UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Proyecto *Sistema de Registros de Salud e Inspección Sanitaria - SIRESA*

Curso: PROGRAMACIÓN III

Docente: *Ing. Juan Manuel Choque Flores*

Integrantes:

Huaman Rivera, Roberto Carlos ***2021071077***

Tacna – Perú
2025

Proyecto Sistema de Registros de Salud e Inspección Sanitaria
Descripción del proyecto final

Versión 1.*0*

¿Que son microservicios?

La arquitectura de microservicios es un estilo de diseño de software en el que una aplicación se estructura como un conjunto de servicios pequeños, autónomos y enfocados en resolver funciones específicas del sistema. A diferencia de la arquitectura monolítica, donde todas las funcionalidades están integradas en un solo bloque de código, los microservicios permiten separar la lógica del negocio en módulos individuales que pueden desarrollarse, desplegarse y escalarse de forma independiente.

Cada microservicio:

Tiene su propia base de datos o repositorio de datos (cuando es necesario).

Se comunica con otros servicios a través de protocolos ligeros como HTTP (REST), mensajería (RabbitMQ, Kafka) o gRPC.

Puede estar desarrollado en un lenguaje diferente al de los demás microservicios.

Puede ser implementado en un contenedor (como Docker), lo que facilita su gestión y despliegue.

Este enfoque refleja una evolución natural en el desarrollo de software moderno, especialmente en entornos donde se requiere escalabilidad, flexibilidad y mantenimiento ágil.

Para que sirven (Utilidad)

La arquitectura de microservicios se ha popularizado debido a los múltiples beneficios que ofrece, especialmente en entornos de desarrollo ágil y en aplicaciones distribuidas que necesitan crecer de forma dinámica. Algunas de sus utilidades clave incluyen:

1. Escalabilidad

Permite escalar solamente los componentes del sistema que requieren más recursos, sin necesidad de duplicar toda la aplicación.

2. Desarrollo independiente

Diferentes equipos pueden trabajar de forma simultánea en distintos microservicios sin interferir en el desarrollo de los demás, acelerando la entrega de nuevas funcionalidades.

3. Despliegue continuo

Al estar desacoplados, se puede actualizar o desplegar un microservicio sin afectar el resto del sistema, lo que permite implementar mejoras o correcciones de manera continua.

4. Resiliencia

El fallo de un microservicio no necesariamente compromete el funcionamiento del sistema completo. Se pueden aislar errores y manejar la tolerancia a fallos de forma granular.

5. Uso de tecnologías diversas

Cada microservicio puede ser desarrollado con la tecnología más adecuada para su función, lo que mejora el rendimiento y la eficiencia.

6. Facilidad de mantenimiento

Al ser módulos pequeños, es más fácil entender, probar y modificar el código, lo que reduce errores y facilita la evolución del sistema.

4 ejemplos de microservicios (puesta en marcha)

1. Microservicio de Autenticación

Objetivo: Gestionar el registro, inicio de sesión y validación de identidad de los usuarios mediante tokens (como JWT - JSON Web Token).

Tecnologías comunes: Node.js + Express, Spring Boot, Python (FastAPI), Auth0.

Caso práctico:

En una app Flutter para gestión de inspecciones, se implementa un microservicio exclusivo para login. El cliente (app) solicita credenciales, y este servicio genera un JWT que se usará para autorizar otras peticiones en los distintos servicios del sistema.

2. Microservicio de Gestión de Usuarios y Roles

Objetivo: Administrar información de usuarios, perfiles, roles, permisos y estados.

Tecnologías comunes: FastAPI, NestJS, Laravel Lumen, Spring Boot.

Caso práctico:

En una plataforma educativa, el microservicio de usuarios gestiona docentes, estudiantes, administrativos, y los permisos de cada uno. Cada vez que otro servicio necesita validar si un usuario tiene acceso, consulta este microservicio.

3. Microservicio de Notificaciones

Objetivo: Enviar correos electrónicos, notificaciones push o SMS a los usuarios ante eventos relevantes.

Tecnologías comunes: Node.js con Nodemailer o Firebase, Go, servicios externos como Twilio o SendGrid.

Caso práctico:

Cuando un decano aprueba una solicitud académica en una app de gestión estudiantil, el microservicio de notificaciones recibe el evento y envía automáticamente un correo y una notificación push al alumno.

4. Microservicio de Inventario / Gestión de recursos

Objetivo: Controlar el stock de productos, movimientos de entrada y salida, y estados de recursos.

Tecnologías comunes: Java (Spring Boot), .NET Core, MongoDB, PostgreSQL.

Caso práctico:

En un sistema de fiscalización de locales, un microservicio administra el equipamiento (como tablets, alcoholímetros, cámaras) y asigna los recursos a cada inspector según disponibilidad.