

**TEMA DE LA PRÁCTICA:** Diseño cliente-Servidor

**MÓ DULO:** Aplicaciones Distribuidas

**DOCENTE:** Ing. Caiza Caizabuano Jose Ruben

**ESTUDIANTE:** Quishpe López Luis Alexander

**NIVEL:** Sexto Software “A”

**FECHA:** 05/04/2025

1. **OBJETIVOS**

**General:**

* Implementar y documentar un API RESTful que ayude el acceso de datos desde un servidor local (Ubuntu), siguiendo los pasos estudiados en clase.

**Específicos:**

* Desarrollar un API RESTful con todos los endpoints para operaciones CRUD, utilizando la tecnología de ASP.NET Core Web API.
* Configurar un servidor remoto en Ubuntu que permita alojar los datos consumidos por el API, garantizando la comunicación mediante protocolo HTTP y formato JSON.
* Validar y documentar el funcionamiento del API y del servidor mediante pruebas manuales en Postman y en el navegador.

1. **INTRODUCCION**

El diseño de APIs RESTful se ha consolidado como un estándar de desarrollo, con la finalidad de ser un proveedor de servicios web que pueden ser consumidos por otros desarrolladores en distintos tipos de aplicaciones como sitios web, aplicaciones móviles, entre otros [1]. Este estilo arquitectónico, basado en los principios del protocolo HTTP, permite la interoperabilidad entre sistemas distribuidos, facilitando la integración de aplicaciones cliente-servidor. En la parte académica y profesional, su adaptabilidad ha crecido exponencialmente, especialmente en entornos como en la nube y microservicios [2]. En esta práctica se implementará un API RESTful funcional, con el fin de consumir los datos desde un servidor remoto que se implementará en Ubuntu.

Lo importante de este trabajo es aplicar los conceptos y conocimientos impartidos por el docente en clases, donde se priorizará la implementación de la API RESTful en el servidor y que permita el manejo de los datos.

1. **HERRAMIENTAS EMPLEADAS**

* Plataforma estudiantil
* Internet
* Inteligencia Artificial
* Computadora
* Word y apuntes de clase.
* Windows, Ubuntu, SQL server .Net.

1. **DESARROLLO**

A continuación, se describen los pasos realizados para implementar el API RESTful y la integración con el servidor local en Ubuntu.

**Paso 1:** Creación del proyecto API RESTful

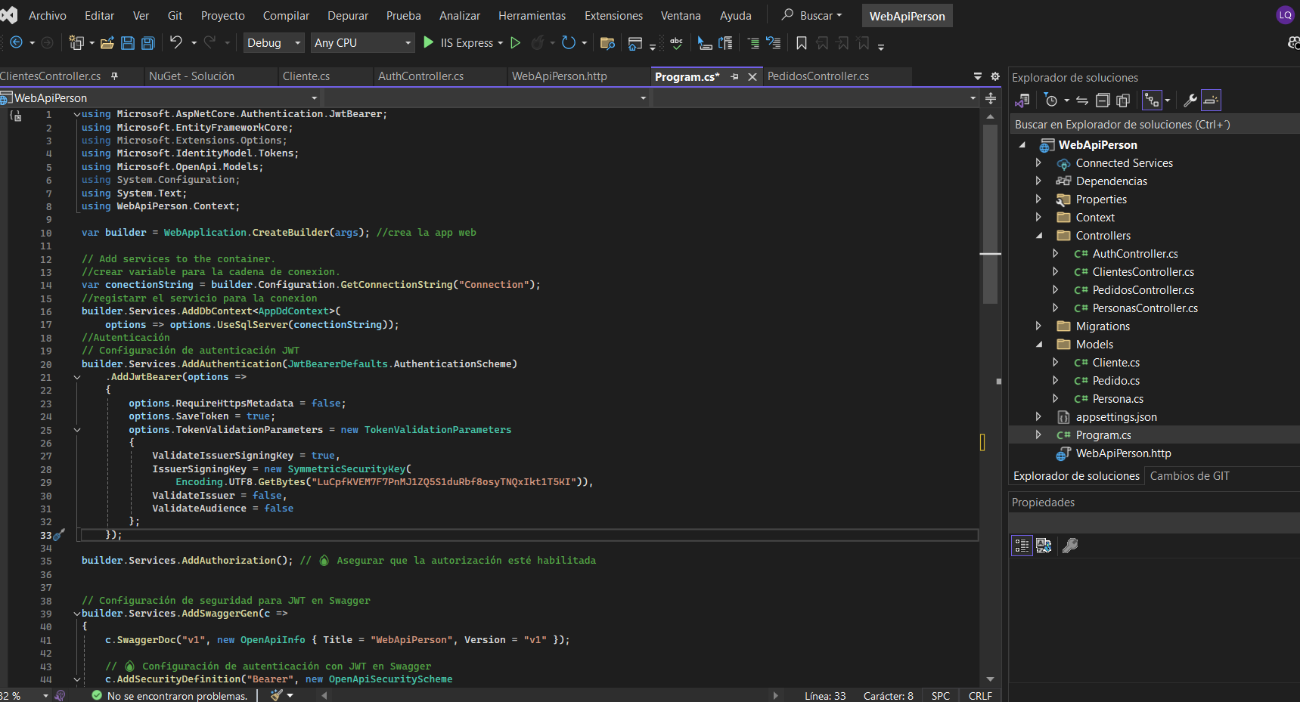
Se desarrolló una API RESTful en Visual Studio aplicando el patrón MVC, definiendo las entidades Persona, Cliente y Pedido como modelos, junto con sus respectivos controladores CRUD. Para la documentación interactiva se implementó Swagger como interfaz de visualización. Adicionalmente, se integró seguridad mediante JWT (JSON Web Tokens), configurando autenticación y autorización en los endpoints críticos.

