

Aplicación Bajo Pruebas

1.1 Nombre de la Aplicación

VynilApp

1.2 Versión

1.0.0

1.3 Descripción

VynilApp es una aplicación web diseñada para amantes de la música y coleccionistas de vinilos que buscan descubrir, compartir y gestionar información sobre álbumes, artistas y coleccionistas.

La plataforma permite a usuarios visitantes explorar un extenso catálogo musical, consultar detalles de álbumes y artistas, y conocer los gustos de distintos coleccionistas. Por otro lado, los usuarios coleccionistas pueden crear y administrar su propia colección, agregar nuevos álbumes y tracks, comentar sobre discos, y compartir sus artistas favoritos con la comunidad.

VynilApp promueve la interacción entre usuarios a través del intercambio de información musical y la construcción de una red de coleccionistas apasionados, convirtiéndose en un espacio colaborativo donde descubrir nuevas joyas del mundo del vinilo.

1.4 Funcionalidades

Criticidad Alta : Ya que permite tener una versión inicial de la APP funcional

1. **HU01:** Consultar catálogo de álbumes: Permite consultar a los coleccionistas el catálogo de álbumes
2. **HU02:** Consultar la información detallada de un álbum: Permite consultar el detalle de un álbum, artista, tracks y comentarios relacionados
3. **HU03:** Consultar el listado de artistas
Permite consultar el listado de artistas en la aplicación
4. **HU04:** Consultar la información detallada de un artista
Permite ver el detalle de información de un artista, Nombre, descripción, imagen y fecha de creación
5. **HU05:** Consultar listado de coleccionistas
Permite ver el listado de todos los coleccionistas creados, nombre, albums coleccionados

6. HU06: Consultar la información detallada de coleccionista

Permite ver información detallada del coleccionista, nombre, teléfono, email, comentarios, álbumes y artistas preferidos

7. HU07: Crear un álbum

Permite Crear un álbum con su información, Nombre, artista, imagen, descripción y fecha de lanzamiento

8. HU08: Asociar tracks con un álbum

Permite al coleccionista agregar tracks a un album, seleccionando un album y agregando la cancion

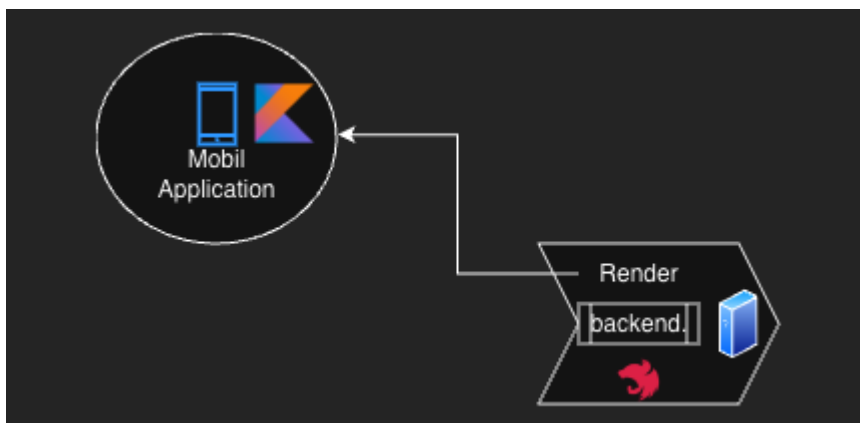
9. HU09: Comentar un Álbum

Permite al coleccionista agregar comentarios a un album.

