2025

FRESH&GO

DE LA TIENDA A TU MESA EN UN CLICK

FRESH&GO 3

MAR ALEJANDRA QUISPE ESCALANTE Y JESUS LACRUZ

Índice

1 Introducción

La Práctica 3 supone la evolución natural de la arquitectura iniciada en la Práctica 2. En esta fase se introduce:

- PostgreSQL como base de datos real
- scripts SQL para crear tablas y datos semilla
- Eliminación de la carpeta /data
- La creación de la API Unificada, encargada de combinar datos de CRM e IoT mediante HTTP

Esta práctica representa el salto de simulación → persistencia real → integración.

2 Cambios y ajustes

Respecto a la Práctica 2:

- Se sustituyen los JSON por datos reales en PostgreSQL.
- Se añaden los scripts crear_tablas.sql y datos_semilla.sql.
- Se reorganiza la carpeta /database.
- CRM e IoT ahora consultan la base de datos.
- Se crea una API Unificada con:
    - controllers,
    - schemas,
    - validación AJV,
    - middleware de errores,
    - endpoints /clientes/detalle/:id y /resumen.

3 Estructura

3.1 BBDD

```
database/
  crear_tablas.sql          # Crea todas las tablas del proyecto
  datos_semilla.sql         # Inserta registros iniciales (clientes, sensores...)
```

3.2 CRM Service (Node.js + SQL)

```
services/crm/
  routes/
    clientes.js             # Rutas REST para clientes
    pedidos.js              # Rutas REST para pedidos
  controllers/
    clientesController.js   # Lógica de consulta a la BBDD
    pedidosController.js    # Controlador de pedidos con SQL
  database/
    connection.js           # Configuración de conexión PostgreSQL
  schemas/
    cliente.schema.json     # Validación del cliente
    pedido.schema.json      # Validación del pedido
  middleware/
    errorHandler.js         # Manejo estandarizado de errores
  index.js                  # Levantamiento del servicio CRM
```

3.3 IoT Service (FastAPI + SQL)

```
services/iot/
 routes/
   sensores_router.py      # Endpoints de sensores
   lecturas_router.py      # Endpoints de lecturas IoT
   vehiculos_router.py     # Endpoints de vehículos
 controllers/
   sensores_controller.py  # Lógica y filtros aplicados a sensores
   lecturas_controller.py  # Lecturas con filtros de fechas
   vehiculos_controller.py # Estado y consulta de vehículos
 database/
   connection.py          # Conexión PostgreSQL para IoT
 schemas/
   sensor_schema.py       # Validación Pydantic del sensor
   lectura_schema.py      # Validación de lecturas
   vehiculo_schema.py     # Validación de vehículos
 exceptions/
   error_handler.py       # Manejo de excepciones en FastAPI
 main.py                 # Arranque del servicio IoT
```

3.4 API Unificada

```
services/api-unificada/
 controllers/
   clienteController.js     # Combina datos CRM + IoT para detalle del cliente
   resumenController.js     # Estadísticas agregadas del sistema
 lib/
   crmClient.js           # Peticiones HTTP al servicio CRM
   iotClient.js           # Peticiones HTTP al servicio IoT
   validator.js           # Configuración AJV para validar el schema
   cache.js               # Sistema de cacheo de respuestas
   logger.js              # Registro de errores e integración
 schemas/
   cliente-detalle.schema.json  # Validación unificada del detalle del cliente
 index.js                 # Configuración principal y rutas
 README.md                # Documentación del módulo
```

4. Funcionamiento General

En esta práctica:

- CRM e IoT consultan su propia base de datos PostgreSQL.
- La API Unificada combina información de ambos mediante HTTP.
- Se valida la salida con AJV y el schema unificado.
- Los endpoints permiten obtener un detalle completo y un resumen global.
- El sistema se comporta como un ecosistema de microservicios real

