

Fresh&Go: Evolución de la Arquitectura en la Práctica 3

Esta presentación detalla los avances y la reestructuración arquitectónica implementada en la Práctica 3 del proyecto Fresh&Go, marcando un paso crucial hacia un sistema más robusto y escalable.





De la Simulación a la Persistencia de Datos

Práctica 2: Datos Simulados

Los datos de los microservicios CRM e IoT se gestionaban mediante archivos JSON estáticos, operando de forma independiente y sin persistencia real.

Práctica 3: Base de Datos PostgreSQL

Introducción de PostgreSQL para una gestión de datos persistente y relacional, mejorando la integridad y disponibilidad de la información.

API Unificada

Emergencia de una capa de integración centralizada que coordina el acceso a la información de los distintos microservicios.

Estructura de la Base de Datos PostgreSQL

La Práctica 3 introduce una base de datos PostgreSQL, la cual almacena de forma persistente la información crítica del sistema Fresh&Go. Esta base de datos es gestionada por scripts SQL específicos.

Scripts SQL Esenciales

- `crear_tablas.sql`: Define la estructura de las tablas y sus relaciones.
- `datos_semilla.sql`: Inserta datos iniciales para pruebas y desarrollo.

Tablas Principales

- Sensores y Lecturas (IoT)
- Clientes (CRM)

La lógica interna de los microservicios CRM e IoT se mantiene, pero ahora se conectan a esta base de datos relacional.

Script SQL: Tablas

LECTURAS

```
-- Tabla de Lecturas
CREATE TABLE lecturas (
    id VARCHAR(50) PRIMARY KEY,
    sensor_id VARCHAR(50) NOT NULL REFERENCES sensores(id) ON DELETE CASCADE,
    ubicacion_id VARCHAR(50) NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    temperatura NUMERIC(5, 2) NOT NULL,
    latitud NUMERIC(10, 7) NOT NULL,
    longitud NUMERIC(10, 7) NOT NULL,
    altitud NUMERIC(7, 2),
    estado VARCHAR(50) NOT NULL CHECK (estado IN ('normal', 'alerta', 'critico')),
    alerta_activa BOOLEAN DEFAULT false,
    tiempo_fuera_rango INTEGER DEFAULT 0,
    cadena_rota BOOLEAN DEFAULT false,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

SENSORES

```
-- Tabla de Sensores
CREATE TABLE sensores (
    id VARCHAR(50) PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    ubicacion_id VARCHAR(50) NOT NULL REFERENCES vehiculos(id) ON DELETE CASCADE,
    tipo_alimento VARCHAR(50) NOT NULL CHECK (tipo_alimento IN ('congelado', 'refrigerado', 'delicado')),
    rango_min NUMERIC(5, 2) NOT NULL,
    rango_max NUMERIC(5, 2) NOT NULL,
    umbral_alerta NUMERIC(5, 2) NOT NULL,
    umbral_critico NUMERIC(5, 2) NOT NULL,
    intervalo_lectura INTEGER DEFAULT 300,
    activo BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Script SQL: Tablas

CLIENTES

```
-- Tabla de Clientes
CREATE TABLE clientes (
    id VARCHAR(50) PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    direccion VARCHAR(500),
    telefono VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Script SQL: Datos semilla

SENSORES

```
-- Sensores (ACTUALIZADO: congelado, refrigerado, delicado)
INSERT INTO sensores (id, nombre, ubicacion_id, tipo_alimento, rango_min, rango_max, umbral_alerta, umbral_critico, intervalo_lectura) VALUES
('SENS001', 'Sensor Vehículo 1 - Zona Congelados', 'VEH001', 'congelado', -22, -18, -15, -12, 300),
('SENS002', 'Sensor Vehículo 1 - Zona Refrigerados', 'VEH001', 'refrigerado', 0, 4, 4, 7, 300), >
('SENS003', 'Sensor Vehículo 1 - Zona Delicados', 'VEH001', 'delicado', 0, 2, 2, 3, 180),
('SENS004', 'Sensor Vehículo 2 - Zona Congelados', 'VEH002', 'congelado', -22, -18, -15, -12, 300),
('SENS005', 'Sensor Vehículo 2 - Zona Refrigerados', 'VEH002', 'refrigerado', 0, 4, 4, 7, 300);
```

LECTURAS