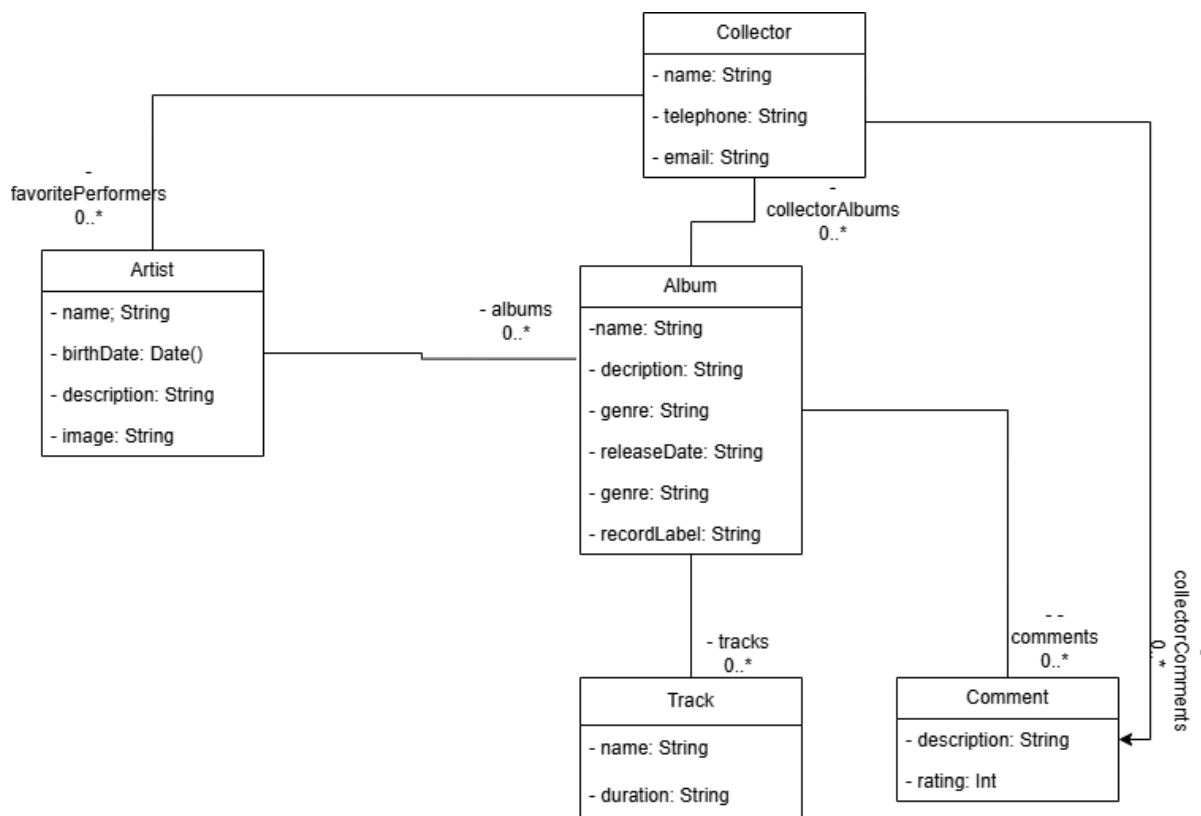
1.5 Diseño arquitectónico

- <https://github.com/apohies/vynilapp-misw4203/wiki/Dise%C3%B1o-arquitectonico>

1.6 Diagrama de Contexto



1.7 Modelo de Dominio



1.8 Modelo de GUI

- <https://www.figma.com/proto/f2HzPu4iXzNS79KWFG10p9/VINILOS-UNIANDES?node-id=0-1&p=f&t=9f369i2WxBRia18q-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=302%3A279&show-proto-sidebar=1>

2 Contexto de la Estrategia de Pruebas

2.1 Objetivos

OBJ-01: Realizar pruebas exploratorias manuales sobre funcionalidades nuevas de criticidad alta, media y baja en la aplicación Vinyl, durante cada semana, a medida que se desarrollan las funcionalidades. Con el fin de garantizar que se está cumpliendo con lo requerido en las historias funcionales.

OBJ-02: Construir scripts de prueba funcionales end to end positivos para los casos de prueba principales utilizando la herramienta Espresso con el fin de tener una base de pruebas automatizadas que permitan garantizar el correcto

funcionamiento de la aplicación en sprints posteriores a medida que se desarrollan las funcionalidades.

OBJ-03: Construir scripts de prueba funcionales end to end negativos para los casos de prueba principales utilizando la herramienta Espresso con el fin de tener una base de pruebas automatizadas que permitan garantizar el correcto funcionamiento de las diferentes validaciones, entradas incorrectas o acciones inválidas para garantizar el correcto funcionamiento de la aplicación en futuros sprints a medida que se añaden nuevas funcionalidades..