Gráfico 1. Configuración del program.cs

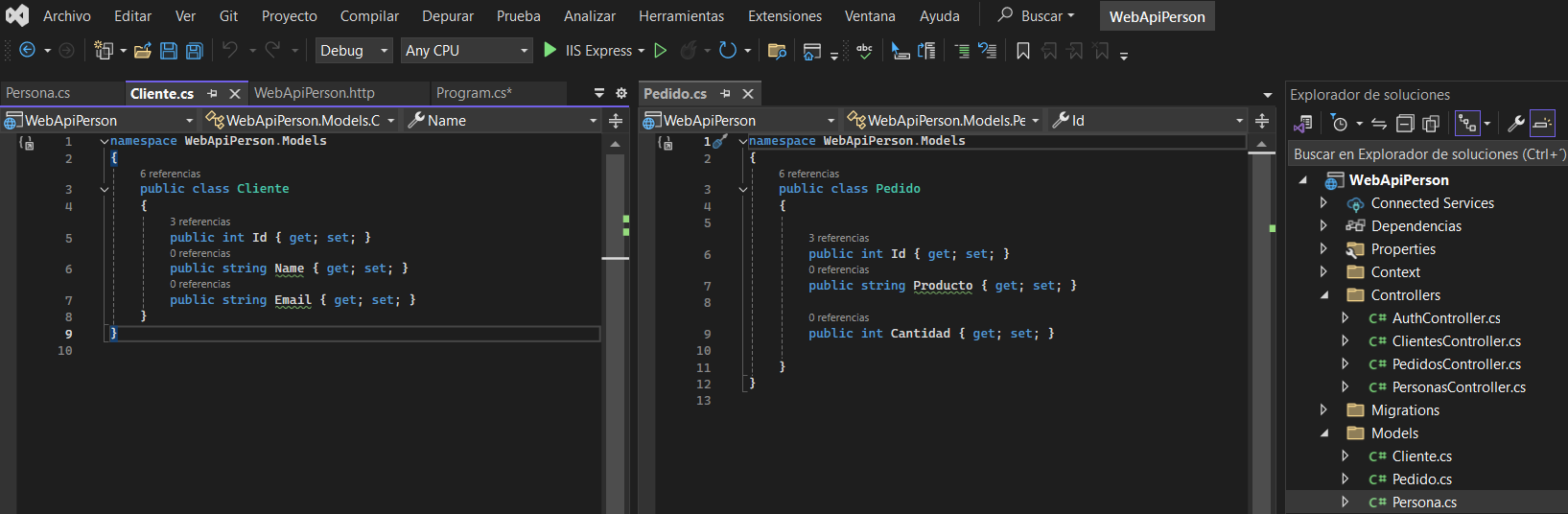
Creación modelos que funcionarán como de entidades estructuradas de manera que se organiza y actúa como intermediario entre la lógica de negocios y la base de datos.

Gráfico 2. Entidades: clientes, pedidos y personas.

.

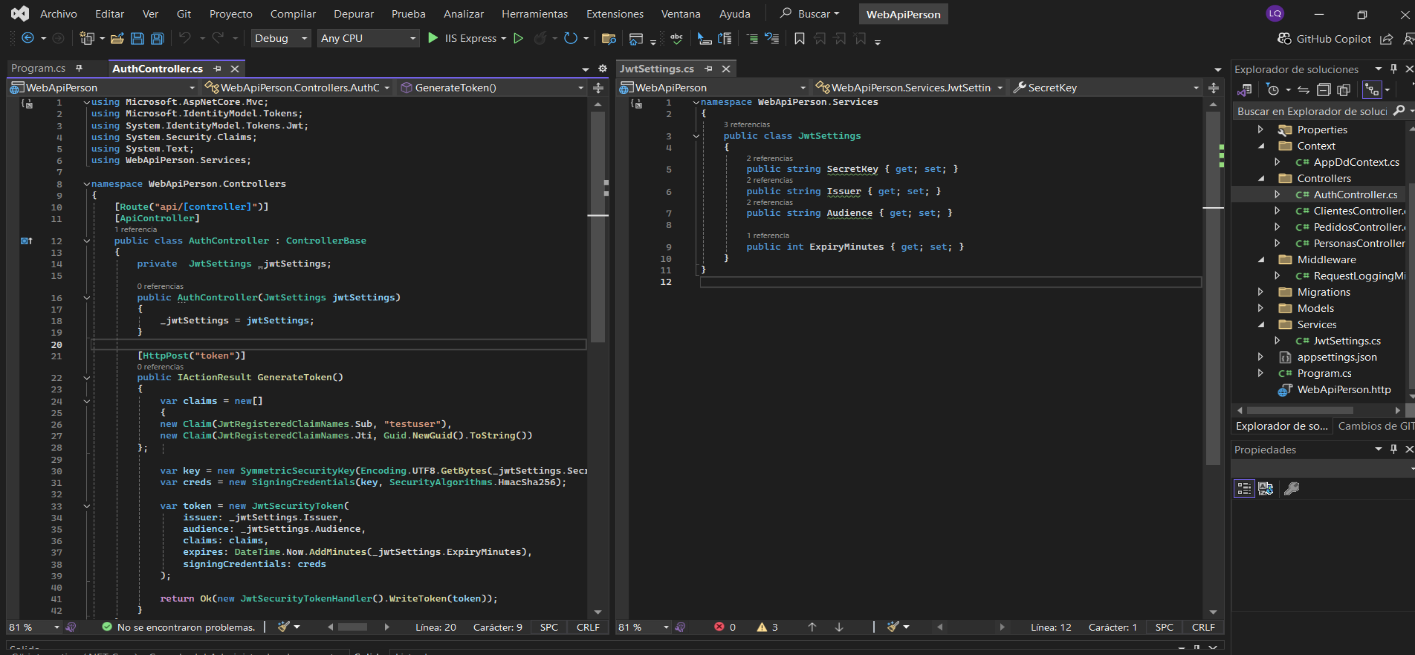
Creación de los controllers de cada entidad que maneja la API RESTful. Cada uno contiene todos los métodos de acceso para peticiones generando respuestas desde la base de datos.

