UNIVERSIDAD DE LOS ANDES

MAESTRIA EN INGENIERIA DE SOFTWARE

MISO

INGENIERIA DE SOFTWARE PARA DISPOSITIVOS MOVILES

ENTREGA FINAL MOVILES

DOCUMENTO DE ARQUITECTURA

Información del Equipo de Desarrollo

Proyecto	Desarrollo móvil	
Grupo	Grupo	
Integrantes	**Nombre**	**Rol**
	Juan David Portilla Montealegre	Product Owner
	Vladimir Alexander Yirsa Aperador	Scrum Máster
	Andrés Eduardo Cárdenas Jaramillo	Desarrollador Senior Backend Frontend
	Juan Sebastián Sánchez L.	Desarrollador Senior Frontend

Contexto del problema

Se requiere diseñar una aplicación que se encargue de mostrar a las personas que sean amantes de la música, una gran lista de canciones, discos y además ver quien tiene gustos similares o escucha el mismo artista y género. Para esto se decide diseñar una aplicación móvil que se va a encargar de ofrecer un servicio de administración de discos musicales, álbumes y de dar a conocer al público en general los discos extraños y perdidos que existan.

- El sistema permitirá al usuario interactuar con la aplicación por medio de una aplicación móvil en el sistema operativo Android.
- El sistema permitirá al usuario crear artistas, canciones y coleccionistas.
- El sistema permitirá al usuario obtener la información de un artista registrado, las personas que están registradas y que son coleccionistas de cierto tipo de música.

Puntos importantes para tener en cuenta

- La arquitectura ha sido desarrollada para las historias de usuario a implementar durante el desarrollo del curso de dispositivos móviles en la maestría en ingeniería de software de la Universidad de Los Andes.
- La arquitectura planteada hace énfasis en una aplicación desarrollada específicamente en el sistema operativo Android.
- La arquitectura se plantea en un sistema basado en una nube hibrida en una combinación de on premise y AWS, sin posibilidad de ser implementada en la nube pública. Sin embargo, si se desea adaptar, no debe tomar más de 200 horas/hombre.
- La arquitectura planteada es de una aplicación que no permite la interacción entre los usuarios, y no debe depender necesariamente de conexión a internet para funcionar correctamente, debido a que guardaría la información importante en la memoria interna del dispositivo.
- La información usada en la aplicación será almacenada directamente en el dispositivo donde la aplicación será instalada.

