Manual de Instalación del Sistema De Ventas

Versión 1.0

Contenido

Contenido ii

Historial de revisiones ii

1. Introducción 1

1.1 Propósito 1

1.2 Audiencia 1

1.3 Referencias 1

2. Ambiente operativo 1

2.1 Restricciones de implementación 2

2.2 Documentación del usuario 2

Historial de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Autor** | **Fecha** | **Comentarios** | **Versión** |
| Steven Lizarzaburu | 30/03/2025 |  | 1.0 |
|  |  |  |  |

# Introducción

## Propósito

Este manual de instalación tiene como objetivo proporcionar instrucciones detalladas para la instalación, configuración y puesta en marcha del **Sistema de Ventas**. Está dirigido a desarrolladores, administradores de sistemas y personal técnico encargado de implementar el sistema en los entornos de desarrollo, pruebas y producción.

El documento explica los requisitos previos, el proceso de instalación paso a paso, configuraciones necesarias y verificación de la instalación para garantizar un funcionamiento óptimo del sistema

## Audiencia

Este documento está dirigido a:

* **Desarrolladores y Arquitectos de Software:** Para instalar, configurar y mantener el sistema.
* **Project Managers:** Para supervisar la instalación y comprender el entorno técnico.
* **Administradores de Sistemas:** Para configurar el entorno operativo.
* **Testers y QA:** Para validar la instalación y configuración.

## Referencias

Este manual se basa en la siguiente documentación de soporte:

* [Visual Studio 2022](https://visualstudio.microsoft.com/es/vs)
* [.NET 9](https://dotnet.microsoft.com/es-es/download/dotnet/9.0)
* [Node.js v20.18.0](https://nodejs.org/es/blog/release/v20.18.0)
* [Angular CLI 18.0.1](https://www.npmjs.com/package/@angular/cli/v/18.0.1)
* [Visual Studio Code](https://code.visualstudio.com/)
* [SQL Server 2019](https://www.microsoft.com/es-es/sql-server/sql-server-downloads)
* [SQL Server Management Studio (SSMS)](https://learn.microsoft.com/en-us/ssms/download-sql-server-management-studio-ssms)
* [Git](https://git-scm.com/)
* [GitHub](https://github.com/)

# Ambiente operativo

## Restricciones de implementación

* Cumplimiento con normativas locales y regulaciones de protección de datos.
* Uso obligatorio de .NET 9, Angular 18, Node 20.18.0 y SQL Server 2019.

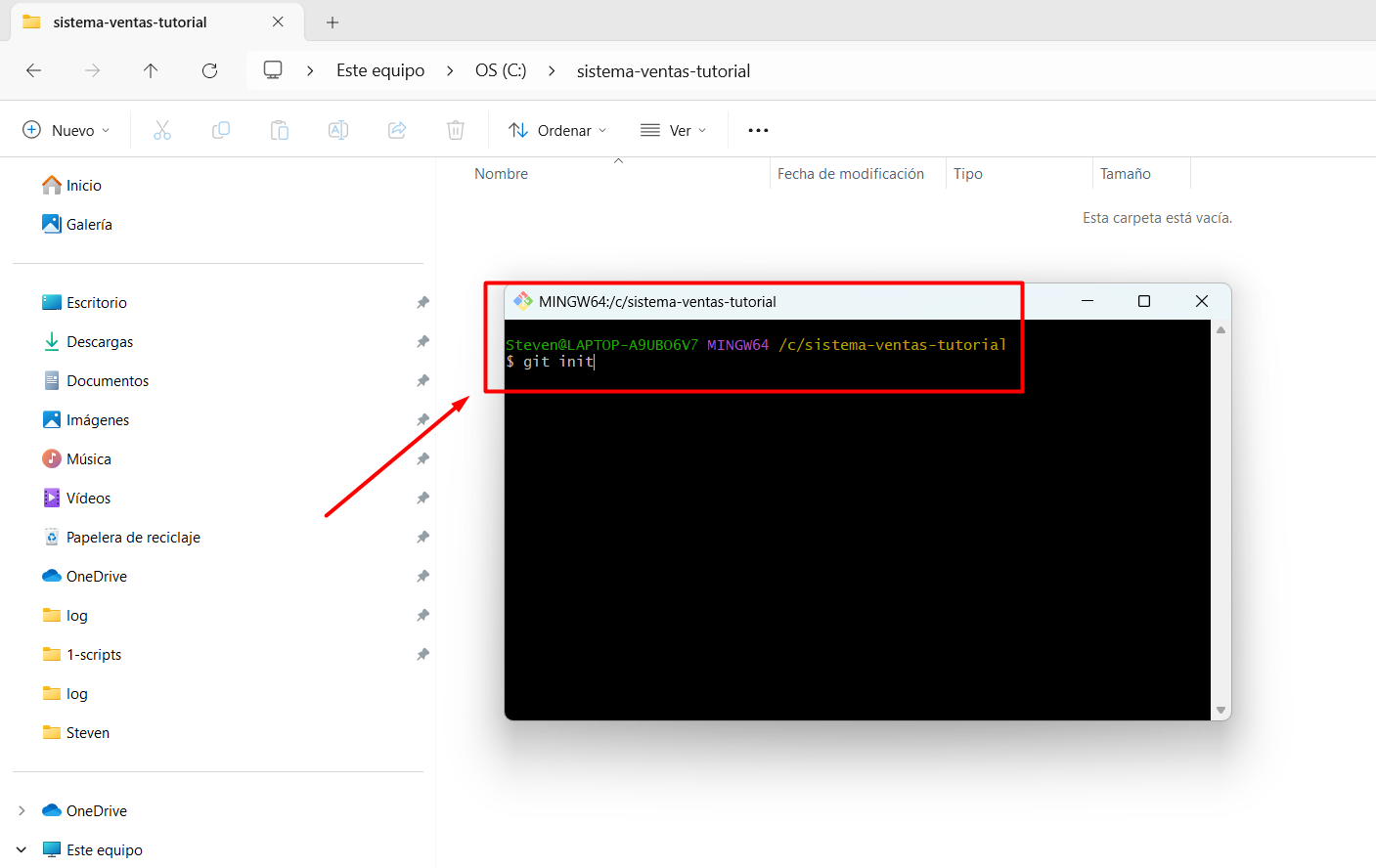
## Documentación del usuario

1. *Crea una carpeta y accede a ella. Haz clic derecho y selecciona* ***Open Git Bash Here****. Esto abrirá la ventana de Git para clonar el repositorio.*

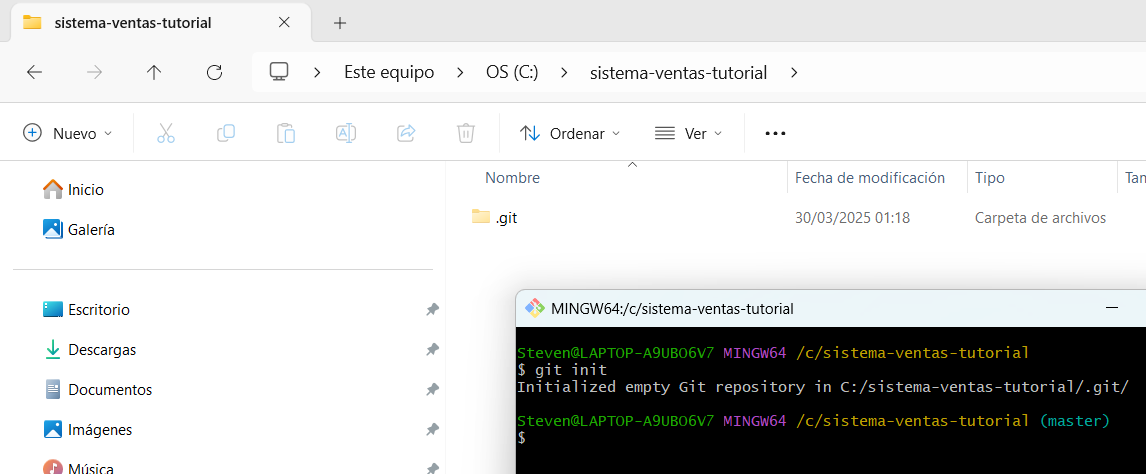
Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

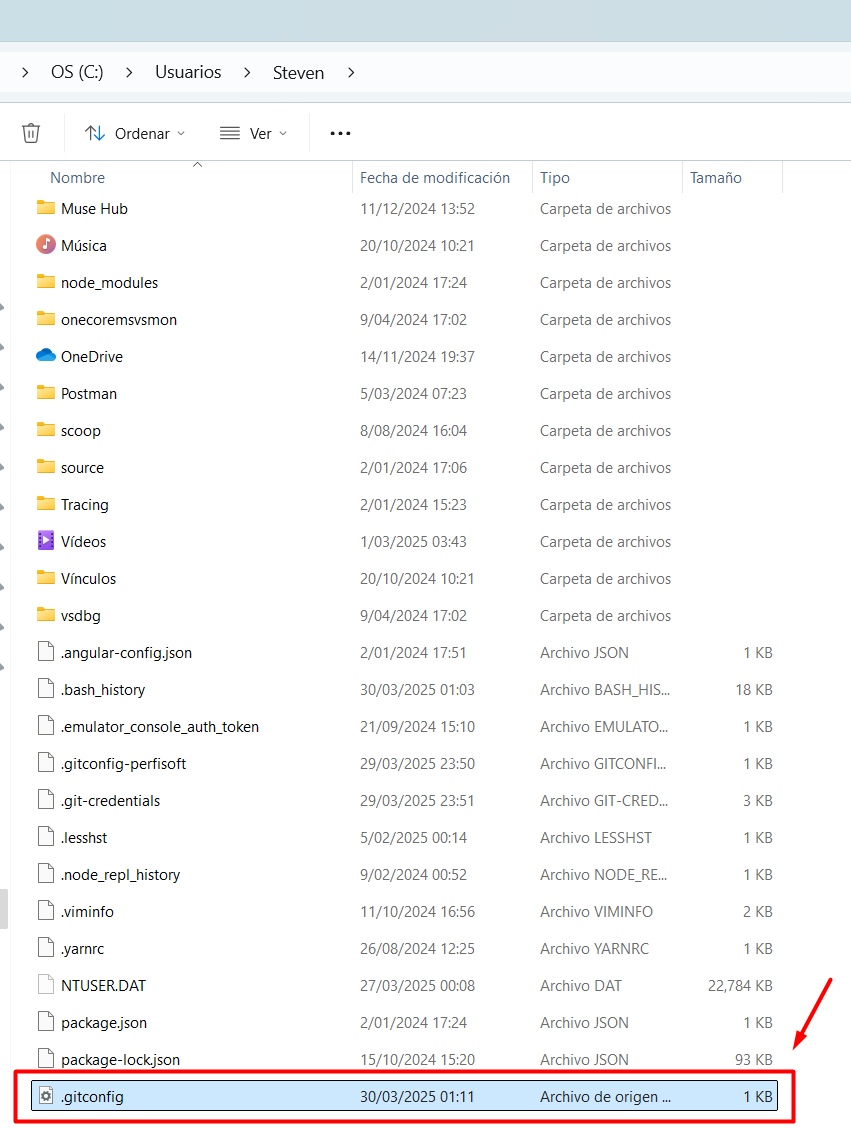
1. Escribe el comando **git init** y presiona **Enter** para inicializar un nuevo repositorio local.



(*Resultado esperado luego de la ejecución del comando)*



1. Ubica el archivo ***gitconfig*** para configurar tu correo electrónico. (***Es obligatorio usar el mismo correo que se agregó al repositorio al momento de adquirir el sistema)*.**



*(En mi caso, tengo configurado el parámetro* ***email*** *con mi correo personal, el cual está agregado al grupo del repositorio, lo que me permite clonar el proyecto con acceso a los recursos)*

Texto

El contenido generado por IA puede ser incorrecto.

1. Copiamos la URL del repositorio ([**https://github.com/slizarzaburupezua/sistemadeventas.git**](https://github.com/slizarzaburupezua/sistemadeventas.git)) para clonarlo en nuestro equipo local.

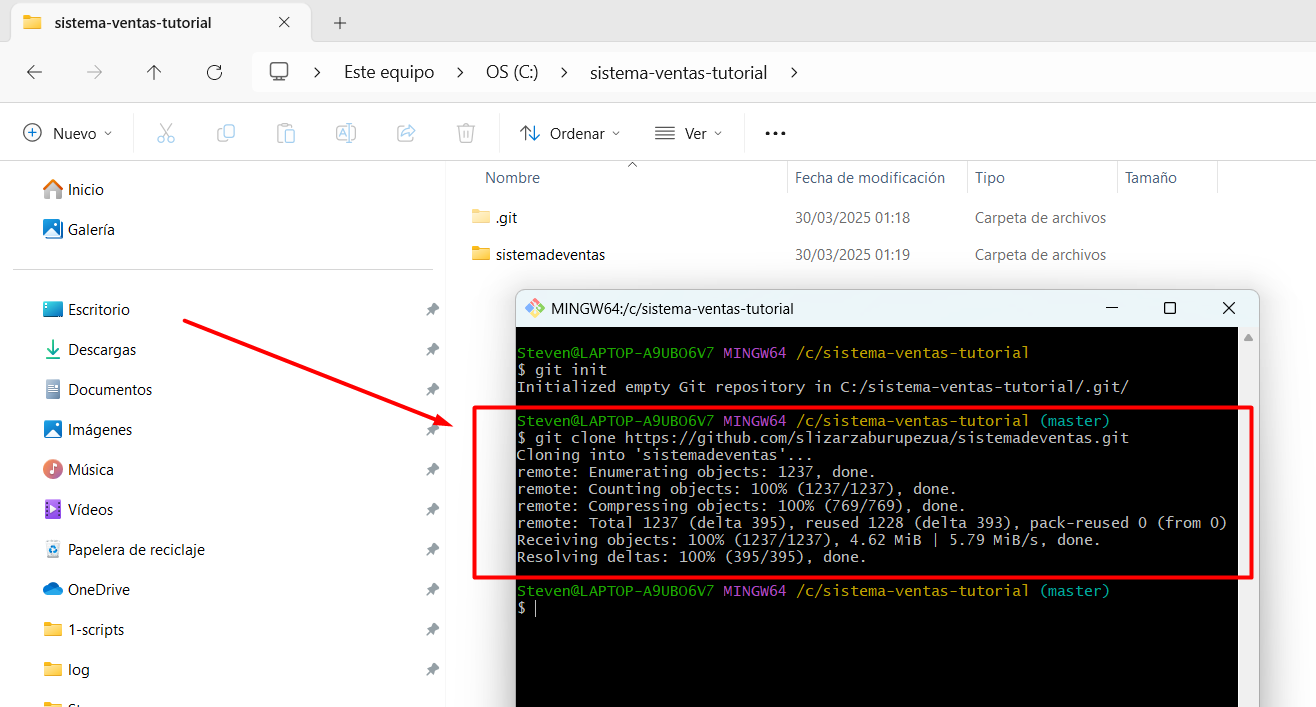
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

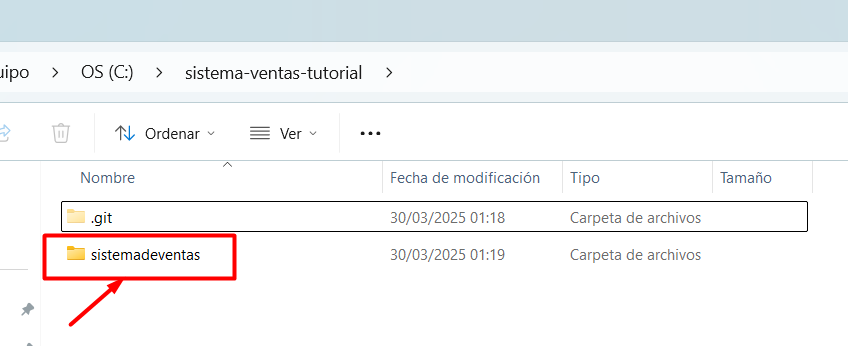
1. Las siguientes imágenes representan el resultado esperado luego de ejecutar el comando.