Gráfico 3. Controllers: personas, pedidos, clientes

**Paso 2:** JWT (Json Web Token)

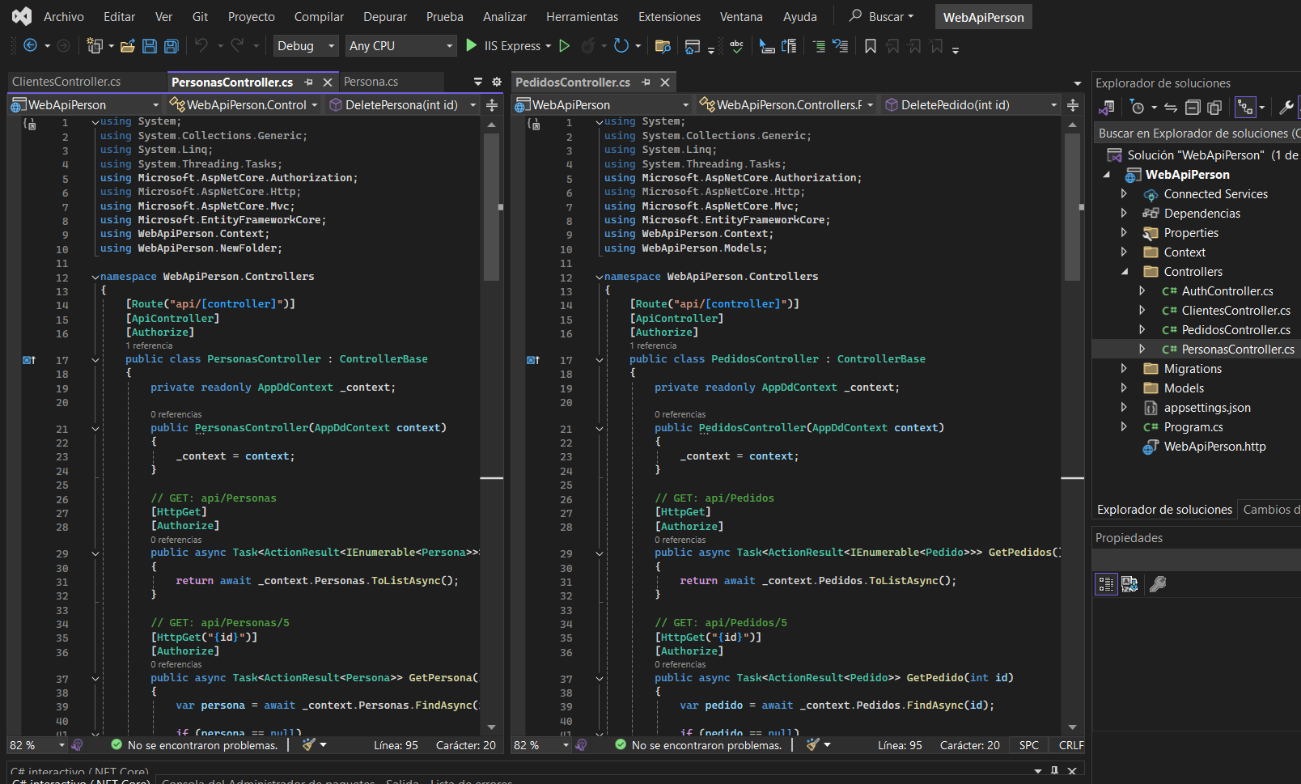
La clase JwtSettings.cs y AuthController.cs contiene la implementación y configuración para la autenticación JWT, es utilizado para firmar los tokens de accesos a las distintos endspoints.

Gráfico 4. Implementación JWT

**Paso 3:** Creación de la Base de Datos

Se agrega el paquete NuGet “Microsof.EntityFrameworkCore.SqlServer” mappear las clases (modelos) a tablas SQL Server, generando automáticamente la base de datos mediante migraciones. Para esto se necesita la configuración de la cadena de conexión esto se hace en el appsettings.json, sin la necesidad de scribir SQL permite tener el esquema directamente en la base de datos.

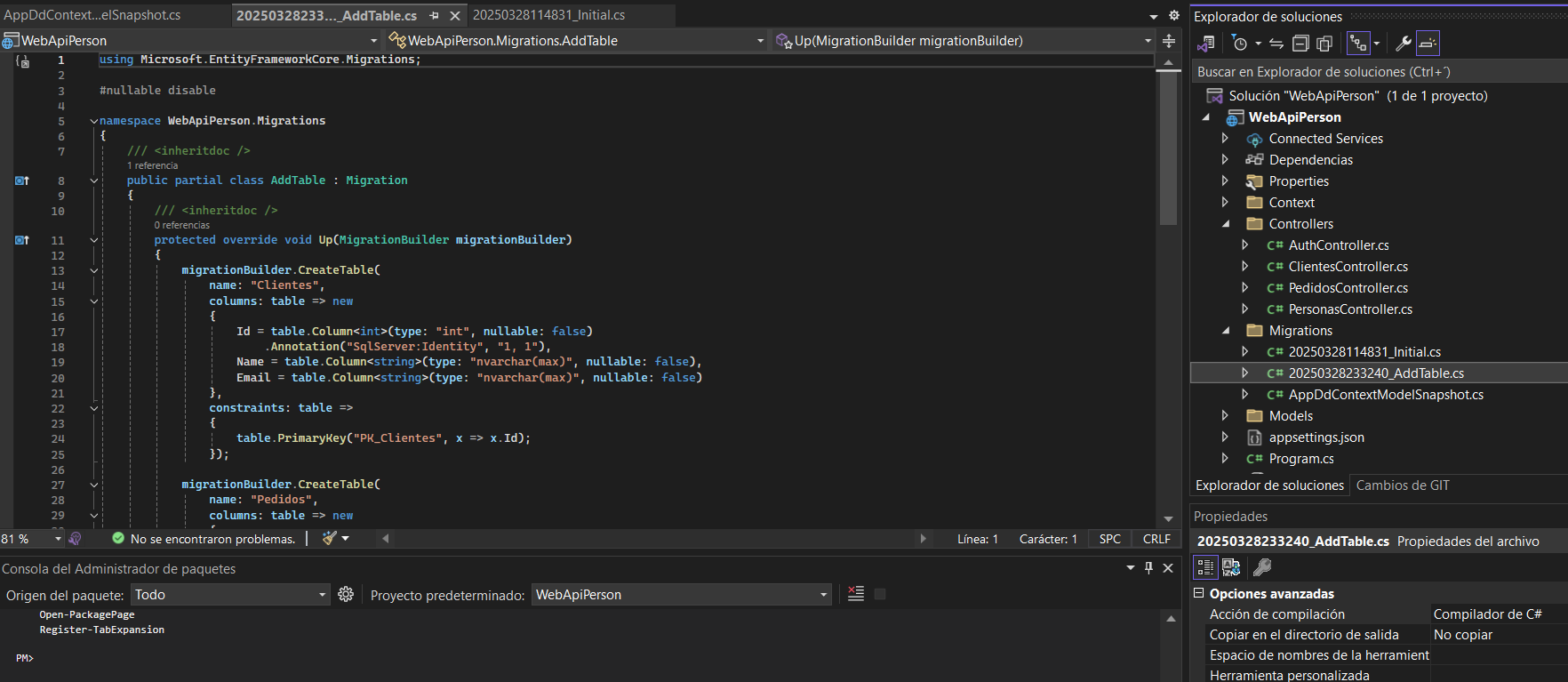


Gráfico 5. Migración y creación de la base de datos.