Estrategia de Arquitectura

Vista de Contexto

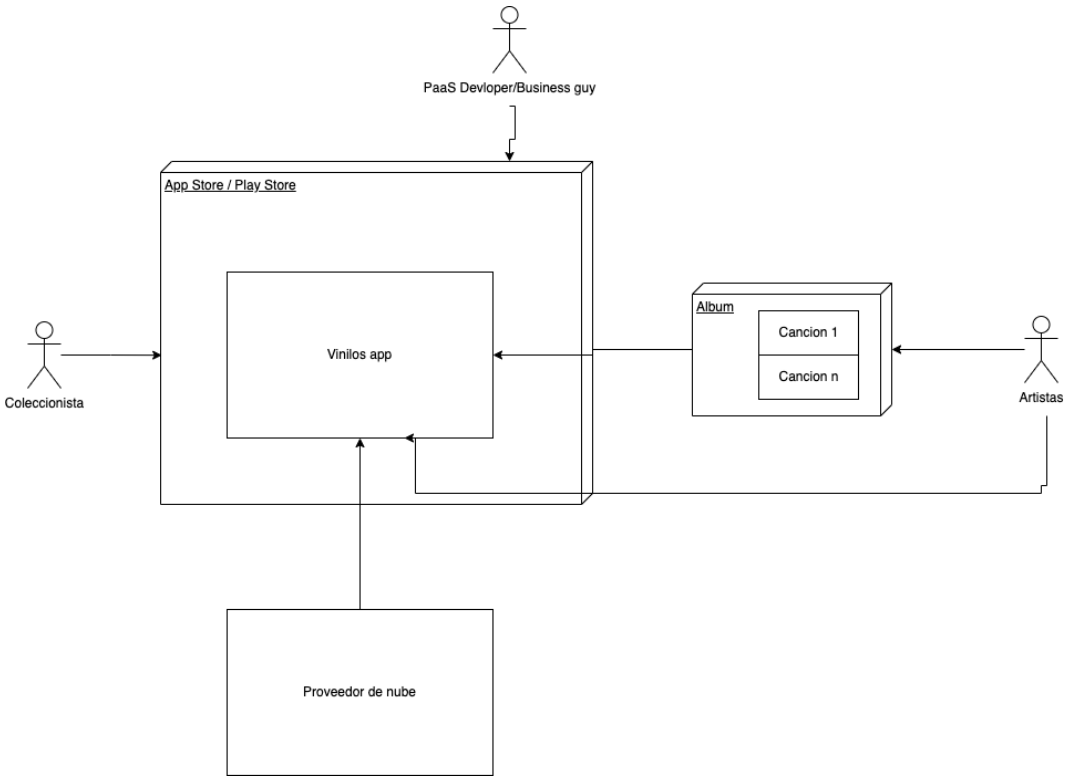

Diagrama 1. Vista de contexto

En la vista de contexto descrita anteriormente, se pueden observar los diferentes actores que actúan sobre la aplicación de Vinilos, el coleccionista, es el usuario de la aplicación que puede crear y administrar su perfil dentro de la administración, llevar a cabo una colección de artistas y álbumes que considere tener dentro de su cuenta. Los artistas son actores que influyen indirectamente sobre la aplicación Vinilos, ya que su negocio se centra en crear nuevos álbumes de música y vender reproducciones de sus discos, dar conciertos sin tener interés en ser parte activa de la aplicación Vinilos.

En el despliegue de la aplicación se encuentra la plataforma de App store y Play store donde la aplicación se estará disponible para que el usuario pueda descargarla y de la misma forma el proveedor de nube que dará las herramientas necesarias para que la funcionalidad de la aplicación vinilos pueda ser altamente escalable en el futuro del desarrollo de la aplicación.

