**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**



**Trabajo Grupal U3:**

**Patrones de microservicios**

Curso:  Patrones de Software

Docente: Ing. Patrick Cuadros Quiroga

Integrantes:

|  |  |
| --- | --- |
| **Aranda Reyes Diego Andre** | **(2019063855)** |
| **Mamani Lima, Erick Mauricio** | **(2020066321)** |
| **Condori Loayza, Helbert Andres** | **(2020067571)** |
| **Cotrina Caceres, Sebastian Airton** | **(2020067569)** |
| **Paz Huaychani, Frank Kevin** | **(2019063321)** |
| **Gallegos Laucata, Alvaro Jhoel** | **(2017057608)** |

**Tacna – Perú**

**2023**

Contenido

[I. INTRODUCCIÓN 3](#_Toc139896864)

[II. DESARROLLO 3](#_Toc139896865)

[DATABASE PER SERVICE PATTERN 4](#_Toc139896866)

[API GATEWAY PATTERN 4](#_Toc139896867)

[HEALTHCHECK API 5](#_Toc139896868)

[III. CONCLUSIONES 5](#_Toc139896869)

# INTRODUCCIÓN

Los microservicios ofrecen varios beneficios en el desarrollo de software. Al dividir una aplicación en componentes pequeños y autónomos, los equipos de desarrollo pueden trabajar de manera más eficiente y en paralelo. Cada microservicio puede ser implementado, probado y desplegado de forma independiente, lo que agiliza el ciclo de desarrollo y permite actualizaciones más rápidas. Además, al tener una base de código más pequeña y centrada en una función específica, los microservicios son más fáciles de entender, mantener y escalar.

La arquitectura de microservicios también fomenta la reutilización de componentes. Dado que cada servicio se enfoca en una tarea específica, estos pueden ser utilizados por diferentes aplicaciones dentro del ecosistema empresarial. Esto reduce la duplicación de esfuerzos y facilita la creación de nuevas aplicaciones a partir de servicios existentes, lo que a su vez acelera el tiempo de lanzamiento al mercado. Asimismo, los microservicios promueven la resiliencia y la tolerancia a fallos. Al tener servicios independientes, un fallo en un microservicio no afecta a los demás. Además, la escalabilidad horizontal permite agregar o eliminar instancias de servicios según las necesidades de carga, lo que garantiza un rendimiento óptimo incluso en momentos de alta demanda.

Sin embargo, adoptar una arquitectura basada en microservicios requiere un enfoque cuidadoso. La gestión de múltiples servicios implica la necesidad de orquestar y coordinar las comunicaciones entre ellos, lo que puede aumentar la complejidad. Además, se deben implementar estrategias de control de versiones y manejo de datos para garantizar la consistencia en todo el sistema.