Interfaz de usuario gráfica, Texto, Aplicación

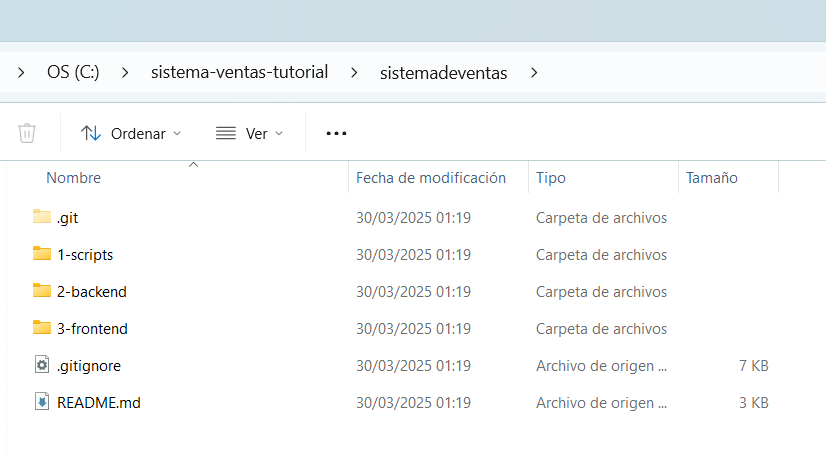
El contenido generado por IA puede ser incorrecto.



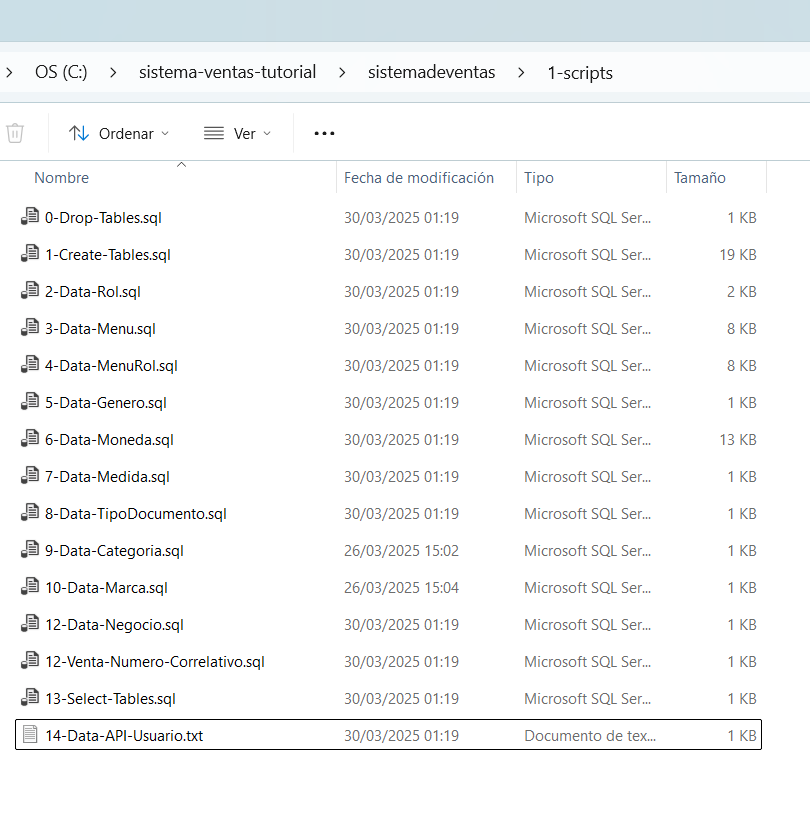
1. Veremos una carpeta llamada **"sistemadeventas"**, que contendrá todos los archivos necesarios para iniciar la instalación del sistema.

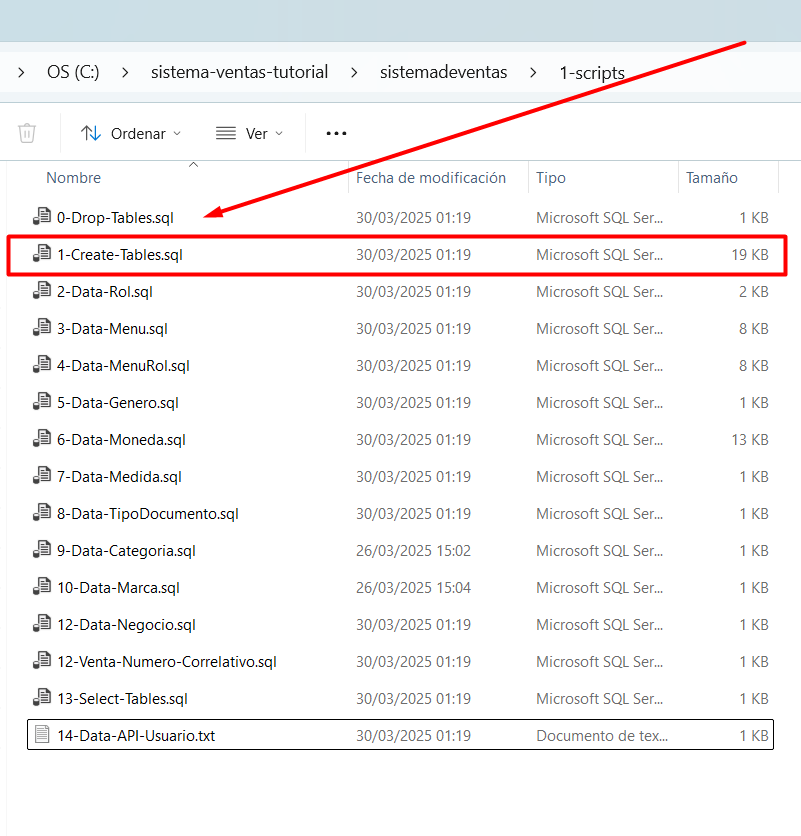


1. Ingresamos a la carpeta y veremos los recursos del proyecto que iremos desglosando cada uno de ellos hasta llegar a la ejecución del sistema en nuestra maquina local.

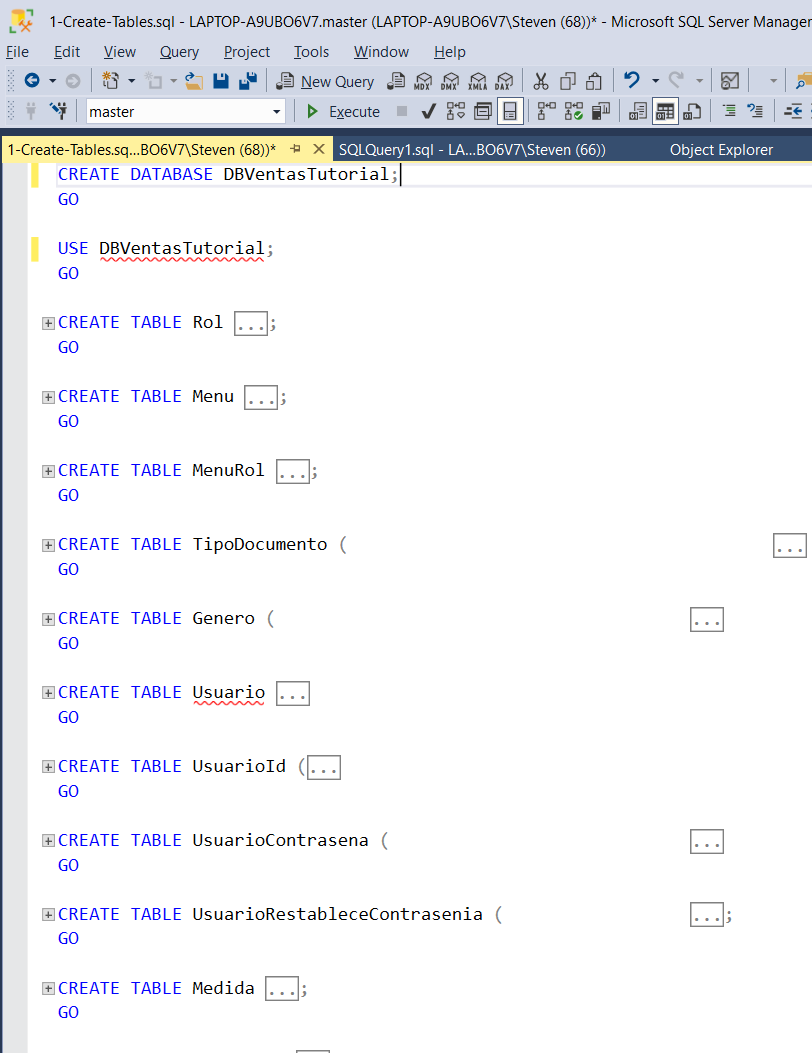


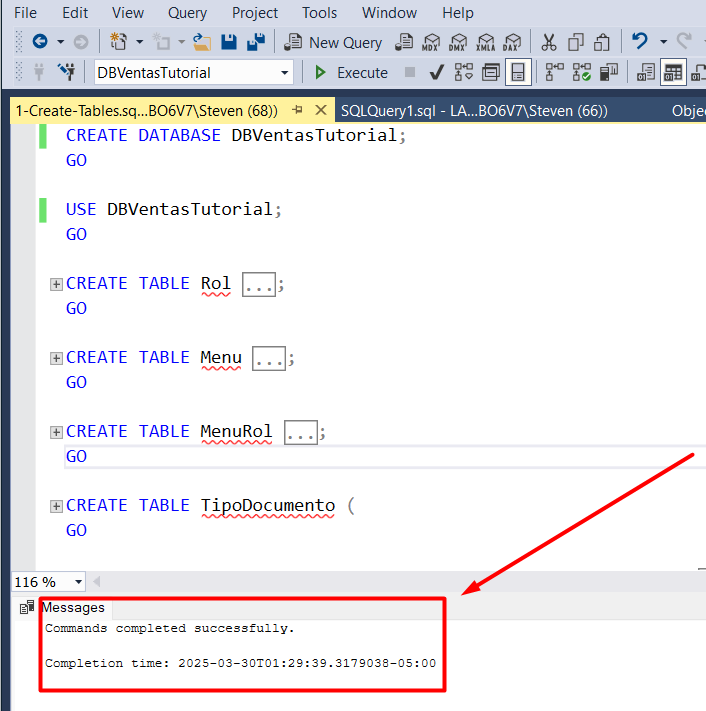
1. Ingresamos a la carpeta **"1-scripts"**, donde veremos todos los scripts que ejecutaremos, en orden, en **SQL Management Studio**. Estos scripts crearán la base de datos, tablas, relaciones y los parámetros iniciales del Sistema.



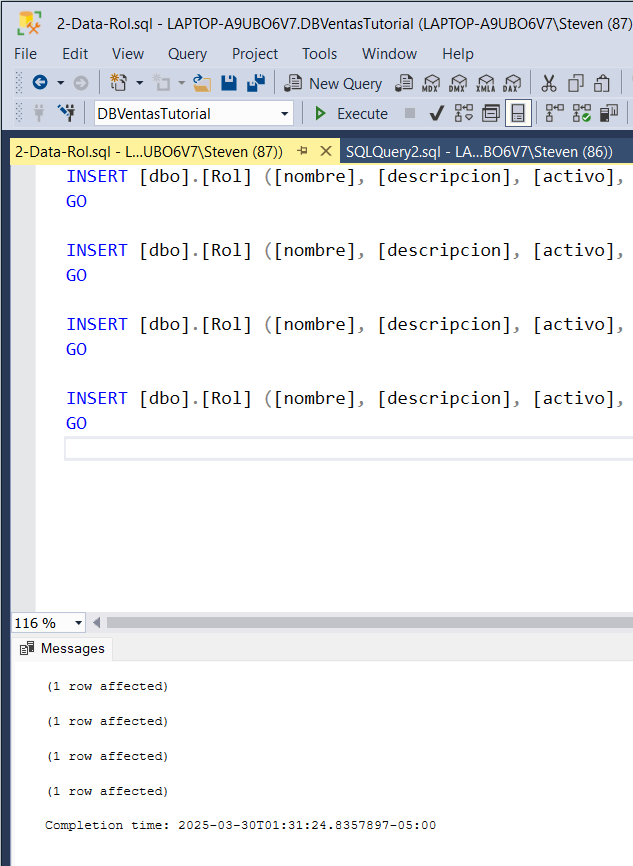
1. Abriremos el siguiente script “**1-Create-Tables.sql**” en nuestro SQL y lo ejecutaremos según el orden de las imágenes.

1. Las siguientes imágenes mostrarán la ejecución de los scripts en orden. Como ejemplo, la base de datos la **"DBVentasTutorial"**.