5. Simulación

Nota: Esta práctica requiere la instalación de requirements.txt

En terminal

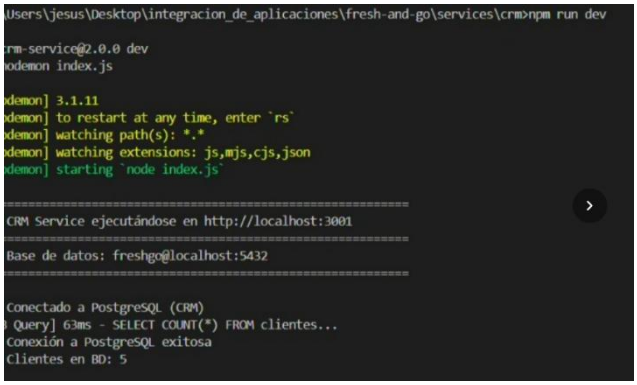


ILUSTRACIÓN 1: EJECUCIÓN DE CRM EN CMD

En el caso del servicio CRM, el arranque se realiza ejecutando npm run dev, lo que activa Nodemon y permite una monitorización automática de los archivos con reinicio ante cualquier cambio. En la terminal se muestra que el CRM está corriendo en el puerto 3001 y conectado exitosamente a la base de datos PostgreSQL en el puerto 5432. Se realizan consultas como el conteo de clientes para verificar el estado de la conexión y los datos iniciales en la base, lo que garantiza que el servicio está correctamente enlazado y que la información puede ser consultada y gestionada desde el microservicio CRM sin inconvenientes (Ilustración 1)

ILUSTRACIÓN 2: EJECUCIÓN DE IOT DE WSL

Para levantar el servicio IoT, se inicia el servidor con Uvicorn en el puerto 8001, como se puede observar en la terminal. Durante el arranque, el sistema crea el pool de conexiones a PostgreSQL y lo inicializa correctamente, asegurándose de que la base de datos esté disponible para las operaciones del microservicio. El proceso muestra mensajes indicando que tanto el servidor como el pool de conexiones están activos y que la aplicación ha iniciado sin problemas, lo que confirma que el entorno está listo para recibir peticiones y gestionar los datos IoT de manera eficiente (Ilustración 2)

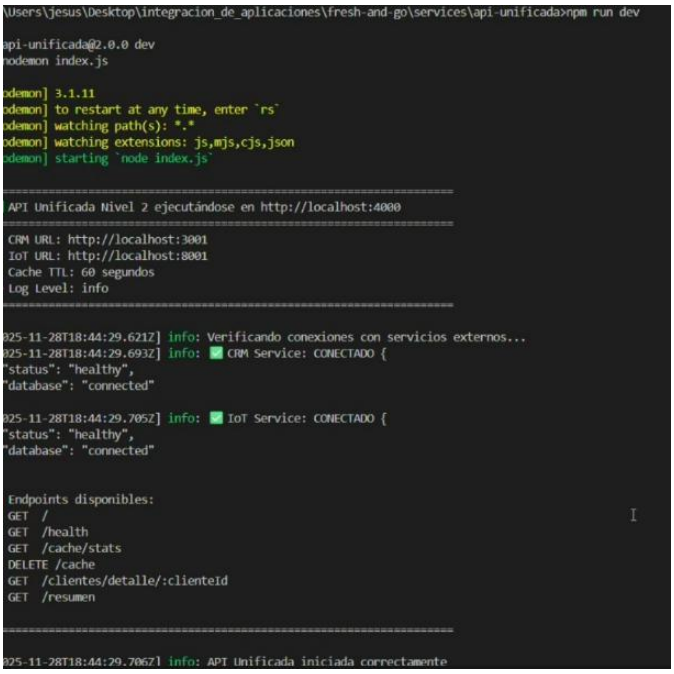


ILUSTRACIÓN 3: EJECUCIÓN DE API-UNIFICADA

Por último, la API unificada se pone en marcha cuando CRM e IoT están levantados y conectados correctamente a PostgreSQL, funcionando como un punto central de integración. En el proceso de inicio, la API verifica la salud y el estado de los microservicios externos y certifica la conexión con sus bases de datos. Al estar ambos servicios conectados y operativos, la API unificada expone sus propios endpoints para consultar datos agregados y facilitar la interacción con la información combinada de CRM e IoT, centralizando el acceso y mejorando la eficiencia de la infraestructura global (Ilustración 3)

