

# Estrategia de Pruebas

La presente estrategia de pruebas corresponde a la Iteración 1 del desarrollo de la aplicación de Vinilos. Para efectos de este desarrollo se cuenta con el back desplegado en [Heroku](#) el cual no estará sujeto a modificaciones. El proceso de desarrollo se centra en el Front de la aplicación.

## 1. Aplicación Bajo Pruebas

**1.1.** Nombre de la aplicación: Vinilos

**1.2.** Versión:1.0

**1.3.** Descripción:

Se quiere desarrollar una aplicación que proporcione al coleccionista aficionado la posibilidad de adquirirlos nuevos o usados, ya sea mediante compra o intercambio.

Se espera que por medio de este sistema los usuarios visitantes y los coleccionistas puedan conocer los vinilos disponibles. Por cada uno, la información de los músicos, las canciones y, para los artistas, los premios que el artista ha recibido (si es que tiene alguno). Adicionalmente, un coleccionista debe poder comprar, vender o intercambiar álbumes.

Un coleccionista puede ser comprador, vendedor o ambos. La información del coleccionista debe incluir su nombre, información de contacto (teléfono y correo electrónico) la cual es privada para el sistema, y sus artistas favoritos

La funcionalidad principal de la aplicación es permitir que un coleccionista, que cumpla el rol de vendedor, pueda registrar los álbumes que tiene para vender o hacer trueque; y quien cumpla el rol de comprador, pueda realizar la compra o el intercambio.

La aplicación debe ser muy llamativa para los usuarios e incluir una galería en donde sea fácil filtrar y buscar por distintos criterios como el nombre del album, del artista, de la casa discográfica o del género.

Cada álbum debe caracterizarse por un nombre, la imagen de la carátula, la fecha de salida al mercado, una descripción, el género, la casa discográfica, el artista (o lista de artistas) y el listado de tracks. Los usuarios registrados en la aplicación pueden hacer comentarios sobre los álbumes.

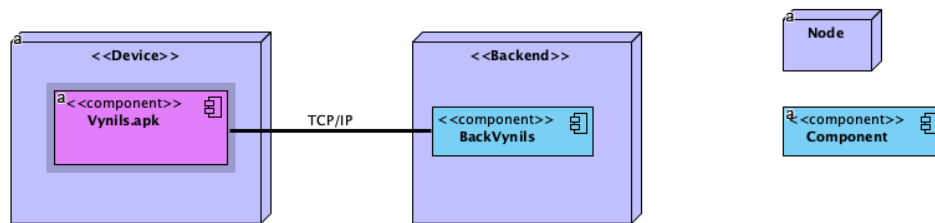
Del artista (que puede ser un músico o una banda) se requiere información como su nombre, una fotografía y un texto con una breve descripción. Si es una banda se debe conocer la fecha de formación, mientras que si es un músico se requiere su fecha de nacimiento.

También es importante conocer los premios que ha recibido el artista y la organización que otorga los premios.

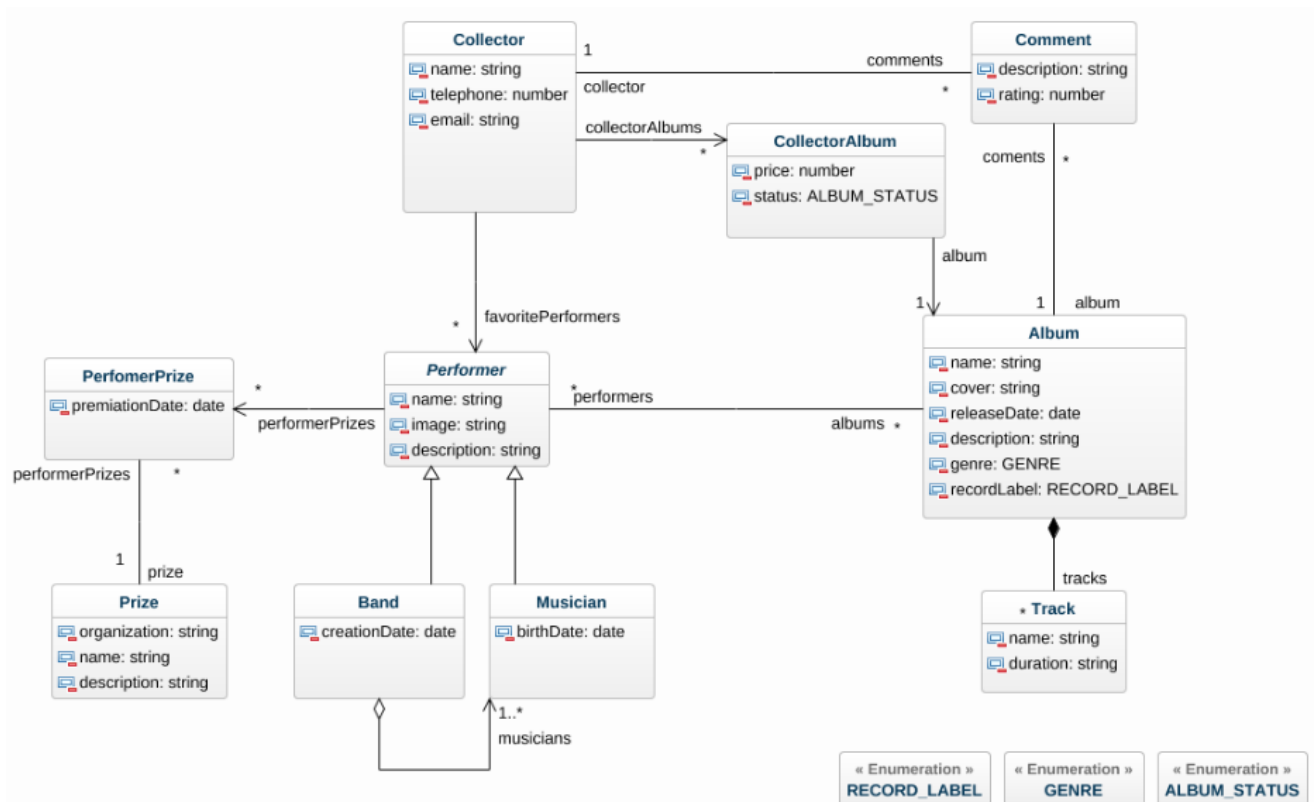
**1.4.** Funcionalidades Core:

- Albums
- Bands
- Collectors
- Comments
- Musicians
- Prices
- Tracks

**1.5.** Diagrama de Arquitectura (despliegue):



## 1.6. Diagrama de Datos:



## 1.7. Diseño UI/UX:

A continuación, se presenta el link donde se puede consultar y visualizar el [Diseño UI/UX](#)

## 2. Contexto de la estrategia de pruebas

### 2.1. Objetivos:

Para la presente estrategia de definieron los siguientes objetivos de acuerdo con las historias de usuario incluidas en el alcance de esta iteración:

#### Historias de Usuario:

- Consultar Catálogo de Álbumes
- Consultar Información Detalla de Álbum
- Consultar listado de Artistas
- Consultar información detalla de Artistas
- Consultar listado de coleccionistas

- Crear Premio
- Consultar la información detallada del coleccionista
- Crear Album

### **Objetivos:**

- Validar que cada uno de los requerimientos de los usuarios son tenidos en cuenta y desarrollados de la mejor forma (validación interna).
- Validar que cada una de las historias de usuario desarrolladas cumple con la tarea definida por los usuarios (validación externa).
- Garantizar la calidad de cada una de las funcionalidades del sistema implementando herramientas para mitigar los errores en desarrollo.
- Validar que se está construyendo el software a la medida de los requisitos de los usuarios y que cada una de las funcionalidades cumple con lo estipulado (validación interna).
- Verificar que el software construido cumple con las especificaciones de diseño y sigue estándares internos aplicando las buenas prácticas adoptadas.
- Analizar de forma estática cada uno de los componentes implementados mediante revisiones de código por parte de revisores pares del equipo de desarrollo.
- Analizar los escenarios de las pruebas manuales basados en los requerimientos y criterios de aceptación iniciales, garantizando que el sistema cumple con lo establecido.
- Implementar pruebas manuales en cada una de las funcionalidades basándose en los criterios de aceptación (AAT).
- Verificar el correcto funcionamiento de las historias de usuario desarrolladas mediante pruebas automatizadas creadas en Espresso.
- Estimar el desempeño de la aplicación haciendo uso de la funcionalidad *Profiler* disponible en Android Studio.
- Identificar y corregir problemas de calidad del código mediante un análisis estático del código realizado con la herramienta *Lint* disponible en Android Studio.
- Identificar el crashes y excepciones generadas en la aplicación mediante la implementación de Pruebas de Reconocimiento (Monkey y Firebase).
- Identificar las limitaciones que se pueden presentar al usar la aplicación mediante la generación de reportes de accesibilidad.

### **2.2. Duración de la iteración de pruebas:**

Aunque para Iteración se dispone de 2 semanas, el equipo realizó la planeación con el objetivo de realizar todas las actividades durante la primera semana esto para evitar, en lo posible, dejar actividades pendientes para la última semana.

La semana empieza con la realización de la retrospectiva del Sprint 2. A continuación se realiza el desarrollo de las historias pendientes (HU07 y HU08). Al finalizar el proceso de desarrollo se inicia el proceso de pruebas que incluye Pruebas de Reconocimiento, Pruebas de Accesibilidad y Pruebas de Desempeño. Finalmente, de acuerdo a los resultados obtenidos se realizan Micro-optimizaciones en el código.

	Mon, May 15 03:00:00	Tue, May 16 02:00:00	Wed, May 17 02:00:00	Thu, May 18 03:00:00	Fri, May 19 03:00:00	Sat, May 20 02:00:00	Sun, May 21 00:00:00
18:00	Retrospectiva Sprint 2 (Without pro) 01:00:00					Pruebas de Desempeño (Without pro) 01:00:00	
19:00						Micro-Optimizaciones (Without pro) 01:00:00	
20:00	HU07-Desarrollo Back (Without pro) 01:00:00	HU07-Desarrollo Front (Without pro) 01:00:00	HU08-Desarrollo Back (Without pro) 01:00:00	HU08-Desarrollo Front (Without pro) 01:00:00	HU07-Pruebas Espresso (Without pro) 01:00:00		
21:00	HU07-Desarrollo Back (Without pro) 01:00:00	HU07-Desarrollo Front (Without pro) 01:00:00	HU08-Desarrollo Back (Without pro) 01:00:00	HU08-Desarrollo Front (Without pro) 01:00:00	HU08-Pruebas Espresso (Without pro) 01:00:00		
22:00				Pruebas de Reconocimiento (Without pro) 01:00:00	Pruebas de Accesibilidad (Without pro) 01:00:00		
23:00							

## 2.3. Presupuesto de pruebas:

### 2.3.1. Recursos Humanos

Role	Experiencia Previa	Tiempo disponible
Desarrollador	– Experiencia en desarrollo de aplicaciones para backend en lenguajes Java, C#, Delphi y para frontend en Angular y Typescript, manejo de bases de datos relacionales SQL y Oracle, y experiencia en ambientes de nube con AWS.	4H
Desarrollador	– Experiencia en desarrollo Backend en lenguajes como Java, C#, .Net Core y Python, manejo de bases de datos como MySQL, PL/SQL, DB2 y PostgreSQL	4H
Desarrollador	– Experiencia en desarrollo de software con Typescript y Java, y del ciclo del software en general.	4H

Desarrollador	– Sin experiencia previa	4H
---------------	--------------------------	----

### 2.3.2. Recursos Computacionales

Dispositivos	Objetivo	Capacidad Computacional	Tiempo computacional
Laptop Personal	Ejecución de Pruebas Manuales	– Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz, 2304 Mhz, 4 procesadores principales, 8 procesadores lógicos	4 horas
Laptop Personal	Ejecución de Pruebas Manuales	– Processor 2,6 GHz 6-Core Intel Core i7 – Graphics Intel UHD Graphics 630 1536 MB – Memory 16 GB 2667 MHz DDR4 – MscOS Ventura 13.3.1	4 horas
Laptop Personal	Ejecución de Pruebas Manuales	– Processor 2,6 GHz 6-Core Intel Core i7 de seis núcleos – Graphics AMD Radeon Pro 5300M 4 – Intel UHD Graphics 630 1536 MB – Memory 16 GB 2667 MHz DDR4 – MscOS Ventura 13.3.1 –	4 horas
Laptop Personal	Ejecución de Pruebas Manuales	– Processor 2,6 GHz 6-Core Intel Core i7 – Graphics Intel UHD Graphics 630 1536 MB – Memory 16 GB 2667 MHz DDR4 – MscOS Ventura 13.3.1	4 horas