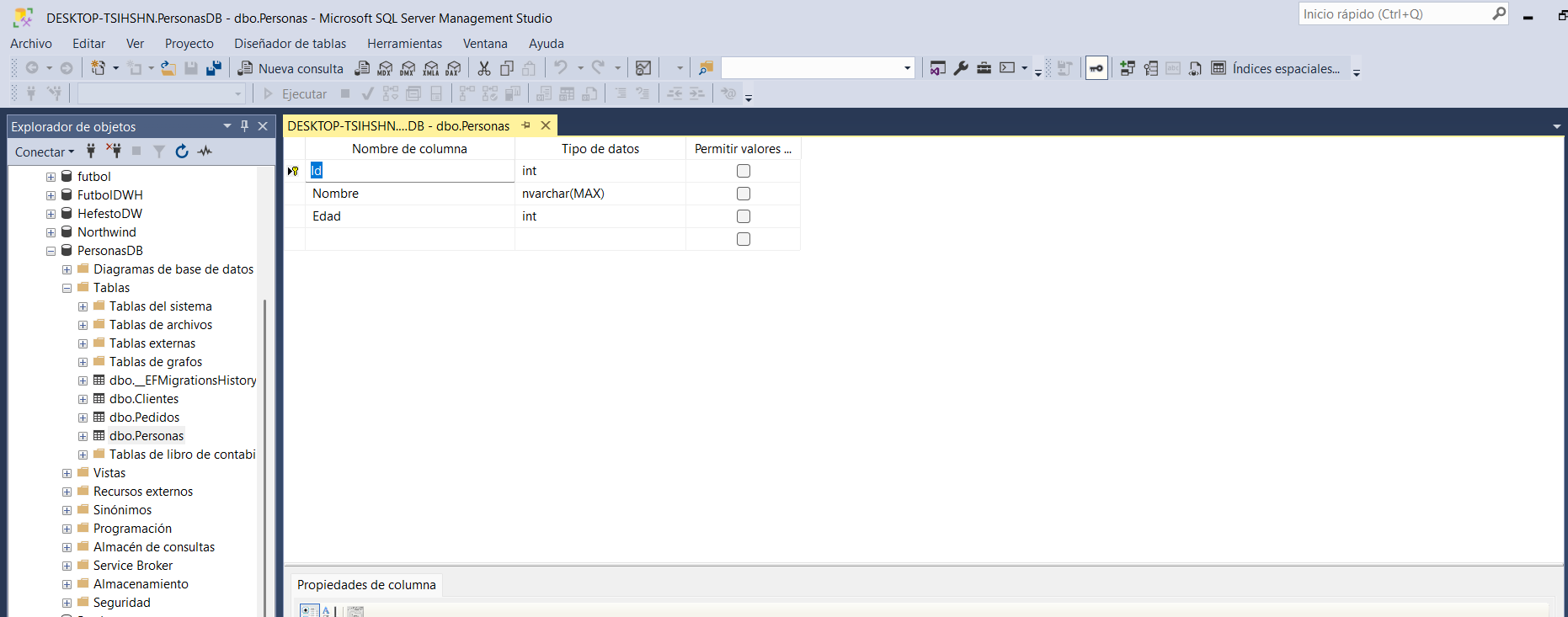
**Paso 4:** Verificación de la base de datos.

Gráfico 6. Base de Datos y tablas.

**Paso 5:** Funcionamiento de la API con Swagger

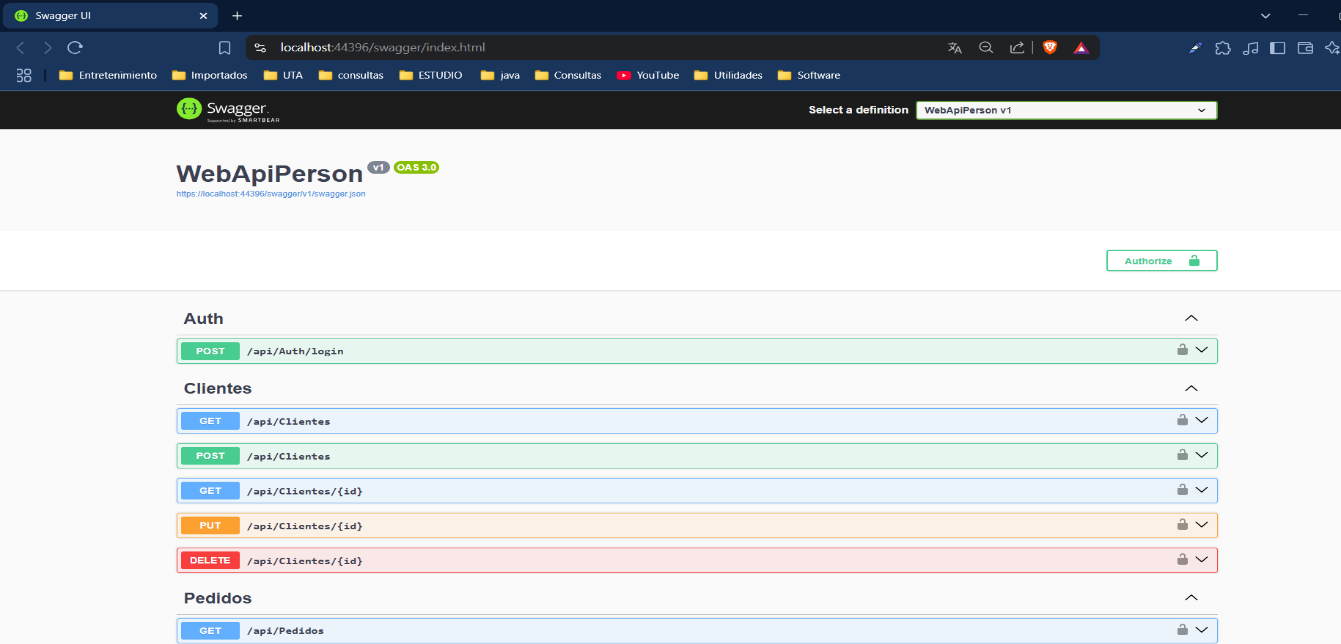
Una vez configurado la base de datos y los controladores. Se ejecuta la aplicación para verificar el funcionamiento de la API RESTful, se utiliza la Vista el SwaggerUI. Como se observa en la imagen aparecen los diferentes endpoints que contiene el proyecto organizado por controladores además de la ruta la cual se manejan. Incluido el JWT para seguridad.

Gráfico 7. Funcionamiento de endopoints de la API RESTful.