Home API Unificada

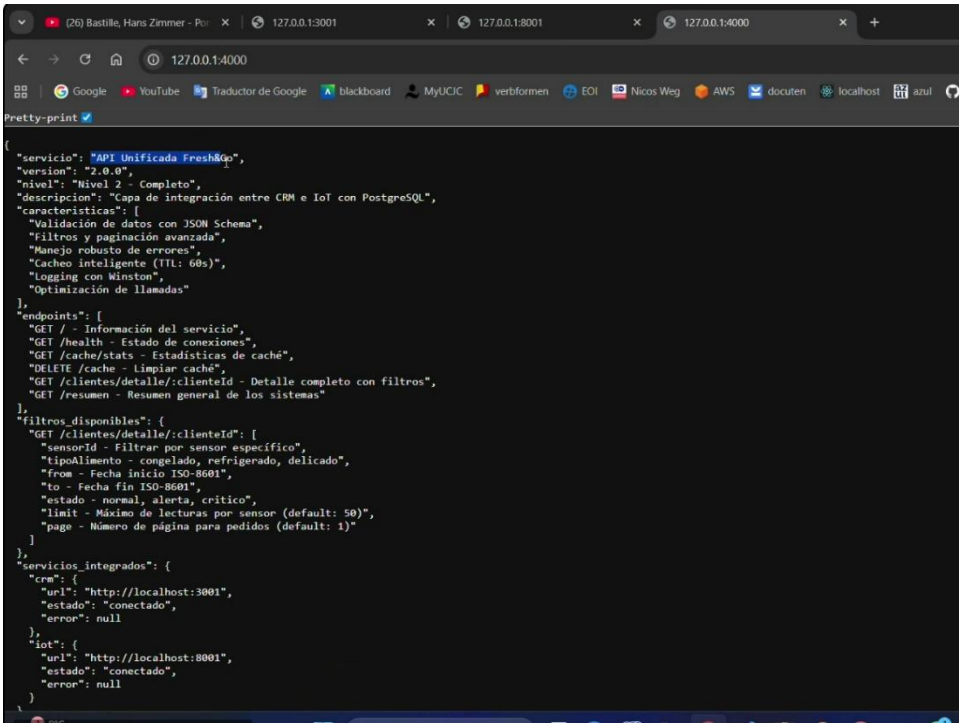


ILUSTRACIÓN 4: ENPOINT INICIAL DE API UNIFICADA

La página principal de / de la API Unificada Fresh&Go ofrece una visión completa de su configuración y capacidades, mostrando la versión, nivel de integración y características clave como validación de datos con JSON Schema, filtros avanzados, paginación automática, manejo robusto de errores y caché inteligente, además de logging centralizado y optimización de llamadas. Se listan los endpoints principales para consultar información general, estado de conexiones, detalles de clientes con distintos filtros y estadísticas, junto al resumen global del sistema, lo que facilita la gestión eficiente de datos integrando tanto el CRM como el IoT. El apartado de servicios integrados confirma la conexión y buena salud de los microservicios en sus respectivas URLs, asegurando que todas las funcionalidades de integración están disponibles y operativas desde este único punto de acceso. (Ilustración 4)