### 2.4. TNT (Técnicas, Niveles y Tipos) de pruebas:

A continuación, se especifican las técnicas, niveles y tipos de pruebas que serán implementadas en el presente presupuesto de pruebas:

Técnica	Nivel	Tipo	Objetivo
Pruebas Automatizadas	Sistema	Funcional/Positiva	Validar que cada uno de los requerimientos de los usuarios son tenidos en cuenta y desarrollados de la mejor forma (validación interna).
Profiler	Sistema	No Funcional	Estimar el desempeño de la aplicación haciendo uso de la funcionalidad <i>Profiler</i>

Plantilla elaborada por

**THE SW DESIGN LAB**

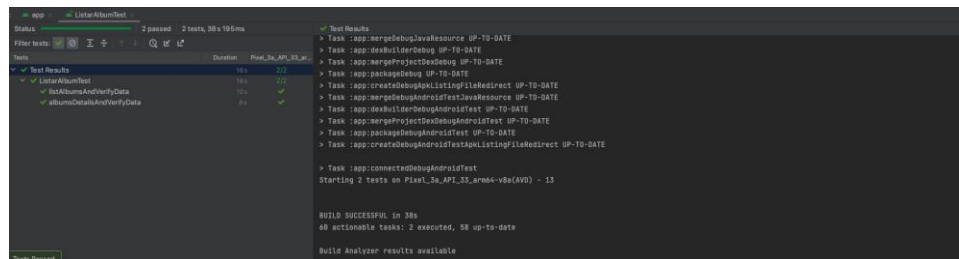
			disponible en Android Studio.
<b>Lint</b>	Sistema	No Funcional	Identificar y corregir problemas de calidad del código mediante un análisis estático del código realizado con la herramienta <i>Lint</i> disponible en Android Studio.
<b>Monkey</b>	Sistema	Funcional/Caja Negra	Identificar el crashes y excepciones generadas en la aplicación mediante la implementación de Pruebas de Reconocimiento (Monkey y Firebase).
<b>Firebase</b>	Sistema	Funcional/Caja Negra	Identificar el crashes y excepciones generadas en la aplicación mediante la implementación de Pruebas de Reconocimiento (Monkey y Firebase).
<b>Reporte de Accesibilidad</b>	Sistema	Funcional	Identificar las limitaciones que se pueden presentar al usar la aplicación mediante la generación de reportes de accesibilidad.

### 3. Resultados

#### 3.1. Resultados Pruebas Espresso

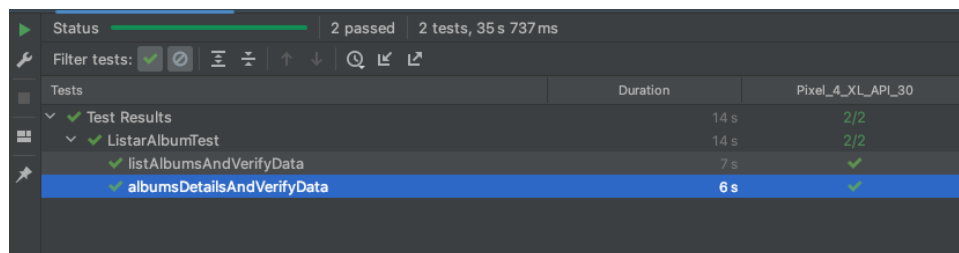
##### HU01- Consultar catálogo de álbumes

Se implementa prueba para listar los álbumes, se realiza E2E que navega a la lista de álbumes y verifica que se encuentre el total de álbumes y se evalúan atributos específicos de los álbumes presentes en la lista.



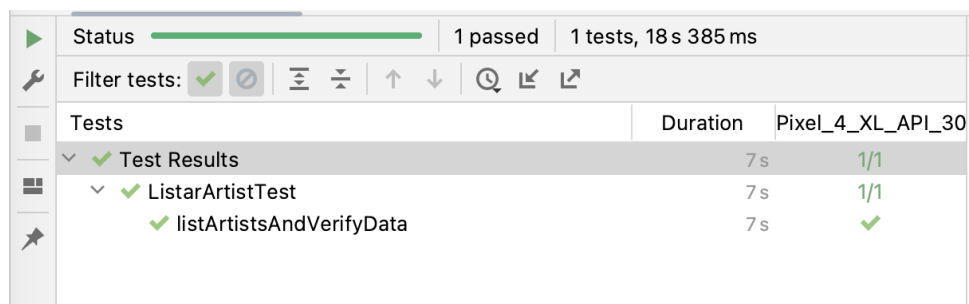
##### HU02 – Detalle del álbum

Se implementan las pruebas sobre la lista de los detalles del álbum. Las pruebas se realizan E2E de tal forma que la prueba inicia en la pantalla principal y navega hasta el detalle del álbum seleccionado, una vez en esta pantalla valida que los datos del álbum corresponden a los que se parametrizaron en la prueba.



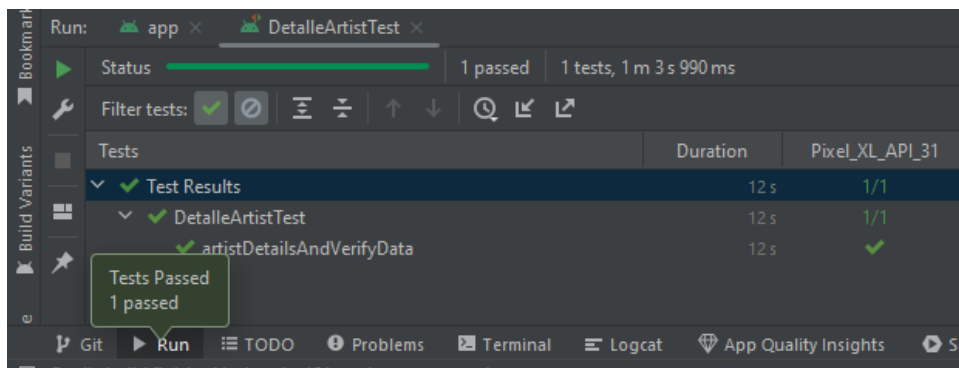
##### HU03- Consultar el listado de artistas

Se implementa las pruebas para la funcionalidad del listado de artistas. Las pruebas E2E se realizan de tal forma que permita consultar el listado de los artistas que llegan de la consulta al API del backend, y según la cantidad de artistas que lleguen en la consulta se procede a pintar los ítems de los artistas. Esta prueba valida el contenido de la consulta y valida que sea consistente con los parámetros indicados en la prueba.



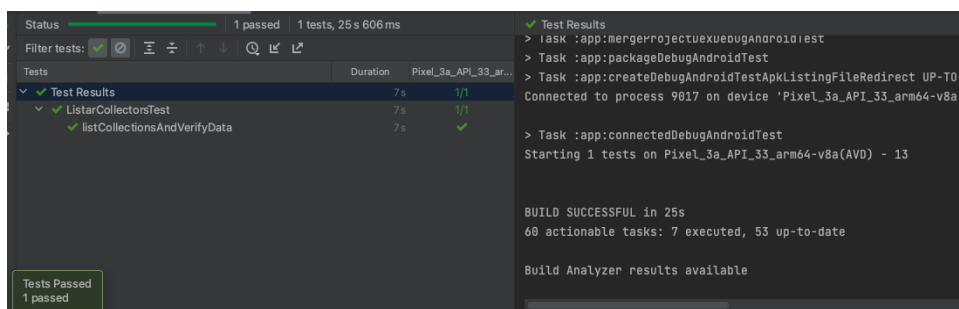
##### HU04- Consultar la información detallada de un artista

Se diseñan y ejecutan pruebas sobre el detalle de los artistas. Las pruebas se realizan E2E de tal forma que la prueba inicia en la pantalla principal y navega hasta el detalle del artista seleccionado. En la pantalla de detalle de artista se valida que el contenido de dicha pantalla sea consistente con los parámetros indicados en la prueba.



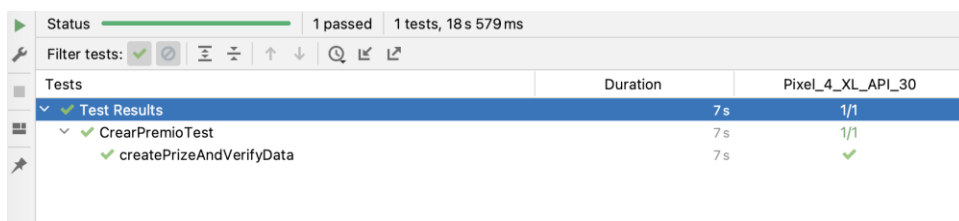
#### HU05- Consultar listado de coleccionistas

Se implementa prueba para listar los coleccionistas, se realiza E2E que navega a la lista de coleccionistas y verifica que se encuentre el total de coleccionistas y se evalúan atributos específicos de los coleccionistas presentes en la lista.



#### HU06- Crear Premio

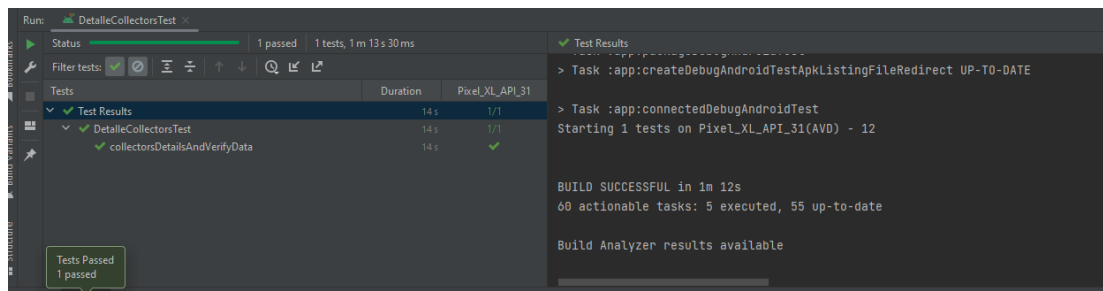
Se implementa prueba automatizada para crear premio en la aplicación. Esta funcionalidad hace parte del desarrollo de la historia de usuario HU06. En la prueba, se recorre la página o actividad y se encuentra coincidencia con los elementos que en ella se encuentran, tales como nombres de los campos de texto que ingresa el usuario como también el nombre de los botones de interacción para crear premio o cancelar el proceso.



#### HU07- Consultar la información detallada de coleccionista

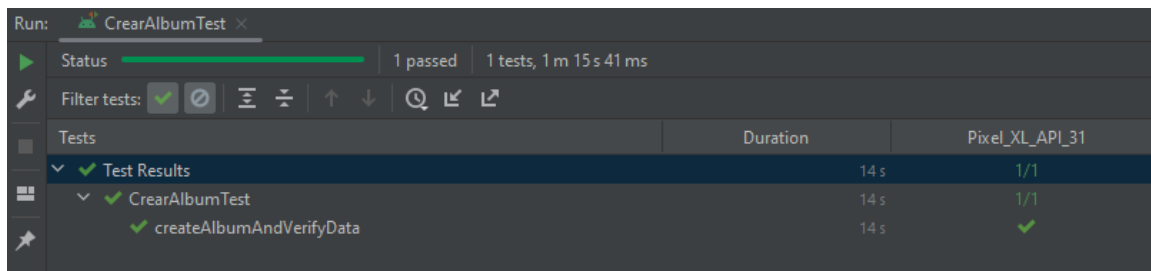
Se implementa prueba automatizada para la historia de *Consultar información detallada de coleccionista*. Las pruebas se realizan E2E de tal forma que la prueba inicia en la pantalla principal, pasa al listado de coleccionistas y navega hasta el detalle del coleccionista seleccionado. En la pantalla de detalle de coleccionista se valida que el contenido de dicha pantalla sea consistente con los parámetros indicados en la prueba.





