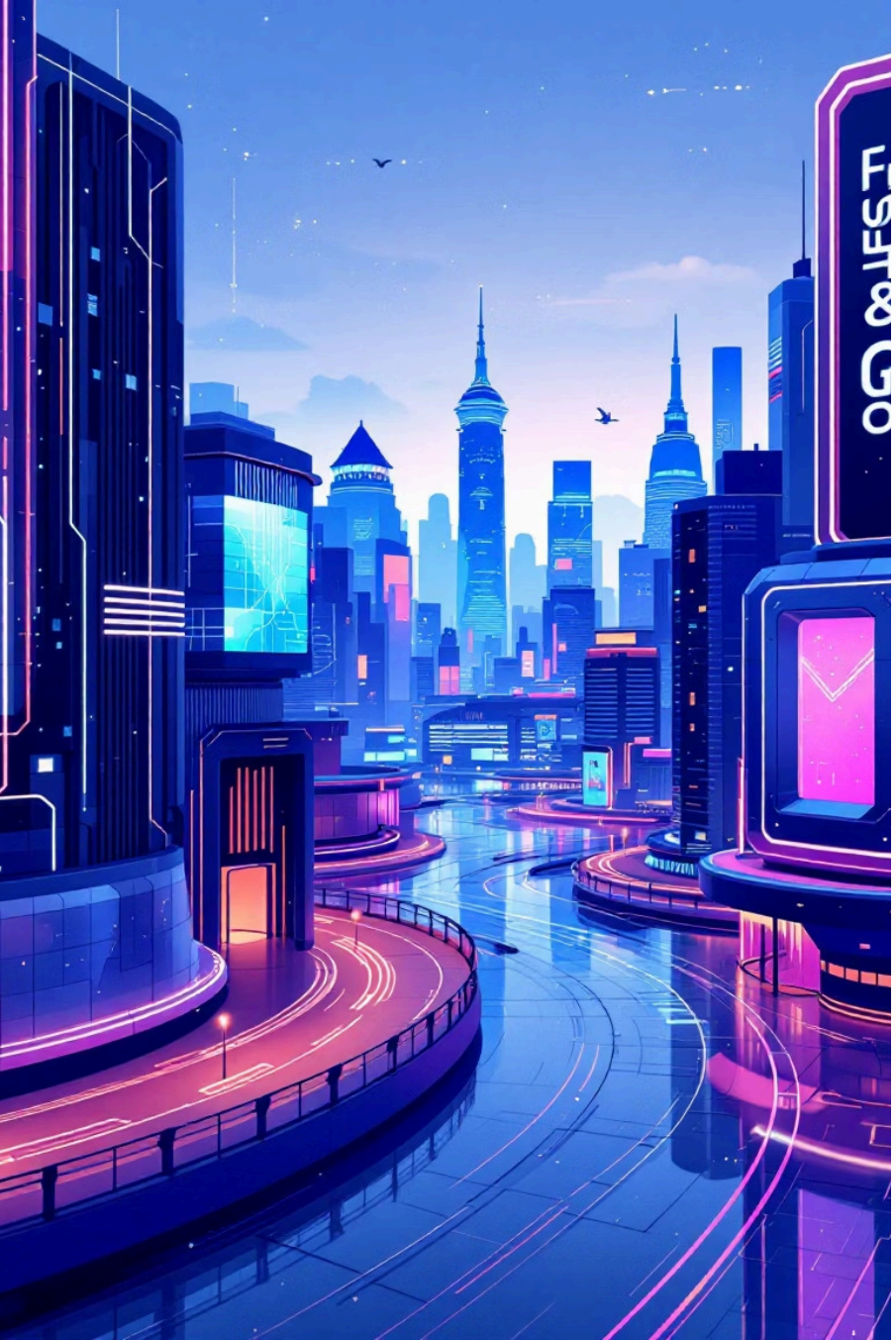


Fresh&Go: Avance Hacia Microservicios Reales

Esta presentación detalla la transición, simulación y desafíos de la Práctica 2 del proyecto Fresh&Go, marcando un hito en la evolución de nuestra arquitectura de microservicios

De la Teoría a la Implementación: Práctica 1 vs. Práctica 2

Práctica 1: Solo JSON Schemas

Iniciamos con esquemas JSON, sin microservicios operativos. Los datos eran confusos, con campos redundantes e irrelevantes.

Práctica 2: Salto al Software Real

Implementación de microservicios CRM (Node.js + Express) e IoT (Python + FastAPI), ejecutándose por primera vez en terminales separadas.

Esta fase representa un paso crucial de la conceptualización a la ejecución práctica de nuestra arquitectura.