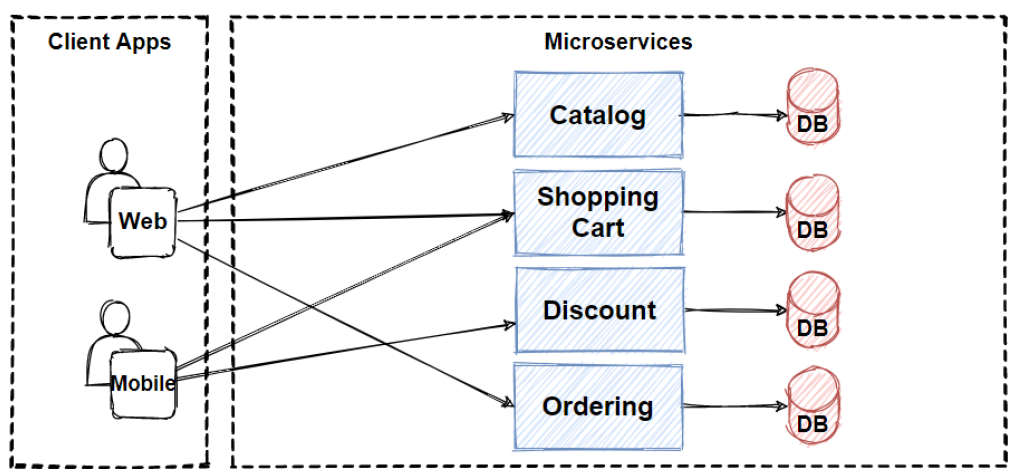
# DESARROLLO

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

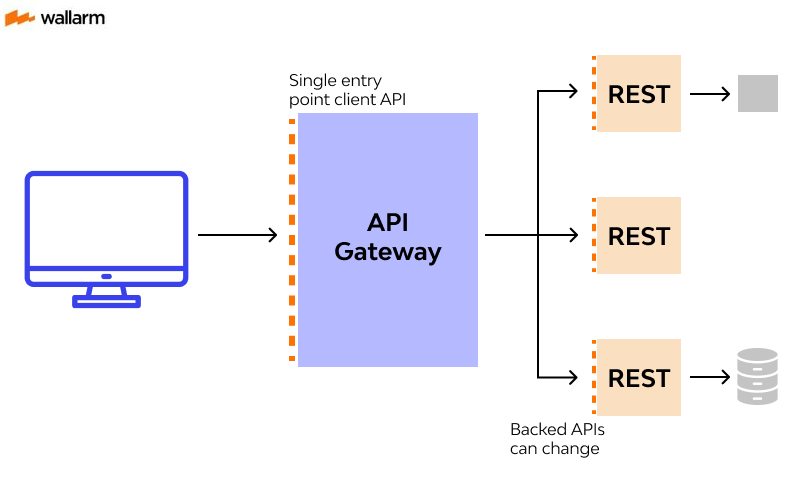
## DATABASE PER SERVICE PATTERN

La implementación del patrón "Database per Service" en una API de servicios implica asignar una base de datos dedicada a cada servicio. Esto permite que cada servicio tenga su propio conjunto de datos y lógica de negocio independiente, mejorando la escalabilidad, el rendimiento y la independencia de cada servicio. Cada base de datos por servicio facilita el mantenimiento, la evolución y la seguridad individual de los servicios, lo que contribuye a una arquitectura modular y eficiente en la API de servicios.



## API GATEWAY PATTERN

La implementación del patrón de "API Gateway" en una API de servicios tiene como objetivo centralizar y simplificar el enrutamiento, la seguridad y la gestión de las solicitudes de los clientes. Al utilizar un componente centralizado como el API Gateway, se simplifica el acceso a la API para los clientes, se gestiona el enrutamiento de las solicitudes a los servicios correspondientes, se implementan políticas de seguridad y se realizan transformaciones de datos según sea necesario. Este enfoque mejora la escalabilidad, la seguridad y la eficiencia de la API, al tiempo que facilita su mantenimiento y evolución.



## HEALTHCHECK API

La implementación del patrón de "Healthcheck API" en una API de servicios tiene como objetivo principal monitorear y reportar el estado y la disponibilidad de los servicios en tiempo real. Al proporcionar una ruta específica para realizar "healthchecks" periódicos, se puede detectar y diagnosticar rápidamente cualquier problema o falla en los servicios, permitiendo una respuesta temprana y eficiente. Esto mejora la experiencia del usuario al garantizar una alta disponibilidad y un rendimiento óptimo de la API, además de facilitar la toma de decisiones informadas sobre la escalabilidad y la optimización de recursos. El patrón de "Healthcheck API" brinda un monitoreo proactivo, detección temprana de fallas y mejor gestión de los servicios en una API de servicios.

Diagrama

Descripción generada automáticamente

# CONCLUSIONES

* La arquitectura de microservicios ofrece agilidad y escalabilidad: Al dividir una aplicación en componentes autónomos, los microservicios permiten a los equipos de desarrollo trabajar de forma independiente y realizar actualizaciones y despliegues de manera más rápida. Además, la capacidad de escalar horizontalmente los microservicios garantizan un rendimiento óptimo incluso en momentos de alta demanda.
* La reutilización de componentes mejora la eficiencia del desarrollo: Con los microservicios, los componentes se pueden reutilizar en diferentes aplicaciones, lo que reduce la duplicación de esfuerzos y acelera el tiempo de lanzamiento al mercado. Esto permite una mayor eficiencia en el desarrollo de nuevas aplicaciones a partir de servicios existentes, lo que a su vez impulsa la innovación y la competitividad.