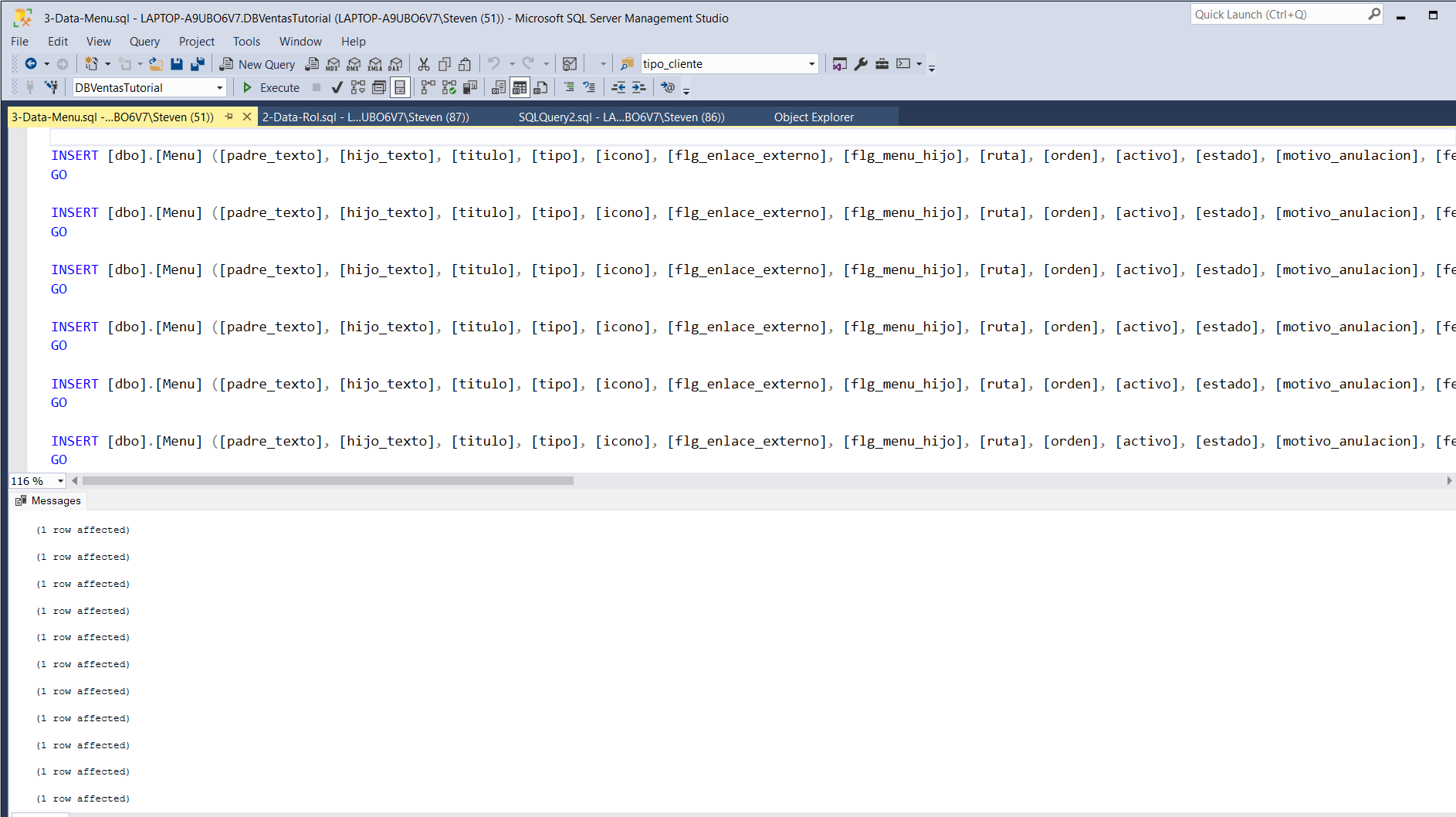


*Resultado esperado luego de la ejecución del primer script “****1-Create-Tables.sql****”*.

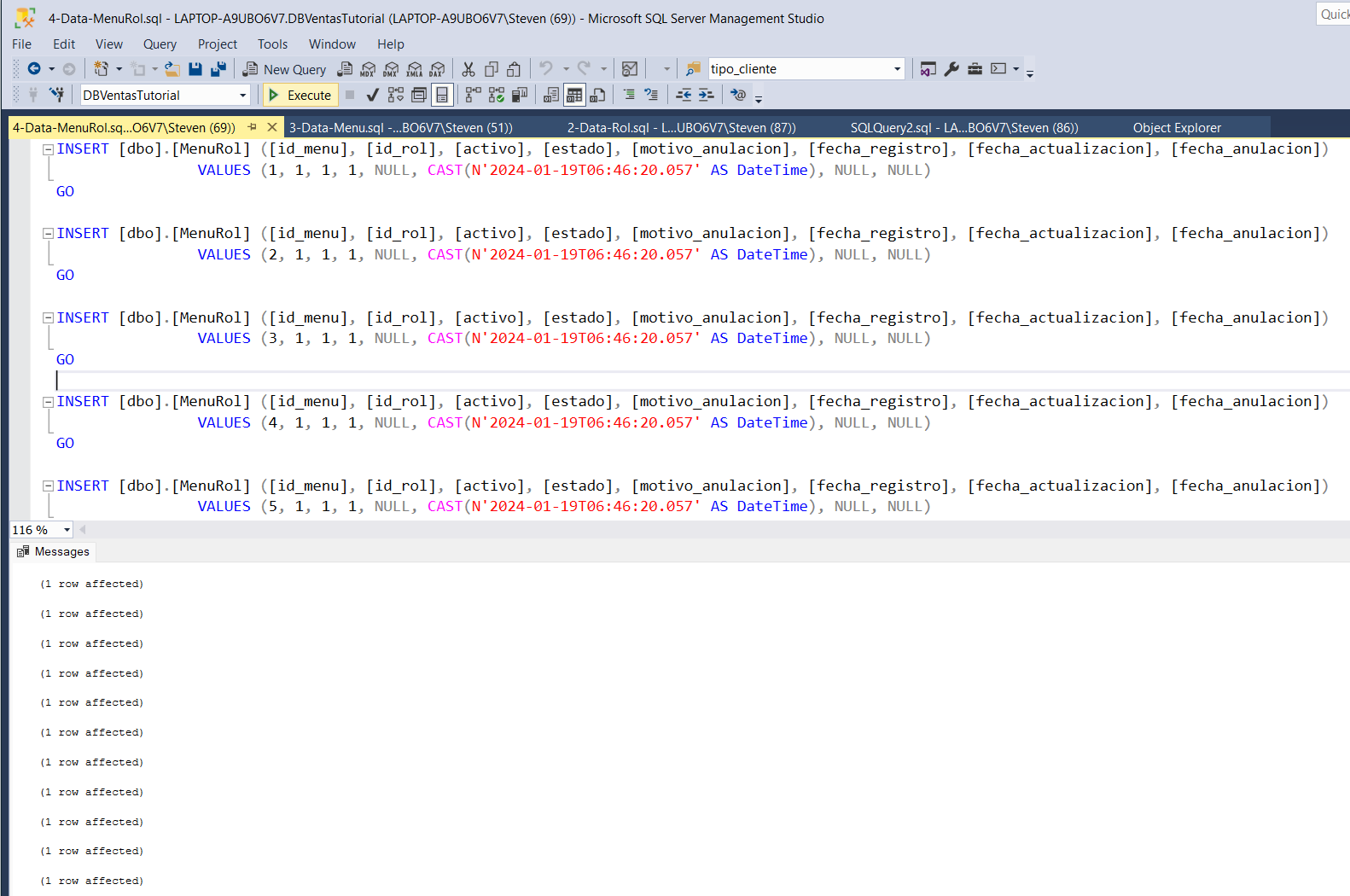
*Resultado esperado luego de la ejecución del primer script “****2-Data-Rol.sql****”.*

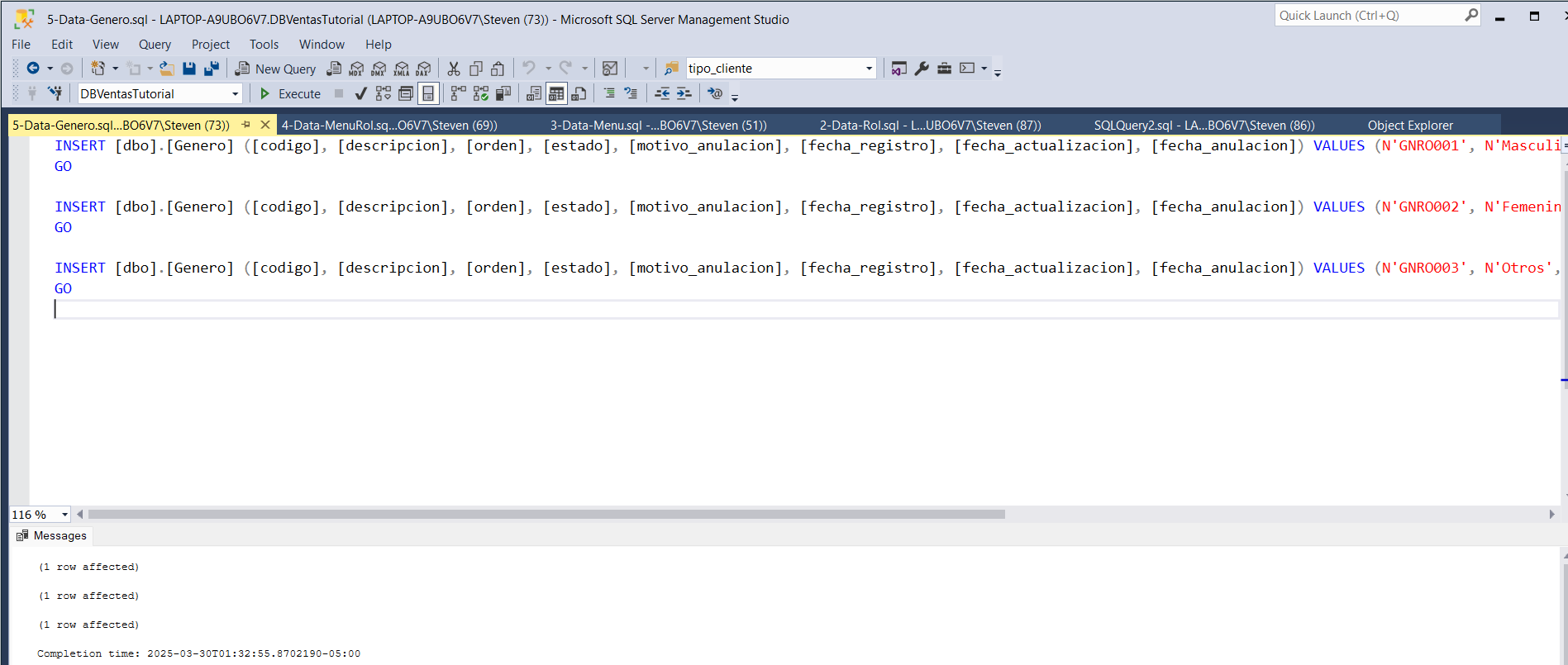
**

*Resultado esperado luego de la ejecución del primer script “****3-Data-Menu.sql****”.*

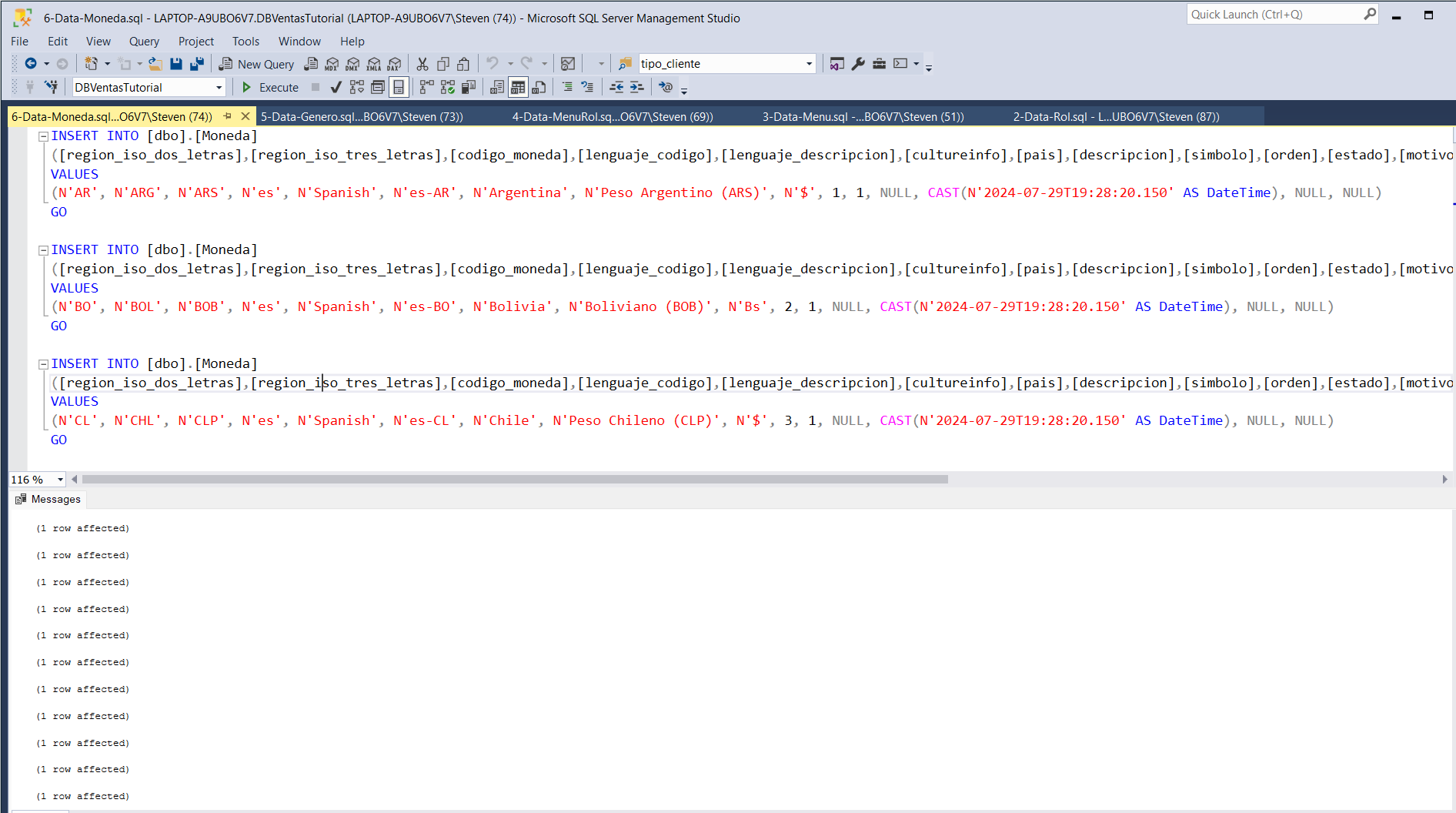


*Resultado esperado luego de la ejecución del primer script “****4-Data-MenuRol.sql****”.*