**Implementación del servidor en Ubuntu conjunto con el proyecto.**

**Paso 1:** Creación de la carpeta compartida

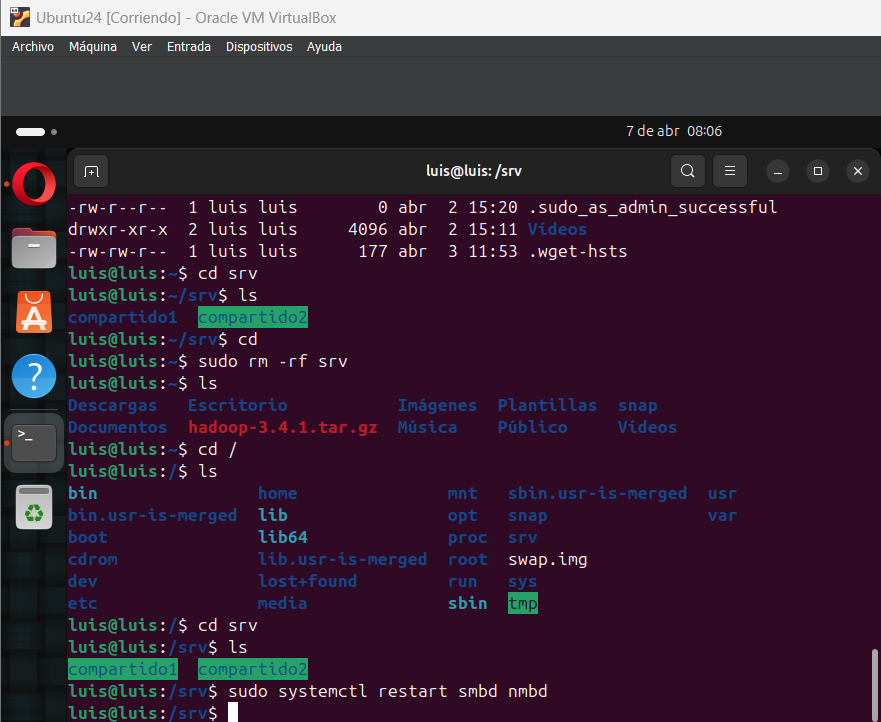
Una vez ingresado a la máquina virtual y con la ayuda de comandos desde la terminal se crea la carpeta compartida, asignando permisos de acceso y usuarios, además de utilizar samba para el servicio compartido.

Gráfico 8. Creación de la carpeta compartida Ubuntu-Windows.

Verificación de la carpeta compartida desde Ubuntu a Windows accediendo desde la ruta **//IP/compartido2**.

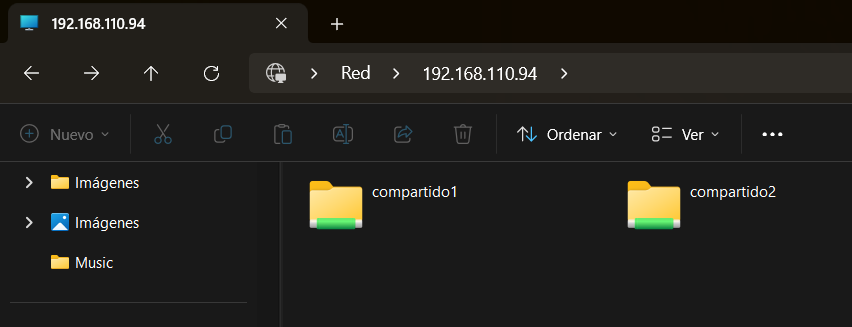


Gráfico 9. Carpeta compartida Windows

**Paso 2:** Publicación de la API

Desde Visual Studio se publica el proyecto dentro de la carpeta compartida para luego utilizar el proyecto desde Ubuntu y levantar el servidor. Configurando la ruta de destino y el Runtime de destino.

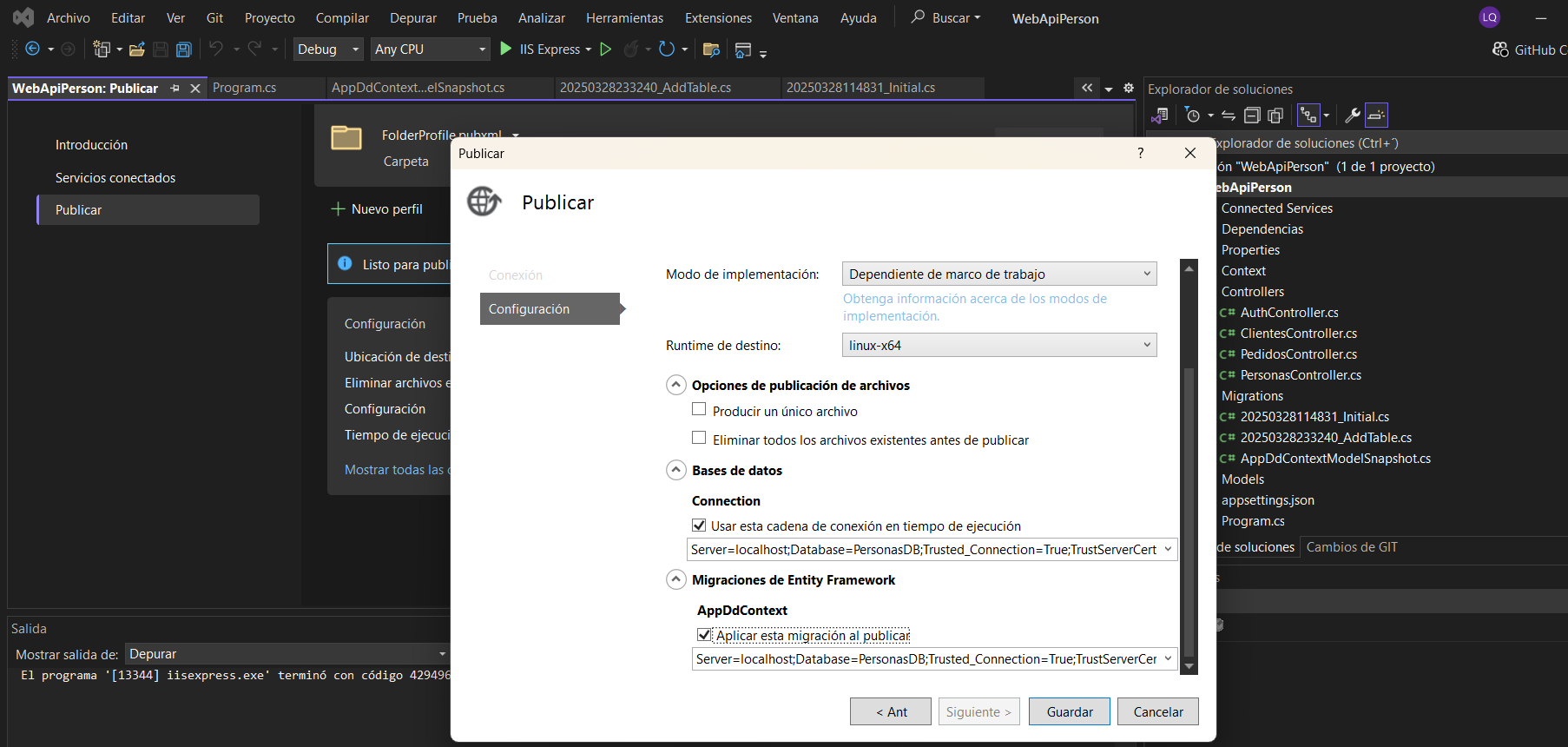


Gráfico 10. Publicación del proyecto dentro de la carpeta compartida.