OBJ-04: Generar un informe funcional consolidado al finalizar cada sprint, con el fin de llevar un registro de los issues/defectos encontrados en cada momento del desarrollo. Esto para priorizarlos y corregirlos a medida que avanzan los diferentes sprints de desarrollo.

OBJ-05: Entrenamiento del equipo en el framework de espresso durante el primer sprint de desarrollo, por medio de la implementación de scripts de prueba. Con el fin de garantizar el correcto manejo de la herramienta en futuros sprints.

OBJ-06: Implementar pruebas unitarias para cada una de las historias de usuario a medida que se vayan desarrollando semana a semana. Esto con el fin de garantizar el correcto funcionamiento de cada uno de los componentes internos de la solución propuesta.

2.2 Niveles de la estrategia de pruebas.

Nivel	Propósito	Herramientas
Unitarias	Validar funciones o clases individuales (sin dependencias reales).	JUnit5
E2E	Probar la interacción entre módulos (por ejemplo, ViewModel + Repository).	Espresso

2.3 Duración de la Iteración de Pruebas

Fecha de inicio de la iteración: 20 de Octubre de 2025

Fecha de fin de la iteración: 30 de Noviembre 2025

Horas laborales por día: 3 horas

Días laborables por semana: 5 días (lunes a viernes)

Horas laborales por semana por persona: 15 horas

Horas totales por persona en toda la iteración: 120 horas (8 semanas × 40 h)

Semana	DESDE	HASTA	Actividad
Semana 3	20 Oct 2025	26 Oct 2025	Planeación de la estrategia, entrenamiento herramientas de Prueba
Semana 4	27 Oct 2025	02 Nov 2025	Diseño de scripts automatizados E2E, UNIT TEST (HU01, HU02,HU03, HU04)
Semana 5	3 Nov 2025	9 Nov 2025	Diseño de scripts automatizados E2E, UNIT TEST (HU05, HU06)
Semana 6	10 Nov 2025	16 Jun 2025	Diseño de scripts

			automatizados E2E, UNIT TEST (HU07, HU08, HU09)
Semana 7	17 Nov 2025	23 Nov 2025	
Semana 8	24 Nov 2025	30 Nov 2025	

2.4 Presupuesto de Pruebas

2.4.1 Recursos Computacionales

Para la ejecución de esta estrategia de pruebas se va a hacer uso de los siguientes recursos

Equipos	
Mac Book Pro	Procesador M3 Ram 18 GB Disco 512
Mac Book Pro	Procesador M3 Ram 18 GB Disco 512
Mac Mini	Ram 32 gigas Procesador M4 pro
ASUS con Windows 11	Procesador Intel Core i5 Ram: 16GB Disco 512 SSD

2.4.2 Recursos para la Contratación de Servicios/Personal

Por ahora no se contempla la contratación de ningún tipo de servicio o personal que apoye la ejecución de la estrategia de pruebas.

2.5 TNT (Técnicas, Niveles y Tipos) de Pruebas

Nivel	Tipo	Técnica	Objetivo
Aceptación	Funcional, caja negra, Positivas / negativas	Manual, exploratorias	OBJ-01
Aceptación	Funcional, caja negra, Positivas,	APIs de automatización	OBJ-02
Aceptación	Funcional, caja negra, negativas,	APIs de automatización	OBJ-03
Componente	Caja blanca, regresión	APIs de automatización	OBJ-06
Componente	Caja blanca, negativas/ positivas	APIs de automatización	OBJ-06

2.6 Alcance

2.6.1 Sprint 1

Para el primer sprint se decidió implementar 4 historias de usuario las cuáles corresponden a las funcionalidades de listar álbumes, ver detalle de un álbum, listar artistas y ver detalle de un artista (HU01, HU02, HU03 y HU04). De acuerdo con lo estipulado por el equipo en cuanto a pruebas unitarias, cada uno de los componentes utilizados que contengan algún tipo de lógica de negocio (ViewModels, Repositories y

Service Adapters) deberán contar con sus respectivas pruebas unitarias y una cobertura superior al 80%. Por otra parte, respecto a las pruebas E2E a realizar, estás corresponderán a las funcionalidades a realizar durante el sprint, las cuáles fueron mencionadas con anterioridad y combinaciones entre las mismas.