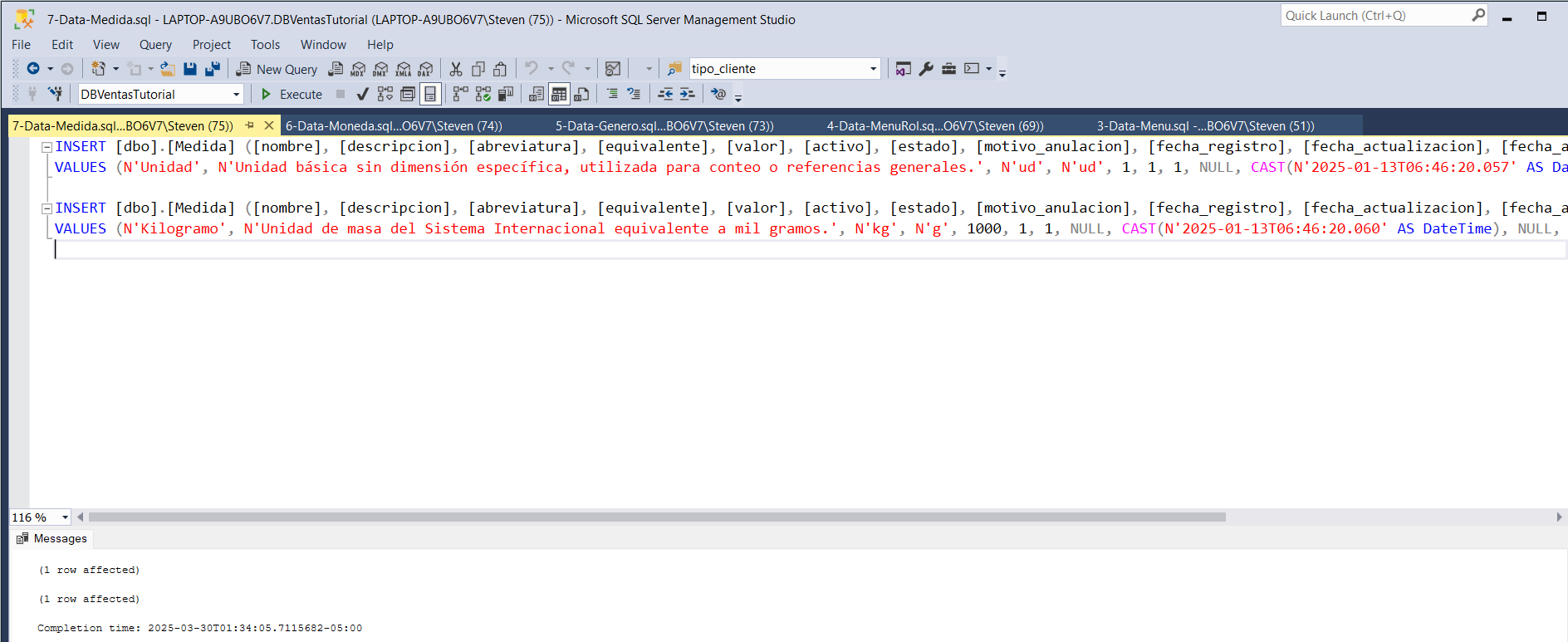


*Resultado esperado luego de la ejecución del primer script “****5-Data-Genero.sql****”.*

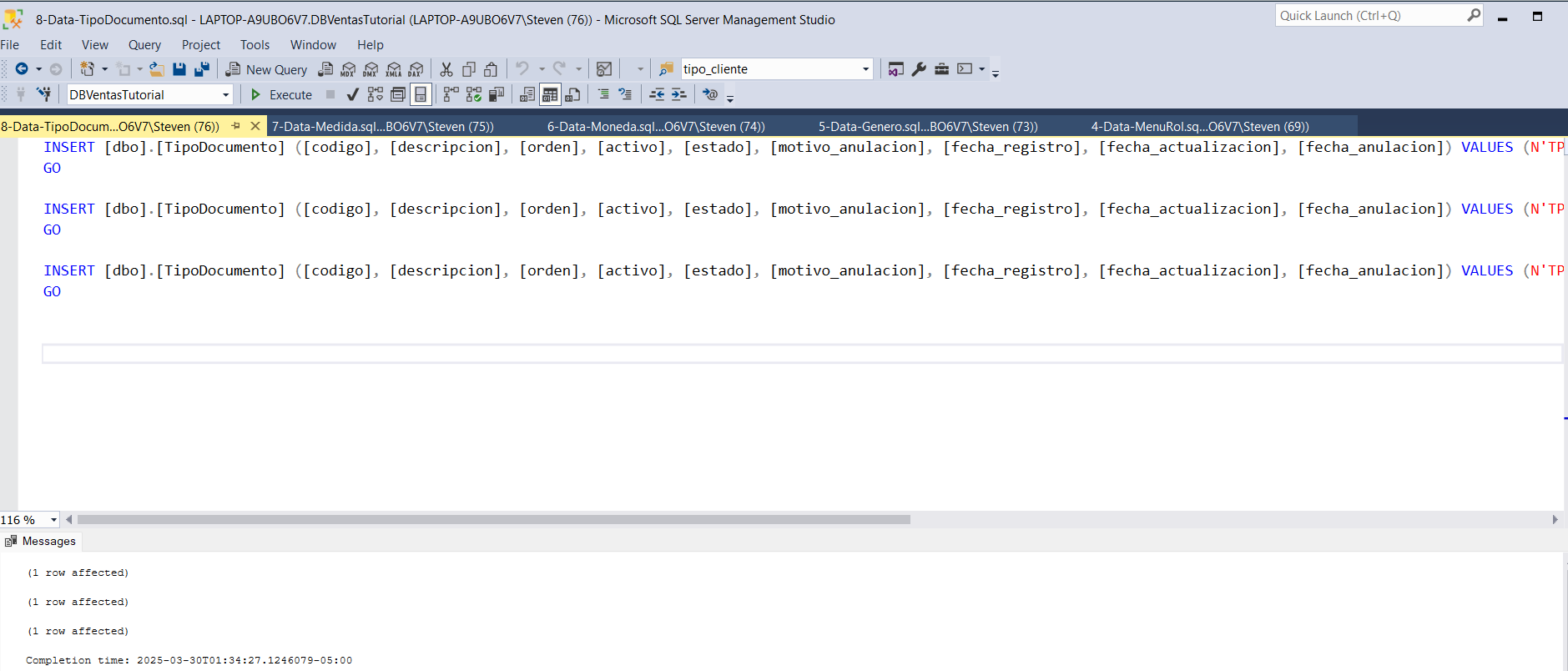
*Resultado esperado luego de la ejecución del primer script “****6-Data-Moneda.sql****”.*



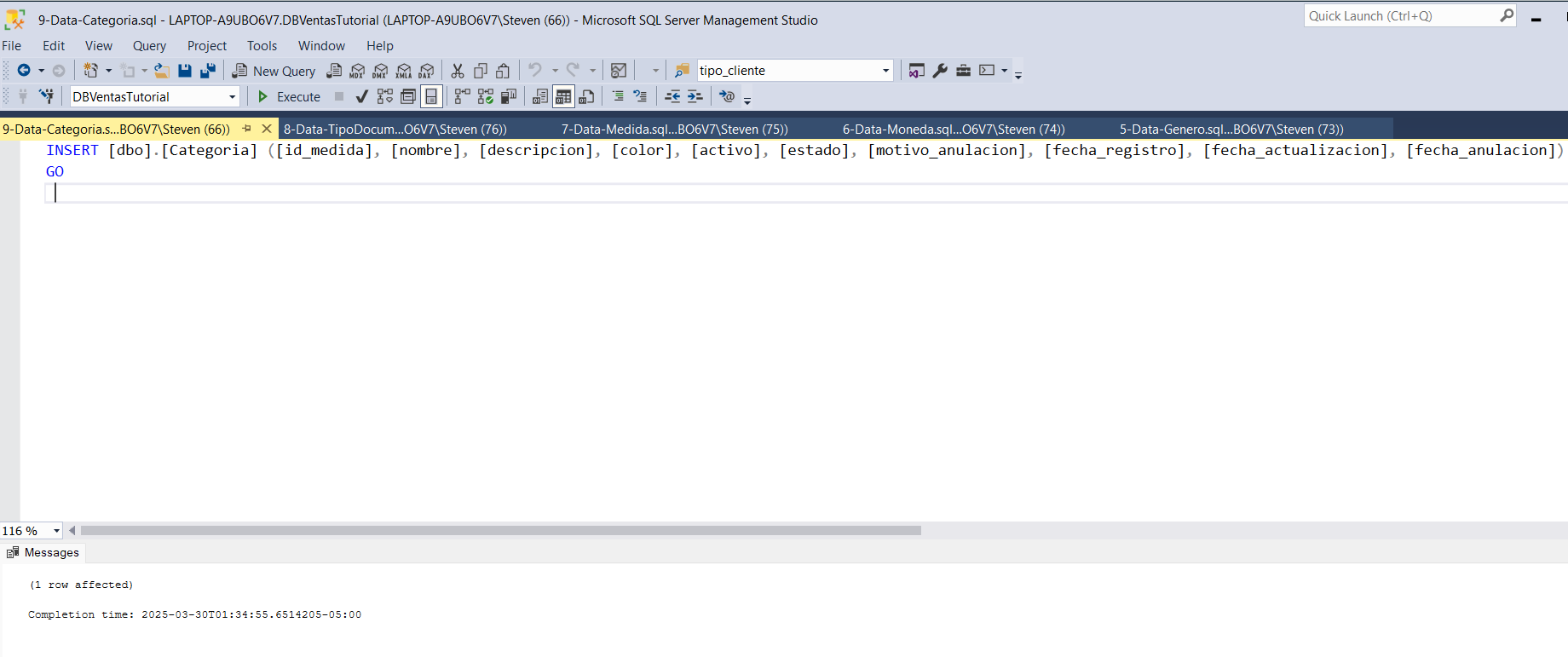
*Resultado esperado luego de la ejecución del primer script “****7-Data-Medida.sql****”.*

**

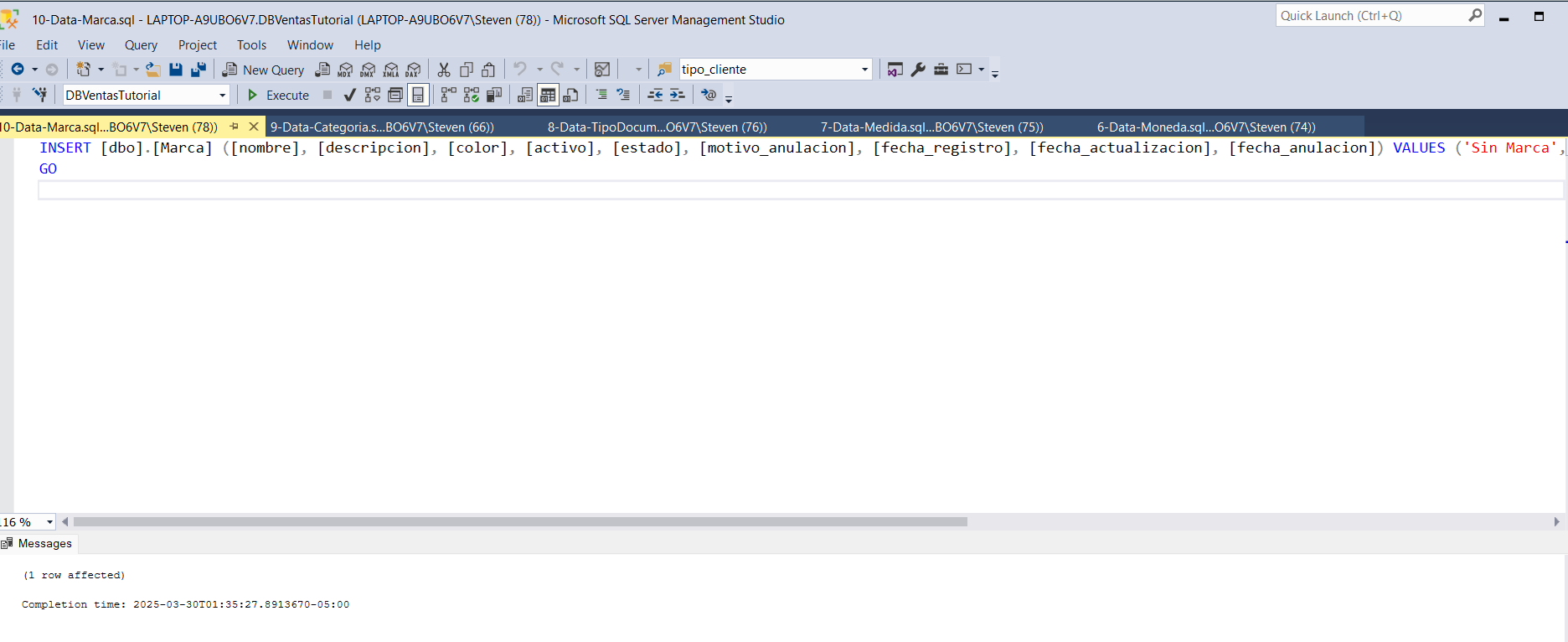
*Resultado esperado luego de la ejecución del primer script “****8-Data-TipoDocumento.sql****”.*

**

*Resultado esperado luego de la ejecución del primer script “****9-Data-Categoria.sql****”.*

