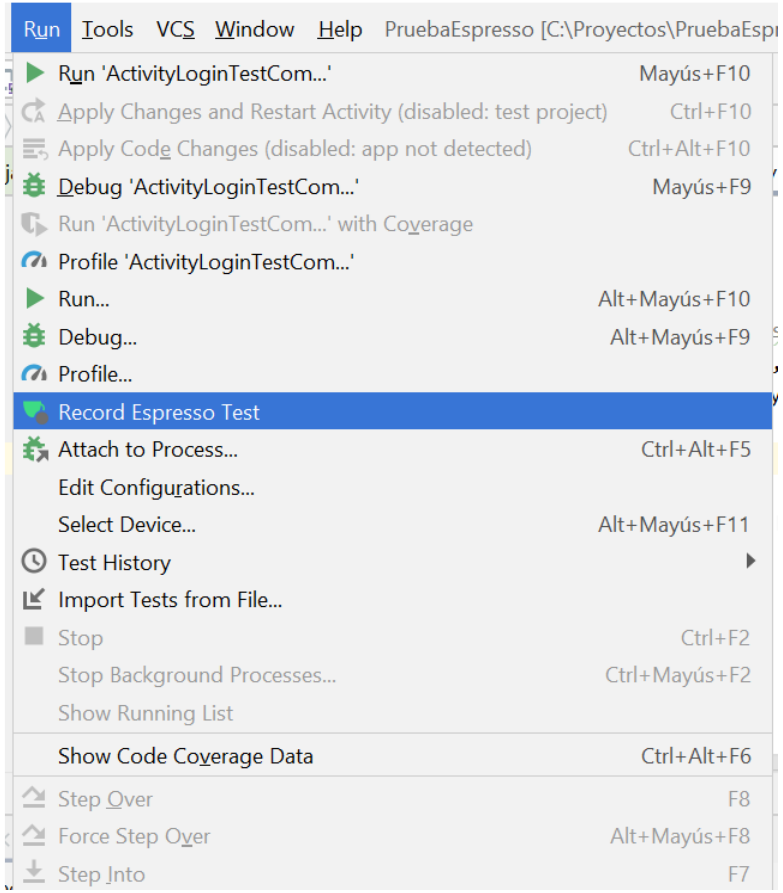
El dispositivo.

- En este momento, interacciona con ella

  para realizar la tarea.

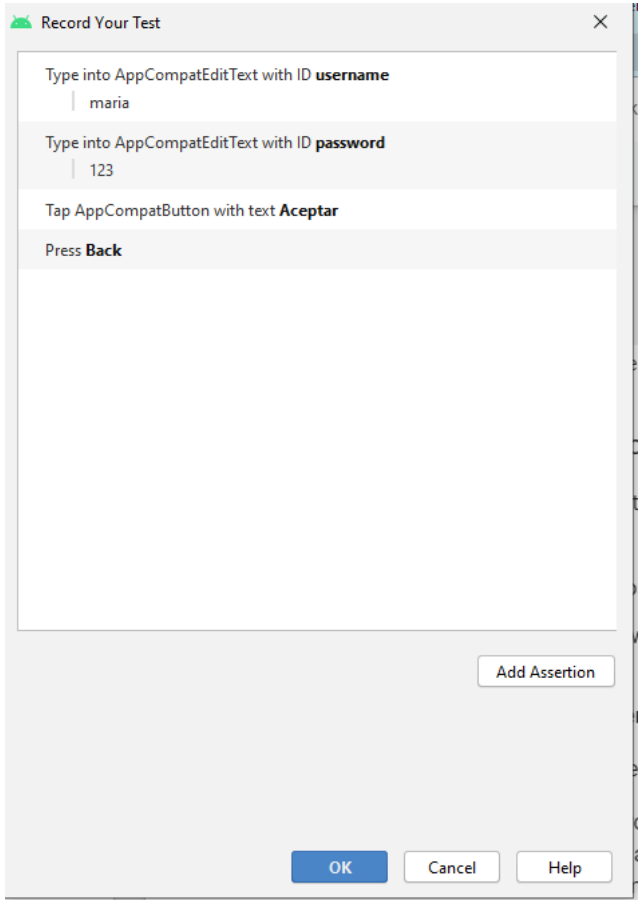
Importante:

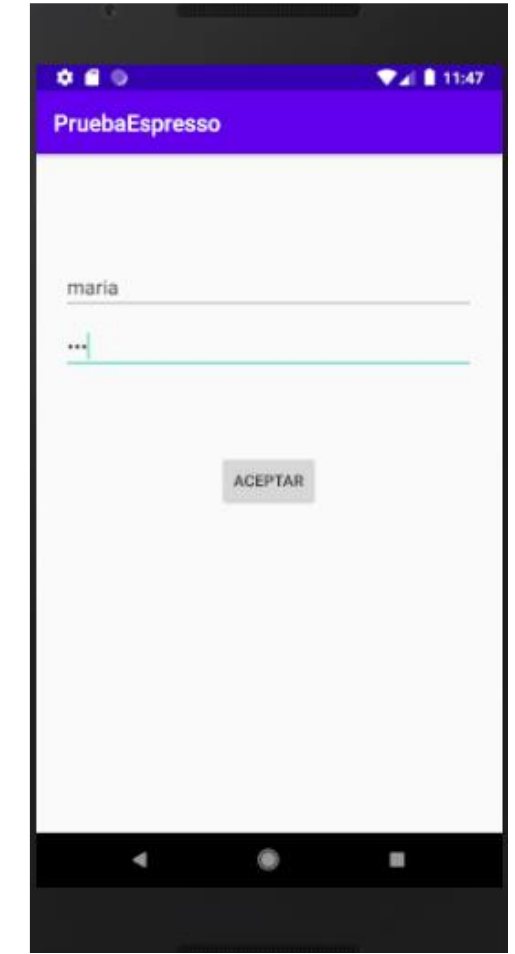
No presiones **Force Close** (Forzar cierre).



Ventana del Test Recorder

- Todas las acciones están quedando registradas en la ventana del Test Recorder

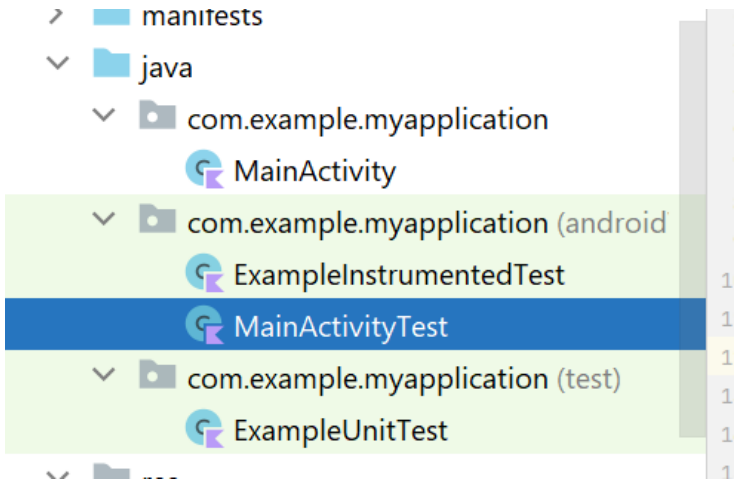


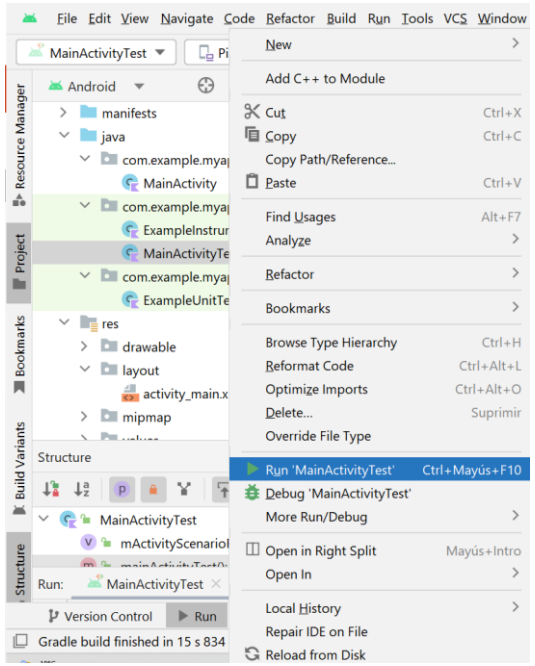


Grabar/ejecución Test

- Pulsar botón Ok de la ventana de registros
- Especificar un nombre de clase test para el test
- Ejecutar el test

Test grabados (creados)