Vista Funcional

Modelo Componente Conector

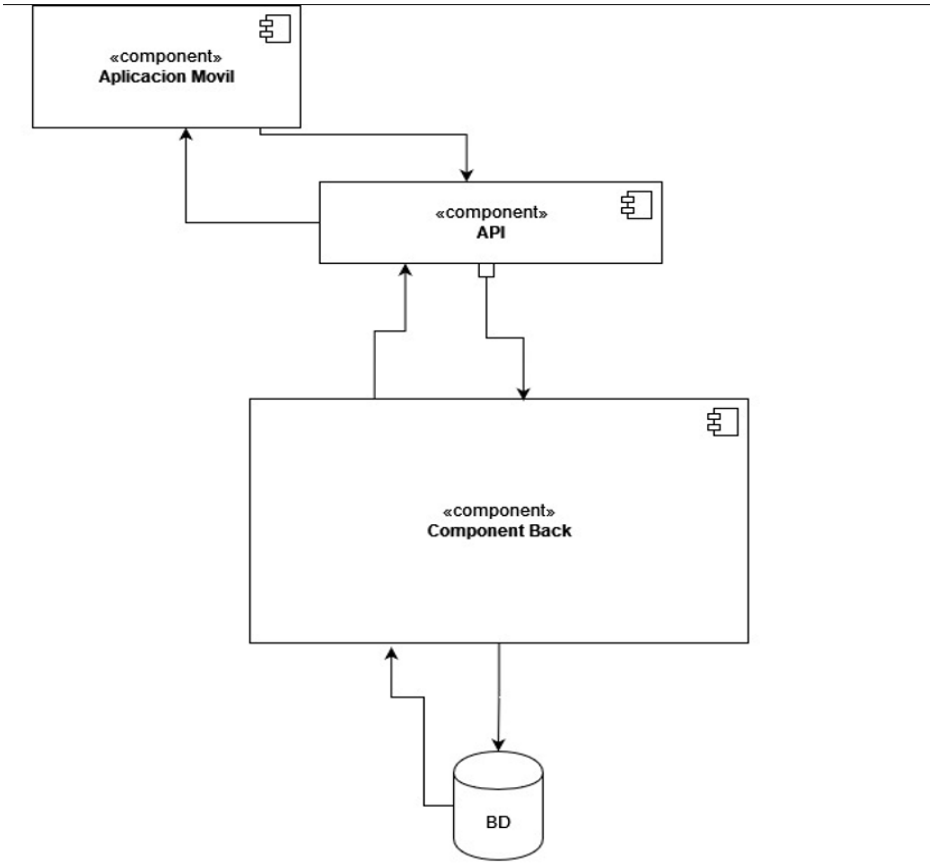

Diagrama 2. Modelo componente conector

En la vista funcional se puede analizar la arquitectura a alto nivel de la aplicación, donde el front-end se conecta con la API Gateway, quien intermedia con el back-end de la aplicación, la comunicación entre los componentes es síncrono, sin embargo, al evitar que la comunicación entre el front y el back sea por medio directo, reduce el acoplamiento de la aplicación.

De manera externa, el back se comunica con la base de datos de la aplicación, la comunicación entre estos dos componentes es síncrona y directa.

Arquitectura de la Aplicación.

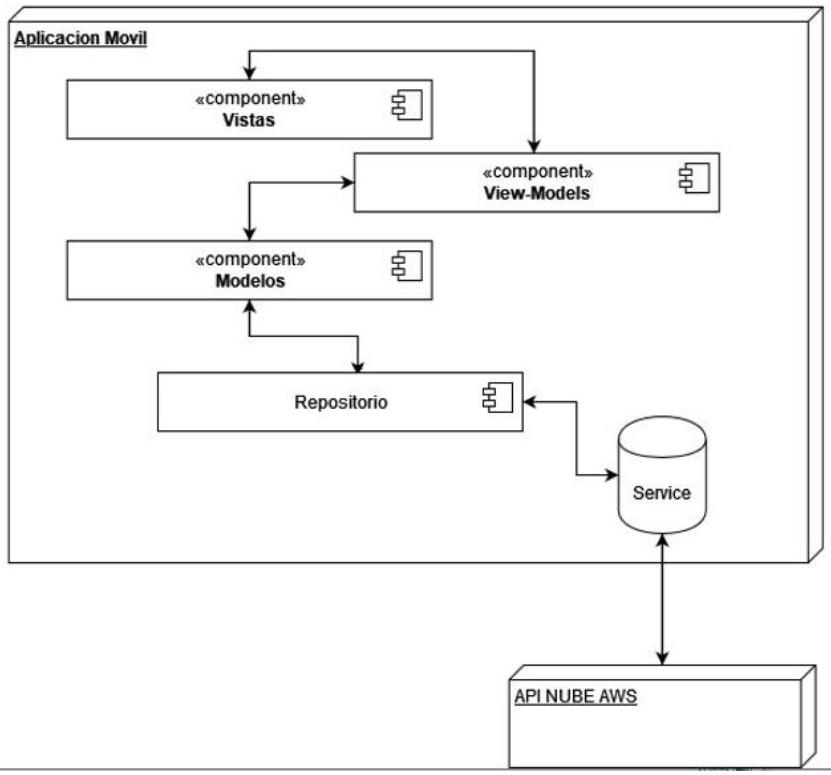

Diagrama 3. Vista funcional ampliada de la aplicación

En este diagrama se puede admirar la vista funcional a bajo nivel, donde se puede identificar el uso de los patrones de arquitectura como el patrón MVVM y repositorio, usados para el desarrollo de la aplicación, permitiendo una mejor comunicación entre los componentes del modelo y desacoplando la interacción entre la vista y el modelo, dejando la interacción en la vista-modelo.