**

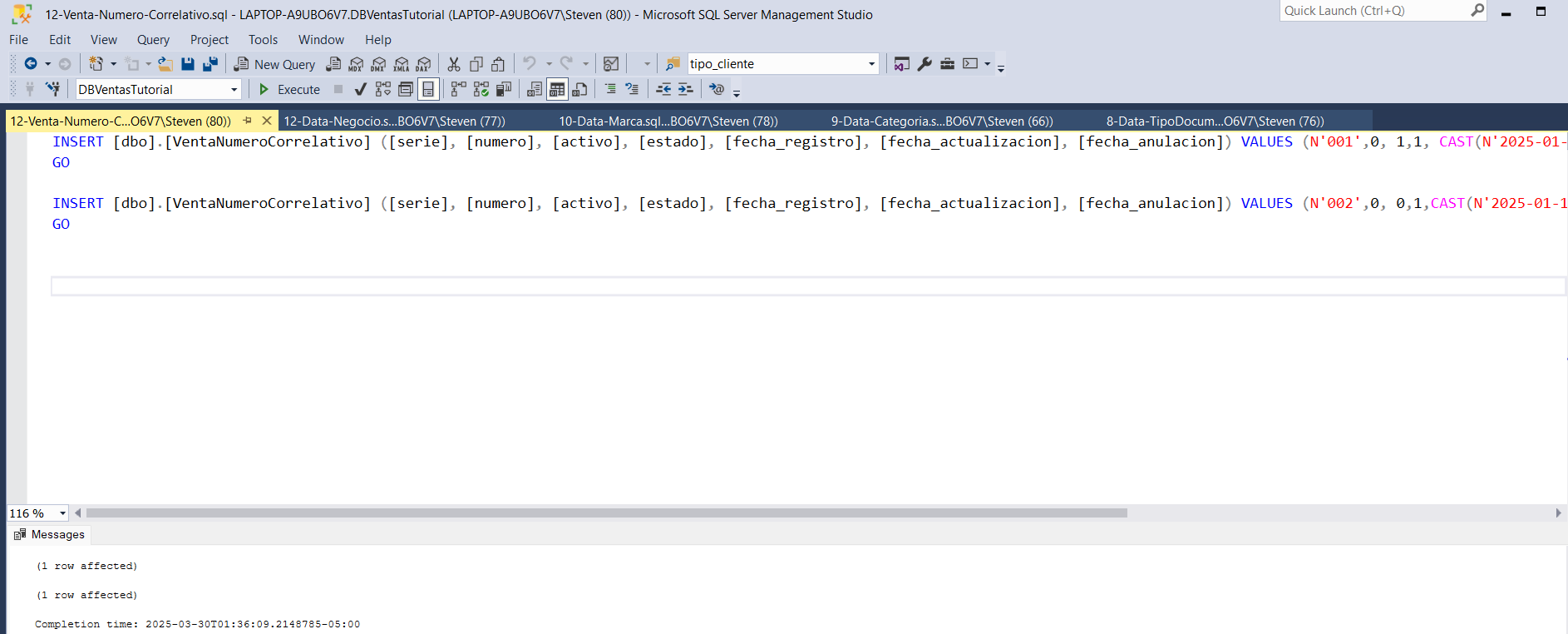
*Resultado esperado luego de la ejecución del primer script “****10-Data-Marca.sql****”.*



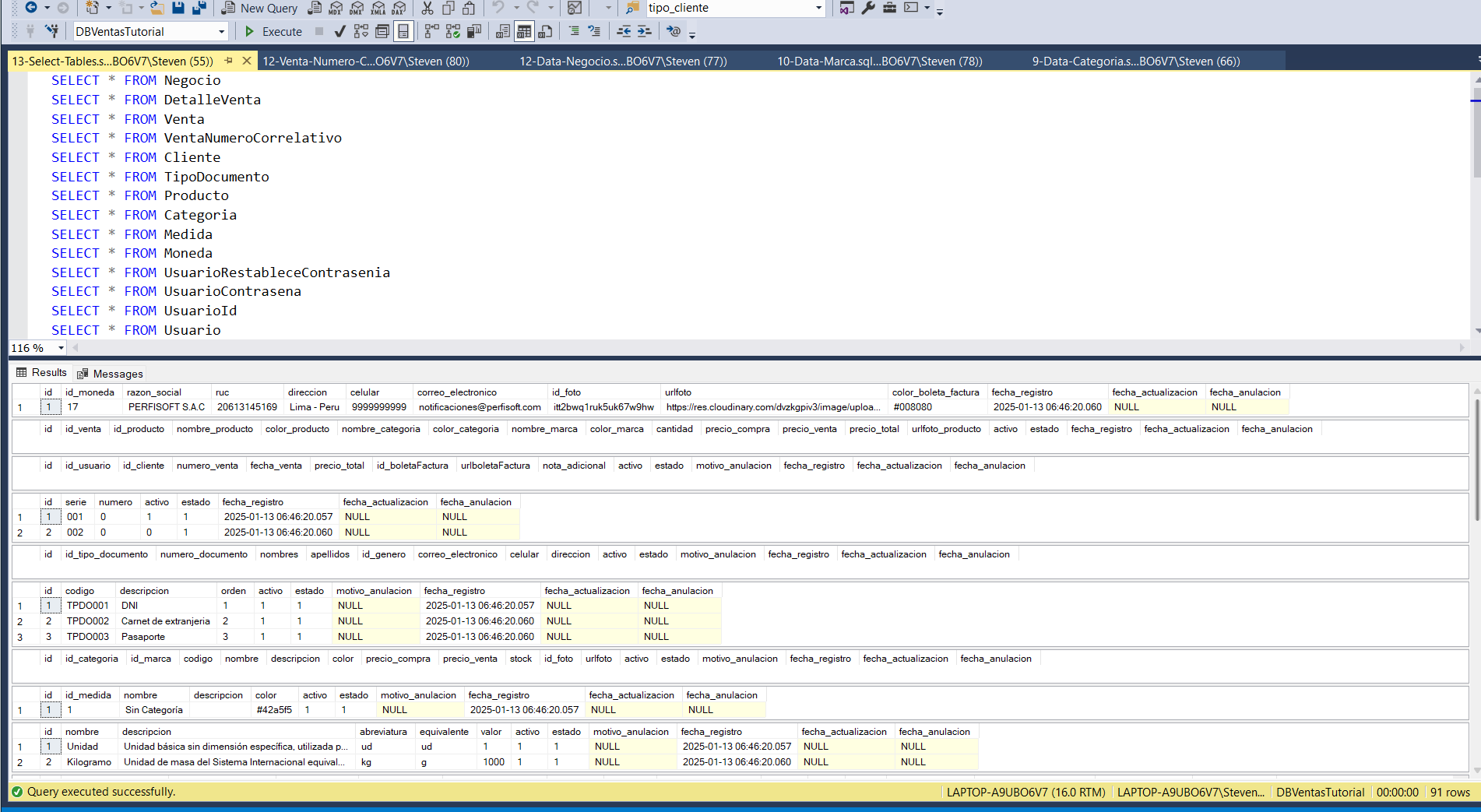
*Resultado esperado luego de la ejecución del primer script “****12-Data-Negocio.sql****”.*

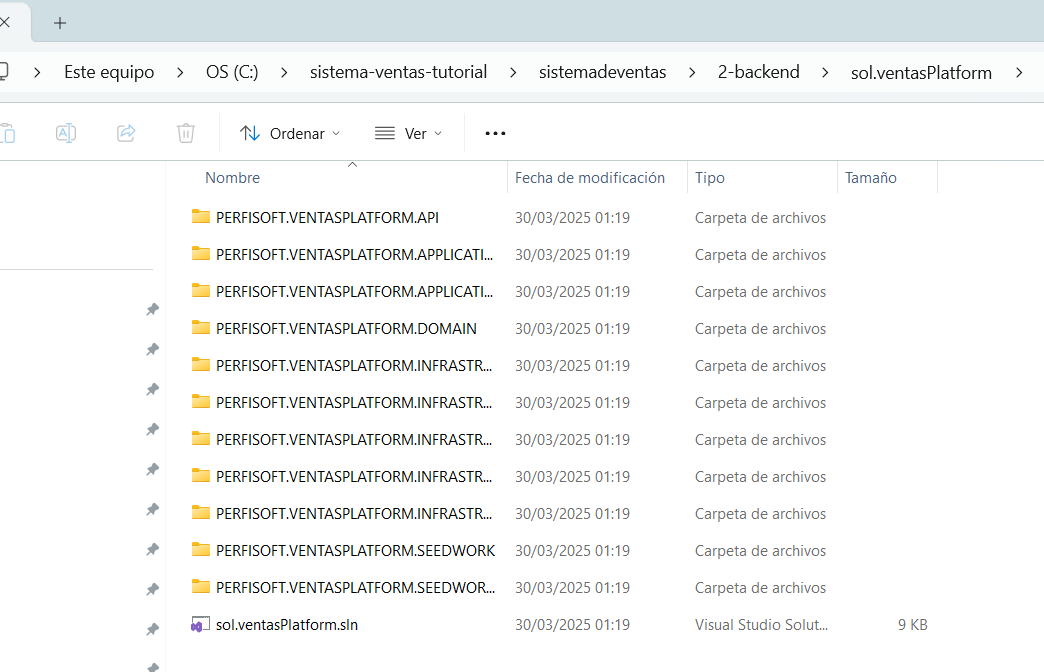


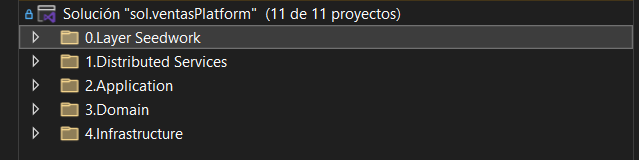
*Resultado esperado luego de la ejecución del primer script “****12-Venta-Numero-Correlativo.sql****”.*



*Resultado esperado luego de la ejecución del primer script “****13-Select-Tables.sql****”.*

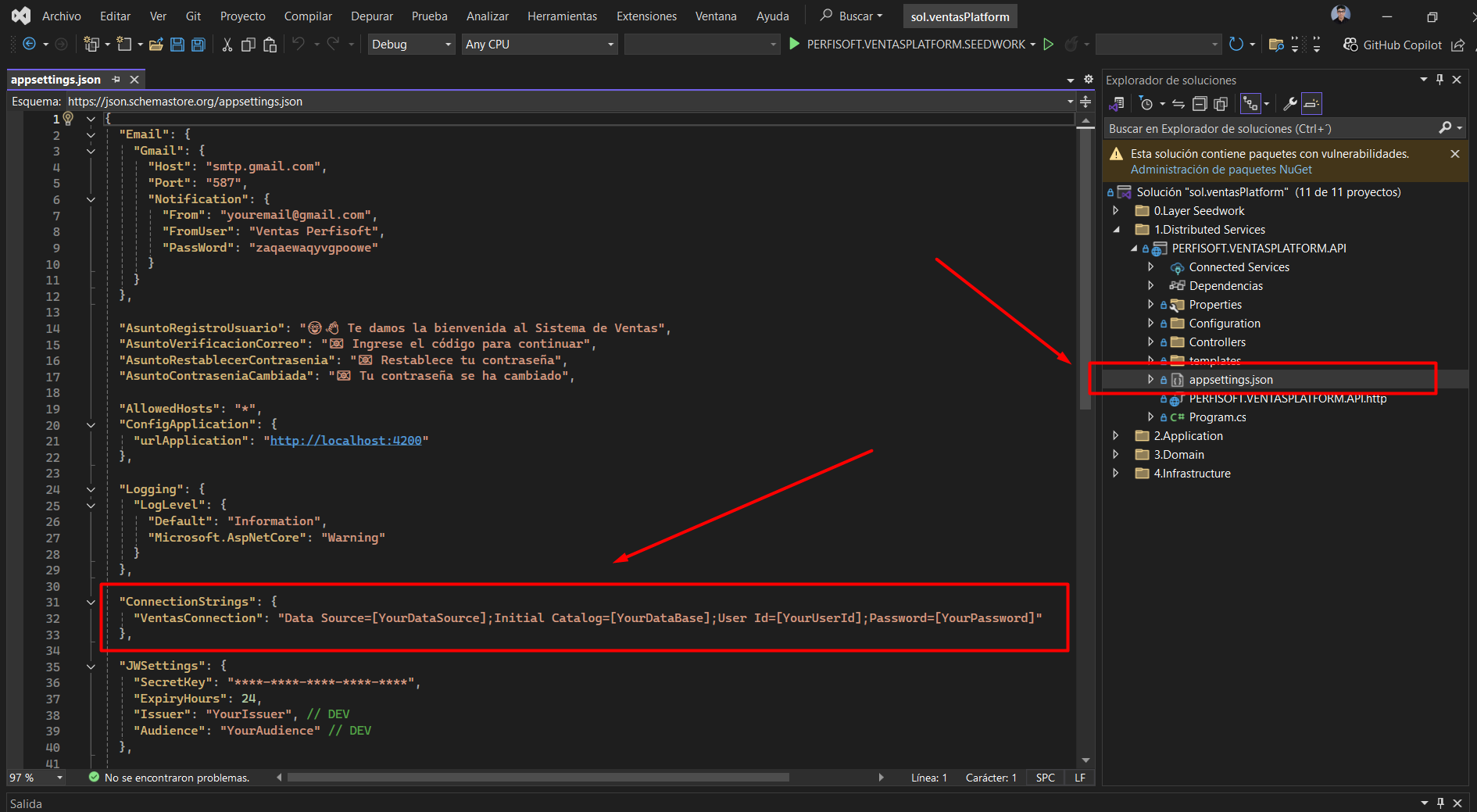


1. Luego de esto, ingresamos a la carpeta **"2-backend"** y hacemos doble clic en la solución **"sol.ventasPlatform.sln"**.

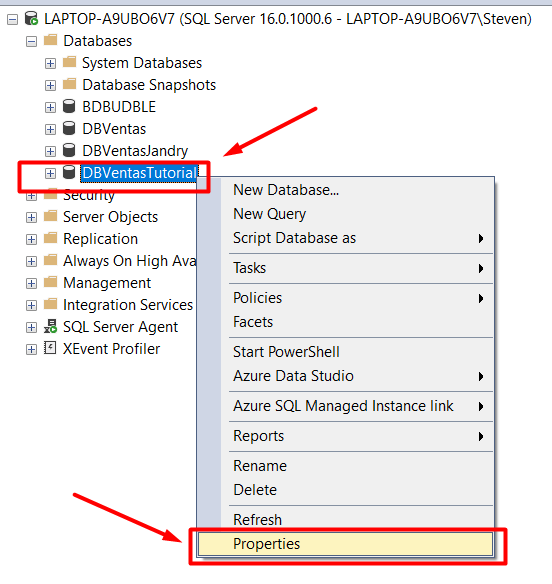
1. *La siguiente imagen muestra la estructura del Backend de la solución del sistema de ventas.*
2. Es hora de configurar nuestra cadena de conexión a la base de datos. Para ello hacemos lo siguiente:

1. Expande la carpeta **"Distributed Services"**.

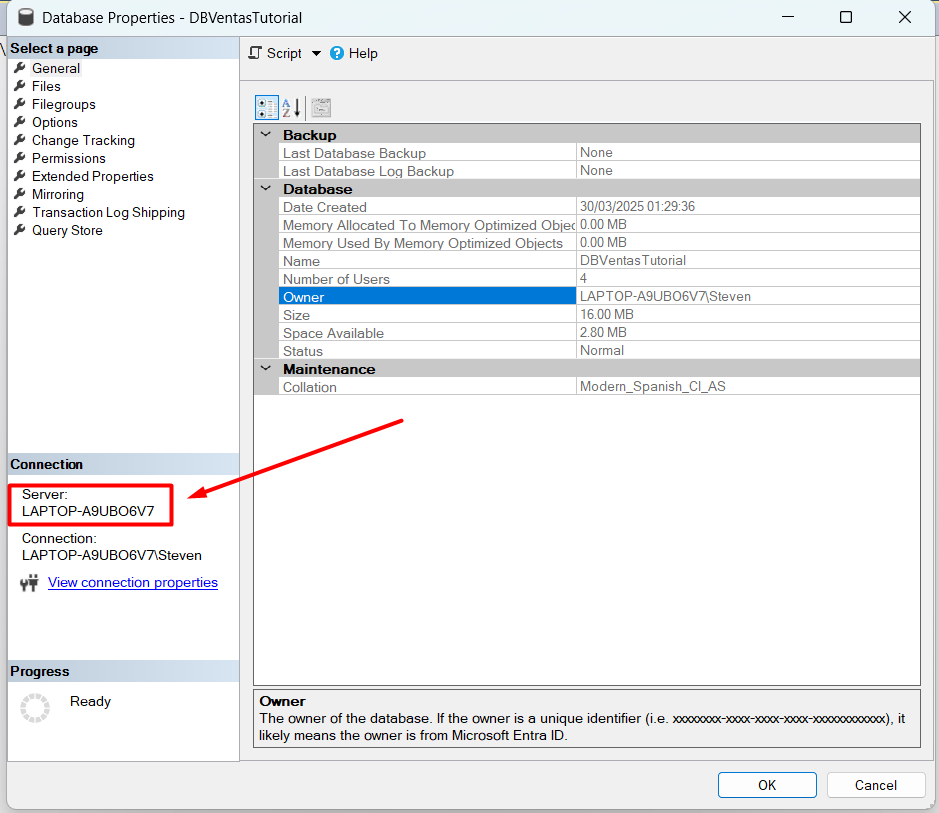
2. Expande **"API PERFISOFT.VENTASPLATFORM.API"**.