Endpoints CRM

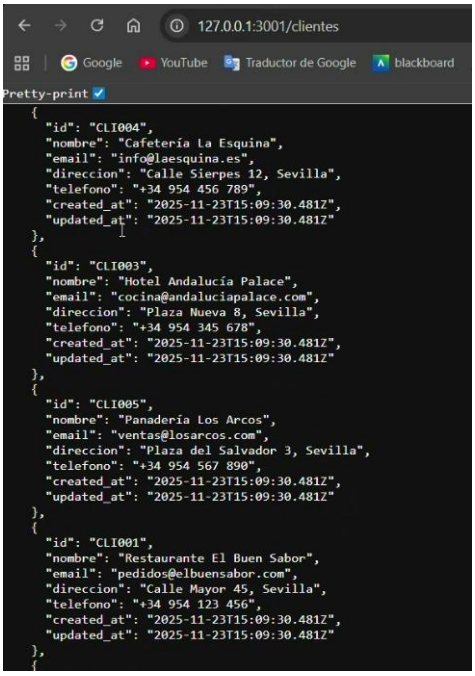


ILUSTRACIÓN 5: ENDPOINT CLIENTES

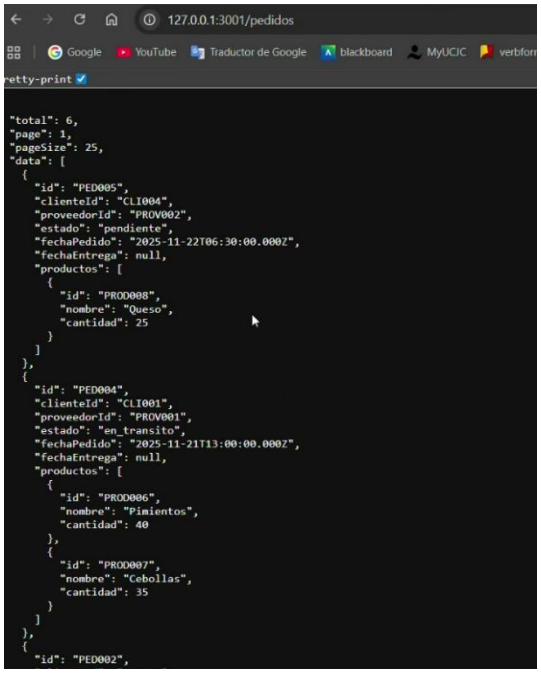


ILUSTRACIÓN 6: ENDPOINT PEDIDOS

El endpoint de clientes en la práctica con API unificada muestra detalles que recuerdan bastante a la versión inicial del CRM, incluyendo el identificador, nombre, email, dirección y teléfono de cada cliente. Ahora, la diferencia clave es que cada registro incorpora también los campos **created_at** y **updated_at**, reforzando el historial y la claridad de cuándo se ha creado y modificado la información de cada cliente. Gracias a estos cambios, la estructura se adapta mejor a los requisitos de auditoría y mantenimiento de datos en sistemas integrados.

Por último, el endpoint de pedidos presenta los pedidos realizados entre clientes y proveedores, mostrando el identificador del pedido, el cliente y proveedor involucrados, estado del pedido, fechas y productos solicitados. A comparación de la versión anterior, mientras se mantienen los principales campos, ahora los productos están organizados por identificador y nombre, y en muchos casos aparece el campo fecha en formato ISO, lo que unifica la gestión temporal de los pedidos. Así, se mejora la integración y el registro cronológico, permitiendo operaciones más precisas y fáciles de auditar entre distintas áreas del sistema