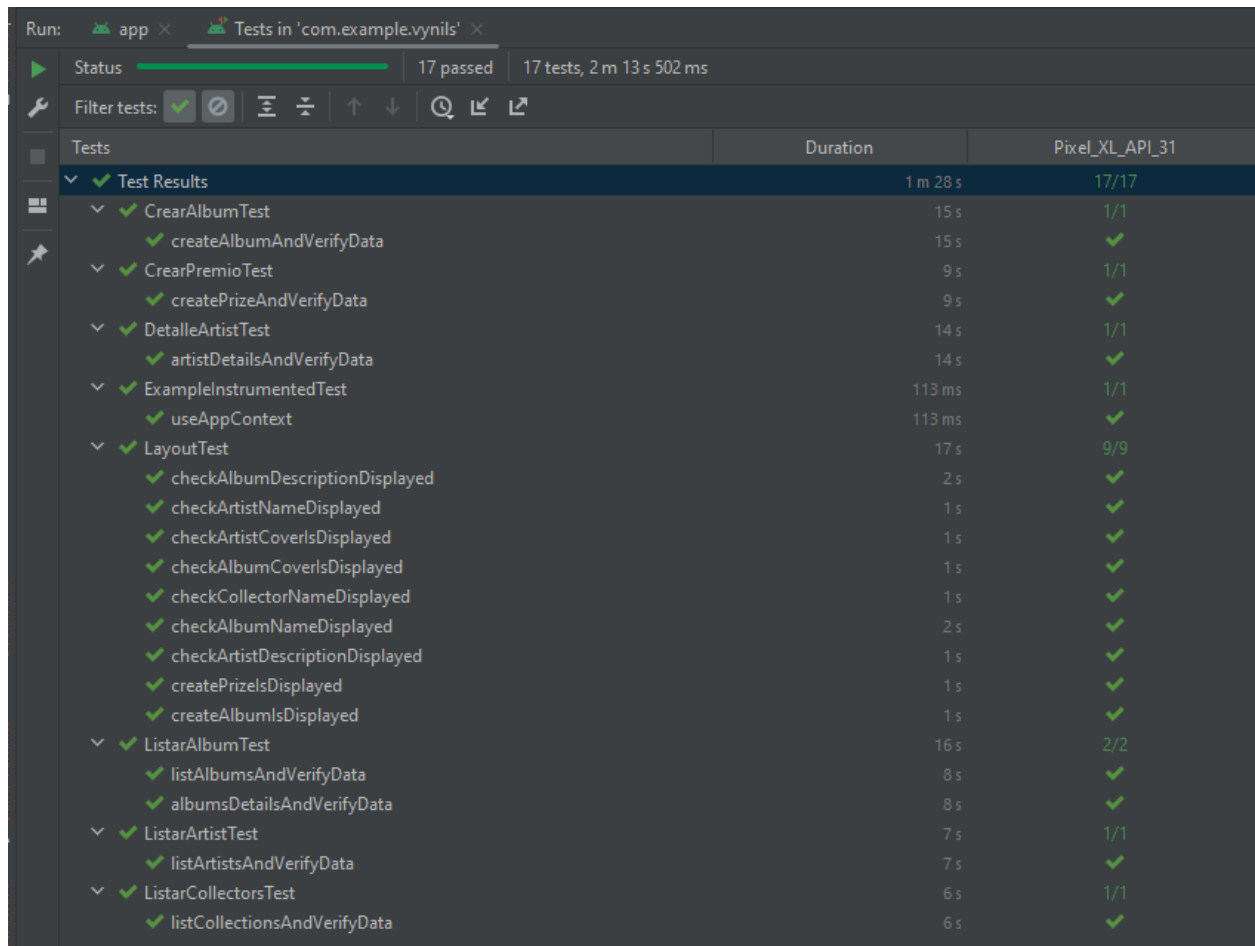
## HU08-Crear Album

Se implementa prueba automatizada para Crear Album en la aplicación. Esta funcionalidad hace parte del desarrollo de la historia de usuario HU08. En la prueba, se recorre la página o actividad y se encuentra coincidencia con los elementos que en ella se encuentran, tales como nombres de los campos de texto que ingresa el usuario como también el nombre de los botones de interacción para Crear Album.



## Total - Resultados de todas las pruebas hasta el momento

A continuación, se muestra el resultado de todas las pruebas que se han implementado hasta el momento en toda la aplicación.



Tests	Duration	Pixel_XL_API_31
✓ Test Results	1 m 28 s	17/17
✓ CrearAlbumTest	15 s	1/1
✓ createAlbumAndVerifyData	15 s	✓
✓ CrearPremioTest	9 s	1/1
✓ createPrizeAndVerifyData	9 s	✓
✓ DetalleArtistTest	14 s	1/1
✓ artistDetailsAndVerifyData	14 s	✓
✓ ExampleInstrumentedTest	113 ms	1/1
✓ useAppContext	113 ms	✓
✓ LayoutTest	17 s	9/9
✓ checkAlbumDescriptionDisplayed	2 s	✓
✓ checkArtistNameDisplayed	1 s	✓
✓ checkArtistCoverIsDisplayed	1 s	✓
✓ checkAlbumCoverIsDisplayed	1 s	✓
✓ checkCollectorNameDisplayed	1 s	✓
✓ checkAlbumNameDisplayed	2 s	✓
✓ checkArtistDescriptionDisplayed	1 s	✓
✓ createPrizesDisplayed	1 s	✓
✓ createAlbumsDisplayed	1 s	✓
✓ ListarAlbumTest	16 s	2/2
✓ listAlbumsAndVerifyData	8 s	✓
✓ albumsDetailsAndVerifyData	8 s	✓
✓ ListarArtistTest	7 s	1/1
✓ listArtistsAndVerifyData	7 s	✓
✓ ListarCollectorsTest	6 s	1/1
✓ listCollectionsAndVerifyData	6 s	✓

### 3.2. Implementación de Corrutinas

De acuerdo con la estrategia formulada por el equipo se determinó que la implementación de corrutinas se realizaría en en las tareas encargadas de consumo de datos del API. A continuación, se presentan algunos fragmentos de código donde se evidencia esta implementación.

```
suspend fun fetchAlbums(): String = suspendCancellableCoroutine { continuation ->
    val request = ApiService.getRequest( path: "albums",
        { response -> continuation.resume(response) },
        { error -> continuation.resumeWithException(error) }
    )
    apiService.instance.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}
```

```

suspend fun fetchArtists(): String = suspendCancellableCoroutine { continuation ->
    val request = ApiService.getRequest( path: "musicians",
        { response -> continuation.resume(response) },
        { error -> continuation.resumeWithException(error) }
    )
    apiService.instance.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}

```

```

suspend fun fetchCollectors(): String = suspendCancellableCoroutine { continuation ->
    Log.d( tag: "API", msg: "Making API call to fetch collectors")
    val request = ApiService.getRequest( path: "collectors",
        { response ->
            Log.d( tag: "API", msg: "API call successful $response")
            continuation.resume(response)
        },
        { error ->
            Log.e( tag: "API", msg: "API call failed $error", error)
            continuation.resumeWithException(error)
        }
    )

    apiService.instance.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}

```

```

suspend fun postPrize(prize: JSONObject): String = suspendCancellableCoroutine { continuation ->
    val request = ApiService.postRequest( path: "prizes", prize,
        { response -> continuation.resume(response.toString()) },
        { error -> continuation.resumeWithException(error) }
    )
    apiService.instance.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}

```

```

suspend fun postPrize(prize: JSONObject): String = suspendCancellableCoroutine { continuation ->
    val request = ApiService.postRequest( path: "prizes", prize,
        { response -> continuation.resume(response.toString()) },
        { error -> continuation.resumeWithException(Exception(error)) }
    )
    apiService.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}

```

```
suspend fun createAlbum(albumDTO: CreateAlbumDTO): String = suspendCancellableCoroutine { continuation ->
    val jsonRequest = JSONObject(Gson().toJson(albumDTO))
    val request = ApiService.postRequest( path: "albums", jsonRequest,
        { response ->
            continuation.resume(response.toString())
        },
        { error -> continuation.resumeWithException(Exception(error)) }
    )
    ApiService.add(request)

    continuation.invokeOnCancellation { it: Throwable?
        request.cancel()
    }
}
```

### 3.3. Resultados de Micro-optimización

#### 3.3.1. Implementación de patrón singleton, correcciones en la formación de las URLs y mejor manejo de errores

**Patrón Singleton:** La clase ApiService actualmente crea una nueva RequestQueue cada vez que es instanciada. Es una buena práctica seguir el patrón Singleton para este tipo de clase de servicio. De esta manera, sólo se crea una instancia de RequestQueue que se reutiliza en toda la aplicación. Esto puede ahorrar memoria y permitir una gestión centralizada de las peticiones.

**Construcción de URL:** Es mejor utilizar la clase Uri.Builder para construir URLs. Reduce el riesgo de URLs malformadas y hace el proceso de construcción de URLs más flexible y legible.

**Gestión de errores:** El ErrorListener de Volley sólo proporciona un VolleyError, pero no indica qué petición ha fallado. Para mejorar la depuración, podemos envolver el error listener en otra función lambda para proporcionar más contexto sobre el error.

#### 3.3.2. Lint y Refactoring

Se realizaron cambios sobre el código una vez que se hizo la inspección del código a través de la herramienta que ofrece Android Studio. La inspección del código arrojó warnings sobre archivos que contenían fragmentos de código sin utilizar y archivos incompatibles con la versión de la aplicación.

```
▼ Performance 21 warnings
  > Invalidating All RecyclerView Data 5 warnings
  > Obsolete SDK_INT Version Check 1 warning
  > Unused resources 15 warnings
```

Adicionalmente se hizo una inspección de forma manual en todos los archivos del proyecto y se eliminaron imports sin usar, fragmentos de código comentado y funciones que no tenían ningún llamado. El detalle de el refactor se puede consultar en el siguiente link, correspondiente al Pull Request

[https://github.com/Helena-Pat29/IngSoft\\_Appmoviles\\_CSHS/pull/48/files](https://github.com/Helena-Pat29/IngSoft_Appmoviles_CSHS/pull/48/files)

### 3.4. Buenas prácticas consumo de memoria

#### Cache Manager

Con la nueva implementación de la cache manager antes de realizar la petición del NetworkServiceAdapter, comprobamos si los datos ya están disponibles en la caché. Si lo están, los devolvemos inmediatamente. Si no, hacemos la petición de red como siempre, pero también almacenamos la respuesta en la caché antes de devolverla.

Es importante tener en cuenta que estrategia de almacenamiento en caché sólo almacena datos en memoria, por lo que la caché se borrará cuando se cierre la aplicación o si el sistema decide recuperar memoria. La estrategia aplica para todos los metodos Get actualmente implementados.

### 3.5. Reporte de perfilamiento de la Aplicación.

#### 3.5.1. Captura de Datos.

Con el propósito de realizar el perfilamiento de la aplicación se definen 5 etapas de captura de datos de acuerdo a las historias de usuario a probar así:

Etapa	HU probada
1	HU01 - Consultar catálogo de álbumes HU02 - Consultar la información detalla del álbum
2	HU03 - Consultar el listado de artistas HU04 - Consultar la información detallada de un artista
3	HU05 - Consultar listado de coleccionistas HU07 - Consultar la información detallada de coleccionista
4	HU06 - Crear Premio
5	HU08 - Crear álbum

#### Etapa 1 – Lista y Detalle Álbumes

Las instrucciones de ejecución son las siguientes:

- Ejecutar el perfilamiento con el botón "*profile 'app'*". En caso de que la información de los coleccionistas no cargue al inicio, volver a presionar este botón.
- Seleccionar *Listar Álbumes*. Esperar un tiempo a que el *profiler* registre el evento del clic y la creación del fragmento nuevo.
- Seleccionar el primer álbum. Esperar un tiempo a que se registre el evento y la interfaz muestre el comentario.

#### Etapa 2 – Lista y Detalle Artistas

Las instrucciones de ejecución son las siguientes:

- Ejecutar el perfilamiento con el botón "*profile 'app'*". En caso de que la información de los coleccionistas no cargue al inicio, volver a presionar este botón.
- Seleccionar *Listar Artistas*. Esperar un tiempo a que el *profiler* registre el evento del clic y la creación del fragmento nuevo.
- Seleccionar el primer artista. Esperar un tiempo a que se registre el evento y la interfaz muestre el comentario.

