

**Eco Market SPA.**

**Nombre de los Estudiantes: Araceli Salgado**

**Tomas Jilberto**

**David Albornoz**

**Nombre del Docente: Eduardo Baeza**

**Nombre Asignatura: Full Stack 1**

**Sección: 011D**

# **Índice**

**Diagrama de Arquitectura de Microservicios** [2](#_gjdgxs)

**Estructura del Proyecto** [3](#_30j0zll)

**Base de Datos** [4](#_3znysh7)

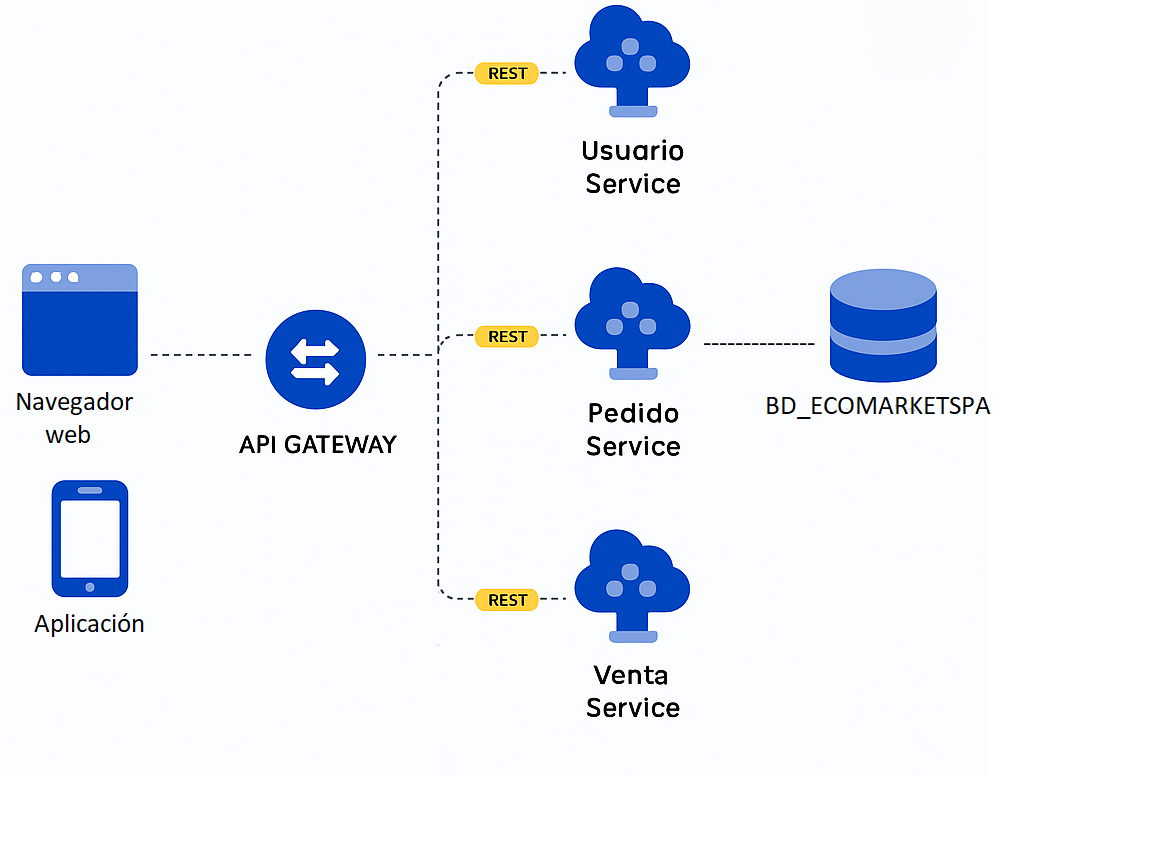
**Implementación de los servicios** [5](#_2et92p0)

**Implementación de Vistas** [6](#_tyjcwt)

**Git-GitHub** [7](#_3dy6vkm)

# **Introducción**

*Diagrama De Arquitectura De Microservicios:*

**

El diagrama muestra la arquitectura basada en microservicios del sistema de EcoMarket SPA, donde los usuarios acceden a través de un navegador web o aplicación móvil, y sus solicitudes son gestionadas por un API Gateway, que enruta las peticiones a los servicios correspondientes mediante APIs REST. Los tres microservicios principales —**Usuario Service**, **Pedido Service** y **Venta Service**— manejan las operaciones CRUD de forma independiente y especializada. El Pedido Service se conecta directamente a la base de datos MySQL llamada **eco\_market**, encargándose de almacenar y gestionar la información relacionada con los pedidos. Esta estructura mejora la escalabilidad, mantenibilidad y permite una gestión más eficiente del sistema.