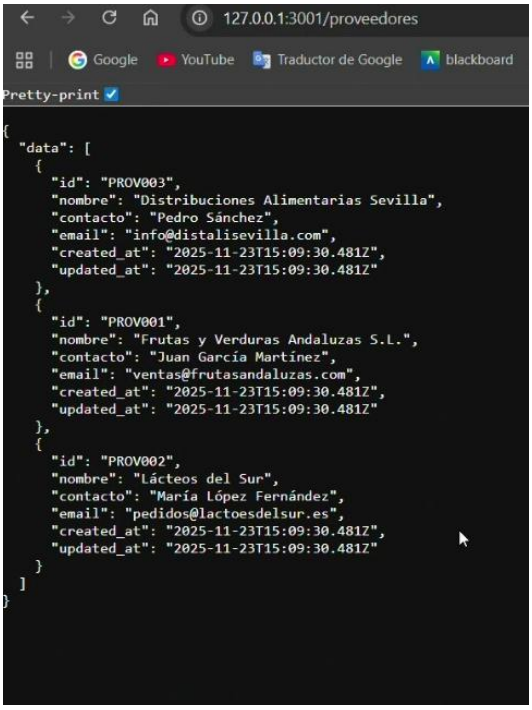


ILUSTRACIÓN 7: ENDPOINT PROVEEDORES

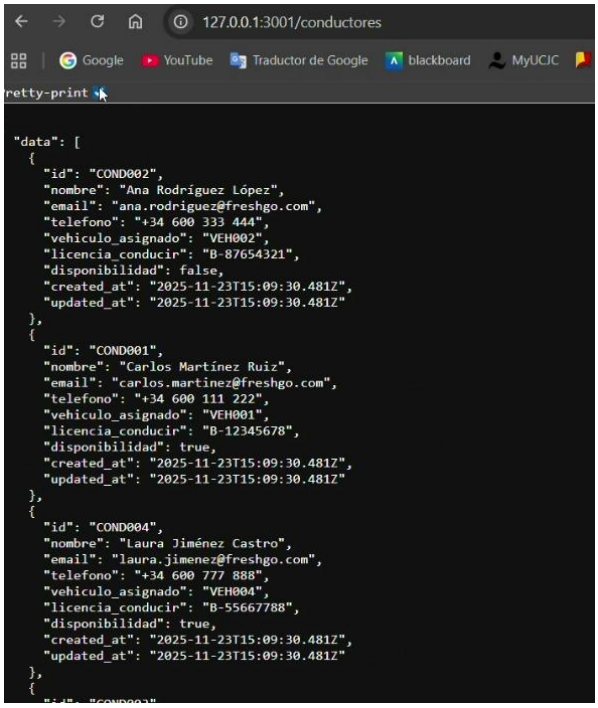


ILUSTRACIÓN 8: ENDPOINT CONDUCTORES

El endpoint de conductores en la API unificada muestra la información de los conductores registrados con datos similares a los que se veían en el CRM, pero ahora incluye campos adicionales como **created_at** y **updated_at**, que sirven para auditar cuándo se creó y se actualizó la información de cada conductor. Siguen estando presentes el identificador, nombre, email, teléfono, vehículo asignado, número de licencia y estado de disponibilidad, pero el añadido de los timestamps mejora el control y trazabilidad sobre posibles cambios o modificaciones en la gestión de conductores.

En el endpoint de proveedores, la API unificada mantiene la base de datos original de proveedores, presentando su identificador, nombre de la empresa, contacto y email. Al igual que en el endpoint de conductores, aparecen los campos **created_at** y **updated_at** para cada proveedor, proporcionando ese valor añadido de trazabilidad temporal sobre la creación o actualización de cada entidad. Esta estructura facilita la sincronización y el seguimiento preciso de toda la información relativa a los proveedores en el sistema, consiguiendo mejorar el control y la gestión de los procesos logísticos.