### **Etapas 3 – Lista y Detalle Coleccionistas**

Las instrucciones de ejecución son las siguientes:

- Ejecutar el perfilamiento con el botón "*profile 'app'*". En caso de que la información de los coleccionistas no cargue al inicio, volver a presionar este botón.
- Seleccionar *Listar Coleccionistas*. Esperar un tiempo a que el *profiler* registre el evento del clic y la creación del fragmento nuevo.
- Seleccionar el primer coleccionista. Esperar un tiempo a que se registre el evento y la interfaz muestre el comentario.

### **Etapas 4 – Crear Premio**

Las instrucciones de ejecución son las siguientes:

- Ejecutar el perfilamiento con el botón "*profile 'app'*". En caso de que la información de los coleccionistas no cargue al inicio, volver a presionar este botón.
- Selecciona Crear Premio. Esperar un tiempo a que el *profiler* registre el evento del clic y la creación del fragmento nuevo.
- Ingresar los siguientes datos de prueba:

Nombre del premio: Premio Test

Organización: Organización Test

Descripción: Esta es la descripción del premio test

- Click en Crear Premio.

### **Etapas 5 – Crear Album**

Las instrucciones de ejecución son las siguientes:

- Ejecutar el perfilamiento con el botón "*profile 'app'*". En caso de que la información de los coleccionistas no cargue al inicio, volver a presionar este botón.
- Selecciona Crear Album. Esperar un tiempo a que el *profiler* registre el evento del clic y la creación del fragmento nuevo.
- Ingresar los siguientes datos de prueba:

Name: Premio New Album Test

Cover: <https://img.freepik.com/vector-premium/emocion-cara-decepcionada-expresion-cabeza-retro-comica> 53562-16413.jpg

Release Date: 2018

Descripción: Esta es la descripción del nuevo album

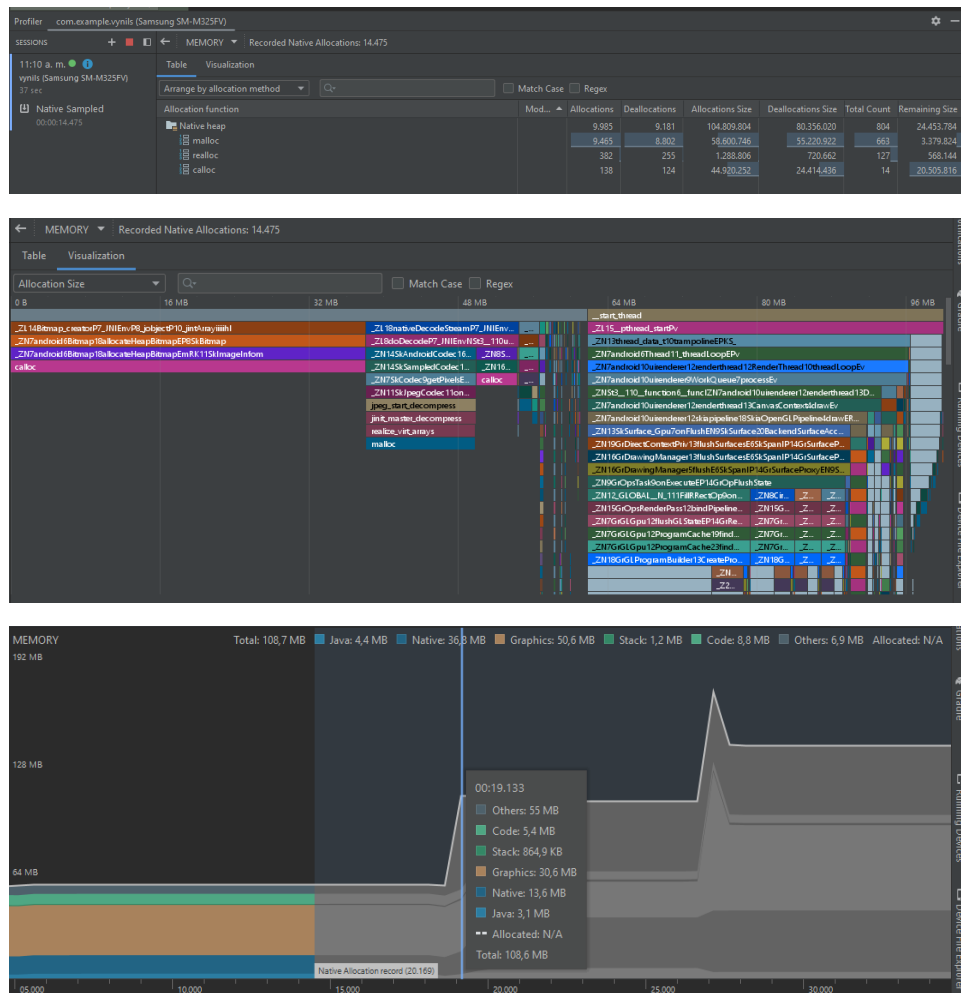
Genre: Rock

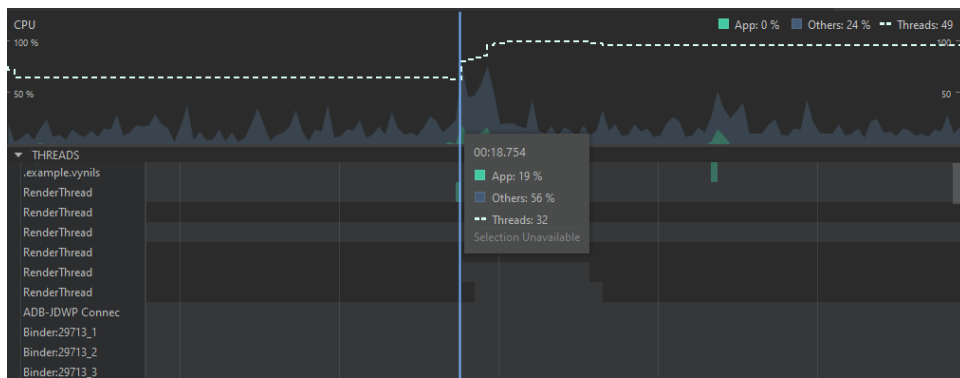
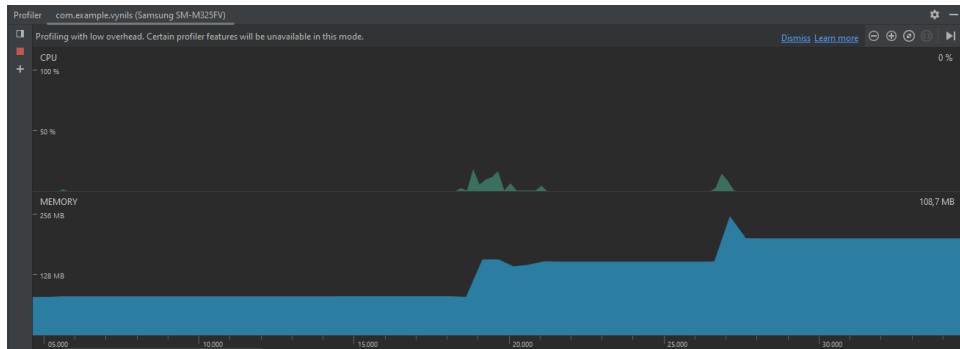
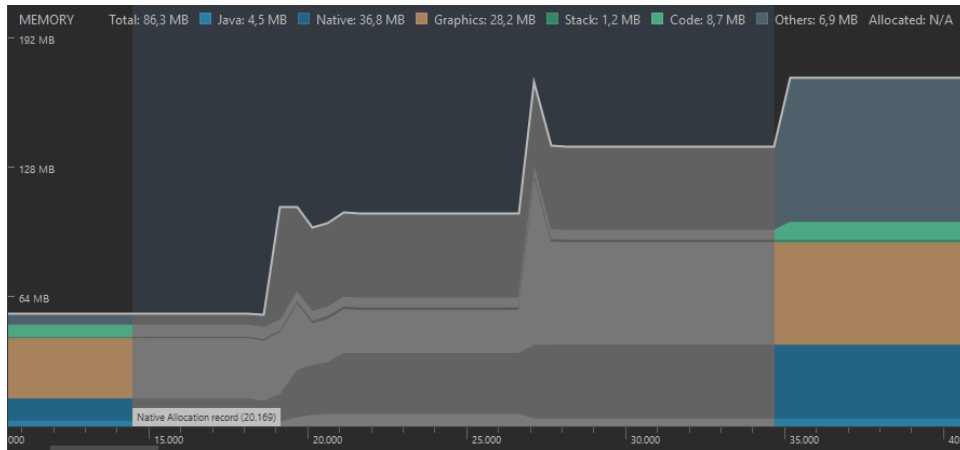
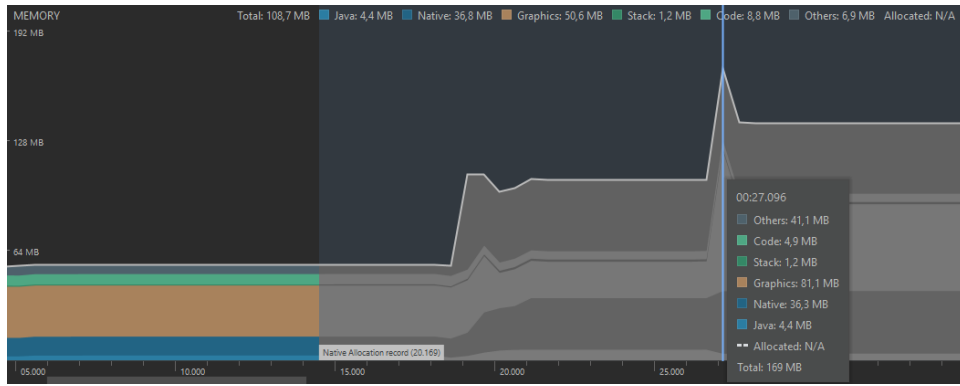
Record Label: EMI

- Click en Crear Album.

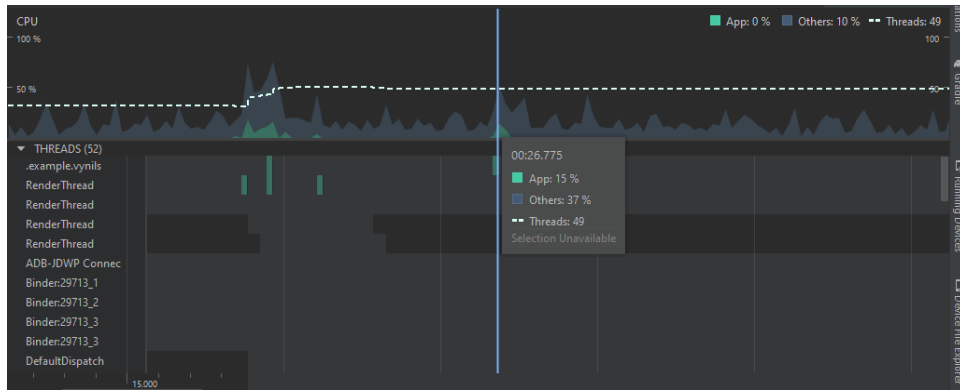
### 3.5.2. Resultados

#### Etaapa 1- Listado y Detalle de Álbumes









## Etapla 2 –Listado y Detalle Artistas

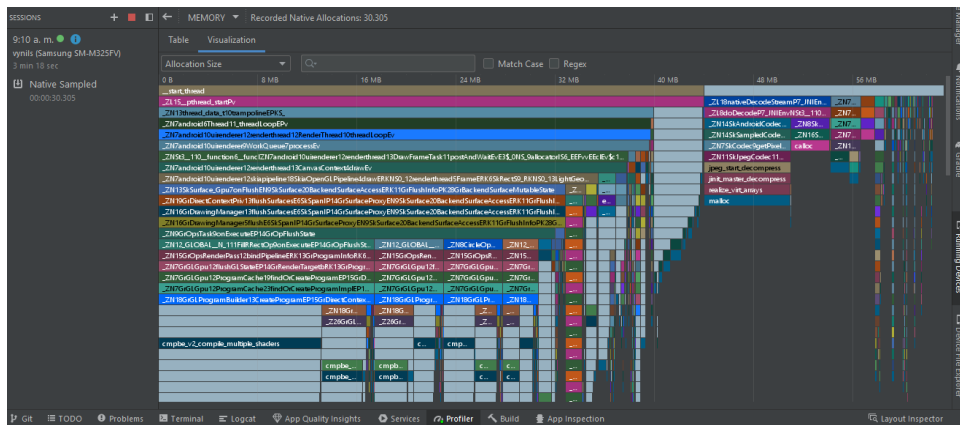
Profiler - com.example.vynils (Samsung SM-M325FV)