*Estructura del Proyecto:*

**Dependencias Ocupadas :**

* Spring web : Permite Construir Servicios web RESTFUL
* Spring data JPA : Facilita el uso de JPA para acceder y manipular datos en la base de datos.
* Spring Boot DevTools : Para recarga automática durante el desarrollo.
* MySQL Driver : Controlador necesario para conectarse a MySQL.

**Componentes Implementados :**

**Utilizamos el siguiente patrón de diseño:**

* Entities: Implementamos 3 Clases llamadas Producto, Venta y Cliente.
* Repository: Implementamos interfaces que manejan operaciones con la base de datos.
* Services: Lógica de negocio (interfaces e implementaciones).
* Controllers: Controladores REST para exponer las peticiones de los clientes

**Operaciones CRUD:**

1. **Producto**

* Descripción: Define los productos que EcoMarket SPA ofrece.
* Atributos: id, nombre, descripción y precio.

**CRUD implementado en**

* ProductoRestController : rutas para crear, modificar, ver detalle y eliminar productos.
* ProductoService + ProductoServiceImpl : lógica de negocio asociada.
* ProductoRepository : interfaz JPA

1. **Venta**

* Descripción: Registra las transacciones de venta realizadas por los clientes.
* Atributos: id, fecha y total.

**CRUD implementado en**

* VentaRestController : operaciones relacionadas con ventas.
* VentaService + VentaServiceImpl : procesamiento de lógica de ventas.
* VentaRepository : interfaz para acceso a datos.

1. **Cliente**

* Descripción: Representa a los clientes del sistema.
* Atributos: id, nombre, gmail y edad.

**CRUD implementado en**

* ClienteRestController : expone rutas REST para gestionar clientes.
* ClienteService + ClienteServiceImpl : contiene la lógica de negocio.
* ClienteRepository : interfaz JPA para acceso a datos.

**Base De Datos:**

* Se utiliza una base de datos MySQL llamada eco\_market
* Las Entidades Producto, Venta y Cliente están mapeadas con JPA y se reflejan como tablas.
* Todas las operaciones CRUD se realizan a través de servicios REST que se comunican con esta base de datos.

**Configuración adicional:**

* Configuración desde Visual Studio Code usando Maven.
* El archivo **application.properties** incluye las credenciales de conexión a MySQL.
* Uso de extensiones “Java Extension Pack” y “Spring Boot Extension Pack”

*Base de Datos:*

**Base De Datos:**

* Motor Utilizado: MySQL
* Herramientas : MySQL Workbench para modelar y Xampp para levantar el servidor.

1. Tabla Producto

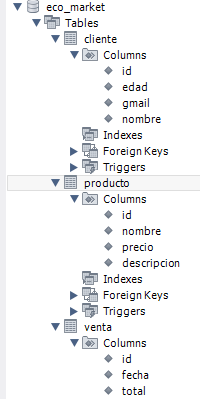
* **Campos Principales:** id (PK), nombre, descripción, precio.

1. Tabla Venta

* **Campos Principales:** id (PK),fecha y total.

1. Tabla Cliente

* **Campos Principales:** id (PK), nombre, gmail y edad.