Rediseño de Datos en CRM: Claridad para el Negocio

```
{
  "id": "COND002",
  "nombre": "Ana Rodríguez López",
  "email": "ana.rodriguez@freshgo.com",
  "telefono": "+34 600 333 444",
  "vehiculoAsignado": "VEH002",
  "licenciaConducir": "0087654321",
  "fechaExpiracionLicencia": "2025-11-30",
  "disponibilidad": false,
  "estado": "activo",
  "experiencia": 8,
  "calificacion": 4.9,
  "rutas": [
    "Sevilla Centro",
    "Triana"
  ]
},
```

En la Práctica 1, los datos JSON estaban sobrecargados de información inútil para el negocio. Inicialmente teniamos:

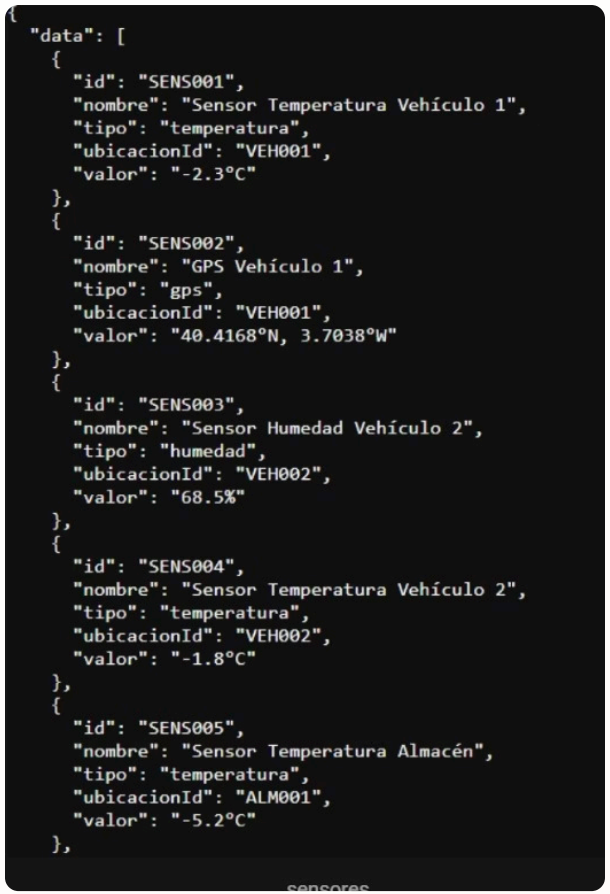
- Campos como experiencia, calificación y las rutas asignadas a un conductor

Para la Práctica 2, hemos limpiado y rediseñado los JSON del Conductor, enfocándonos solo en los campos relevantes para la toma de decisiones empresariales.

- Simplificación de eso en campos estado y disponibilidad, si tiene un pedido asignado no estara disponible

Este enfoque garantiza que nuestros microservicios operen con información precisa y valiosa.

Rediseño de Datos en IoT: Claridad para el Negocio



En la Práctica 1, los datos JSON estaban sobrecargados de información inútil para el negocio. Inicialmente teniamos:

- Sensores de varios tipos: Temperatura, GPS, humedad y presión

Para la Práctica 2, hemos limpiado y rediseñado los JSON del Sensor , enfocándonos solo en los campos relevantes para la toma de decisiones empresariales.

- Un solo sensor con temperatura y GPS

Este enfoque garantiza que nuestros microservicios operen con información precisa y valiosa.

Tecnologías Implementadas en la Práctica 2

Microservicio CRM

- Desarrollado con Node.js y Express.
- Uso de AJV para validación de esquemas.
- Integración de middlewares para gestión de peticiones.

Microservicio IoT

- Implementado en Python con FastAPI.
- Utiliza JSONSchema para la validación de datos.
- Servidor Uvicorn para la ejecución asíncrona.

Estas tecnologías nos permiten construir una arquitectura robusta y escalable.

Diferencias en los microservicios de la Práctica 2

La Práctica 2 opera exclusivamente con datos simulados, cargados directamente desde archivos JSON, prescindiendo de una base de datos real.

Ejecución Separada

El microservicio CRM se ejecuta en una terminal CMD, mientras que el microservicio IoT lo hace en WSL, permitiendo una operación independiente y simulando un entorno real.