Endpoints IoT

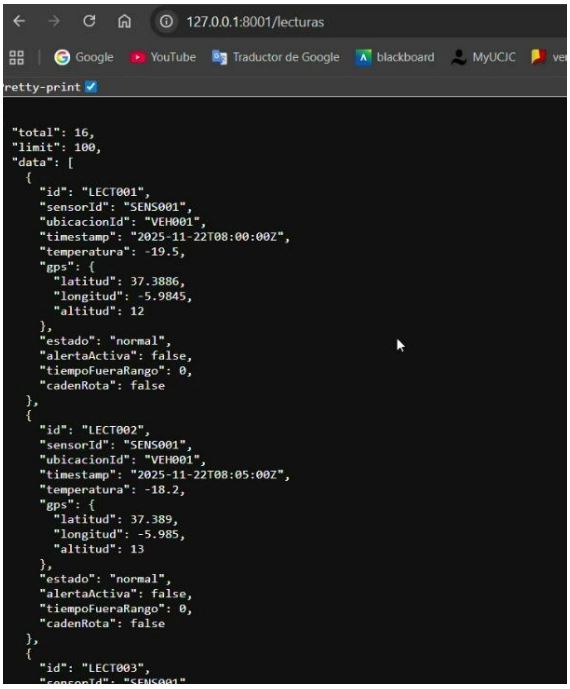


ILUSTRACIÓN 9: ENDPOINT LECTURAS

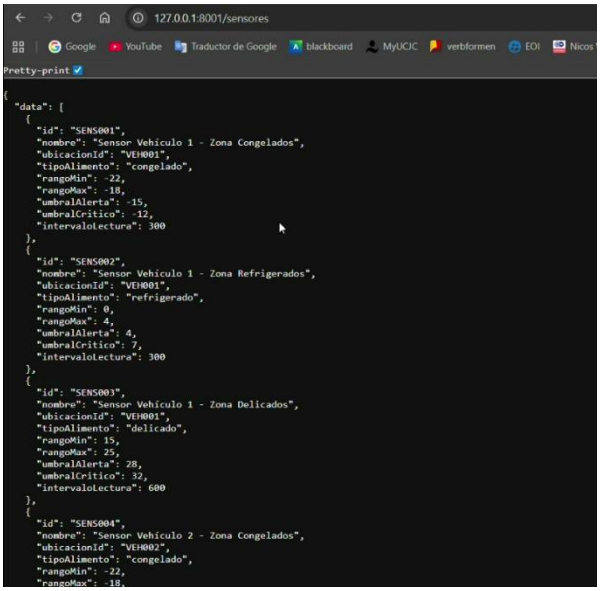


ILUSTRACIÓN 10: ENDPOINT SENSORES

Respecto al endpoint de sensores, sigue permitiendo visualizar la configuración de los diferentes sensores instalados en los vehículos IoT. Se mantiene la información sobre el identificador, nombre descriptivo, ubicación (vehículo), tipo de alimento monitorizado, rangos de operación, umbrales de alerta y críticos y el intervalo de lectura. Sin embargo, ahora tenemos una presentación más estandarizada, lo que facilita la lectura y procesamiento de la información para el control ambiental y de calidad de la carga transportada.

El endpoint de lecturas muestra los datos históricos capturados por los sensores IoT. Cada lectura incluye ahora, de forma clara y ordenada, el identificador del sensor, el vehículo asociado, la fecha y hora exacta con formato de timestamp, los datos ambientales como la temperatura y las coordenadas GPS. Además, se detalla el estado, si hay alerta activa, si la medición está fuera de rango y otros parámetros de control como 'cadenRota'. En comparación con prácticas anteriores, ahora el formato se encuentra mejor estructurado y es más fácil filtrar y analizar la evolución de mediciones en tiempo real o histórico

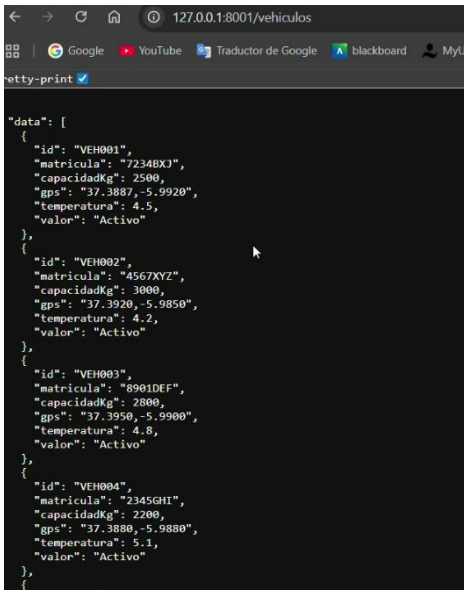


ILUSTRACIÓN 11: ENDPOINT VEHICULOS

En el endpoint de vehículos en IoT, los datos presentados siguen siendo muy similares a los que aparecían antes, mostrando para cada vehículo su identificador, matrícula, capacidad máxima en kilogramos, posición GPS, temperatura y estado ('valor'). Buscando la estructura eficiente para el monitoreo logístico, pero en esta versión de la práctica se ha asegurado que todos los campos sean consistentes y se han depurado posibles diferencias o mejoras en la nomenclatura y valores registrados respecto a versiones anteriores.