Diagrama Flujo de Navegación

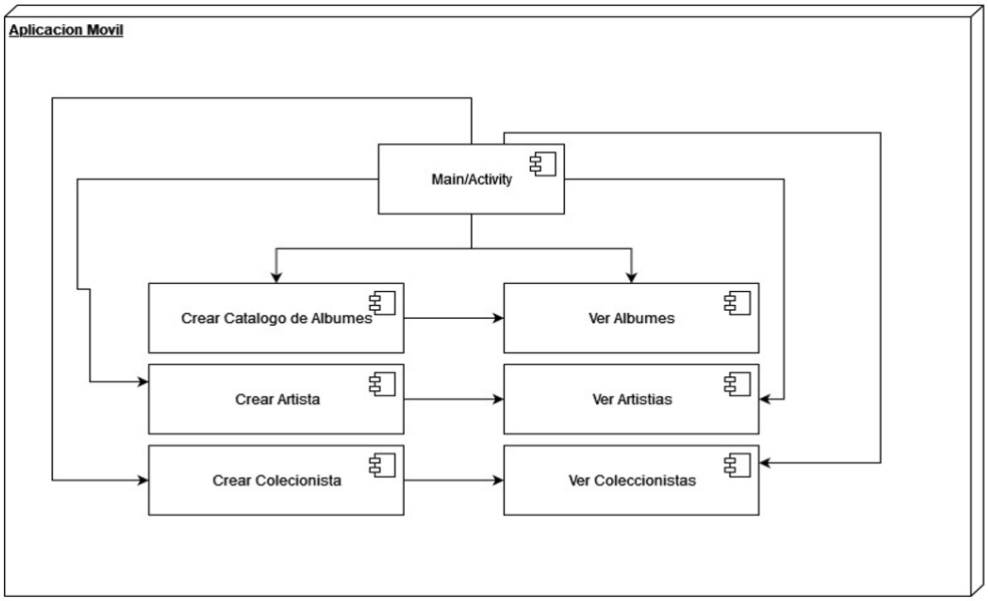

Diagrama 4. Diagrama de flujo de navegación.

En este diagrama se describe las posibles rutas de interacción que tendría el usuario de la aplicación vinilos, y como podría llegar de una actividad a otra por medio de la interacción que puede tener con las secciones y botones de la aplicación.