```
-- Lecturas
INSERT INTO lecturas (id, sensor_id, ubicacion_id, timestamp, temperatura, latitud, longitud, altitud, estado, alerta_activa, tiempo_fuera_rango, cadena_rota) VALUES
-- Sensor 1 (Congelados VEH001)
('LECT001', 'SENS001', 'VEH001', '2025-11-22 08:00:00', -19.5, 37.3886, -5.9845, 12, 'normal', false, 0, false),
('LECT002', 'SENS001', 'VEH001', '2025-11-22 08:05:00', -18.2, 37.3890, -5.9850, 13, 'normal', false, 0, false),
('LECT003', 'SENS001', 'VEH001', '2025-11-22 08:10:00', -14.8, 37.3895, -5.9855, 14, 'alerta', true, 5, false),
('LECT004', 'SENS001', 'VEH001', '2025-11-22 08:15:00', -11.5, 37.3900, -5.9860, 15, 'critico', true, 10, false),
('LECT005', 'SENS001', 'VEH001', '2025-11-22 08:20:00', -10.2, 37.3905, -5.9865, 16, 'critico', true, 20, true),
```

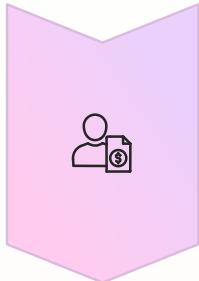
Script SQL: Datos semillas

CLIENTES

```
-- Clientes
INSERT INTO clientes (id, nombre, email, direccion, telefono) VALUES
('CLI001', 'Restaurante El Buen Sabor', 'pedidos@elbuensabor.com', 'Calle Mayor 45, Sevilla', '+34 954 123 456'),
('CLI002', 'Supermercado Fresh Market', 'compras@freshmarket.es', 'Avenida de la Constitución 23, Sevilla', '+34 954 234 567'),
('CLI003', 'Hotel Andalucía Palace', 'cocina@andaluciapalace.com', 'Plaza Nueva 8, Sevilla', '+34 954 345 678'),
('CLI004', 'Cafetería La Esquina', 'info@laesquina.es', 'Calle Sierpes 12, Sevilla', '+34 954 456 789'),
('CLI005', 'Panadería Los Arcos', 'ventas@losarcos.com', 'Plaza del Salvador 3, Sevilla', '+34 954 567 890');
```

Arquitectura de Servicios en la Práctica 3

En esta iteración, los servicios se reorganizan para interactuar con la base de datos persistente y a través de una API unificada, mejorando la cohesión del sistema.