Lista de funcionalidades e2e a probar

E001 - Navegabilidad Bottom Navigation Bar: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que al seleccionar cada una de las opciones del menú inferior (Álbumes, Artistas y Colecciones) lleva a las pantallas correspondientes de forma adecuada. En el caso de colecciones actualmente sólo se tiene una pantalla de preview, pues la funcionalidad aún se encuentra en el backlog.

E002 - Acceso a álbumes: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de álbumes y que al seleccionar un álbum se muestre el detalle del mismo correctamente.

E003 - Acceso a artista: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de artista y que al seleccionar un artista se muestre el detalle del mismo correctamente.

E004 - Álbumes lista vacía: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de álbumes se maneje correctamente el caso en el que se reciba una lista vacía del backend.

E005 - Artistas lista vacía: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de artistas se maneje correctamente el caso en el que se reciba una lista vacía del backend.

E006 - Álbum único: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de álbumes se maneje correctamente el caso en el que se reciba un único álbum del backend.

E007 - Artista único: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de artistas se maneje correctamente el caso en el que se reciba un único artista del backend.

E008 - Álbum estado vacío: Se debe validar que al navegar a la pantalla de detalle de un álbum, se maneje correctamente el caso en el que se reciban listas de comentarios y de canciones vacías.

E009 - Cambios en pantalla: Se debe validar que en la pantalla principal, cuando se den cambios en la pantalla (giros en direcciones que cambien la interfaz), la barra de

bottom navigation se continúe visualizando correctamente para que el usuario pueda seguir navegando.

E010 - Filtro de álbumes: Se debe validar que al navegar a la pantalla de álbumes e ingresar algún valor de búsqueda en la barra superior, se realice el filtrado correctamente.

E011 - Botón de atrás: Se debe validar que al navegar hacia cada una de las pantallas si se hace click en el botón de atrás se regrese a la pantalla correspondiente de forma correcta.

2.6.2 Sprint 2

Para la iteración 2 se desarrollaran las funcionalidades correspondientes a listar coleccionistas, ver detalle de un coleccionista, agregar un álbum al listado y comentar un álbum. De acuerdo con esto las pruebas a desarrollar durante esta iteración estarán enfocadas a dichas funcionalidades. Al igual que durante el sprint anterior, se espera que las pruebas unitarias para los componentes que manejen lógica de negocio superen un 80% de cobertura. Respecto a las pruebas e2e los escenarios que se esperan probar durante el sprint se encuentran a continuación.

Lista de funcionalidades e2e a probar

E012 - Acceso a coleccionistas: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de coleccionistas y que al seleccionar un álbum se muestre el detalle del mismo correctamente.

E013 - Coleccionistas lista vacía: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de coleccionistas y que se maneje correctamente el caso en el que se reciba una lista vacía del backend.

E014 - Coleccionista único: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de coleccionistas y que se maneje correctamente el caso en el que se reciba un único coleccionista del backend.

E015 - Ver detalle de coleccionista: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de detalle de coleccionista dando click en alguno de los coleccionistas listados en la lista de coleccionistas, en caso de no tener imagen el componente debe mostrar una letra con la inicial del nombre del coleccionista así mismo los álbumes del coleccionista deben estar listados con su respectiva portada.

E016 - Crear álbum: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado, probar que se pueda navegar a la pantalla de álbumes y que posteriormente se pueda navegar a la pantalla de crear álbum. Al diligenciar los datos

correctamente, se debe poder crear un nuevo álbum, el cuál se debe mostrar en la lista de álbumes.

E017 Crear álbum caso inválido: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado, probar que se pueda navegar a la pantalla de álbumes y que posteriormente se pueda navegar a la pantalla de crear álbum. Al diligenciar los datos incorrectamente, se deben mostrar los errores correspondientes en la interfaz gráfica para que el usuario pueda realizar dichas correcciones.

E018 - Crear comentario: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado, probar que se pueda navegar a la pantalla de álbumes y que posteriormente se pueda navegar al detalle de un álbum. Una vez en el detalle del álbum, si se llena el campo correspondiente correctamente, se debe poder añadir un comentario a un álbum.