Endpoints Definidos

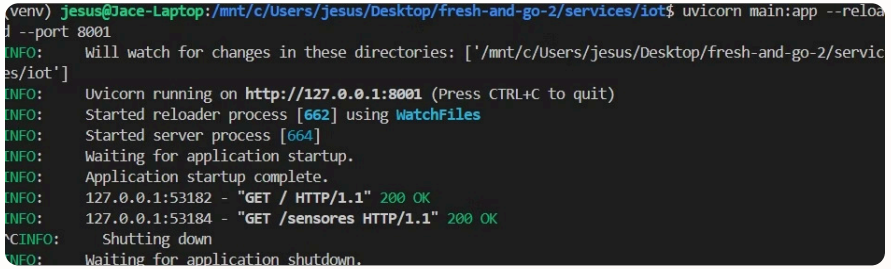
Cada servicio tiene endpoints específicos que permiten interactuar con los datos simulados, facilitando las pruebas y el desarrollo.

Este enfoque nos permite validar la funcionalidad de los microservicios antes de la integración con bases de datos.

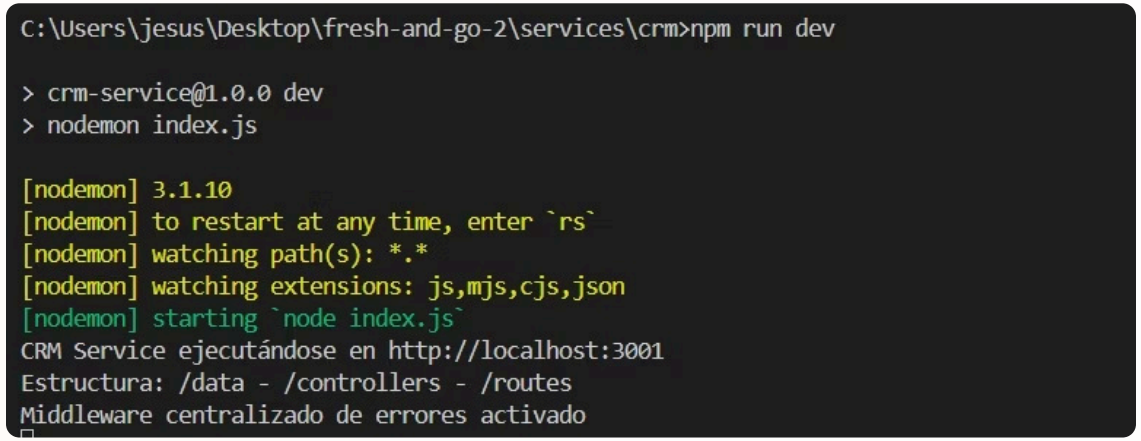
Breve ejecución: Terminal

A continuación, se mostrarán las capturas de la terminal, demostrando la ejecución de ambos microservicios.

IoT



CRM



Estas imágenes validan el funcionamiento y la correcta ejecución de nuestra arquitectura de microservicios.

Problemas Técnicos y Soluciones (Práctica 2)

Datos Excesivos en Consultas

Solución: Rediseño de respuestas para incluir solo datos relevantes, como evitar información del vehículo al consultar un cliente.

JSON con Campos Inútiles

Solución: Rediseño completo de los esquemas JSON para incluir únicamente datos útiles para el negocio.

Problemas GitHub y Soluciones (Práctica 2)

Estructura del repositorio

Al inicio, la estructura del repositorio no era clara, generando confusión sobre la ubicación de archivos

Solución: Tener claro tres ramas; una para código ejecutable (.py, .js, etc), otro para datos simulados (.json, .sql, etc) y otro para la documentación (.md, .pdf, etc)

Commits

En lugar de múltiples pequeños commits por cada cambio

Solución: Aprendimos la filosofía de "commits significativos", consolidamos cambios para reflejar hitos importantes, mejorando la trazabilidad y organización.

Fresh&Go: En camino hacía un producto de Software Real

La Práctica 2 consolida los cimientos de Fresh&Go como un producto de software viable y funcional.

Microservicios Operativos

Ahora contamos con microservicios CRM e IoT funcionando de manera independiente y coordinada.

Datos Limpios y Listos

Los datos están depurados y estructurados, preparados para la migración a una base de datos SQL.

Hacia un Sistema Integral

Fresh&Go comienza a tomar forma como una solución de software completa y coherente.

Próximos Pasos: La Práctica 3

La Práctica 3 se construirá sobre los cimientos de la Práctica 2, integrando bases de datos SQL y una API unificada.

Integración SQL

Migración de datos simulados a bases de datos SQL reales.

API Unificada

Desarrollo de una API que centralice la comunicación con los microservicios.

Arquitectura más completa

Consolidación de la arquitectura de Fresh&Go.

Este será el paso final hacia un sistema de software completamente funcional y robusto.