Microservicio CRM

Conectado directamente a PostgreSQL para gestionar datos de clientes.



Microservicio IoT

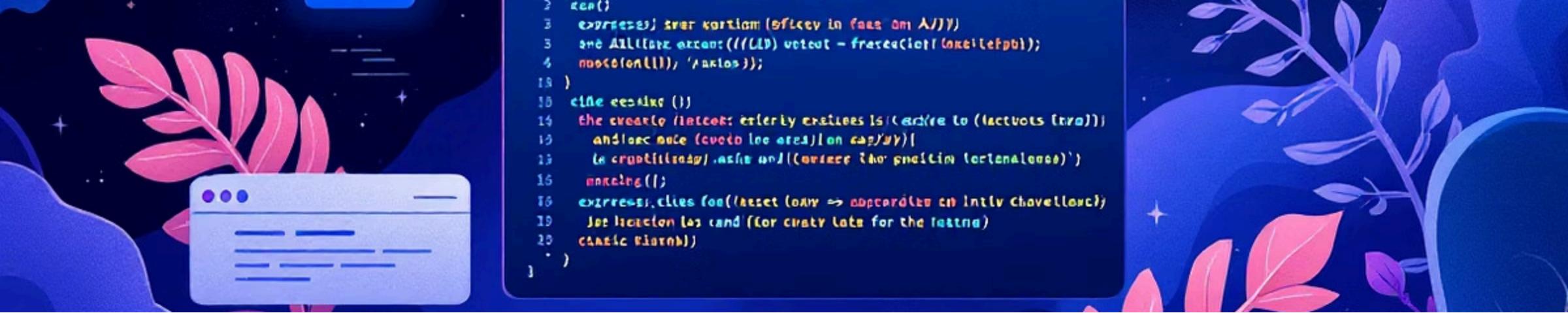
Conectado a PostgreSQL para almacenar y recuperar datos de dispositivos y sensores.



API Unificada

Actúa como orquestador, consumiendo información de CRM e IoT mediante HTTP.





La API Unificada: Capa de Integración

La API Unificada es el componente clave para la consolidación de datos, permitiendo consultas combinadas sin exponer directamente los microservicios individuales o la base de datos subyacente.

Consumo de Servicios

Utiliza `axios` para realizar solicitudes HTTP a los microservicios CRM e IoT.

Validación de Datos

Emplea `AJV` (Another JSON Schema Validator) y middlewares personalizados para asegurar la integridad y el formato correcto de los datos en las peticiones y respuestas.

Tecnologías Clave

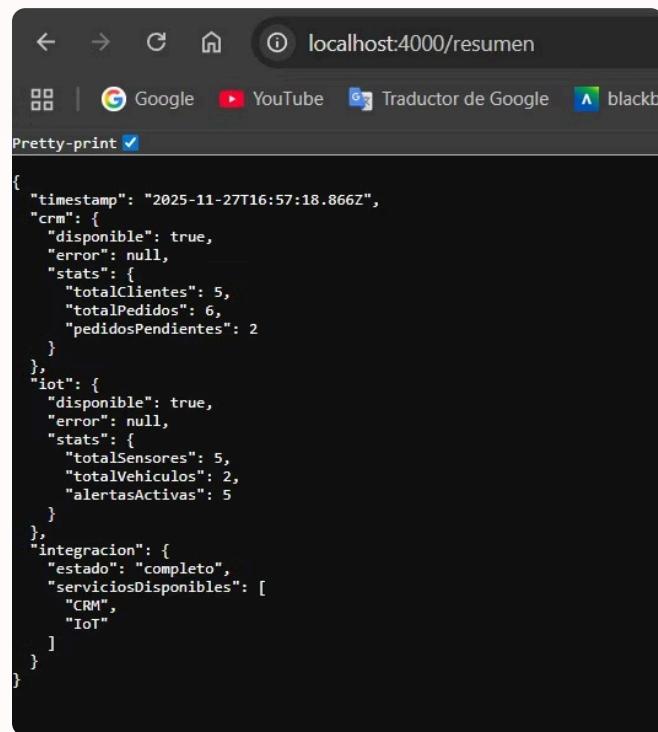
Desarrollada con `Express.js`, aprovechando su robustez para la creación de APIs RESTful.

Endpoints Clave de la API Unificada

Estos endpoints ilustran la capacidad de la API Unificada para ofrecer vistas consolidadas de la información, respondiendo a necesidades de negocio específicas.

/resumen

Ofrece una visión general del estado del sistema, consolidando métricas y datos relevantes de los diferentes microservicios.



A screenshot of a web browser window displaying a JSON response. The URL in the address bar is `localhost:4000/resumen`. The browser interface includes standard navigation buttons (back, forward, search) and a tab bar with links to Google, YouTube, and Traductor de Google. Below the address bar, there's a "Pretty-print" checkbox which is checked. The main content area shows a nested JSON object:

```
{  
  "timestamp": "2025-11-27T16:57:18.866Z",  
  "crm": {  
    "disponible": true,  
    "error": null,  
    "stats": {  
      "totalClientes": 5,  
      "totalPedidos": 6,  
      "pedidosPendientes": 2  
    }  
  },  
  "iot": {  
    "disponible": true,  
    "error": null,  
    "stats": {  
      "totalSensores": 5,  
      "totalVehiculos": 2,  
      "alertasActivas": 5  
    }  
  },  
  "integracion": {  
    "estado": "completo",  
    "serviciosDisponibles": [  
      "CRM",  
      "IoT"  
    ]  
  }  
}
```

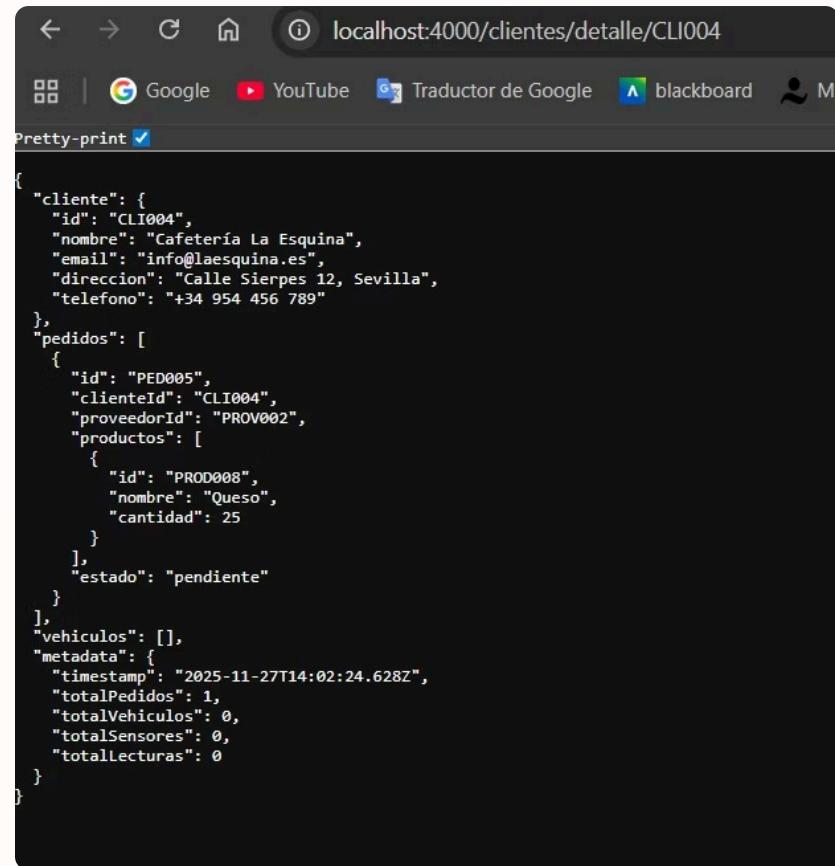
La API centraliza la lógica de negocio para consultas complejas, facilitando el desarrollo de futuras interfaces de usuario o aplicaciones cliente.

Endpoints Clave de la API Unificada

Estos endpoints ilustran la capacidad de la API Unificada para ofrecer vistas consolidadas de la información, respondiendo a necesidades de negocio específicas.

/clientes/detalles/:id

Permite obtener una vista completa de un cliente específico, combinando sus datos del CRM con la información de sus dispositivos IoT asociados



A screenshot of a web browser window displaying a JSON response. The URL in the address bar is `localhost:4000/clientes/detalle/CLI004`. The browser tabs include Google, YouTube, Traductor de Google, blackboard, and My. The JSON response is pretty-printed and shows the following data:

```
{  
  "cliente": {  
    "id": "CLI004",  
    "nombre": "Cafetería La Esquina",  
    "email": "info@laesquina.es",  
    "direccion": "Calle Sierpes 12, Sevilla",  
    "telefono": "+34 954 456 789"  
  },  
  "pedidos": [  
    {  
      "id": "PED005",  
      "clienteId": "CLI004",  
      "proveedorId": "PROV002",  
      "productos": [  
        {  
          "id": "PROD008",  
          "nombre": "Queso",  
          "cantidad": 25  
        }  
      ],  
      "estado": "pendiente"  
    }  
  ],  
  "vehiculos": [],  
  "metadata": {  
    "timestamp": "2025-11-27T14:02:24.628Z",  
    "totalPedidos": 1,  
    "totalVehiculos": 0,  
    "totalSensores": 0,  
    "totalLecturas": 0  
  }  
}
```

La API centraliza la lógica de negocio para consultas complejas, facilitando el desarrollo de futuras interfaces de usuario o aplicaciones cliente.

Problemas Técnicos y Soluciones (Práctica 3)

Durante la implementación de la Práctica 3, surgieron diversos desafíos técnicos que requirieron ajustes y soluciones específicas para asegurar la correcta integración y funcionamiento del sistema.

01

Conexión entre Servicios

Ajustes finos en las configuraciones de red y los parámetros HTTP para que los microservicios se comunicaran eficientemente.

02

Errores de Rutas y Validación

Corrección de errores iniciales en las definiciones de rutas y en la lógica de validación de datos, impactando directamente en la estabilidad de la API Unificada.

03

Adaptación del JSON Schema

Revisión y adaptación del JSON Schema unificado para que reflejara con precisión la nueva estructura de datos proveniente de PostgreSQL.

04

Migración de Datos

Incidencias relacionadas con la transición de datos de la estructura JSON a la base de datos SQL, incluyendo la compatibilidad de tipos y la integridad referencial.

Gestión del Repositorio en GitHub