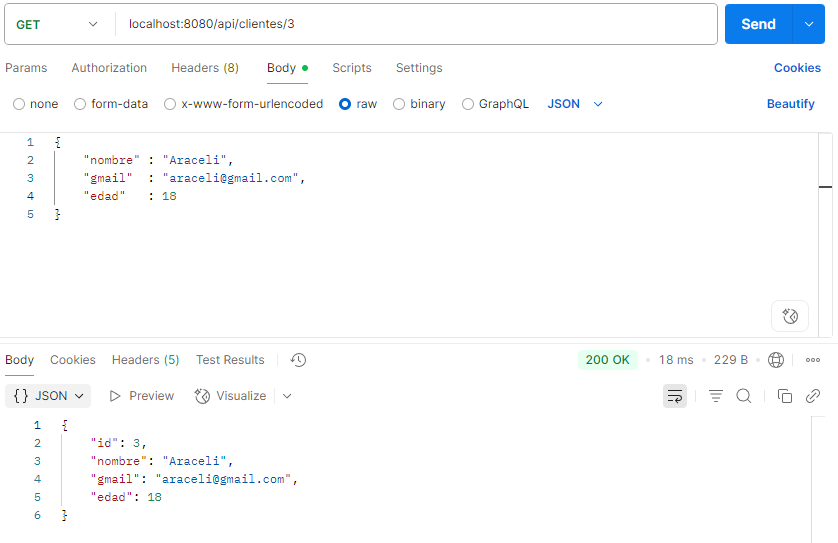
**Implementacion de servicios FULL REST:**

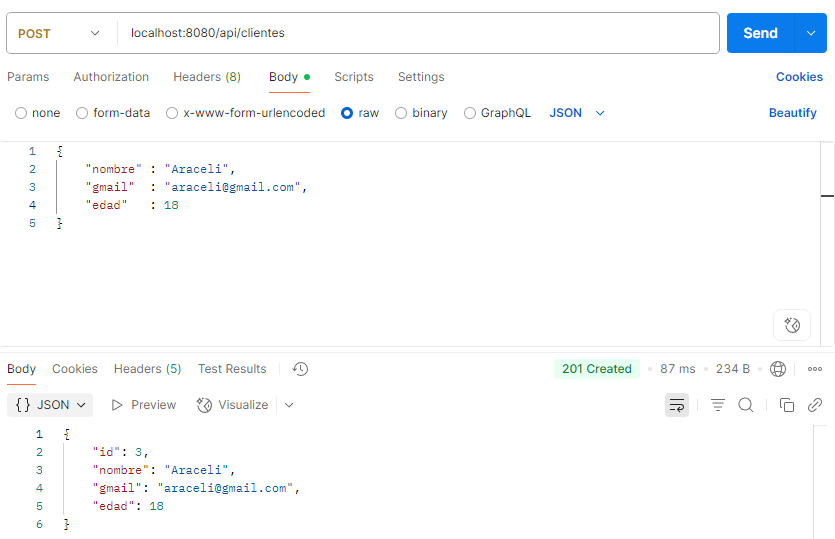
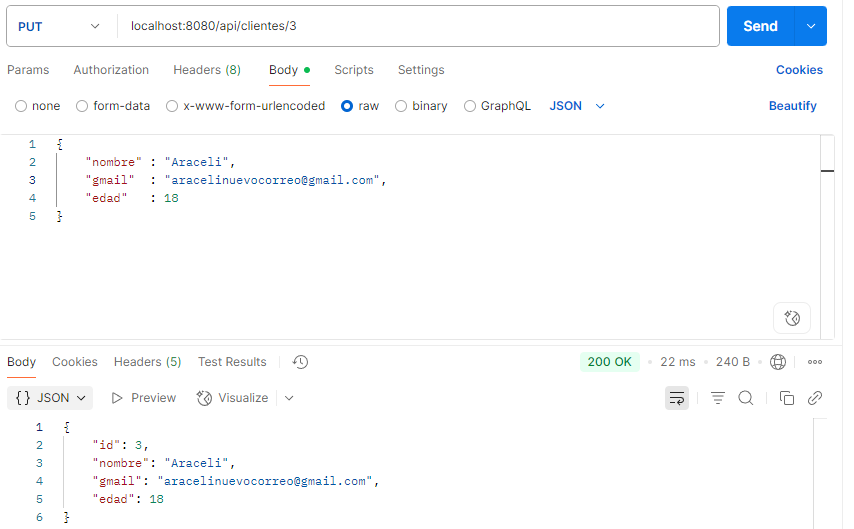
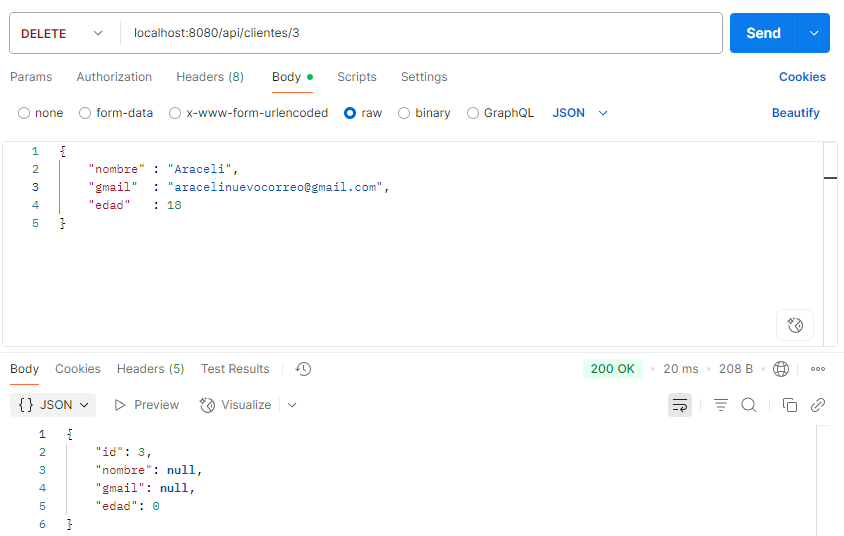
En el desarrollo del proyecto, se definieron e implementaron tres servicios REST principales utilizando el framework Spring Boot: **producto**, **venta** y **cliente**. A continuación, se describe brevemente cada uno de ellos y se presenta evidencia de su funcionamiento mediante capturas de pantalla tomadas desde Postman.