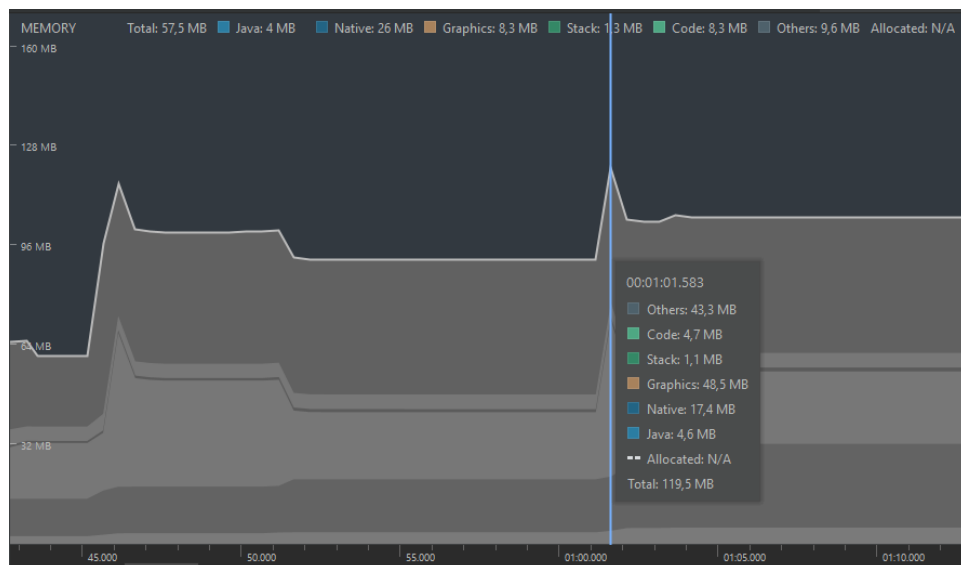
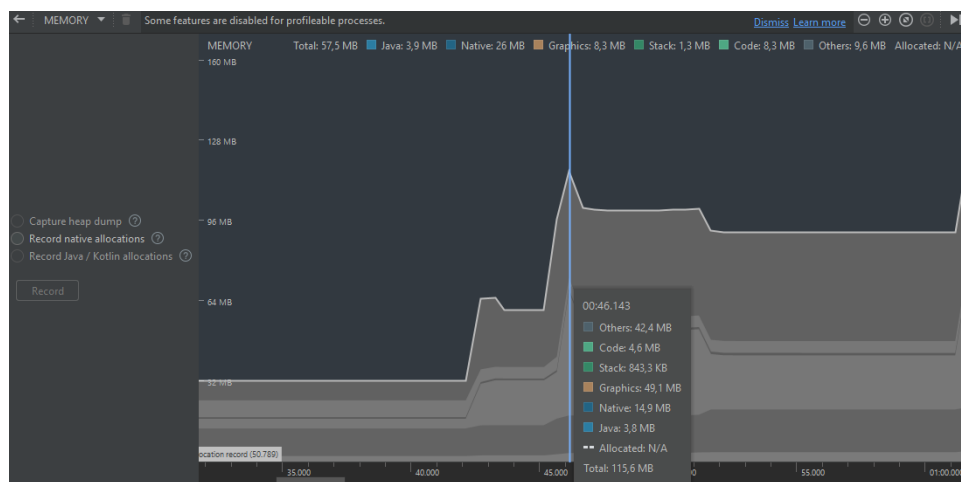
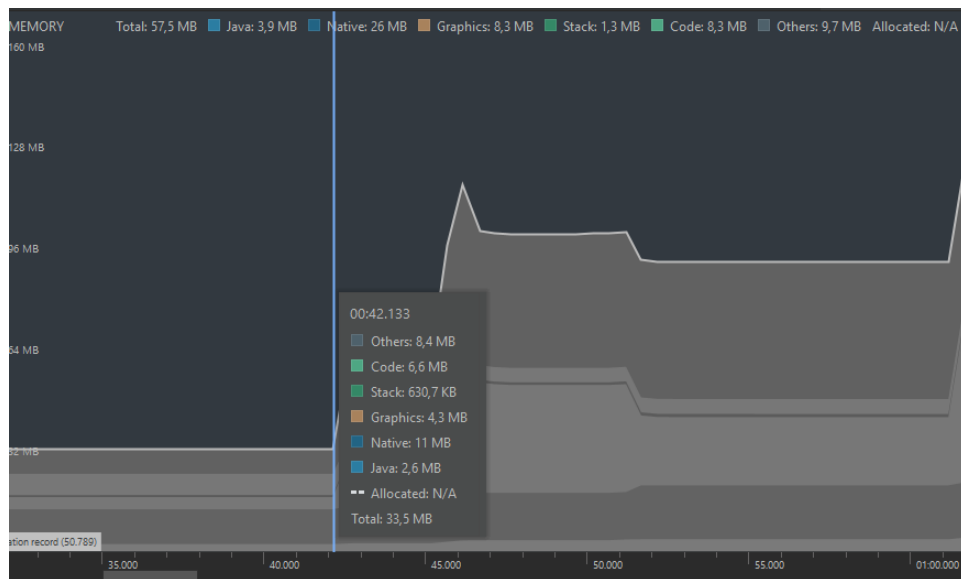
SESSIONS + MEMORY Recorded Native Allocations: 30,305

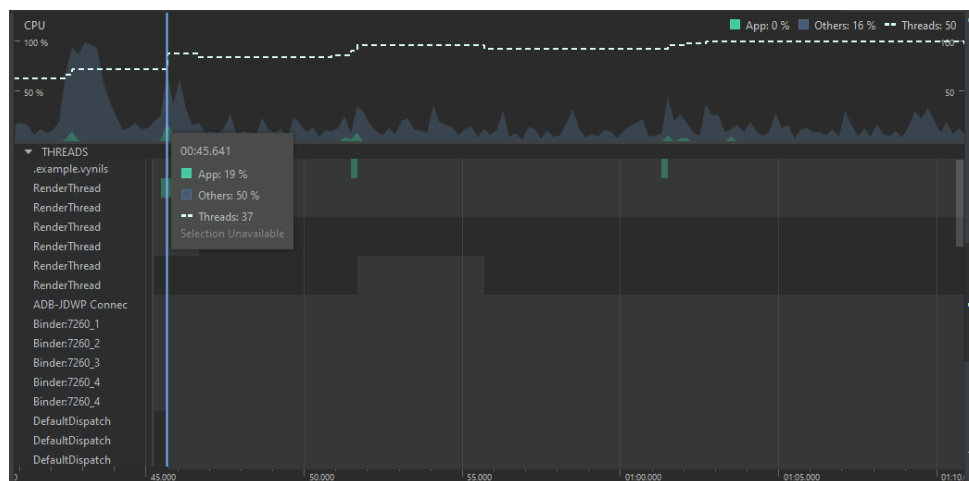
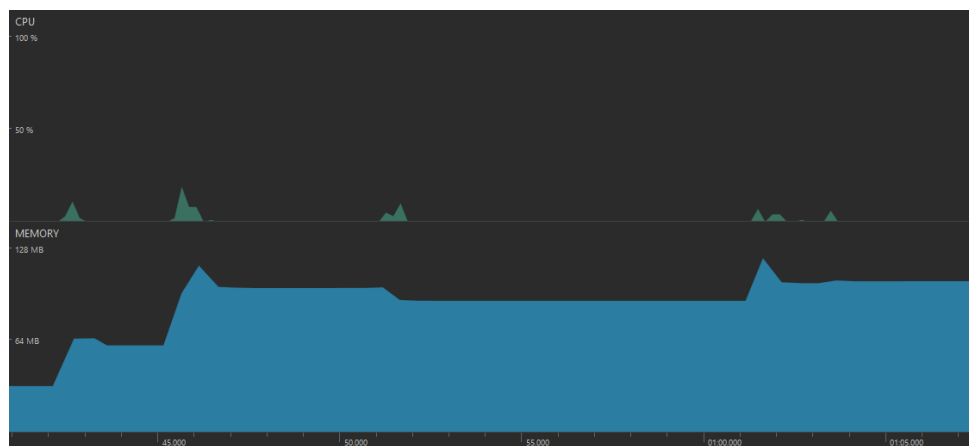
9:10 a. m. vynils (Samsung SM-M325FV) 2 min 13 sec

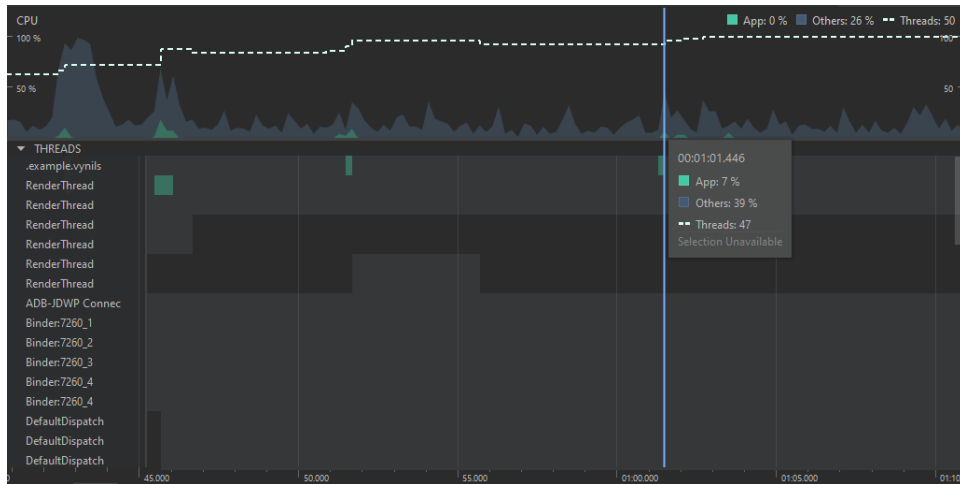
Native Sampled 00:00:30.305

Allocation function	Mod...	Allocations	Deallocations	Allocations Size	Deallocations Size	Total Count	Remaining Size
Native heap		11,278	10,464	66,424,770	61,026,486	814	5,408,284
malloc		10,650	9,961	60,113,458	56,440,464	689	3,673,024
calloc		144	133	4,813,340	3,599,120	11	1,214,220
realloc		484	370	1,507,942	986,902	114	521,040









### Etapas 3 – Lista y Detalle Coleccionistas

Profiler: com.example.vynils (Samsung SM-M325FV)

SESSIONS: 9:21 a. m. vynils (Samsung SM-M325FV) 1 min 2 sec

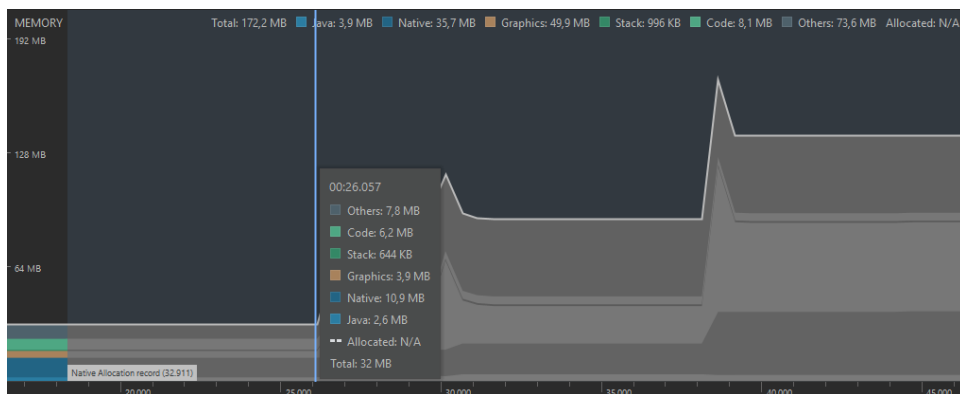
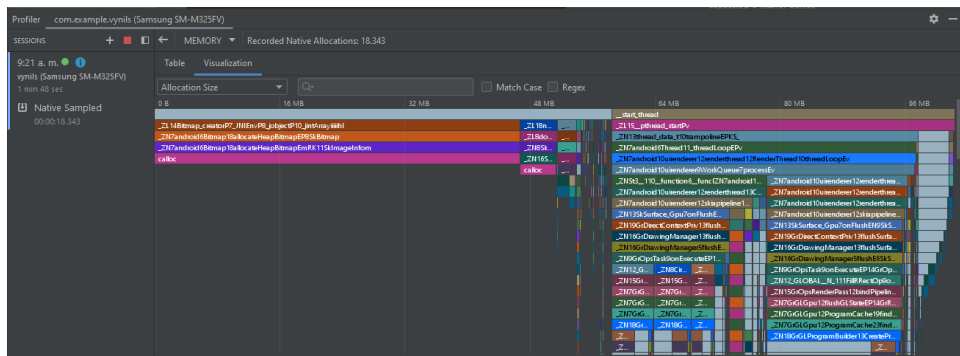
Native Sampled 00:00:18.343