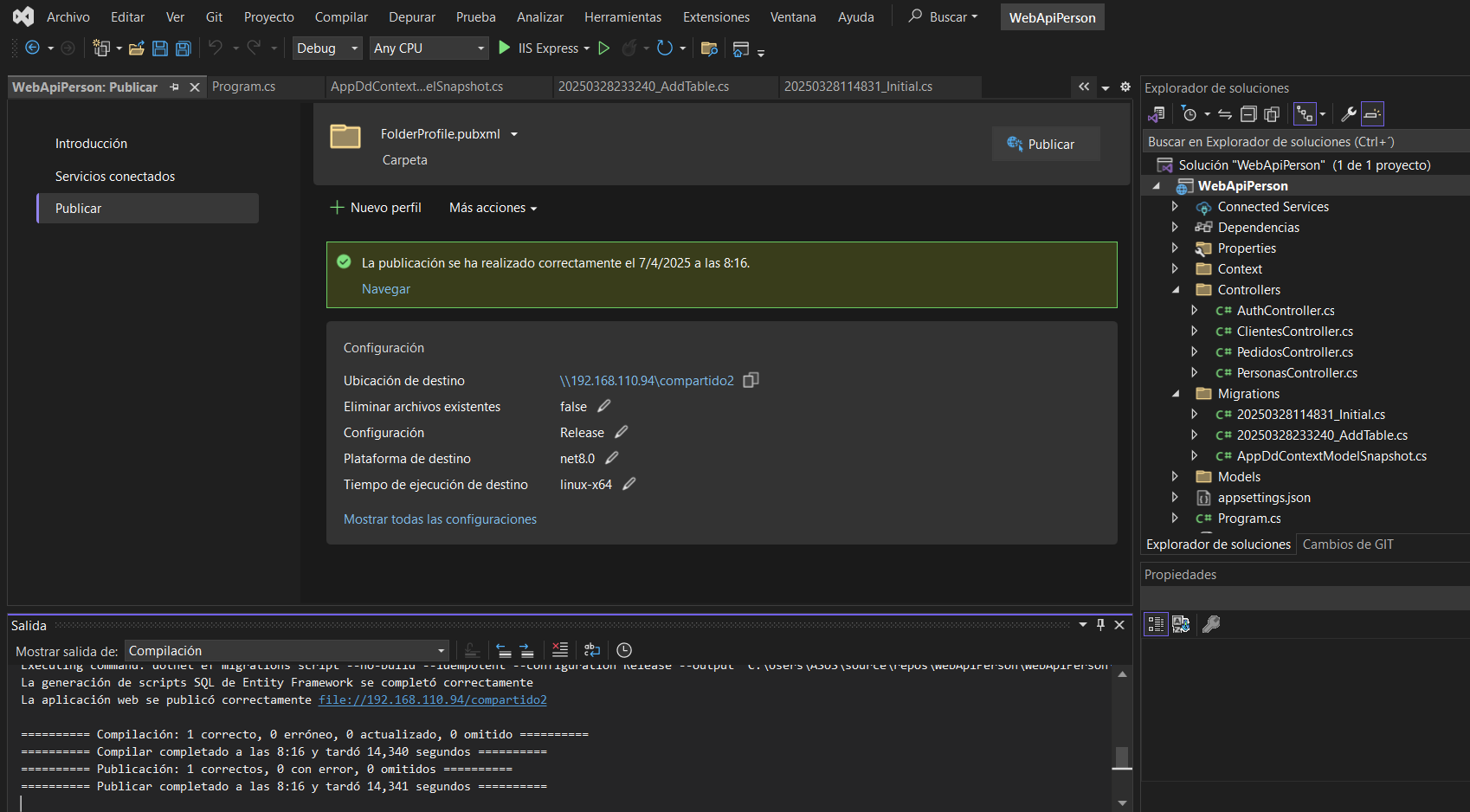


Gráfico 11. Proyecto publicado.

Ejecución del proyecto en Ubuntu para probar el funcionamiento de la API RESTful. Para ejecutar el API RESTful se debe colocar el siguiente comando: dotnet WebApiPerson.dll

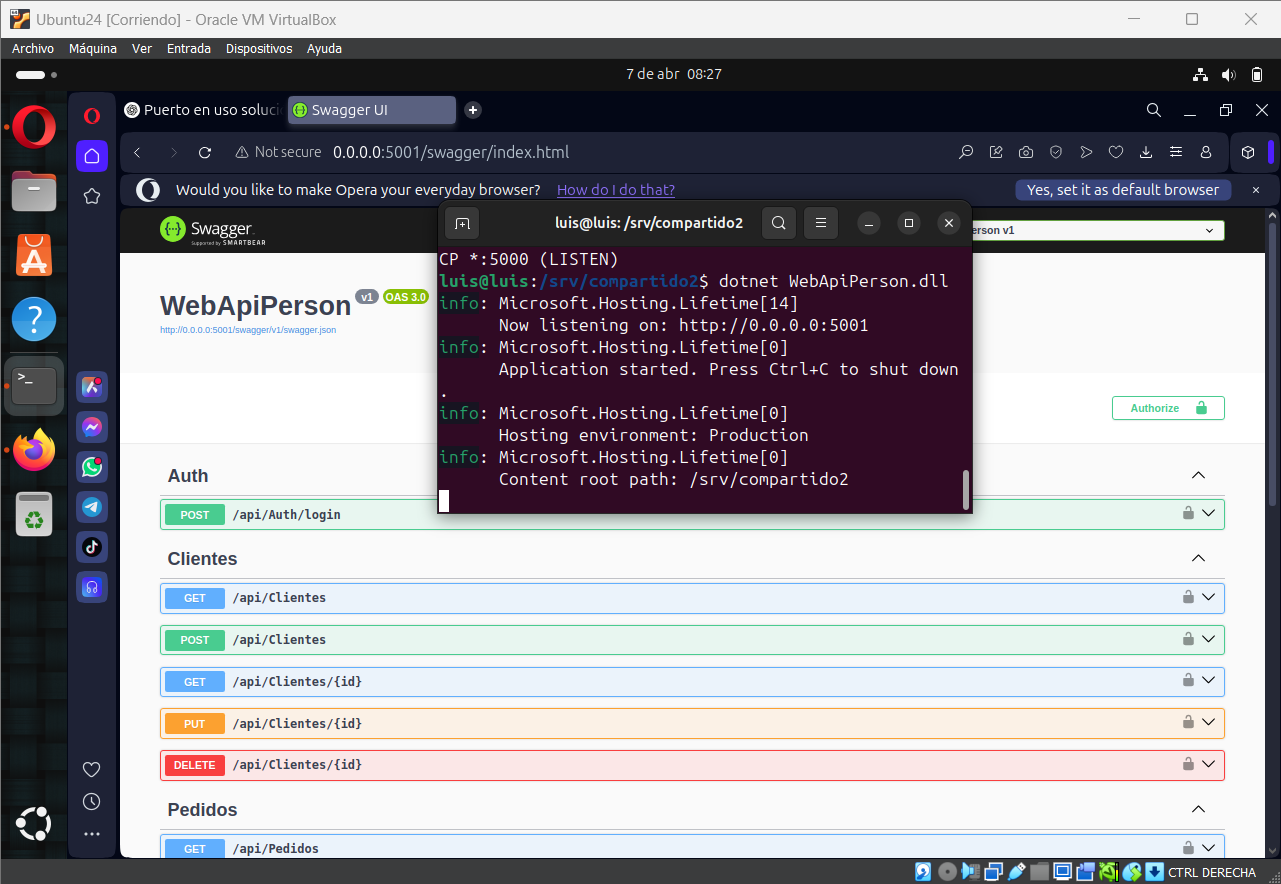


Gráfico 12. Ejecución de la API RESTful

Prueba de funcionamiento de la API RESTful desde el servidor local.