#### 1. Servicio de Cliente

Este servicio permite gestionar la información de los clientes. Se implementaron operaciones CRUD completas:

* **GET**: Obtener lista de clientes o un cliente por su ID.



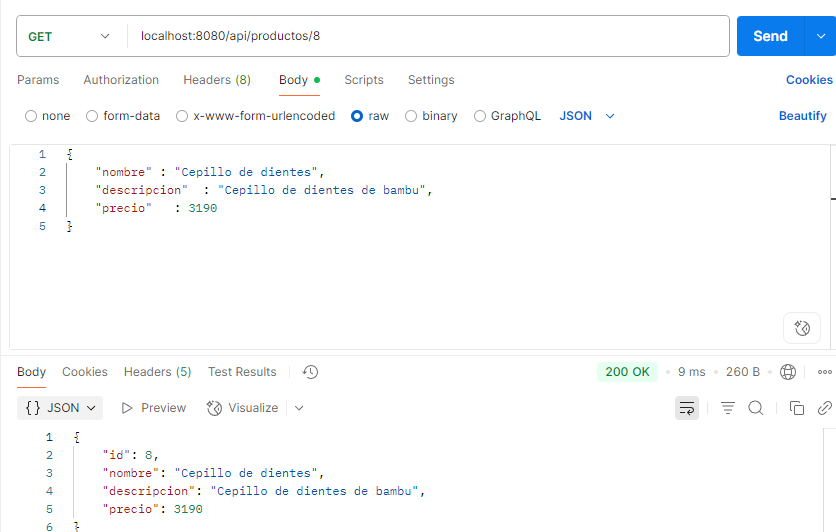
* **POST**: Crear un nuevo cliente.
* 
* **PUT**: Actualizar los datos de un cliente existente.  
  
* **DELETE**: Eliminar un cliente del sistema.  
  