3. Selecciona el archivo **"appsettings.json"**.

1. Para configurar correctamente la cadena de conexión, ubicamos nuestra base de datos creada en SQL, hacemos clic derecho sobre ella y seleccionamos **"Properties"**.

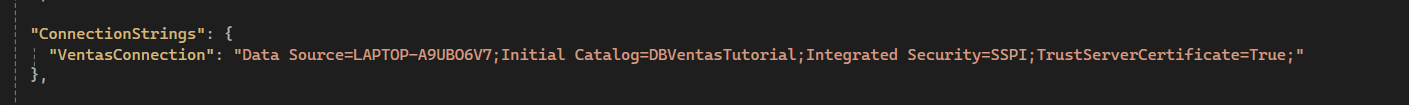


1. Ubicamos la etiqueta **"Server"** y copiamos su valor. En mi caso es **LAPTOP-A9UBO6V7.**

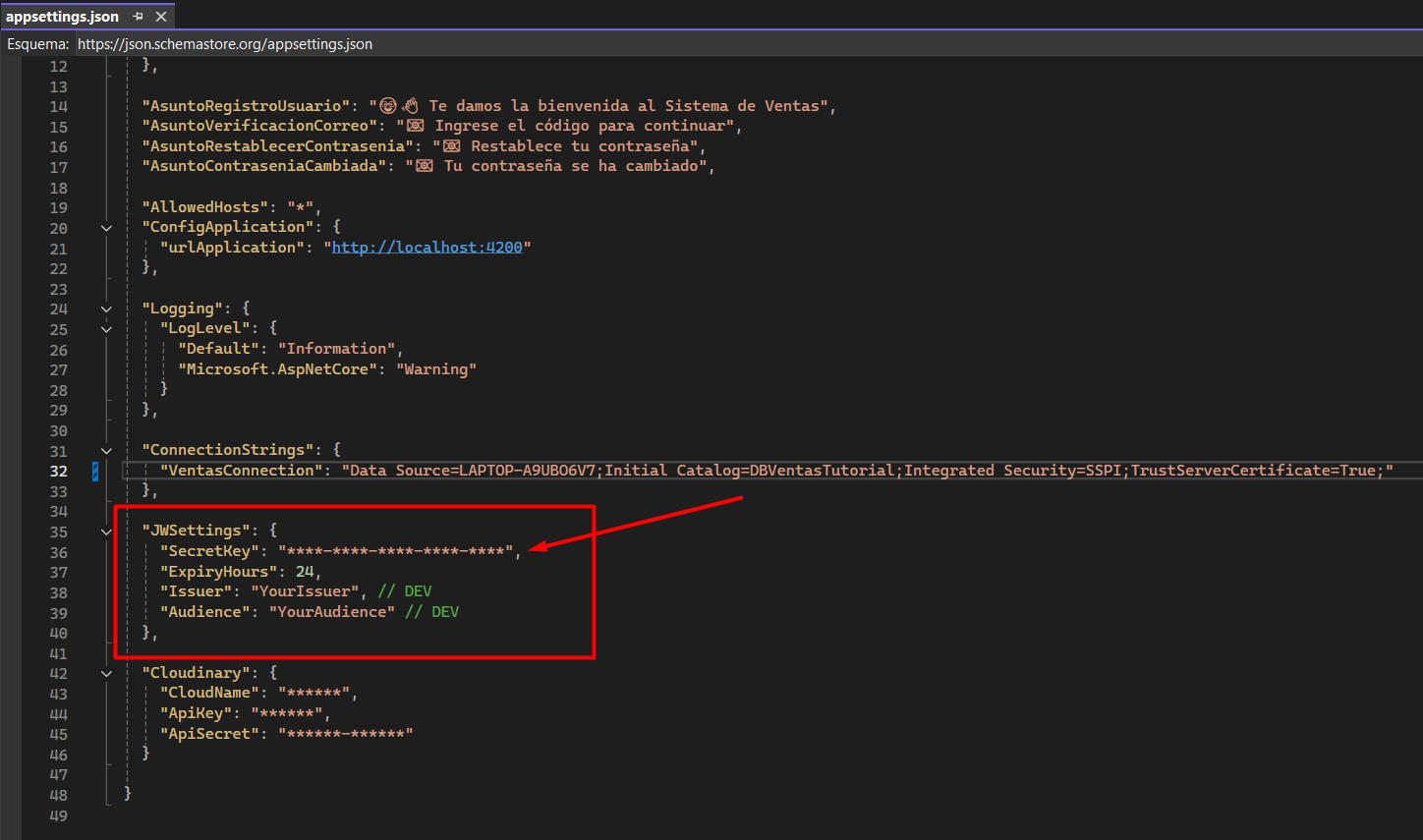


1. Configuramos los valores de la siguiente manera:
2. **Data Source:** Ingresa el valor del servidor que copiamos anteriormente.
3. **Initial Catalog:** Ingresa el nombre de la base de datos. En este ejemplo, es **"DBVentasTutorial"**.
4. Si tu base de datos requiere **usuario y contraseña**, agrégalos en los campos correspondientes **[YourUserId]** and **[YourPassword].** En este caso, como la ejecución será local, usa la siguiente configuración:

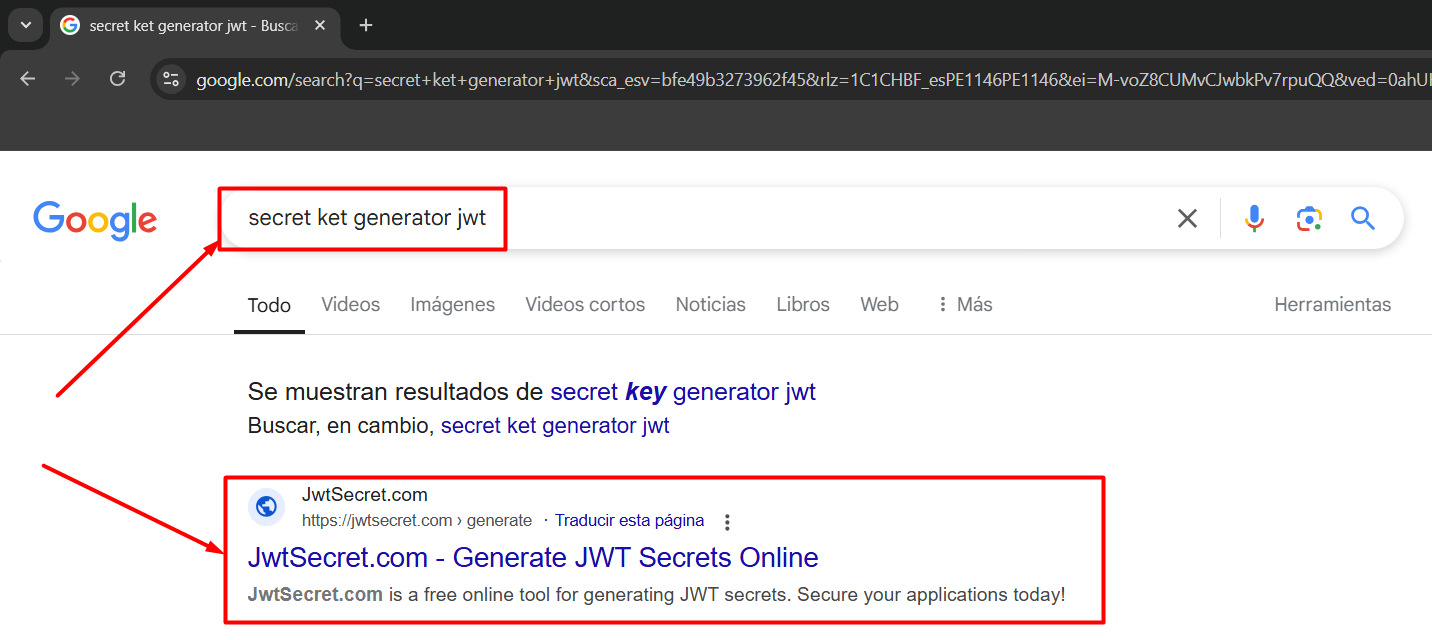
**Integrated Security=SSPI; TrustServerCertificate=True;**

*La siguiente imagen muestra cómo debería estar configurado nuestra cadena de conexión* ***en un entorno local***

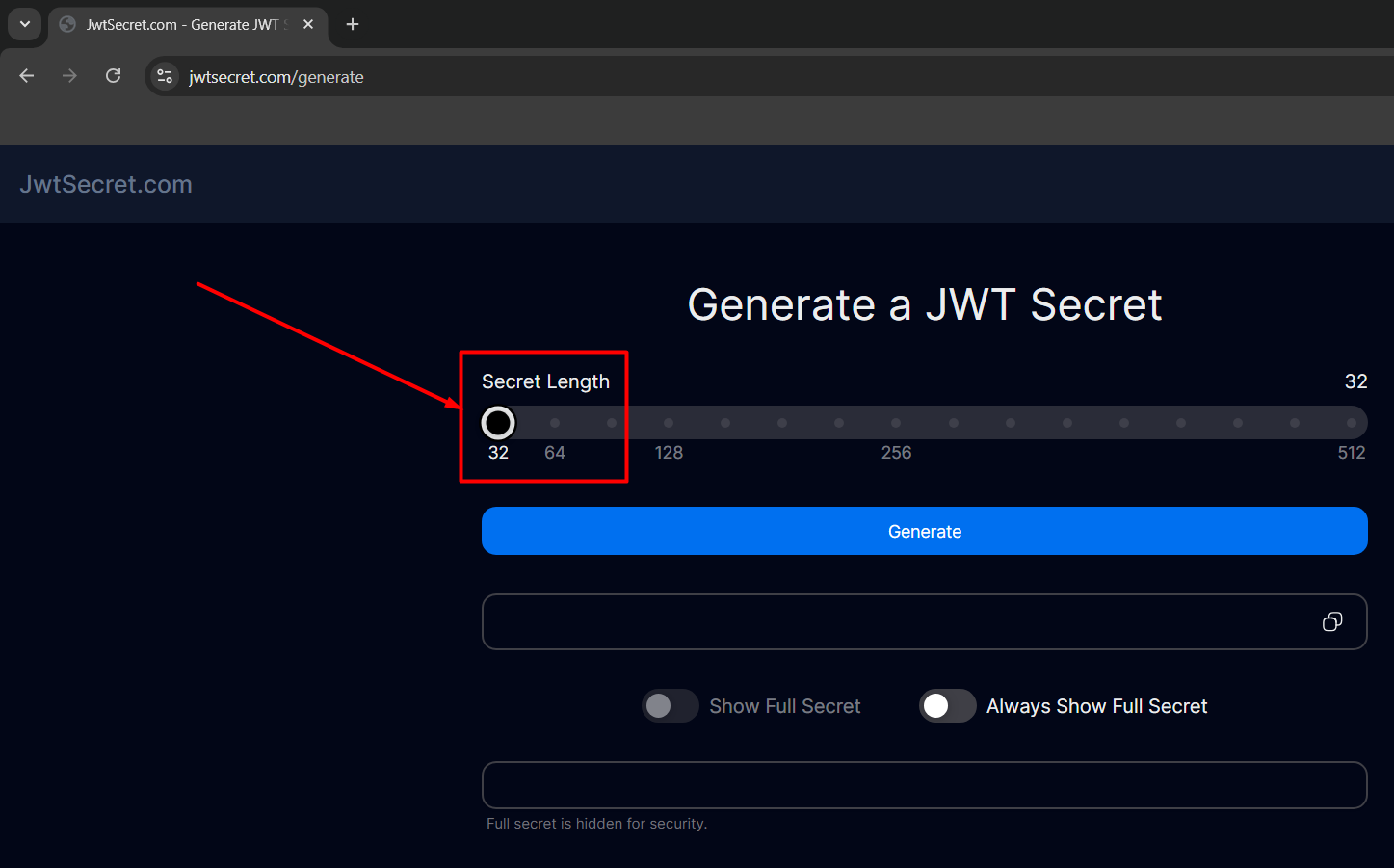
1. Ahora configuraremos el token generando nuestra **SecretKey**.



1. Ingresamos a nuestro navegador favorito y escribimos “**Secret Key Generator JWT**”. El enlace directo es <https://jwtsecret.com/generate>.