Vista de Información

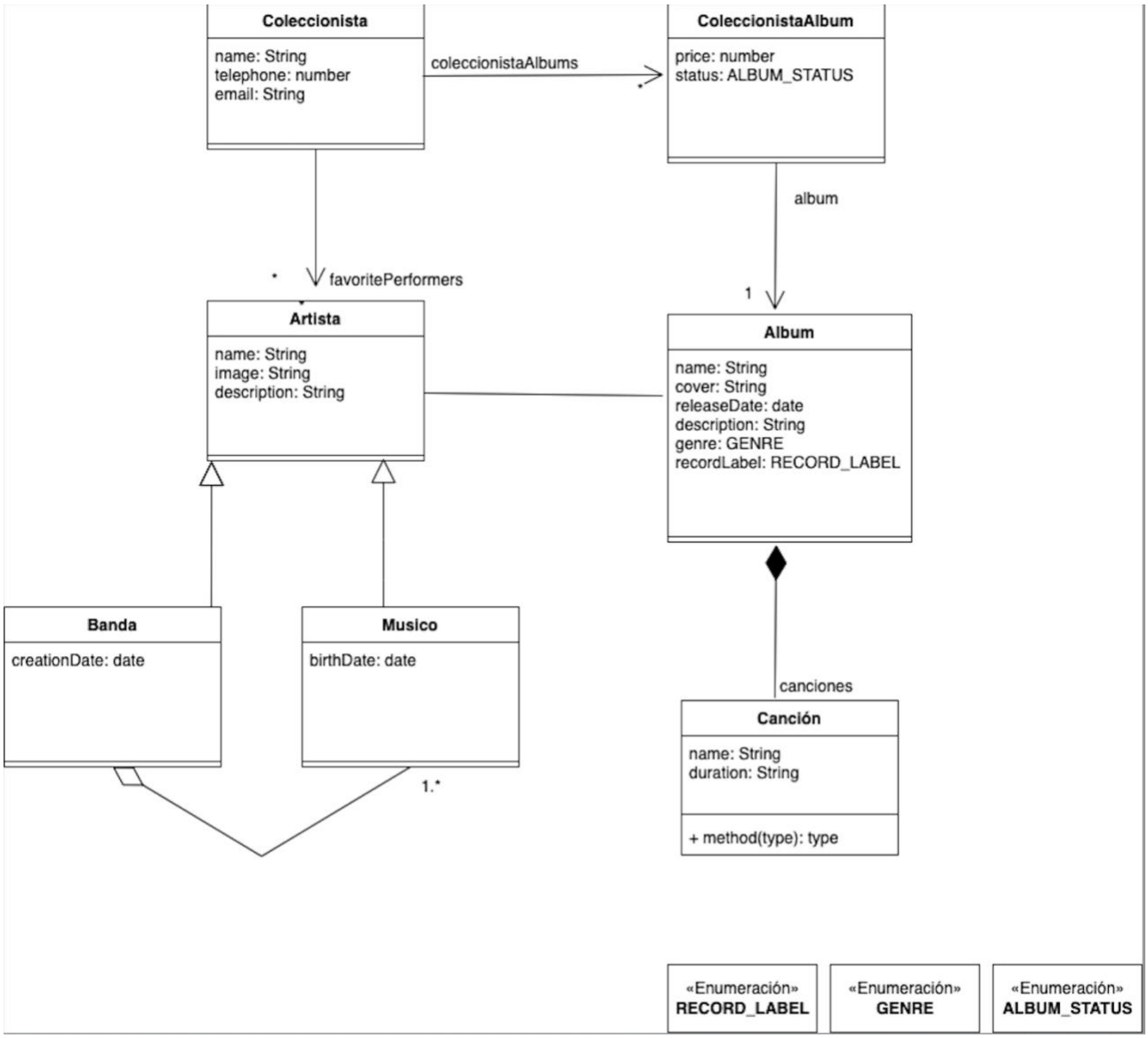

Diagrama 5. Vista de información.

En este diagrama se evidencia la forma en la que el flujo de información dentro de la aplicación se va a ver administrado por los diferentes componentes de la aplicación en su forma terminada, también como las clases, atributos y métodos tienen la información que se va a ver utilizada para el despliegue del usuario en la aplicación.

Un punto importante para tener en cuenta es que, aunque en nuestro componente backend contemplamos el uso de la separación entre musico y banda, en nuestro frontend y en la aplicación como tal, nos decidimos para usar solamente “Artista” para contemplar ambos de la misma forma, la razón de esto fue para facilidad de desarrollo y poder contemplar a ambos tipos de artista como uno solo. Además, en el mock up interactivo que desarrollamos, vimos a ambos como artista individual.