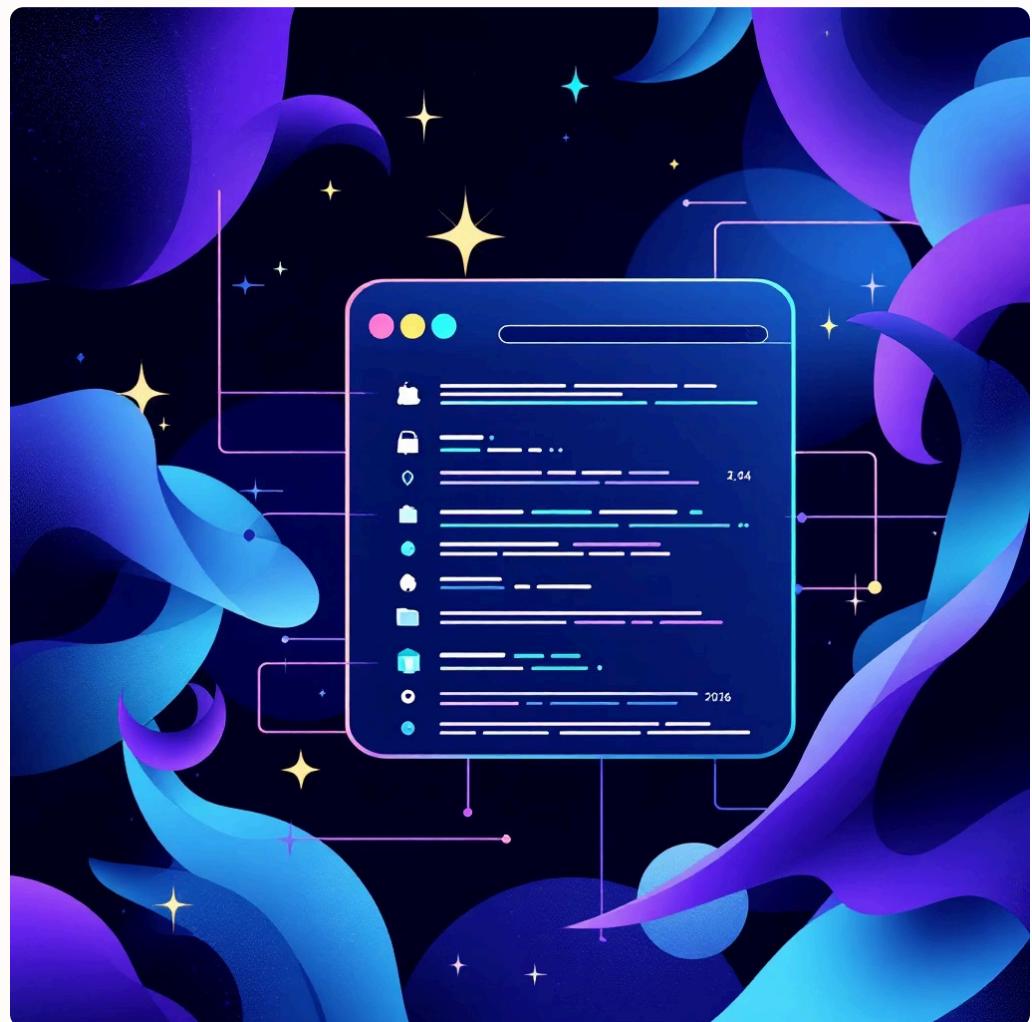
La organización y versionado en GitHub fue fundamental para gestionar el desarrollo colaborativo y el seguimiento de los cambios introducidos en la Práctica 3.

Estructura de Carpetas

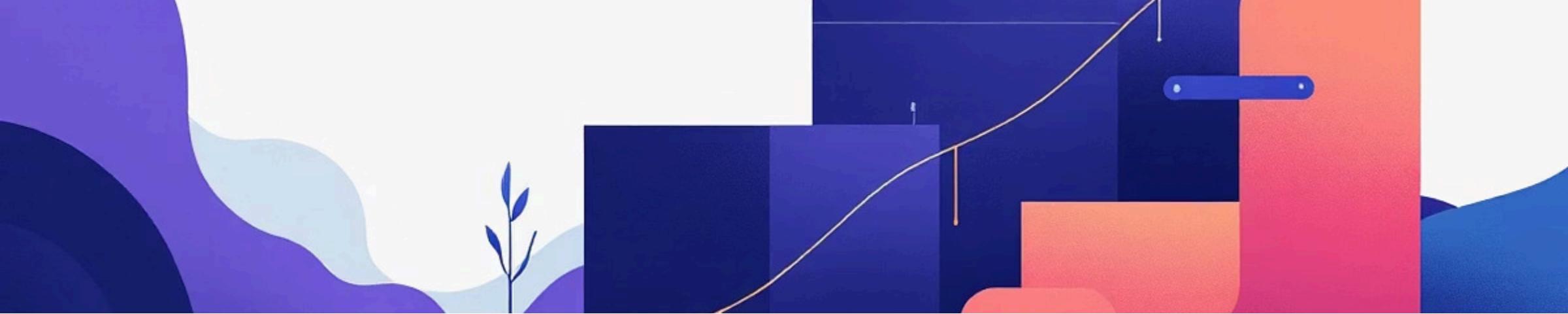
- Reorganización para alojar los nuevos scripts SQL y la API Unificada

Control de Versiones

- Gestión de branches para el desarrollo de nuevas funcionalidades.
- Realización de commits descriptivos para trazar el progreso y los ajustes con el objetivo de no repetir los errores de la Práctica 2



Estos esfuerzos aseguran un entorno de desarrollo ordenado y eficiente.



Conclusiones de la Práctica 3

La Práctica 3 representa una fase crítica en la maduración del proyecto Fresh&Go, sentando las bases para una arquitectura más robusta y escalable.

Resumen General

Hemos pasado de la simulación a la persistencia de datos con PostgreSQL y hemos establecido una API Unificada esencial para la integración.

Evolución Continua

Cada práctica construye sobre la anterior, añadiendo capas de complejidad y funcionalidad que acercan el sistema a su visión final.

Importancia de la API Unificada

La API Unificada es ahora el centro de orquestación, clave para futuras expansiones y la interacción con otras aplicaciones.

Hacia el Futuro: Fresh&Go

La relación progresiva entre las prácticas, desde la simulación hasta la persistencia y la integración, es fundamental para el desarrollo sostenible de Fresh&Go.

Simulación (P2)

Entender la lógica inicial con datos controlados.

Expansión (Futuro)

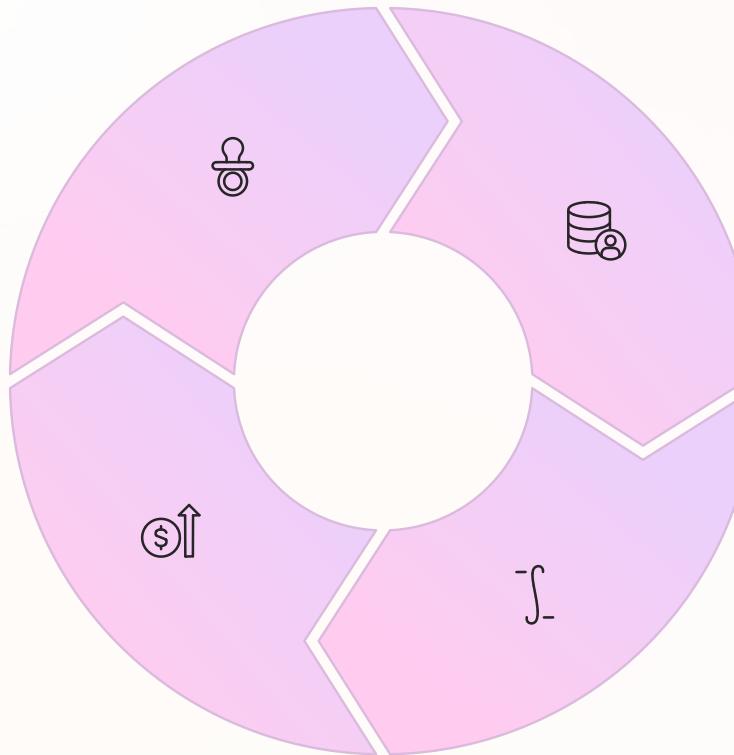
Facilitar la incorporación de una interfaz.

Persistencia (P3)

Implementar almacenamiento real y gestión de datos.

Integración (P3)

Conectar servicios mediante la API Unificada.



Estos avances permiten seguir construyendo el sistema Fresh&Go en prácticas posteriores.