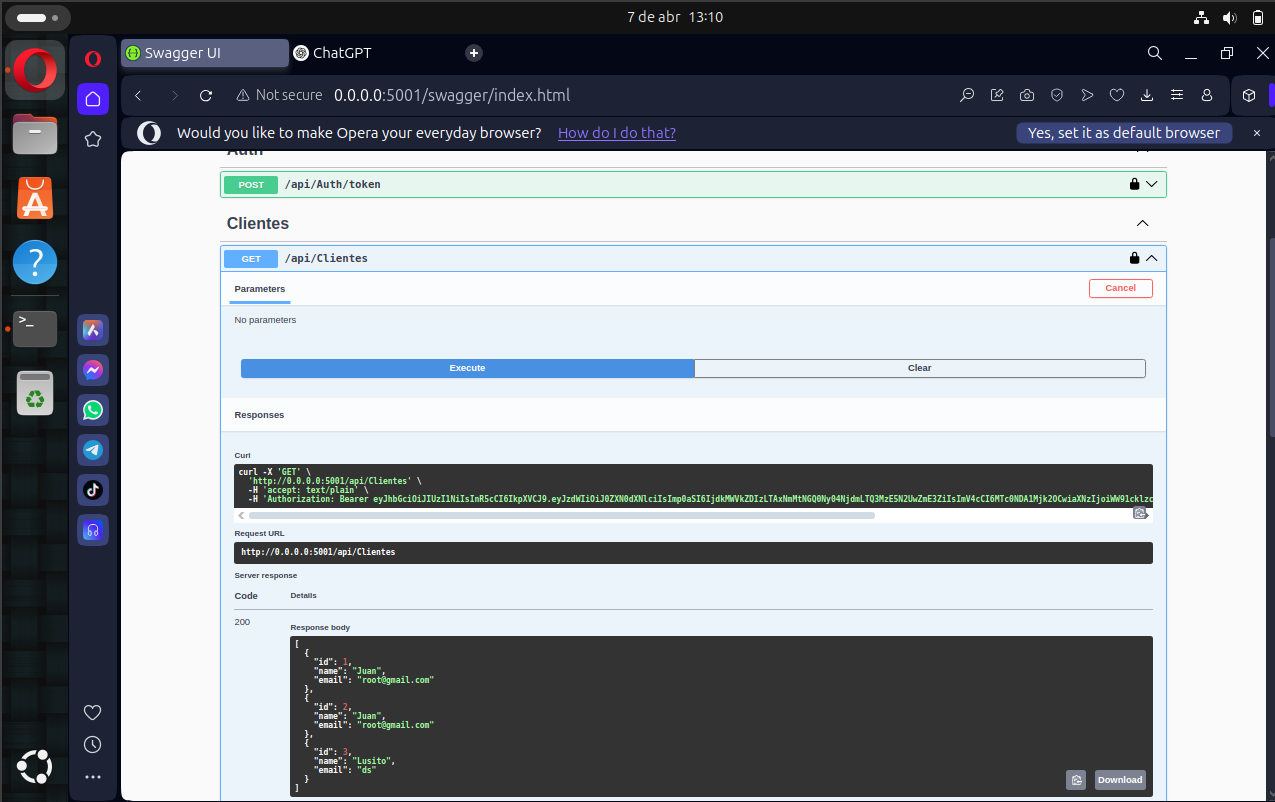


Gráfico 13. Api Restfull iniciado desde Ubuntu servidor local.

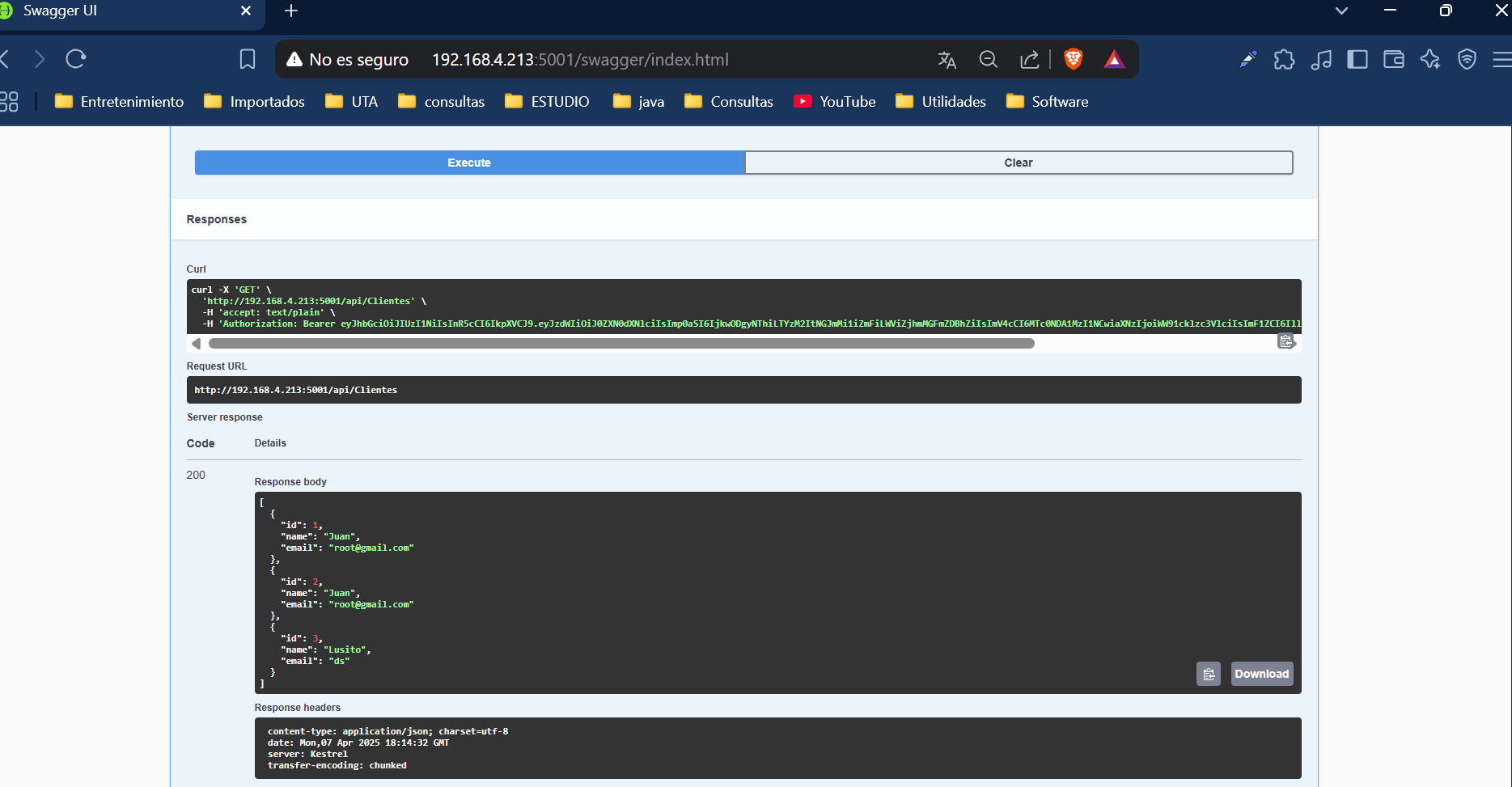
Prueba de funcionamiento de la API RESTful desde Windows manejado por la dirección <http://192.168.4.213:5001/Sagger>