Pruebas de desempeño

Para analizar el desempeño de la aplicación en los dispositivos Android se realizarán 2 actividades:

- En primer lugar, se utilizará la funcionalidad de Android linter con el objetivo de buscar posibles malas prácticas al momento de desarrollar las funcionalidades. Posterior a esto, se buscará corregir la mayor cantidad de smells posibles correspondientes a desempeño.
- En segundo lugar, se utilizará la funcionalidad de profiling de Android Studio, después de haber cargado una gran cantidad de datos al backend de la aplicación. Con el profiling, se buscará encontrar posibles clases o funcionalidades "greedy" en lo respectivo a uso de Memoria y CPU. En caso de encontrarse alguna ocurrencia se priorizará su corrección.

2.6.3 Sprint 3

Para la iteración número 3, únicamente se desarrollará la funcionalidad correspondiente a agregar tracks de un álbum para permitir al usuario actualizar el catálogo. De acuerdo con la estrategia que se ha seguido en los anteriores sprints, se deben desarrollar pruebas unitarias para cubrir esta funcionalidad en la capa de lógica de negocio. Asimismo, se deben desarrollar las pruebas E2E correspondientes. Los escenarios a probar son los siguientes:

Lista de funcionalidades e2e a probar

E019 - Añadir track al álbum: La aplicación se inicia correctamente. Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de detalle de un álbum. En la pantalla de detalle de un álbum se debe poder añadir un nuevo track al álbum. Una vez el usuario haga click en el botón de guardar, la información se debe almacenar correctamente en el backend.

E020 - Añadir un track al álbum sólo con caracteres especiales: Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de detalle de un álbum. En la pantalla de detalle de un álbum se debe poder añadir un nuevo track al álbum. La información del nombre del álbum se debe diligenciar únicamente con caracteres especiales. La aplicación debe informar al usuario que este es un nombre inválido pues sólo contiene caracteres especiales.

E021 - Añadir un track al álbum con más de 50 caracteres: Una vez el contenido inicial se encuentra cargado se debe probar que se pueda navegar a la pantalla de detalle de un álbum. En la pantalla de detalle de un álbum se debe poder añadir un nuevo track al álbum. La información del nombre del álbum se debe diligenciar con un nombre que tenga más de 50 caracteres. La aplicación debe informar al usuario que este es un nombre inválido pues excede el límite de caracteres permitidos.

Pruebas de desempeño

Al igual que durante la iteración anterior, se realizarán 2 actividades para analizar el desempeño de la aplicación:

- En primer lugar, se utilizará la funcionalidad de Android linter con el objetivo de buscar posibles malas prácticas al momento de desarrollar las nuevas funcionalidades. Posterior a esto, se buscará corregir la mayor cantidad de smells posibles correspondientes a desempeño.
- En segundo lugar, se utilizará la funcionalidad de profiling de Android Studio, después de haber cargado una gran cantidad de datos al backend de la aplicación. Con el profiling, se buscará encontrar posibles clases o funcionalidades "greedy" en lo respectivo a uso de Memoria y CPU. En caso de encontrarse alguna ocurrencia se priorizará su corrección.

Pruebas aleatorias y de exploración sistemática

Con el objetivo de encontrar posibles errores causados por corner cases se realizarán 2 actividades de pruebas:

- En primer lugar, se realizarán dos sesiones de pruebas aleatorias utilizando la herramienta Android Monkey. Para cada una de estas sesiones se ejecutarán al menos 10 escenarios de prueba cada uno con al menos 50 eventos aleatorios. Para tener trazabilidad de cada una de las ejecuciones se llevará el registro de las semillas utilizadas y en caso de encontrarse algún tipo de error el identificador el issue en Github. Un ejemplo de la tabla se puede ver a continuación.

id	Resultado	Semilla	Issues encontrados
EJ01	Exitoso/Fallido		

- En segundo lugar, se realizará una sesión de pruebas de exploración sistemática con el objetivo de buscar evaluar el correcto funcionamiento de la aplicación a profundidad. En esta sesión se ejecutarán 10 escenarios de GUI/Ripping cada uno con al menos 50 eventos. Al igual que con las pruebas de exploración sistemática se llevará un registro de las semillas utilizadas y los errores encontrados. El formato de tabla a utilizar será el mismo que el formato anterior.

2. 7 Referencias

[Repositorio GITHUB](#)