Endpoints API Unificada

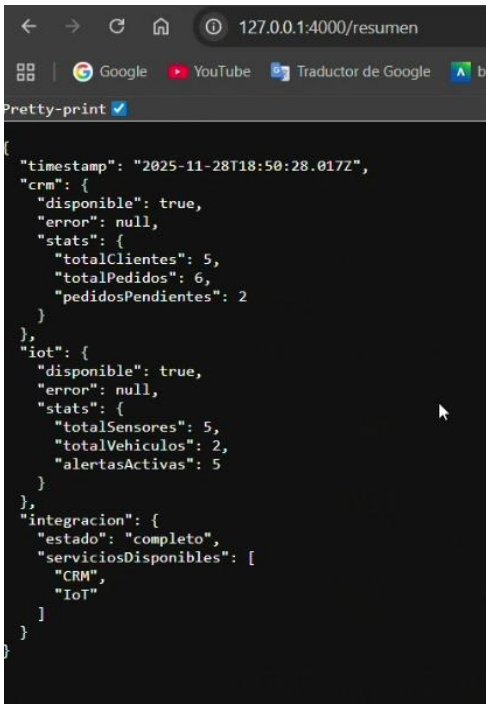


ILUSTRACIÓN 12: ENDPOINT RESUMEN

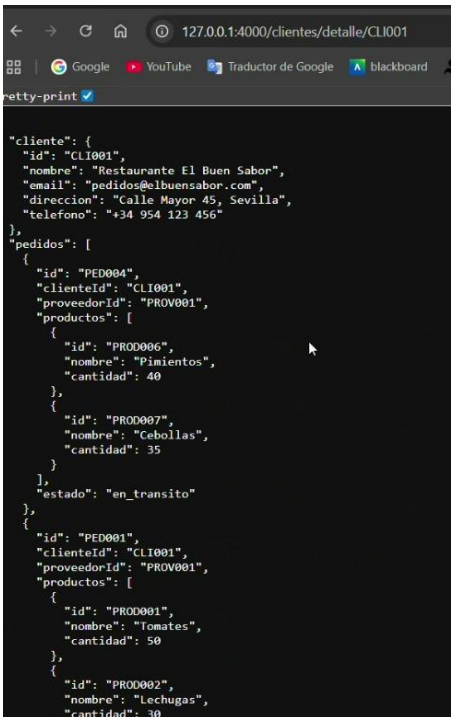
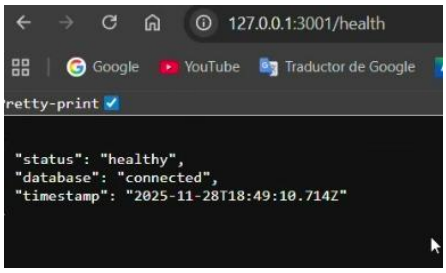