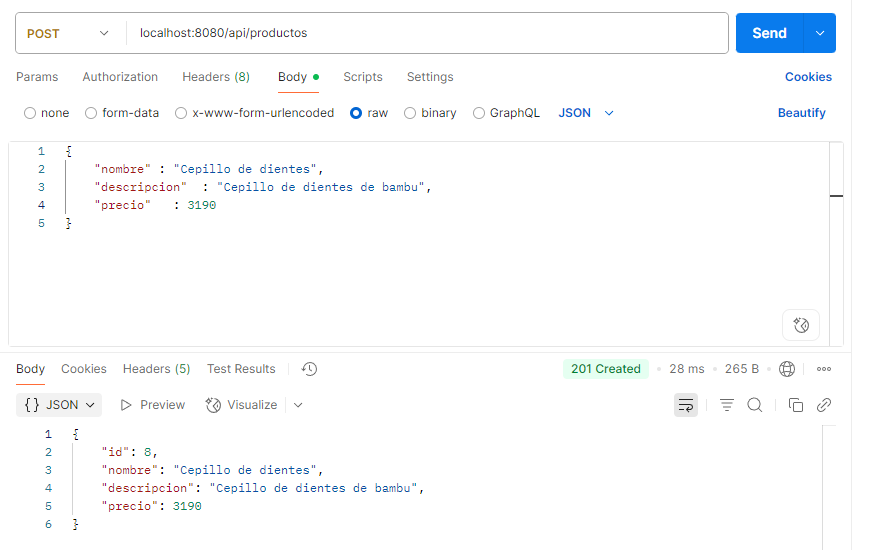
#### 

#### 2. Servicio de Producto

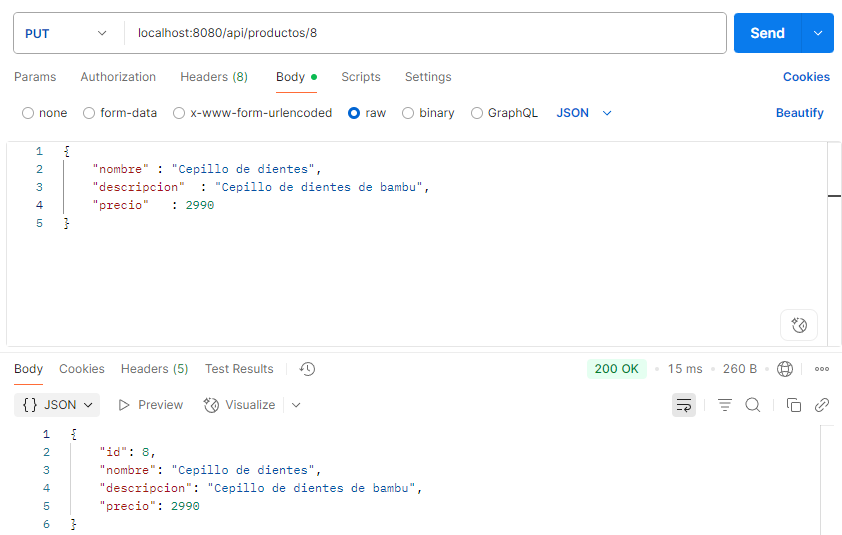
Este servicio administra los productos disponibles. Incluye:

**GET**: Listar todos los productos o consultar uno específico.

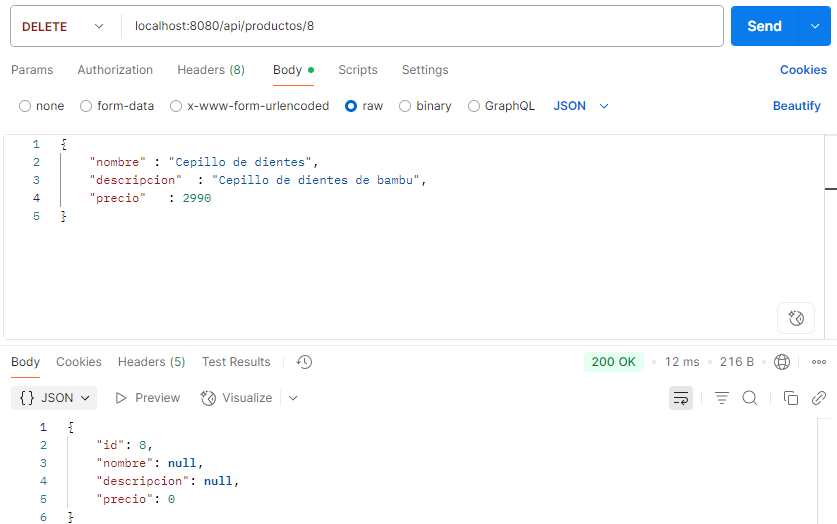


**POST**: Registrar un nuevo producto.

**PUT**: Editar información de un producto.



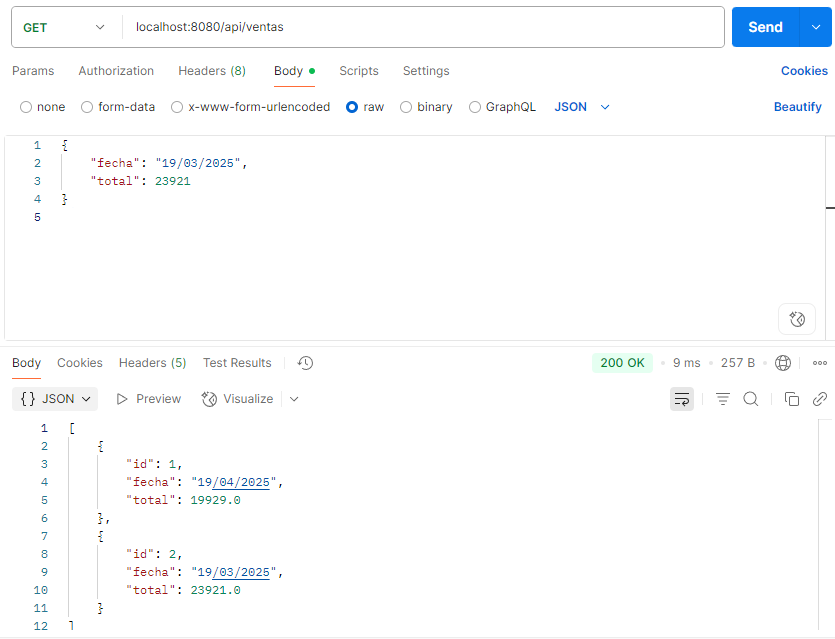
**DELETE**: Eliminar un producto del catálogo.



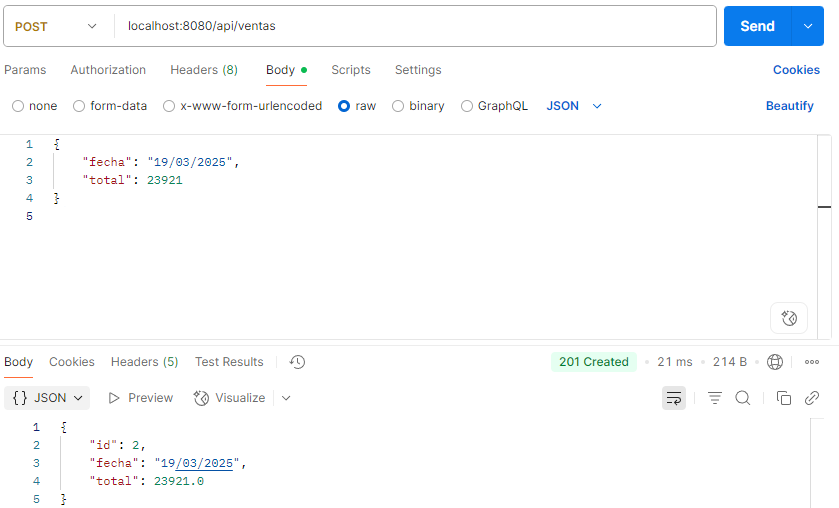
#### 3. Servicio de Venta

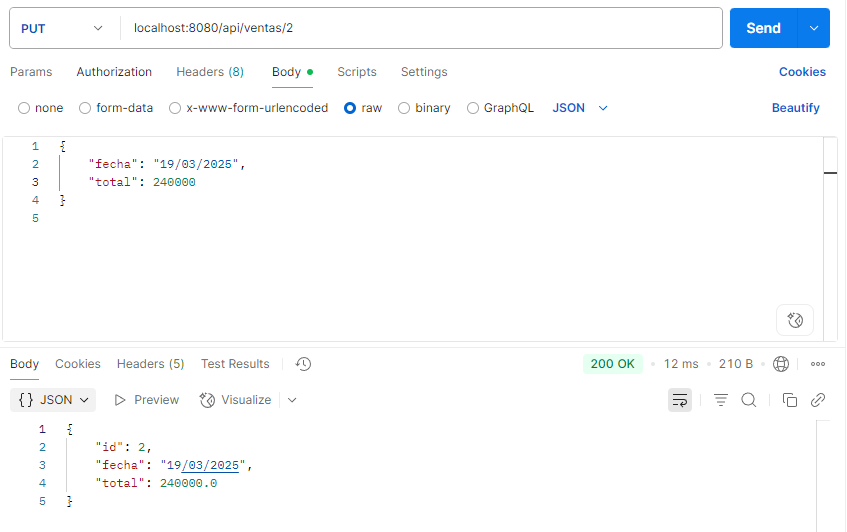
El servicio de venta registra y gestiona las transacciones. Las operaciones implementadas son:

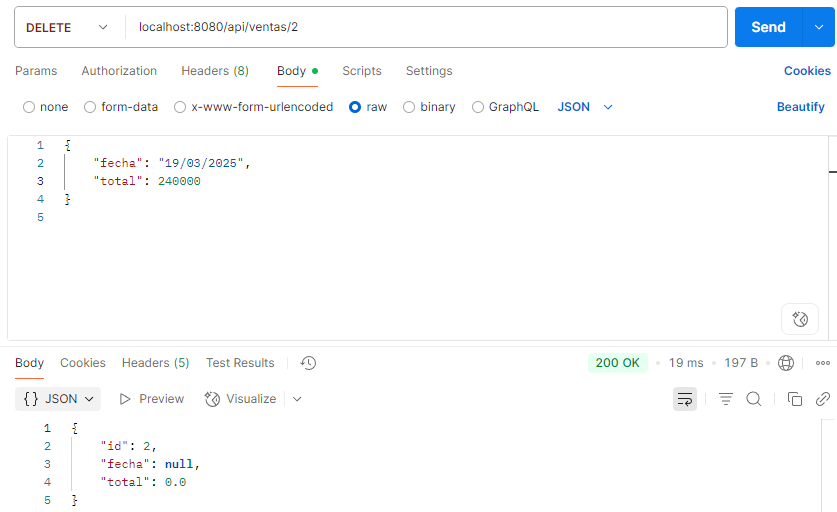
**GET**: Consultar todas las ventas o una en específico.

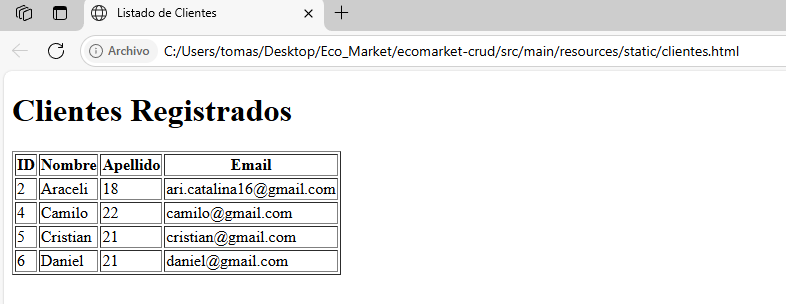


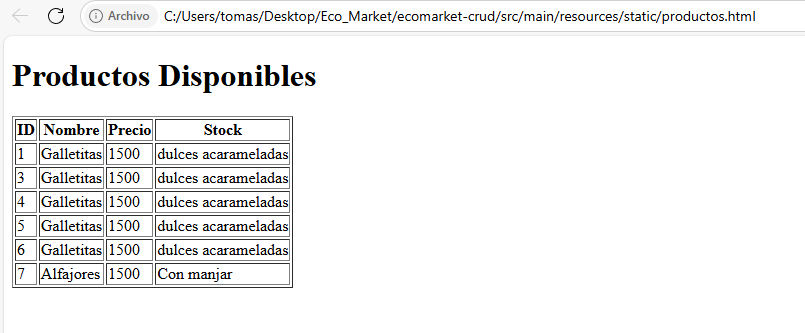
**POST**: Registrar una nueva venta asociada a un cliente y productos.