1. Movemos el botón a **32** de “**Secret Length**”.

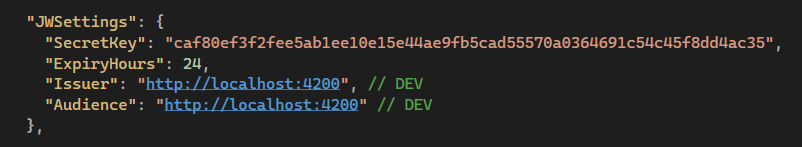


1. Presionamos el Botón “**Generate**” y copiamos el valor generado.

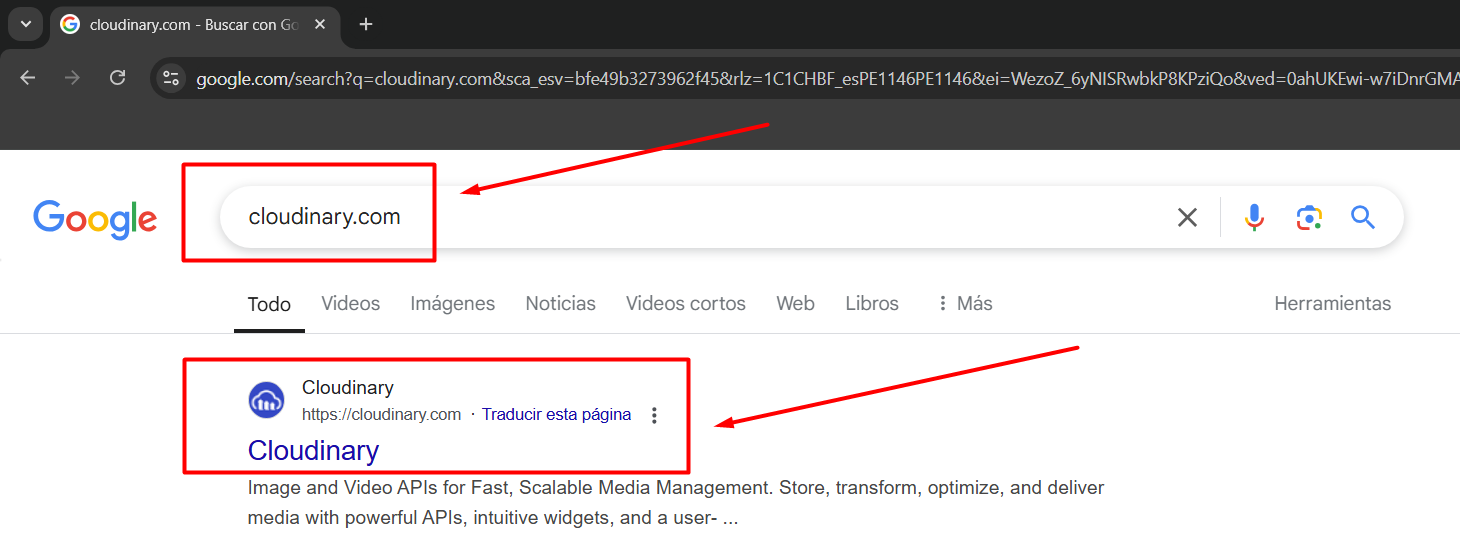


1. Pegamos el valor copiado en la propiedad “**SecretKey**”, como se muestra en la siguiente imagen

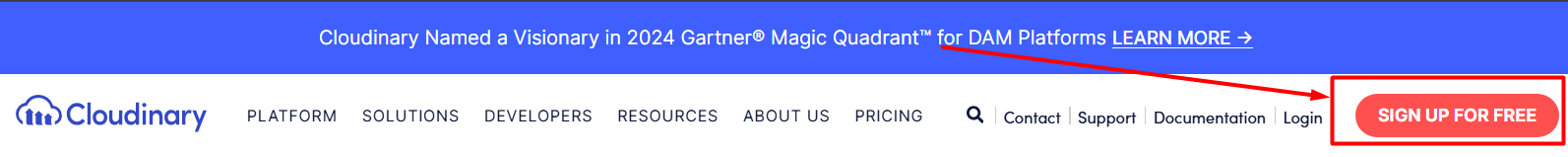
* **ExpiryHours** = Horas en las que expira el token; el usuario deberá volver a iniciar sesión.
* **Issuer** = El emisor, en este caso, se establece como **"**[**http://localhost:4200**](http://localhost:4200)**"**.
* **Audience** = Audiencia, también establecida como **"**[**http://localhost:4200**](http://localhost:4200)**"**.

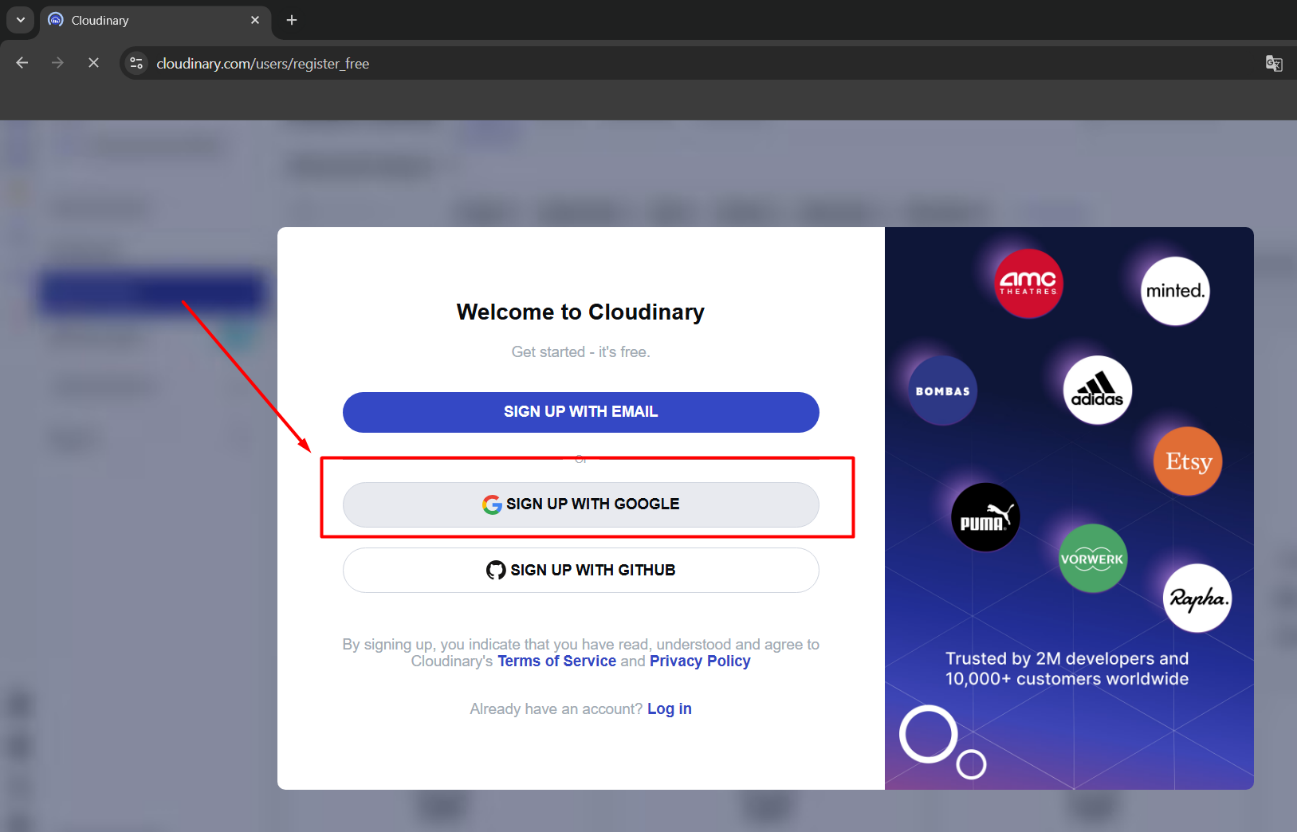
*(****Establecer estos valores son cruciales para poder correr el sistema****)*

1. Ahora necesitaremos crear una cuenta en **Cloudinary** para almacenar las fotos de los productos y las boletas de ventas generadas al realizar una venta. Para ello, ingresamos a nuestro navegador favorito y buscamos **Cloudinary**.

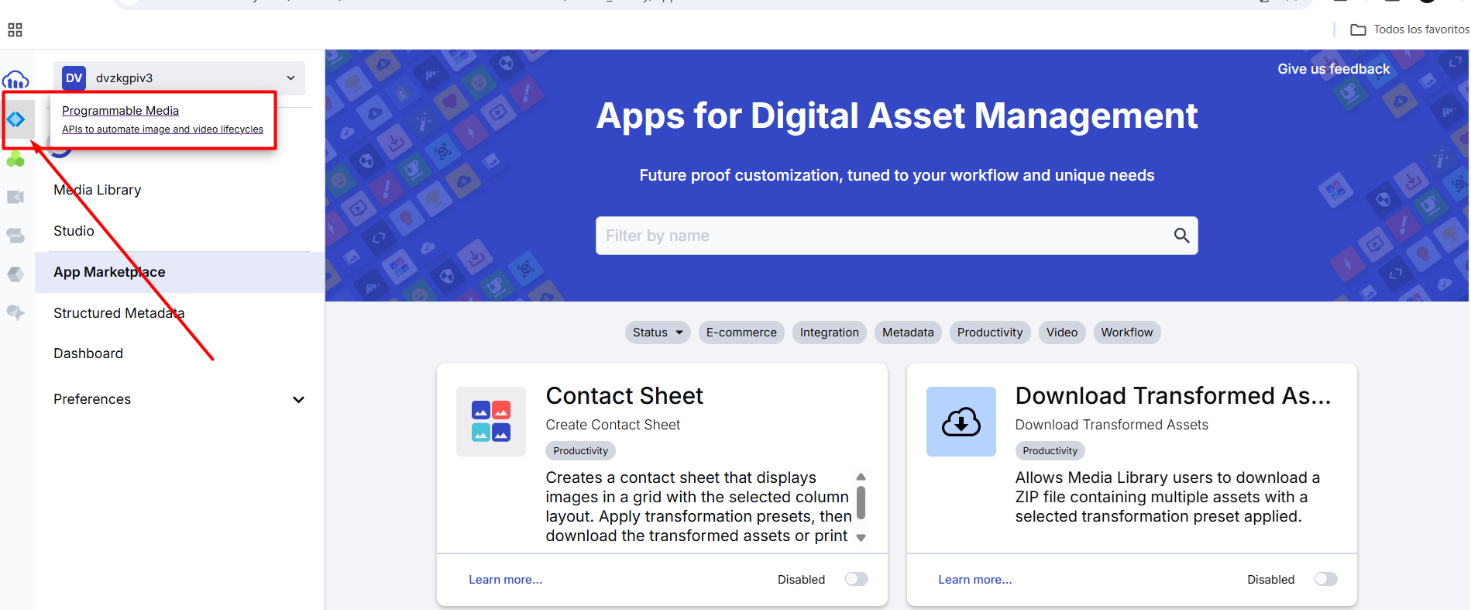
Enlace Directo: <https://cloudinary.com>

1. Nos creamos una nueva cuenta presionando en el botón “**SIGN UP FOR FREE**”

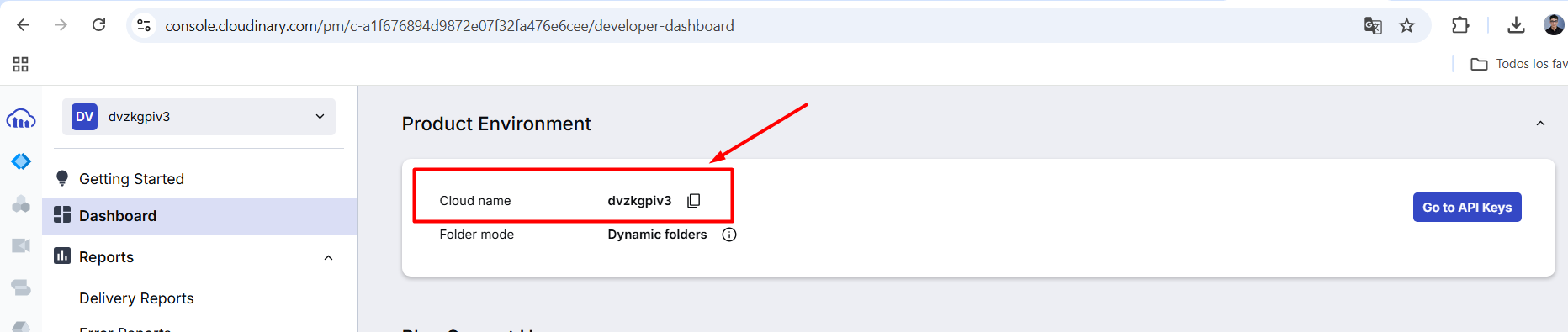




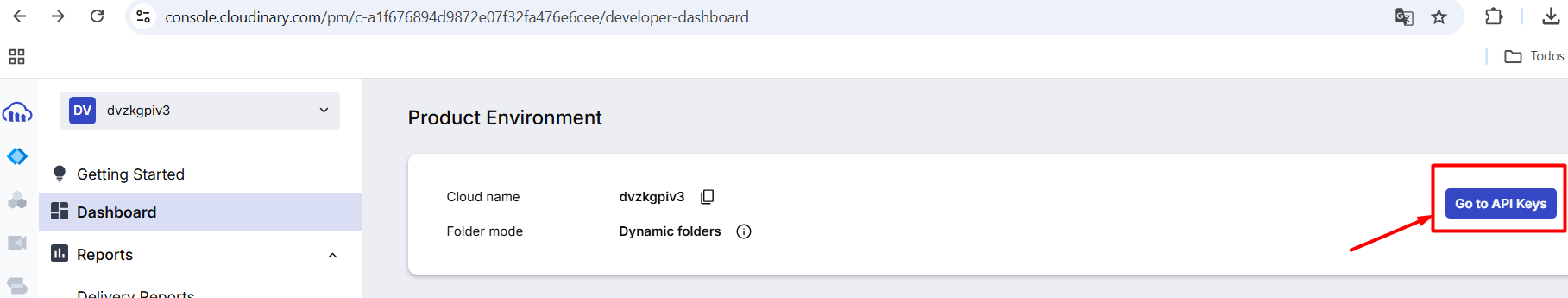
1. Después de registrarnos, nos dirigimos al menú de la izquierda y hacemos clic en **"Programmable Media"**.