Gráfico 14. API RESTful en Windows.

**Paso 4:** Creación del servicio systemd para que la API arranque automáticamente.

Para que la API arranque de manera automática cuando se encienda el servidor hay que crear un archivo de servicio con las siguientes configuraciones:

[Unit]

Description=API Resy en .NET

After=network.target

[Service]

WorkingDirectory=/srv/compartido1

ExecStart=/usr/bin/dotnet /srv/compartido1/MiApiRest.dll

Restart=always

RestartSec=10

# TimeoutSec=30

KillSignal=SIGINT

User=www-data

Environment=ASPNETCORE\_ENVIRONMENT=Production

Environment=DOTNET\_PRINT\_TELEMETRY\_INFORMATION=false

SyslogIdentifier=miapi

[Install]

WantedBy=multi-user.target

Una vez configurado el archivo se inicia el proceso una sola vez con el siguiente comando: **sudo systemctl start miapi**.

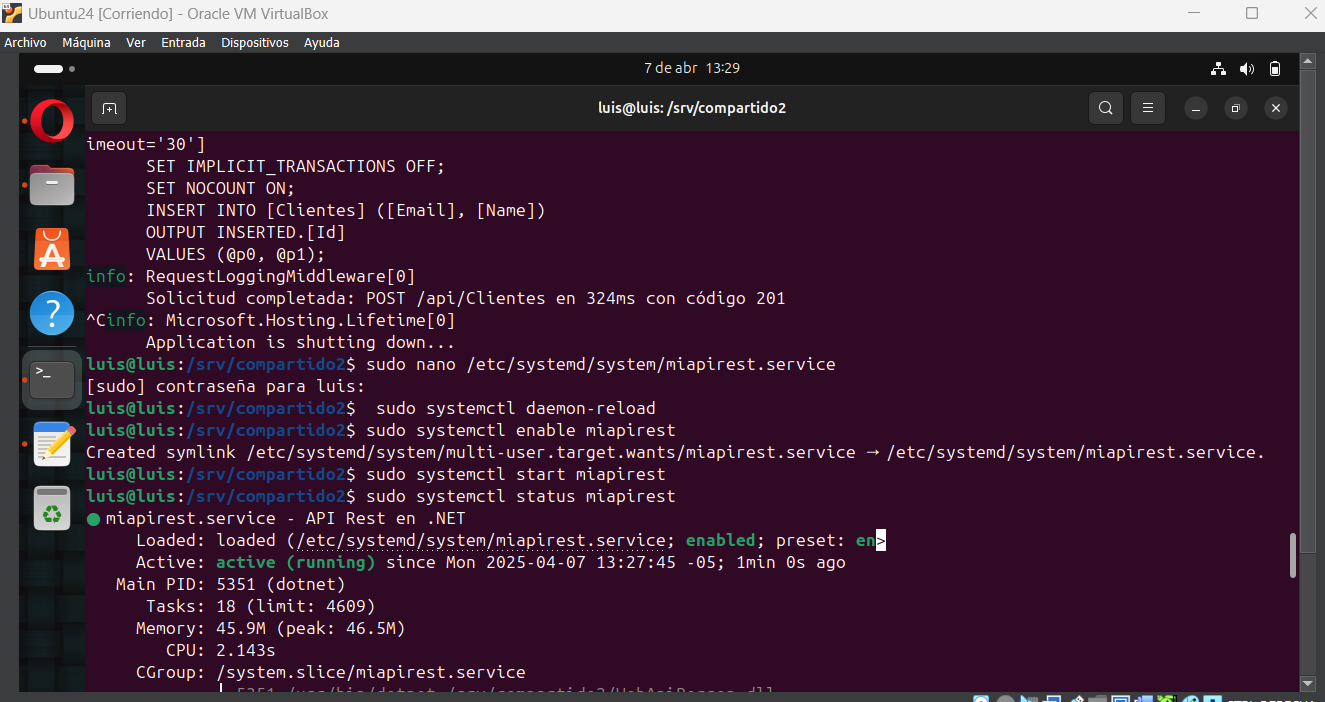


Gráfico 15. Status del servicio running.

1. **CONCLUSIONES**

En conclusión, se implementó con éxito una API RESTful funcional alojada en un servidor Ubuntu, capaz de gestionar las entidades clientes, pedidos y personas con un rendimiento óptimo. La integración de Entity Framework agilizó el desarrollo al automatizar la creación y consulta de la base de datos, eliminando la necesidad de escribir comandos SQL manuales y garantizando un acceso eficiente a los datos.

Además, como capa crítica de seguridad, se implementó JWT (JSON Web Tokens), que autentica y autoriza a los usuarios, protegiendo los endpoints contra accesos no autorizados y mitigando riesgos de ataques externos. Finalmente, se configuró el servidor Ubuntu para exponer la API mediante una dirección IP específica, permitiendo el acceso remoto a los endpoints desde cualquier cliente autorizado.

1. **REFERENCIAS BIBLIOGRÁFICAS**

# Bibliografía

|  |  |
| --- | --- |
| [1] | T. A. V. C. a. K. E. U. Arriaza, «Implementación de API RESTFUL para uso en APP de ofertas (Chaplist), desarrollada con IONIC,» *Ph.D. dissertation, Universidad de San Carlos de Guatemala,* 2016. |
| [2] | S. Newman, «O'Reilly Media Inc. Building microservice,» 2015.  joseru82@hotmail.com |