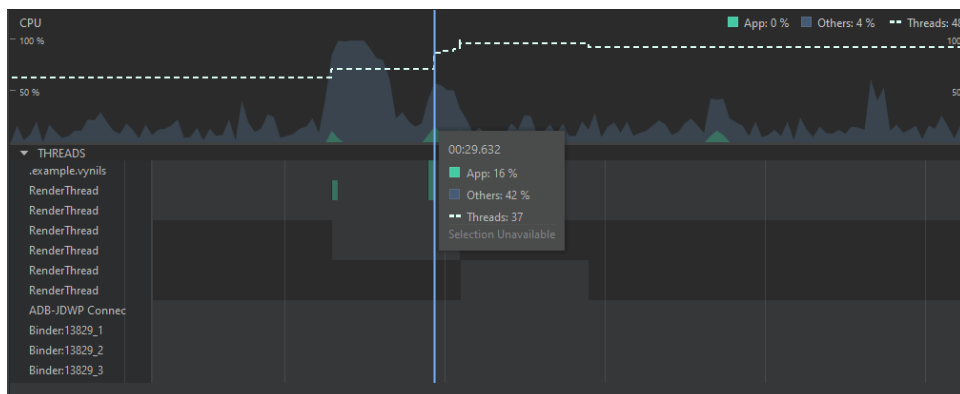
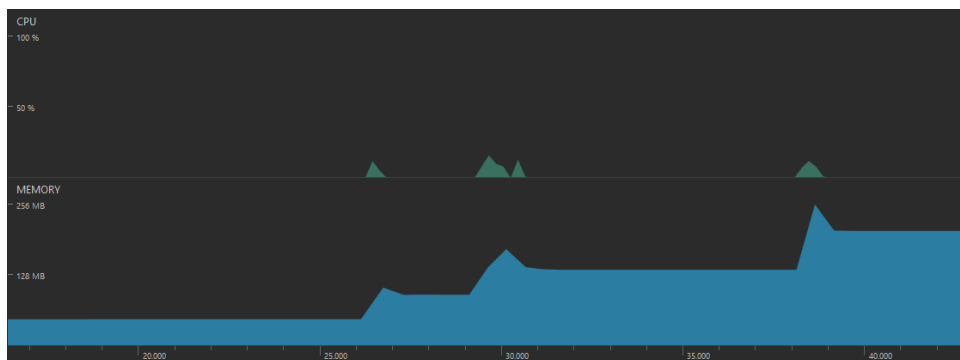
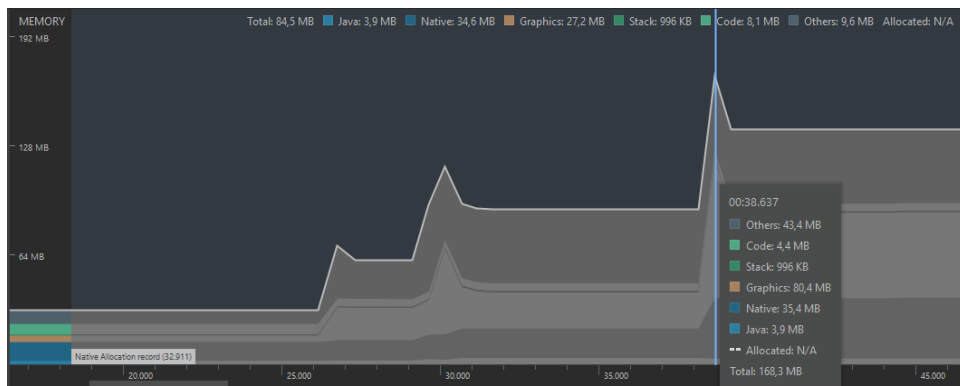
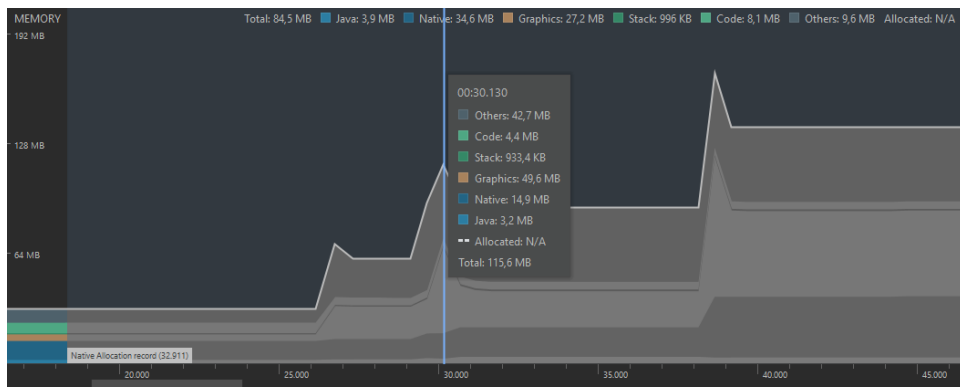
Table Visualization

Arrange by allocation method Q-

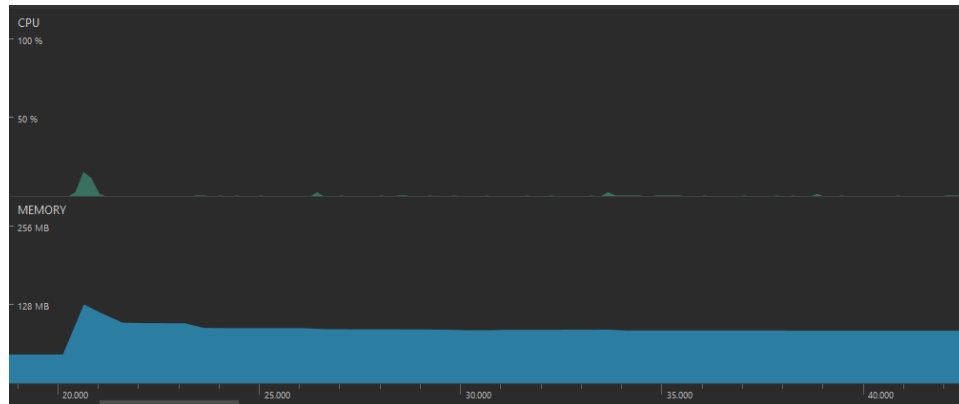
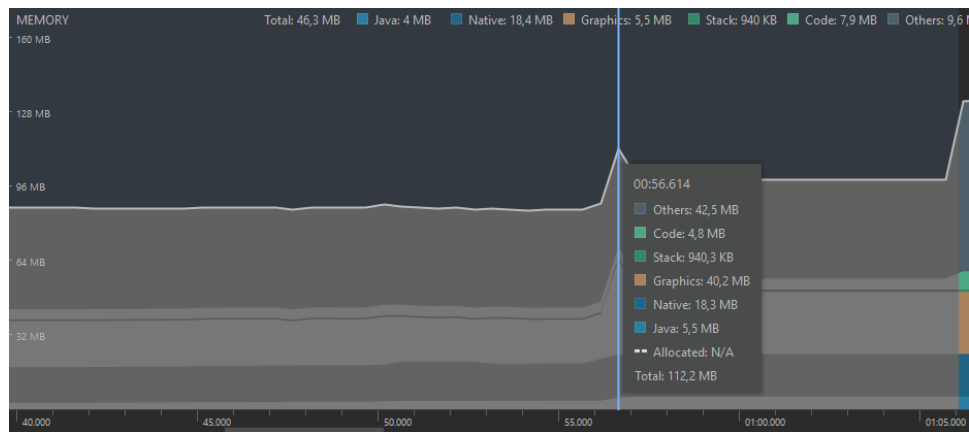
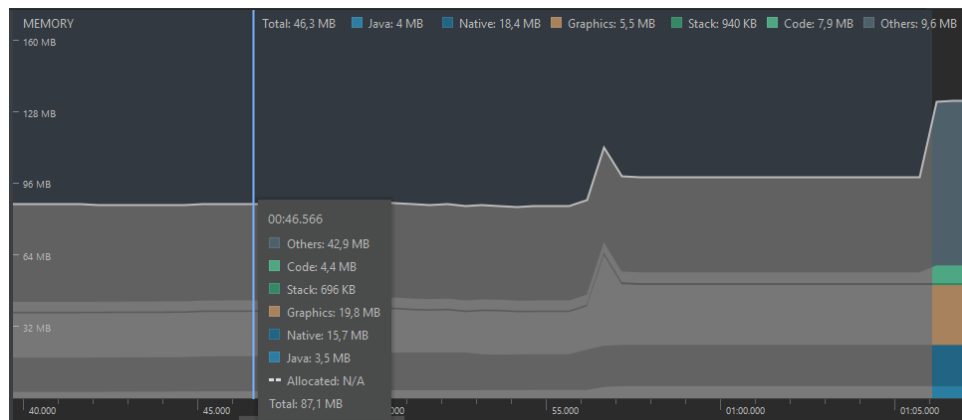
Match Case Regex