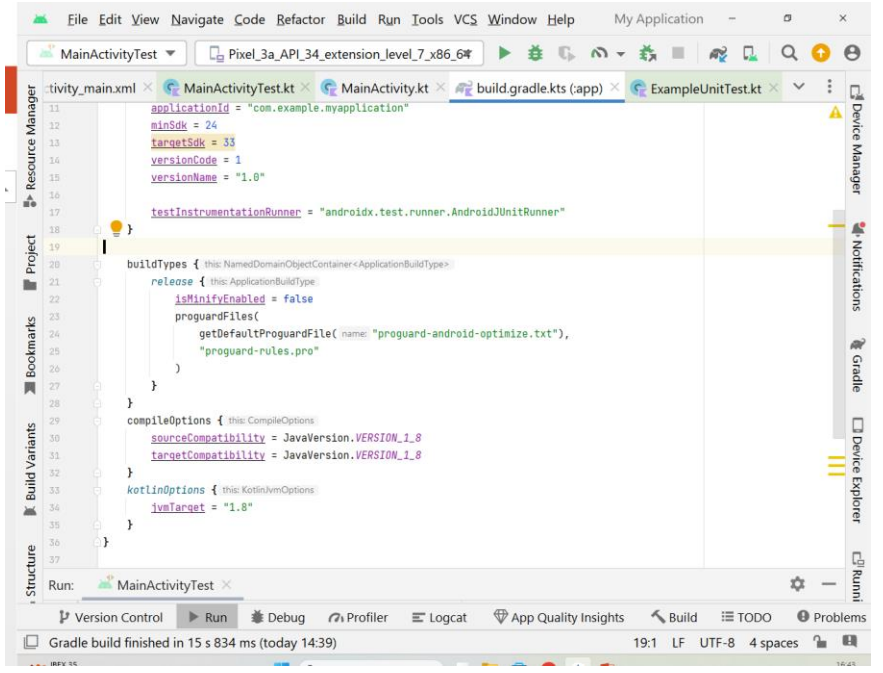


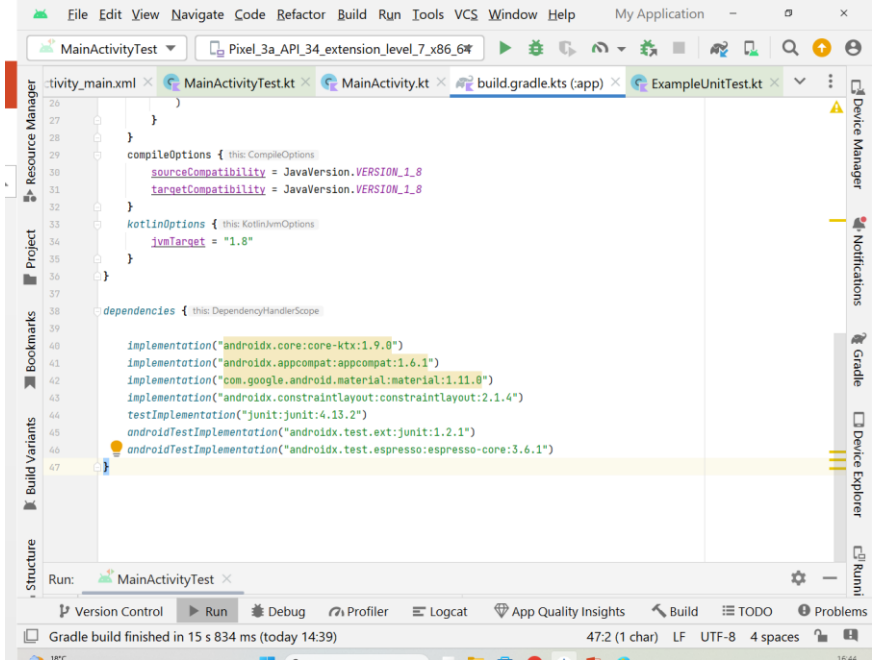


Ejecutar Test

Cambios en el proyecto

- En el AndroidManifiest.xml no hay cambios
- En el proyecto. En androidTest están las pruebas realizadas
- En el build.gradle (Module:app)





Assertions

- Se puede añadir aserciones para verificar los elementos de la UI
- La ventana del Test Recorder viene provista de tres tipos de aserciones que verifican la existencia o contenido de una View de tres formas:
  - **text is**: Chequea el contenido del texto de la View específica
  - **exists:** Comprueba que la View específica está presente en la jerarquía de Views visibles en la pantalla
  - **does not exist**: Comprueba lo contrario de exits

El resultado de la grabadora

- Se obtiene un código legible de una prueba que te permite iniciarte en las pruebas de IU.
- Ese código sirve de partida para ser modificado (si se desea) y adaptarlo a los casos de pruebas específicos que queremos hacer.