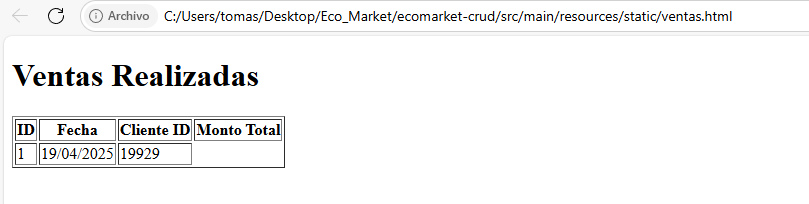


**PUT**: Modificar detalles de una venta.**DELETE**: Eliminar una venta registrada.

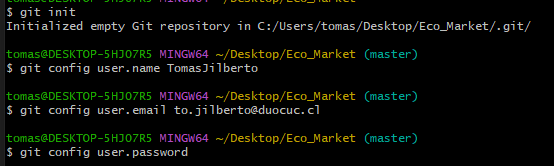


**Datos de cada servicio desde un archivo html:**



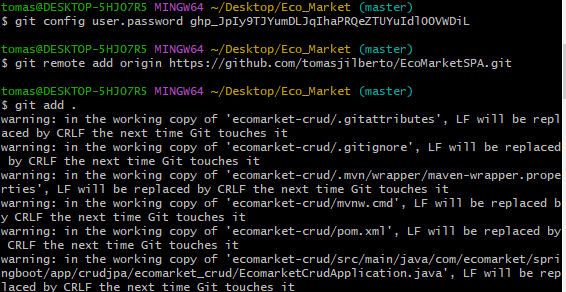


GITHUB



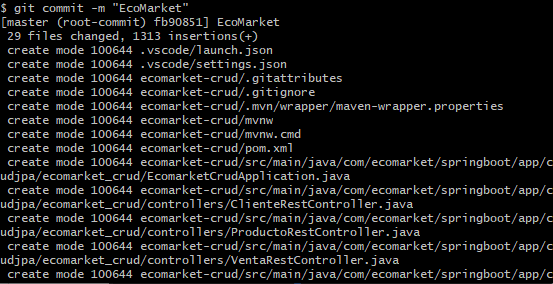
### 1. git init

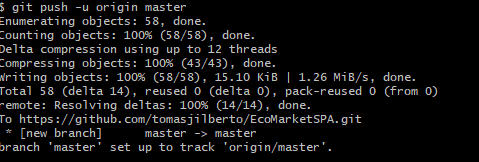
Este comando inicializa un nuevo repositorio Git en la carpeta del proyecto. Permite comenzar a versionar los archivos localmente.



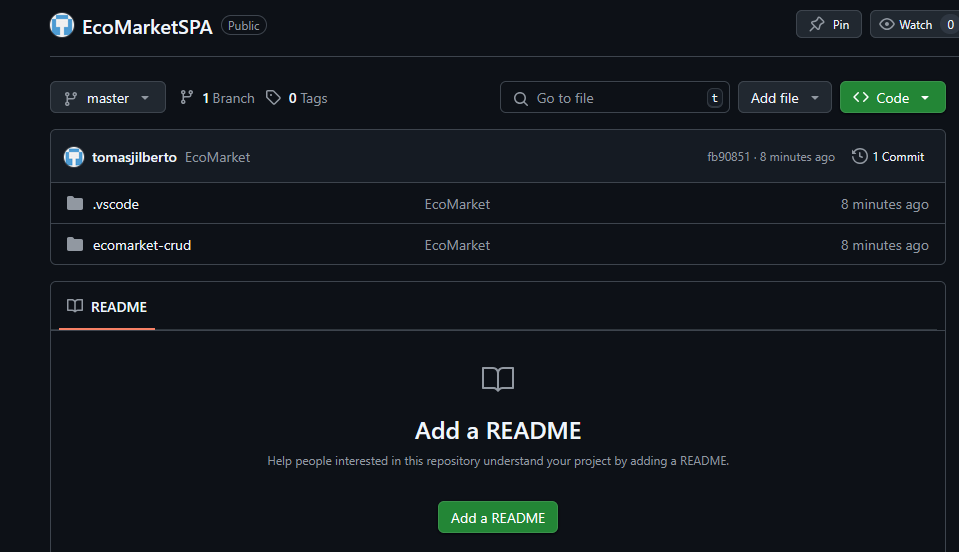
### 2. git add .

Agrega todos los archivos del proyecto al área de preparación para incluirlos en el próximo commit.





Aca se terminan de subir los archivos como se aprecia en GitHub



# 

# 