Vista de Despliegue

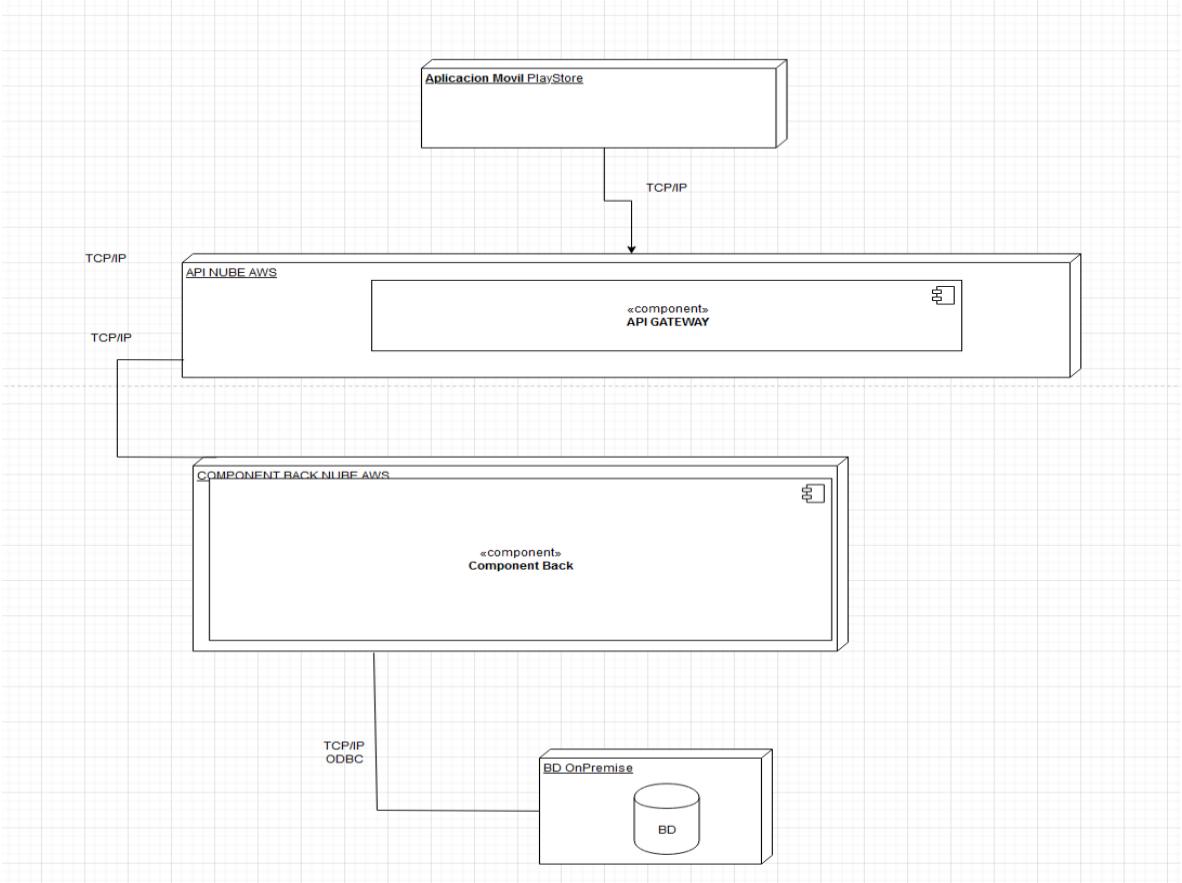

Diagrama 6. Vista de despliegue.

La aplicación estará desplegada en 3 grandes capas, donde en el front están los componentes propios de la interacción y la forma de mostrar la aplicación a los usuarios. Una capa intermediadora API Gateway que permite la comunicación entre el front y el back de una forma desacoplada y eficiente, el bloque back que contiene la lógica de negocio y su respectiva conexión con la base de datos.

Conclusiones

- El lenguaje de desarrollo Kotlin para aplicaciones Android es muy potente y permite diferentes configuraciones y ventajas, entre ellas que permite adaptarse con los diseños propios de Android.
- Es posible dentro de Kotlin implementar diferentes patrones de diseño detallado y patrones de arquitectura para mejorar las aplicaciones desarrolladas con Kotlin.
- Se lograron desarrollar las HU planteadas en el curso de manera satisfactoria.