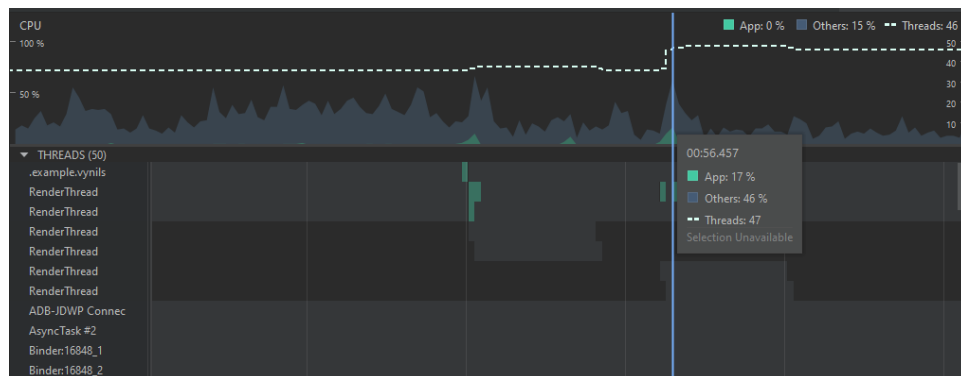
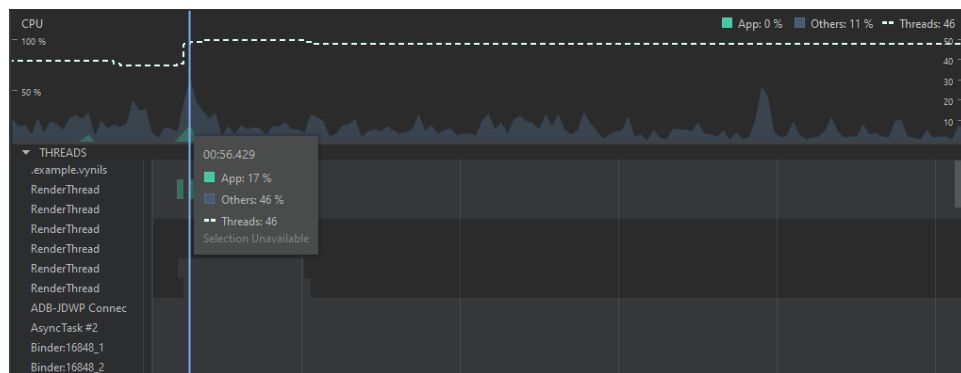
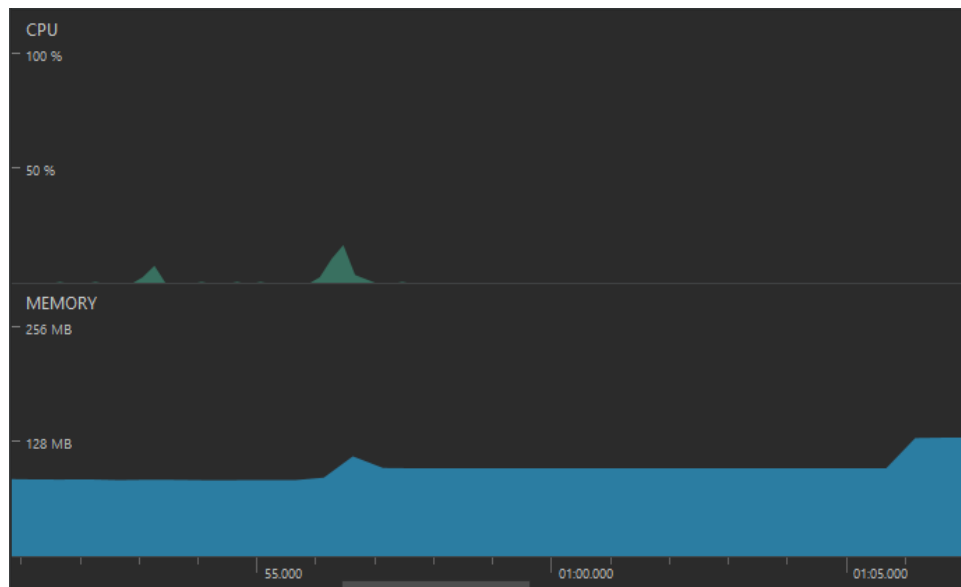
Allocation function	Mod...	Allocations	Deallocations	Allocations Size	Deallocations Size	Total Count	Remaining Size
Native heap		10,919	10,201	107,255,484	83,103,787	718	24,151,697
malloc		10,340	9,740	51,318,641	47,894,849	600	3,323,792
realloc		436	329	1,335,910	888,118	107	467,792
calloc		143	132	54,800,933	34,240,820	11	20,360,113











## Etapas 5 – Crear Album

Profiler: com.example.vynils (Samsung SM-M325FV)

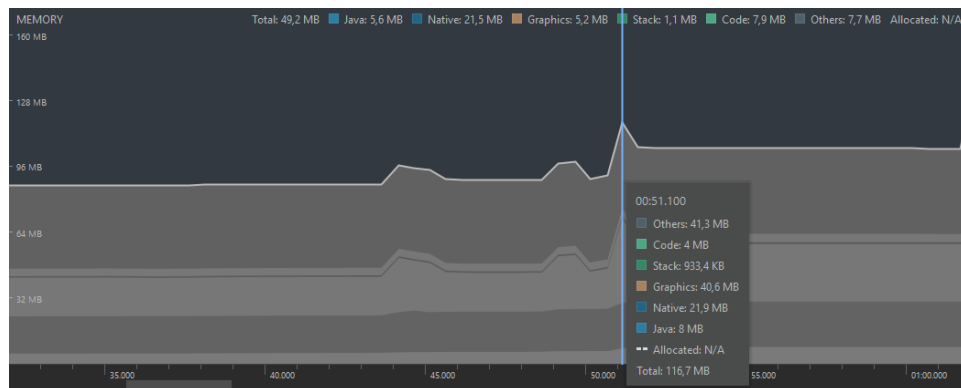
SESSIONS: 11:25 a. m. (vynils (Samsung SM-M325FV) 3 min 5 sec)

Native Sampled: 00:00:14.773

Recorded Native Allocations: 14,773

Allocation function	Mod...	Allocations	Deallocations	Allocations Size	Deallocations Size	Total Count	Remaining Size
Native heap		35,610	34,041	180,890,343	170,936,567	1,569	9,953,776
malloc		33,708	32,347	173,774,311	166,438,223	1,361	7,316,088
calloc		501	469	3,305,768	1,390,448	32	1,915,320
realloc		1,401	1,225	3,810,264	3,087,896	176	722,368







### 3.6. Pruebas de Reconocimiento Monkey y Firebase

#### 3.6.1. Pruebas de Reconocimiento Monkey

##### Ejecución de la prueba

##### Obtener el APK a probar

Para generar el APK, en Android Studio, en el menú de la parte superior seleccione la opción Build > Build Bundle(s)/APK(s) > Build APK(s). Al finalizar el proceso se podrá ver un mensaje el cual indica que se generó el APK deseado y permite localizar el archivo en el sistema de archivos. La alerta se ve de la siguiente forma:

**i Build APK(s)**  
APK(s) generated successfully for 1 module:  
Module 'vynils.app': [locate](#) or [analyze](#) the APK.

### Ejecutar la Prueba Monkey

Se conecta un dispositivo Android a la computadora asegurándose que esté conectado en modo de transferencia de archivos y tenga activadas las opciones de desarrollador. Se verifica que el dispositivo ha sido reconocido y que se estableció un puente ADB ejecutando el siguiente comando en una terminal:

*adb devices*

```
Terminal: Local x + v
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS> adb devices
RF8RA132TVZ      device
```

Para ejecutar pruebas de Monkey, primero es necesario que el APK esté instalado en el dispositivo Android y sea reconocido por el Packet Manager. Para esto, desde la misma terminal se ubica el archivo APK que generó en el primer paso, y ejecute el siguiente comando para instalarlo en el dispositivo:

*adb install app-debug.apk*

```
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS> cd app\build\outputs\apk\debug
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS\app\build\outputs\apk\debug>
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS\app\build\outputs\apk\debug> adb install app-debug.apk
Performing Streamed Install
Success
```

Luego de haberlo instalado, se ejecuta el siguiente comando para iniciar la prueba con 100 eventos igualmente distribuidos:

*adb shell monkey -p com.example.vynils -v 100*

```
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS\app\build\outputs\apk\debug> adb shell monkey -p com.example.vynils -v 100
```

### Resultados

```
PS C:\Users\Helena\AndroidStudioProjects\IngSoft_Appmoviles_CS\SHS\app\build\outputs\apk\debug> adb shell monkey -p com.example.vynils -v 100
bash arg: -p
bash arg: com.example.vynils
bash arg: -v
bash arg: 100
args: [-p, com.example.vynils, -v, 100]
arg: "-p"
arg: "com.example.vynils"
arg: "-v"
arg: "100"
data="com.example.vynils"
:Monkey: seed=1684860446723 count=100
:AllowPackage: com.example.vynils
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
```

```

Terminal: Local +
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: -0.0%
// 6: 25.0%
// 7: 15.0%
// 8: 2.0%
// 9: 2.0%
// 10: 1.0%
// 11: 13.0%
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.example.vynils/.MainActivity;e
nd
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.example.vynils/.MainActivity } in package com.ex
ample.vynils
:Sending Touch (ACTION_DOWN): 0:(271.0,1500.0)
:Sending Touch (ACTION_UP): 0:(273.27286,1515.8326)
:Sending Touch (ACTION_DOWN): 0:(756.0,359.0)
:Sending Touch (ACTION_UP): 0:(751.49384,354.4263)
:Sending Trackball (ACTION_MOVE): 0:(0.0,-1.0)
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.example.vynils/.MainActivity;e
:Sending Touch (ACTION_DOWN): 0:(881.0,1518.0)
:Sending Touch (ACTION_UP): 0:(934.6698,1568.8345)

:Sending Touch (ACTION_DOWN): 0:(1009.0,2067.0)
:Sending Touch (ACTION_UP): 0:(1019.7995,2077.2646)
:Sending Trackball (ACTION_MOVE): 0:(3.0,0.0)
:Sending Touch (ACTION_DOWN): 0:(85.0,160.0)
:Sending Touch (ACTION_UP): 0:(90.74472,165.46361)
:Sending Flip keyboardOpen=false
Got IOException performing flipjava.io.FileNotFoundException: /dev/input/event0: open failed: EACCES (Permission denied)
// Injection Failed
:Sending Trackball (ACTION_MOVE): 0:(1.0,2.0)
:Sending Touch (ACTION_DOWN): 0:(472.0,1803.0)
Events injected: 100
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=1 rotations=0
## Network stats: elapsed time=1973ms (0ms mobile, 0ms wifi, 1973ms not connected)
// Monkey finished

```

### 3.6.2. Pruebas de Reconocimiento Firebase

A continuación, se presentan las pruebas Robo realizadas con Firebase para la aplicación, utilizando el APK generado por Debug:

- Pruebas de reconocimiento: [Artefactos de prueba](#)

## Bucket

El nombre del Bucket es "test-lab-zvp04aff965a-jkt5tjsam4k30".

test-lab-zvp04aff965a-jkt5tjsam4k30

Location

us (multiple regions in United States)

Storage class

Standard

Public access

Value hidden

Protection

None

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

NEW

INVENTORY REPORTS

NEW

Buckets

> test-lab-zvp04aff965a-jkt5tjsam4k30

> web-build\_20230517\_fryp

> redfin-30-en\_US-portrait

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

DOWNLOAD

DELETE

Filter by name prefix only

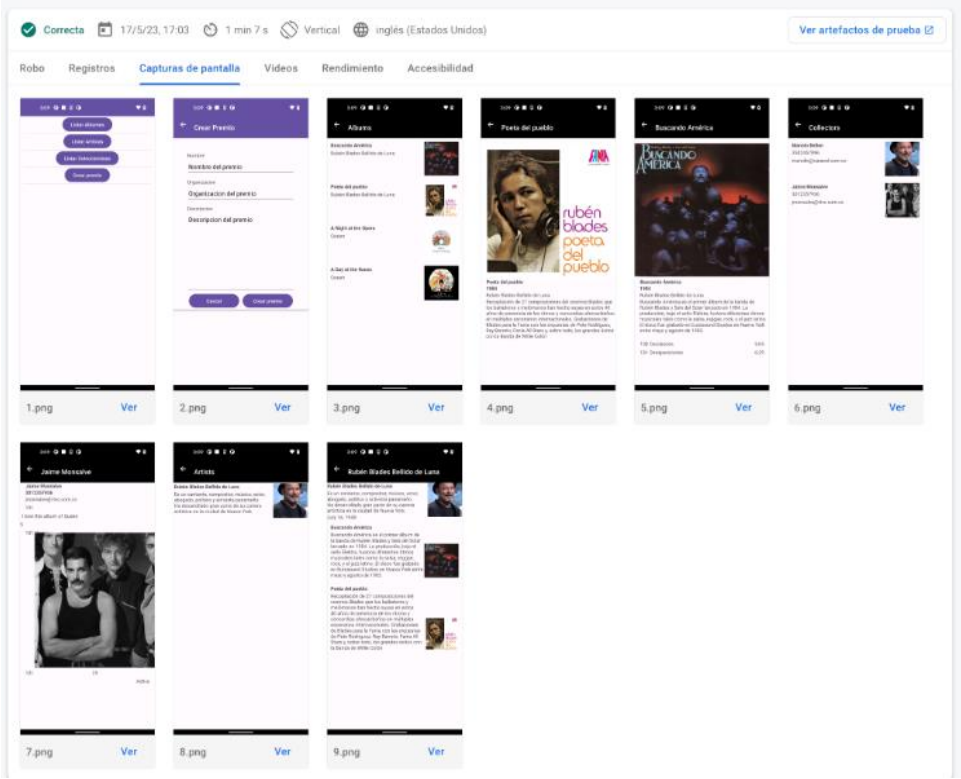
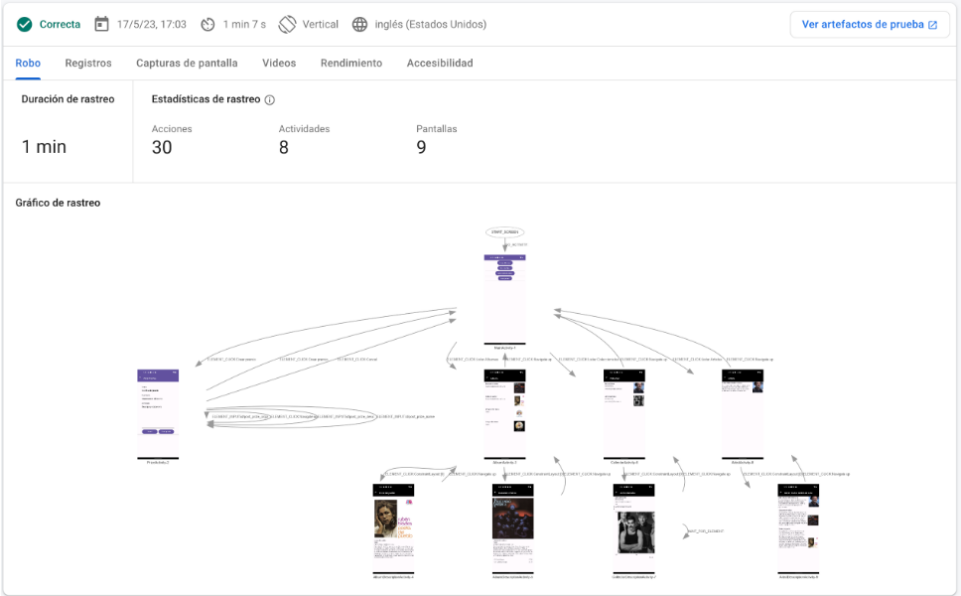
Filter