ILUSTRACIÓN 13: ENDPOINT DETALLES DEL CLIENTE

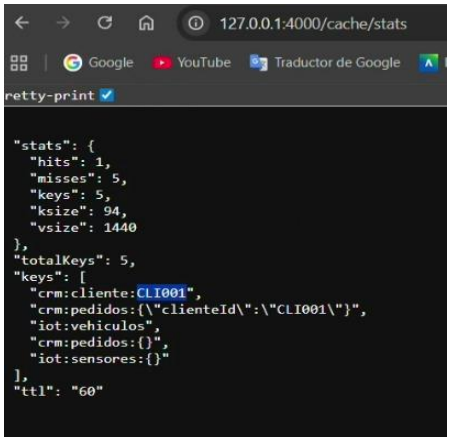
El endpoint /resumen de la API unificada muestra de manera centralizada un estado general de los sistemas CRM y IoT, integrando estadísticas clave como el número total de clientes, pedidos y pedidos pendientes por parte del CRM, y el total de sensores, vehículos y alertas activas del lado de IoT. Todos estos valores aparecen agrupados bajo un mismo registro junto con un timestamp y un estado de integración "completo", lo que permite comprobar rápidamente la disponibilidad y el correcto funcionamiento de ambos sistemas, así como la accesibilidad de los servicios integrados. Esta visión agregada facilita la gestión global y el seguimiento eficiente de recursos y alertas en tiempo real

Por otro lado, el endpoint /clientes/detalle/CLI001 en la API unificada proporciona toda la información relevante de un cliente específico, en este caso "Restaurante El Buen Sabor", integrando sus datos básicos como nombre, email, dirección y teléfono junto con el historial de pedidos asociados. Para cada pedido, se detalla tanto el proveedor y el identificador, como la lista de productos involucrados (por ejemplo, pimientos, cebollas, tomates y lechugas) y el estado actual de la entrega. Este enfoque integrado permite una visión 360º del cliente, asociando directamente su actividad comercial con todos los datos relevantes en el sistema, y aprovechando la potencia de la API para unificar y consultar la información distribuida entre CRM e IoT

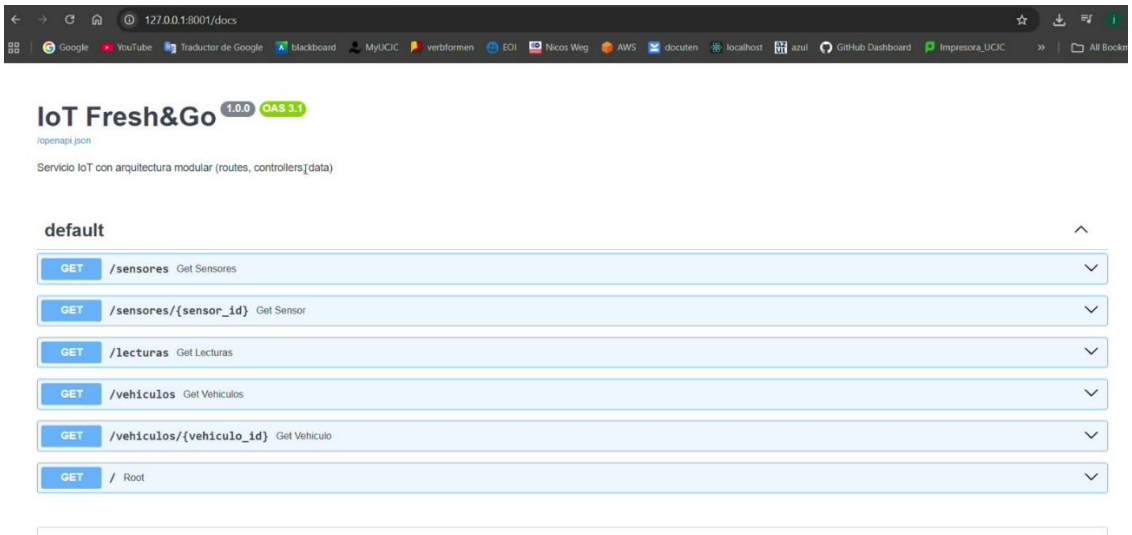
Otros endpoints útiles



Para asegurarnos que esta correctamente conectado el CRM a la BBDD



Una especie de historial donde podemos observar las ultimas busquedas, también tenemos la opción de borrar el historial



Un swagger para mostrar los principales endpoints de forma más bonita para el usuario

FRESH&GO

DE LA TIENDA A TU MESA EN UN CLICK