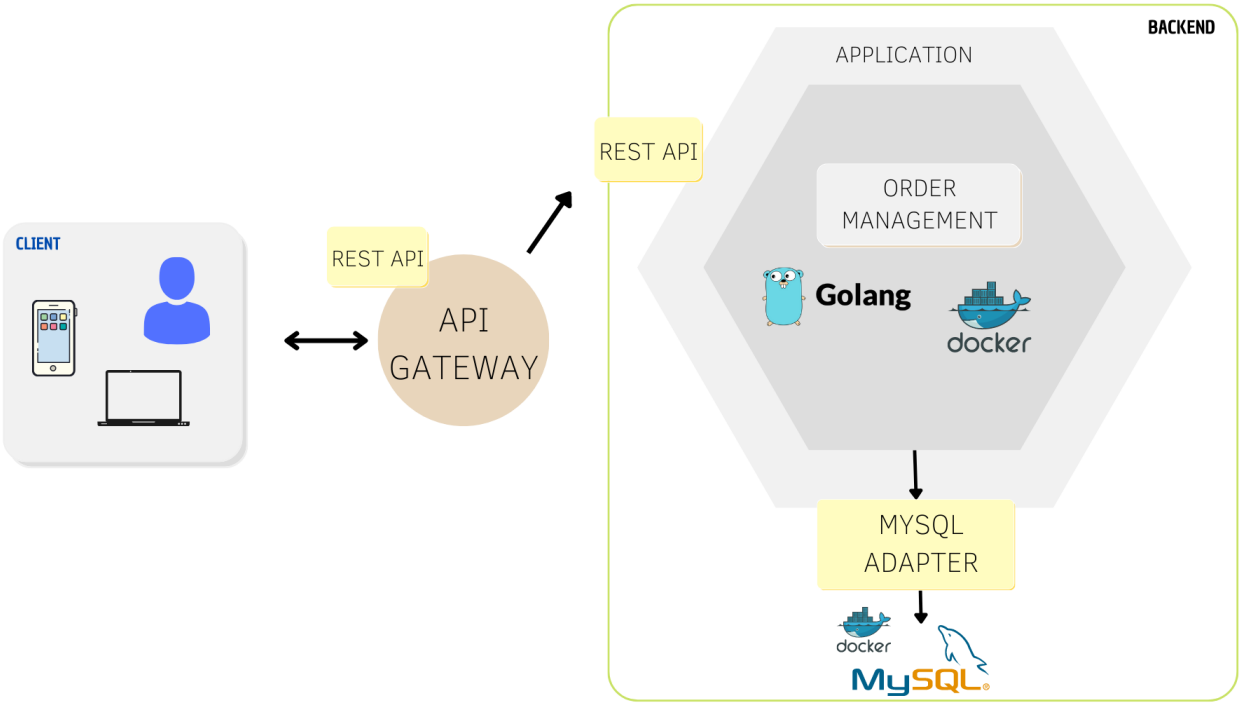

Figura 1: Arquitectura Aplicación

Para la mejora de este microservicio se propone implementar clean architecture, debido a que en comparación con otras arquitecturas como layered architecture, clean nos permitirá realizar una implementación inmediata, realizar cambios sin grandes impactos como por ejemplo cambiar de framework o base de datos de manera sencilla, además es ideal para proyectos que seguirán creciendo con el tiempo ya que layered architecture la aplicación no esta desarrollada orientada al negocio por lo tanto la capa de dominio tiene dependencias con la capa de datos lo que hace que sea menos escalable.

Cabe resaltar la importancia del API GATEWAY en la arquitectura de la aplicación debido a que se reflejan mejoras en acoplamiento, seguridad, rendimiento, entre otros. También se recomienda separar aparte la base de datos

del microservicio de distribución de un listado de órdenes, ya que proporciona mayor seguridad debido a que se puede configurar por cada servicio roles, autenticación, autorizaciones y cifrados de datos; asimismo se podrá seleccionar el gestor más adecuado para las necesidades de cada microservicio. Finalmente, el uso de Docker ayuda a que sea más sencillo manejar los microservicios en contenedores, permitiendo ejecutar múltiples contenedores por instancia y manteniendo la cogerencia entre estas; asimismo admite múltiples lenguajes de codificación y automatiza el monitoreo.

Teniendo en cuenta esto las tecnologías recomendadas a migrar son Golang debido a que este tiene diferentes ventajas con respectos a otros lenguajes de uso frecuente como por ejemplo Python, Nodejs, entre otros. Golang es un lenguaje que cumple con todos los criterios para una buena tecnología de microservicios, es de alto rendimiento de procesamiento, adopta la cultura de automatización, permite la descentralización de componentes, es de gran facilidad encontrar apoyo, además permite mantener una cultura de integración y despliegue continuo. Sin embargo algunas de las características que lo diferencian entre otros lenguajes, son la sintaxis de Golang es relativamente simple, lo que facilidad comenzar a escribir código funcional inmediatamente lo que permitirá que sea fácil para los empleados de este microservicio aprender y comenzar a reescribir el código; además Golang tiene un tiempo de compilación mejor en comparación a otros lenguajes de programación populares y debido a la estricta separación de módulos este promueve componentes independientes para proporcionar una lógica empresarial la cual sigue los patrones de diseño de microservicios; lenguajes como NodeJS y Python nos brindan un desempeño excelente para las tareas de desarrollo sin embargo Golang viene con funciones integradas que son más adecuadas para arquitecturas de microservicios.

Para la elección de Base de datos se debe tener en cuenta inicialmente la estructura de los datos, si siempre se mantendrá una estructura y estos estarán relacionados entre sí será recomendado utilizar bases de datos relaciones, de lo contrario si no se tiene una estructura de datos clara y predefinida será mejor una base de datos no relacional como MongoDB. Contemplando la cantidad de datos que se manejan y la estructura de estos no será necesario recurrir a una base de datos no relacional cuando una relacional lo podría hacer de una manera más eficiente, comparando diferentes motores de bases como PostgreSQL y MySQL; MySQL tiene ventajas en la configuración y es ideal para transacciones de datos sencillas en los que la lectura de datos es lo más importante, por eso teniendo en cuenta la petición requerida en el microservicio será de mejor elección este motor de base de datos.

Referencias

[1] https://www.mindinventory.com/blog/technologies-for-mircroservices-architecture/

[2] https://www.ideamotive.co/blog/building-a-microservice-in-go-golang-business-guide

[3] https://xurxodev.com/por-que-utilizo-clean-architecture-en-mis-proyectos/

[4] https://blog.mdcloud.es/postgresql-vs-mysql-cual-usar-para-mi-proyecto/Rendimiento

[5] https://towardsdatascience.com/microservice-architecture-and-its-10-most-important-design-patterns-824952d7fa41

[6] https://tsh.io/state-of-microservices/ebook