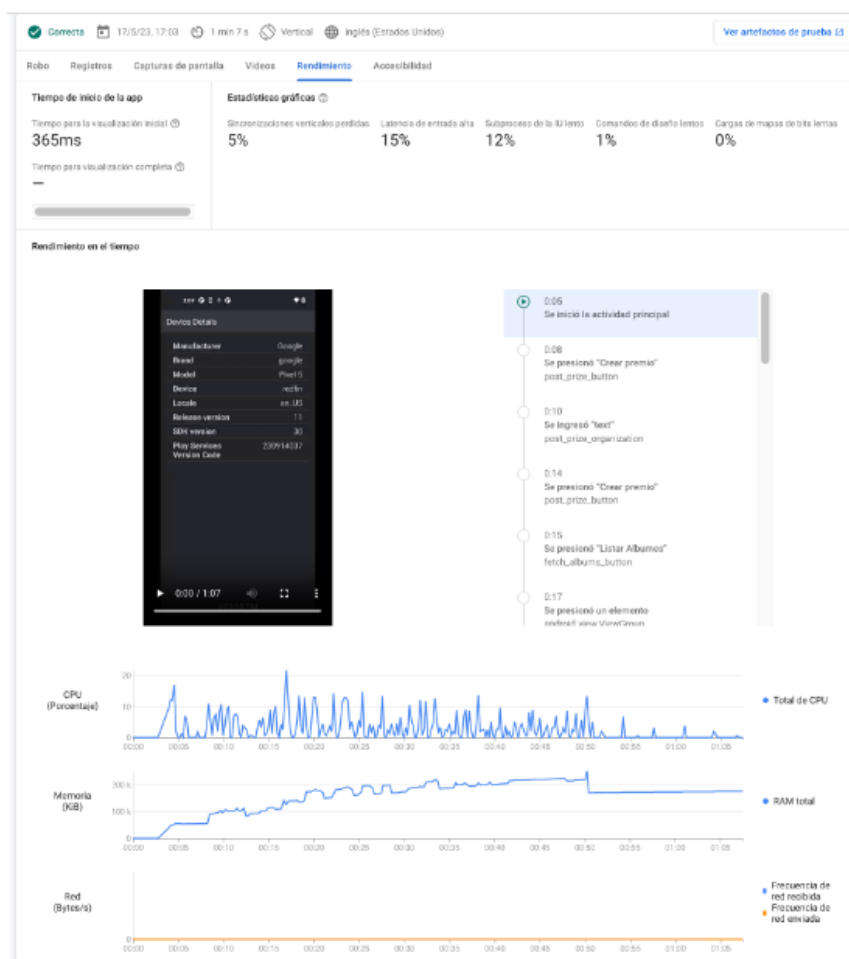
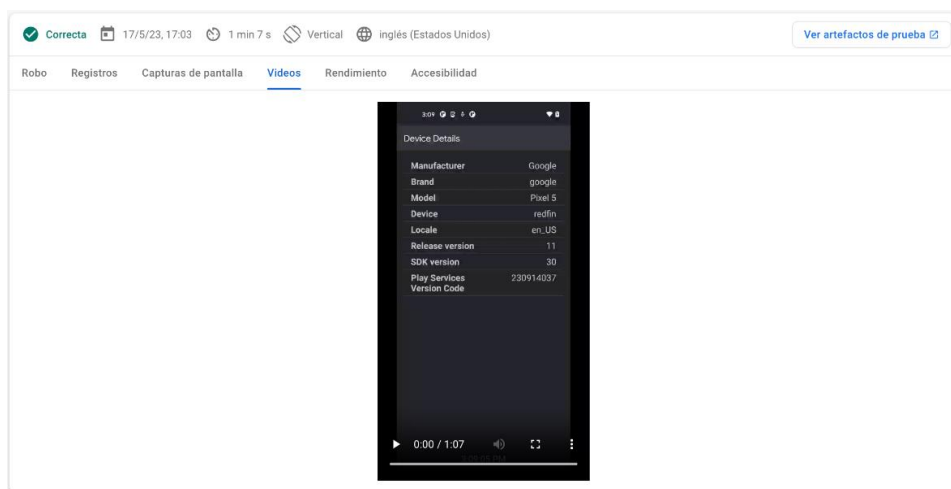
Filter objects and folders

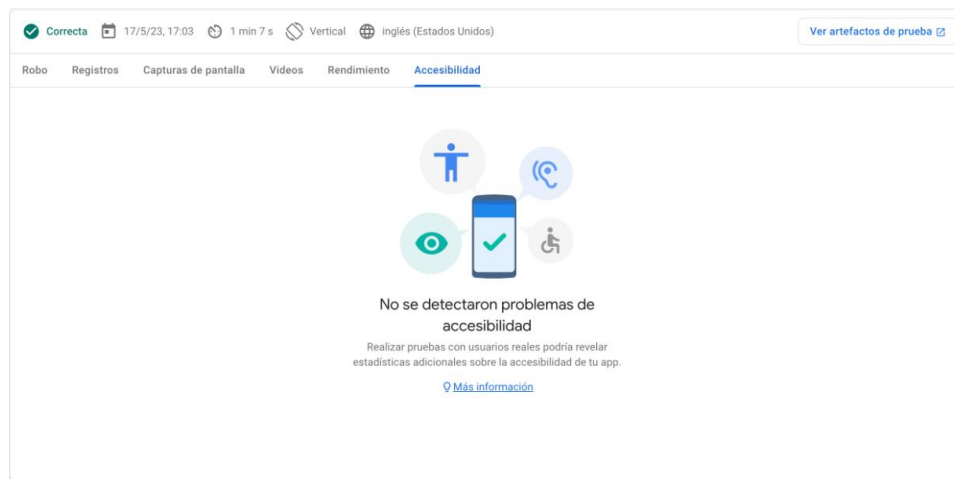
Show deleted data

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expires	
<input type="checkbox"/>	actions.json	14.2 KB	application/json	May 17, 2023, 5:10:54 PM	Standard	May 17, 2023, 5:10:54 PM	Not public	—	Google-managed key	—	
<input type="checkbox"/>	artifacts/	—	Folder	—	—	—	—	—	—	—	
<input type="checkbox"/>	logcat	4.2 MB	text/plain	May 17, 2023, 5:10:48 PM	Standard	May 17, 2023, 5:10:48 PM	Not public	—	Google-managed key	—	
<input type="checkbox"/>	robo_results.pb	150.1 KB	application/octet-stream	May 17, 2023, 5:10:55 PM	Standard	May 17, 2023, 5:10:55 PM	Not public	—	Google-managed key	—	
<input type="checkbox"/>	video.mp4	1.1 MB	video/mp4	May 17, 2023, 5:10:50 PM	Standard	May 17, 2023, 5:10:50 PM	Not public	—	Google-managed key	—	

Pruebas Robo en Firebase







### 3.7. Reporte y Análisis de Accesibilidad

A continuación, se presenta el reporte del análisis de accesibilidad de la aplicación, utilizando las herramientas de Firebase (mediante pruebas Robo, en la sección "Accesibilidad") y la funcionalidad del IDE Android Studio para inspeccionar código.

#### **Accesibilidad con pruebas Robo (Firebase):**

Como se puede evidenciar en la imagen de abajo, al momento de correr el APK en la herramienta de Firebase, utilizando el tipo de prueba Robo, y luego de que termine el análisis del archivo APK en la herramienta, en la última pestaña aparece un listado de los posibles errores de accesibilidad que se tienen en la aplicación. Este es un punto de partida para resolver y solucionar estos issues de accesibilidad.

Correcta 20/5/23, 00:46 1 min 30 s Vertical Inglés (Estados Unidos) [Ver artefactos de prueba](#)

Robo Registros Capturas de pantalla Videos Rendimiento **Accesibilidad**

★ Aquí encontrarás los problemas de accesibilidad que se encontraron en las pruebas automatizadas, además de las recomendaciones asociadas. [Más información](#)

### Descripción general


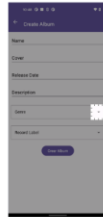
Se encontraron 3 problemas

Tipo de problemas	Advertencias	Problemas menores	Sugerencias
Tamaño de objetivos táctiles	0	0	0
Contraste bajo	0	2	0
Etiquetas de contenido	0	1	0
Implementación	0	0	0

#### Tamaño de los objetivos táctiles (0)

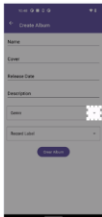
No se encontraron problemas.

#### Contraste bajo (2)

1 ejemplo Ver 2 ejemplos Ver

#### Etiquetas de contenido (1)




1 ejemplo Ver

#### Implementación (0)

No se encontraron problemas.

Luego de resolver todos los issues de accesibilidad, se vuelve a correr el análisis del APK en la herramienta, y el resultado es que no se tienen problemas de accesibilidad. En la imagen de abajo se puede evidenciar el resultado.

Robo Registros Capturas de pantalla Videos Rendimiento **Accesibilidad**



**No se detectaron problemas de accesibilidad**

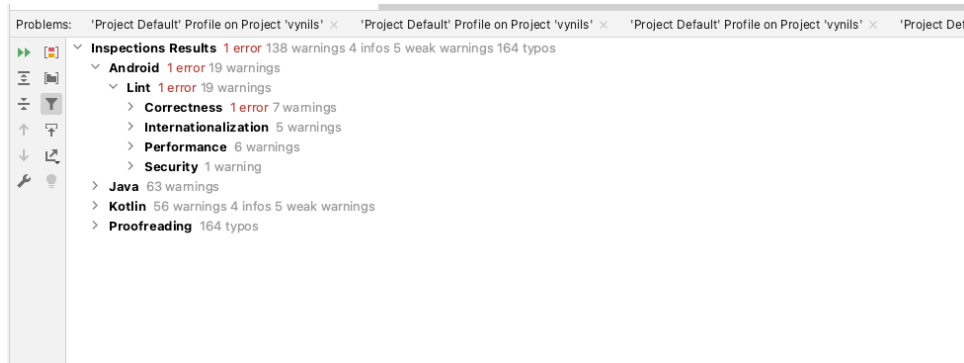
Realizar pruebas con usuarios reales podría revelar estadísticas adicionales sobre la accesibilidad de tu app.

[Más información](#)



### Accesibilidad resultado inspección de código (IDE Android Studio):

Como se puede evidenciar en la imagen de abajo, luego de inspeccionar el código utilizando la funcionalidad del IDE (Code -> Inspect Code -> Analyze), dentro de los resultados no se encuentran errores o advertencias relacionados a la accesibilidad de la aplicación.



Se concluye que, luego de terminar el análisis y corrección de los temas de accesibilidad de la aplicación, se considera esta app como accesible y disponible para personas con y sin discapacidad visual.