Paquetes Espresso

- espresso-core: Es lo básico. Tiene lo fundamental y básico para las Views, concretamente los matchers, actions y assertions.
- espresso-web: Contiene los recursos para el soporte del WebView
- espresso-idling-resource: Recoge la sincronización con trabajos en background
- espresso-intents: Validación de intents enviados por otras apps.

Más información en
https://developer.android.com/training/testing/espresso

Estándares de calidad en las apps de Android

- Permiten obtener aplicaciones móviles con la calidad necesaria para llegar a los clientes finales.
- Permiten garantizar que las apps, funcionen bien en diferentes dispositivos, cumplen las normas de Android para la navegación y diseño, y van a estar preparadas para las oportunidades de promoción en Google Play.
- Se deberían de aplicar antes de publicar la app.
- Android ofrece una serie de estándares y los criterios aplicados en cada de uno de ellos. Para verificar cada criterio, además proporciona pruebas que evalúan cada criterio.
- Más información en https://developer.android.com/quality?hl=es

Estándar 1- Diseño visual e interacción del usuario

- Los criterios aquí incluidos permiten mantener en la aplicación un aspecto visual e interacción estándar a todas las aplicaciones de Android.
- En este estándar se incluyen los siguientes criterios.
  - **Diseño estándar**: UX-B1.
  - **Navegación**: UX-N1, UX-N2, UX-N3.
  - **Notificaciones**: UX-S1, UX-S2.

Estándar 2- Funcionalidad

- Garantizar que la app proporciona la funcionalidad prevista con el nivel de permisos adecuado.
- Los criterios aplicados en este estándar son:
  - **Permisos**: FN-P1 y FN-P2.
  - **Ubicación de la instalación**: FN-L1.
  - **Audio**: FN-A1, FN-A2, FN-A3 y FN-A4.
  - **IU y gráficos**: FN-U1, FN-U2 y FN-U3.
  - **Estado del usuario o la app**: FN-S1 y FN-S2.

Estándar 3- Compatibilidad, rendimiento y estabilidad

- Garantizar la compatibilidad, rendimiento, estabilidad y capacidad de respuesta que los usuarios esperan.
- Los criterios aplicados son:
  - **Estabilidad**: PS-S1.
  - **Rendimiento**: PS-P1 y PS-P2.
  - **Batería**: PS-B1.
  - **Medios**: PS-M1.
  - **Calidad visual**: PS-V1 y PS-V2.

Estándar 4- Seguridad

- Garantizar que las apps manejan los datos de los usuarios y la información personal de manera segura
- Los criterios aplicados son
  - **Datos**: SC-D1, SC-D2, SC-D3 y SC-D4.
  - **Componentes de la app**: SC-P1, SC-P2, SC-P3
  - **Red**: SC-N1, SC-N2, SC-N3.
  - **Bibliotecas**: SC-U1.
  - **WebViews:** SC-W1, SC-W2, SC-W3
  - **Ejecución**: SC-E1.
  - **Criptografía**: SC-C1, SC-C2

Estándar 6- Google Play

- Garantizar que las apps estén listas para su publicación en el Google Play
- Los criterios aplicados son:
  - **Políticas**: GP-P1, GP-P2, GP-P3
  - **Página de detalles de la app**: GP-D1, GP-D2, GP-D3
  - **Asistencia para el usuario**: GP-X1

FAQs sobre estándares de calidad

- Se usan los estándares de calidad?
  - Sí. Se usan porque son fundamentales para garantizar que las aplicaciones que se distribuyen a través de Google Play Store ofrezcan una experiencia óptima a los usuarios.
- Se han de seguir?
  - Los desarrolladores deberían seguir estos estándares, ya que tienen un impacto directo en la aceptación de la aplicación en la Play Store y en la percepción que los usuarios tienen de la misma
- Qué garantizan?
  - Ayudan a mejorar la aceptación de la app en el mercado y asegura que las apps funcionen bien en el amplio ecosistema de dispositivos Android. Las aplicaciones que no siguen estos estándares corren el riesgo de sufrir bajas calificaciones, problemas de rendimiento, o incluso ser eliminadas de la tienda.