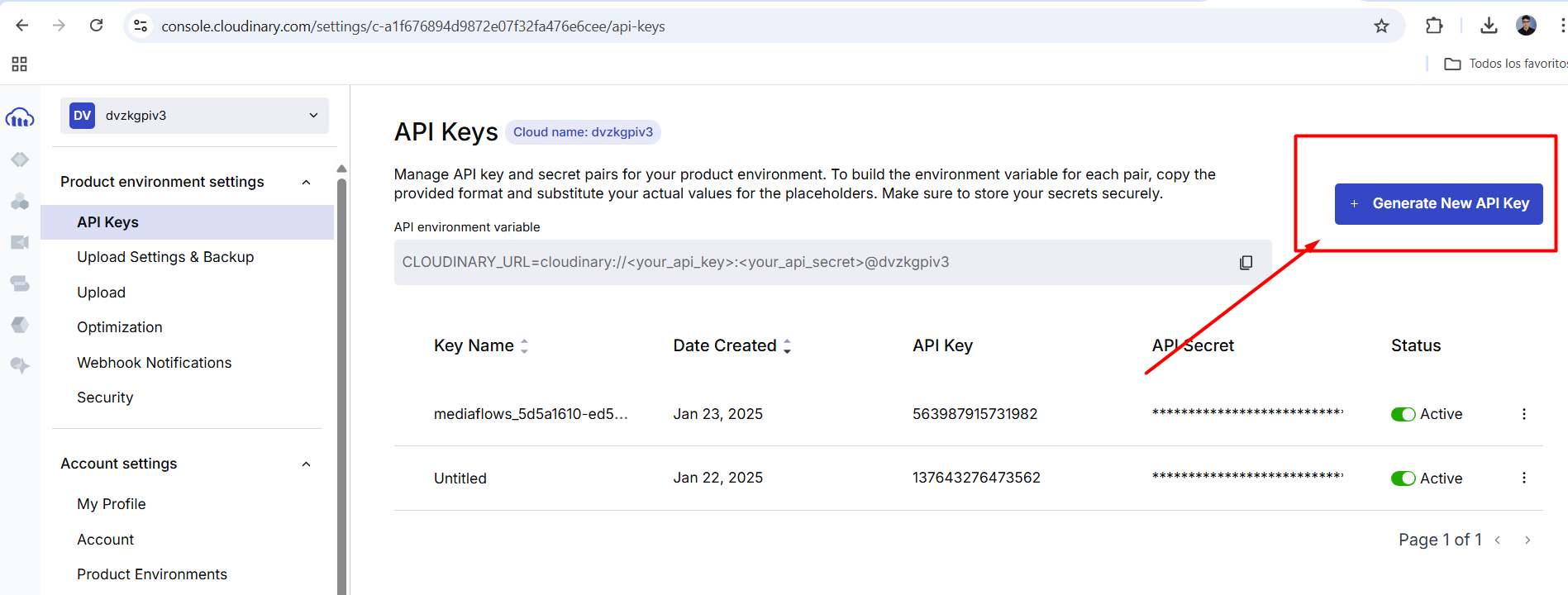
1. Copiamos el valor de la etiqueta “**Cloud Name**” en mi caso es “**dvzkgpiv3**”.



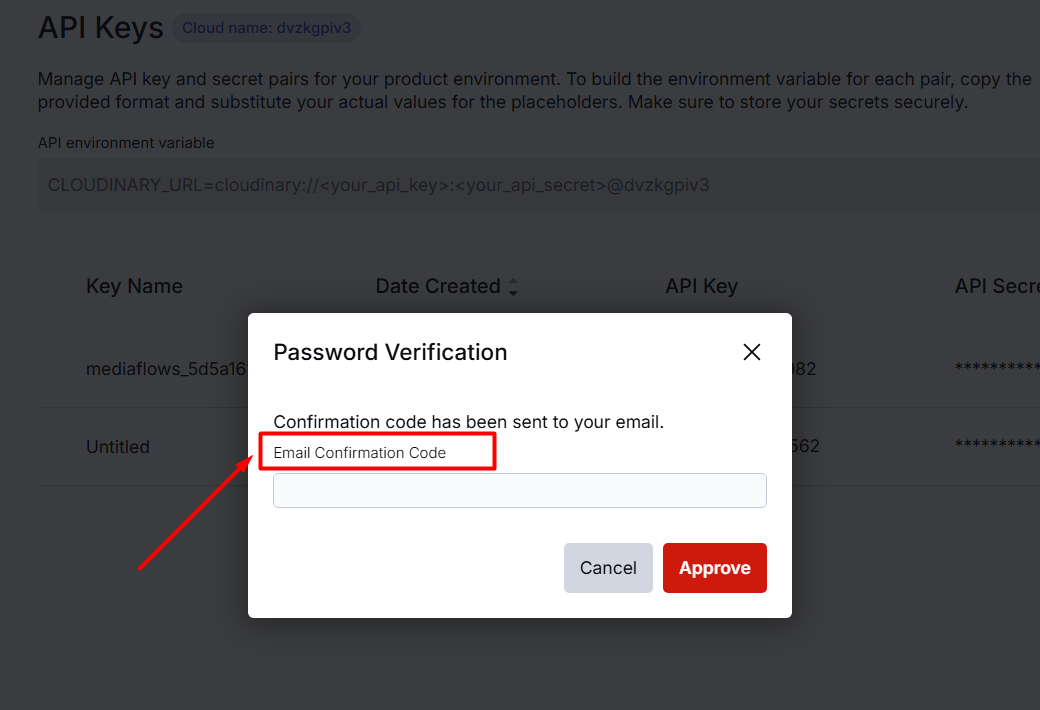
1. Luego damos click al botón donde dice “**Go to API Keys**”



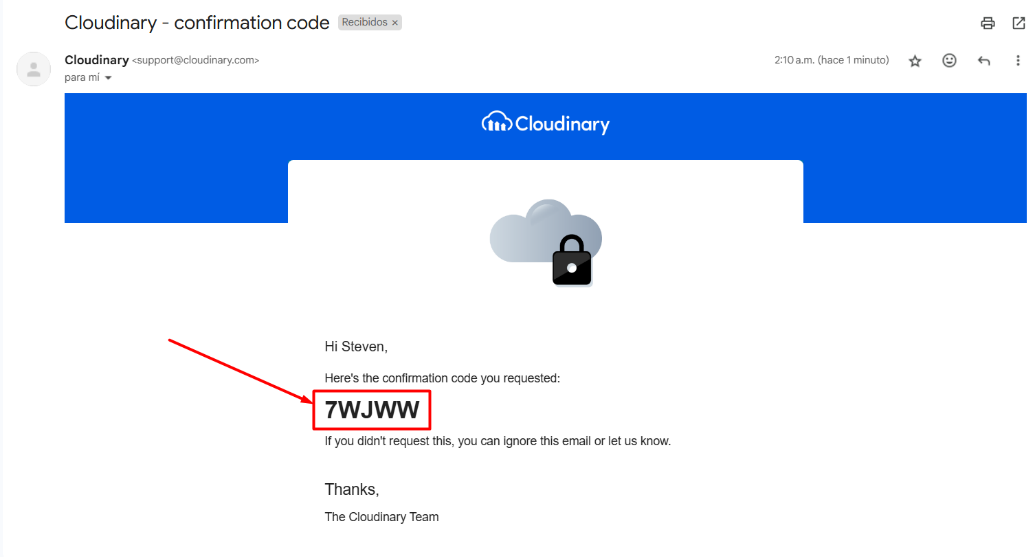
1. Luego damos click al botón donde dice “**Generate New API Key**”



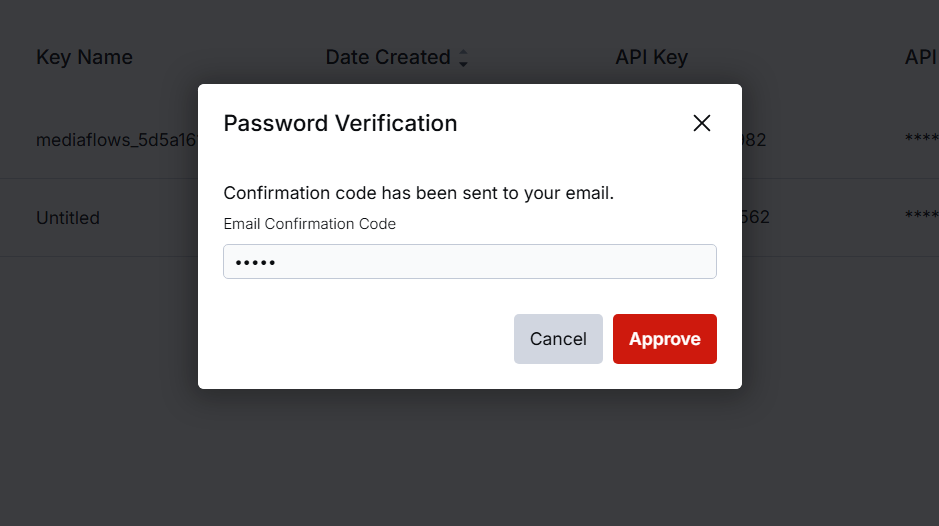
1. Revisamos nuestra bandeja de correo electrónico para obtener el código de confirmación.



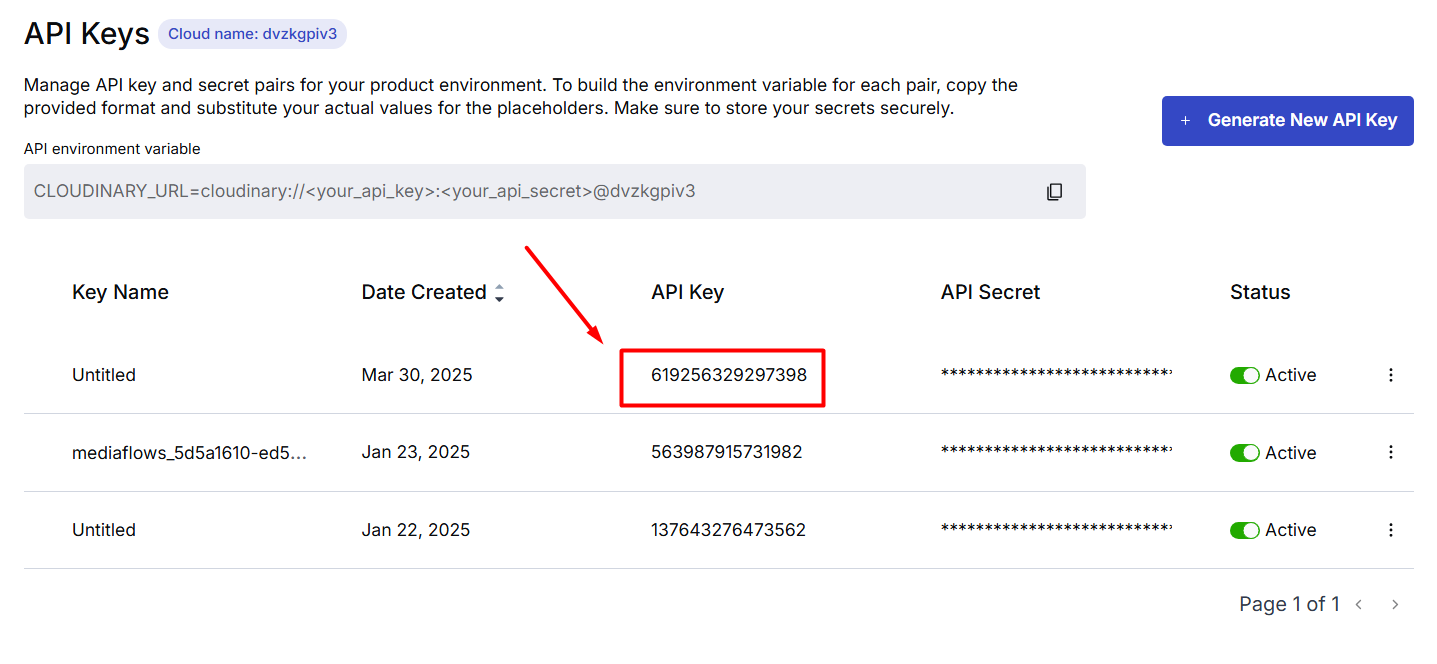
1. *La siguiente imagen muestra el código recibido, el cual debe ingresarse en el cuadro de confirmación*



1. Ingresamos el código y presionamos en “**Approve**”.



1. En la columna **"API KEY"** encontraremos el valor que agregaremos en nuestro **appsettings.json**.

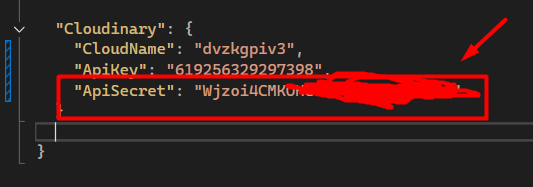
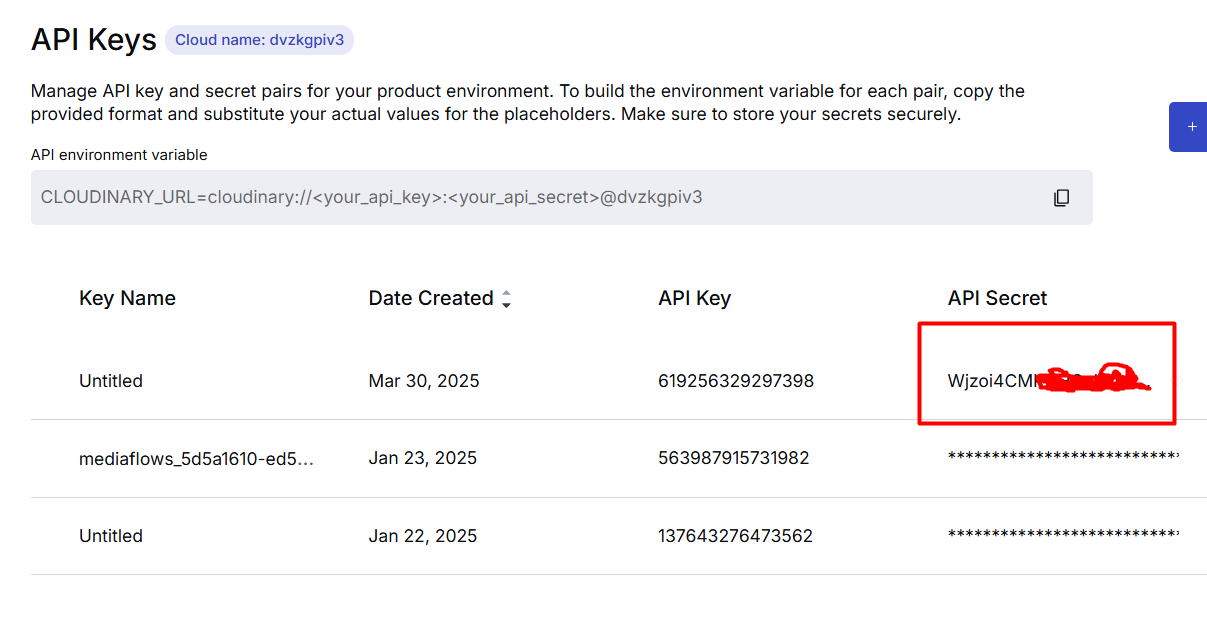


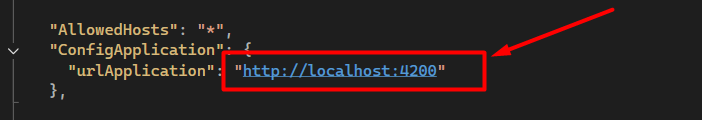
*La siguiente imagen muestra cómo debería quedar nuestro valor del atributo “****ApiKey****” y nuestro “****CloudName****”*

1. Damos clic en el icono del ojito para visualizar el “**API SECRET**”

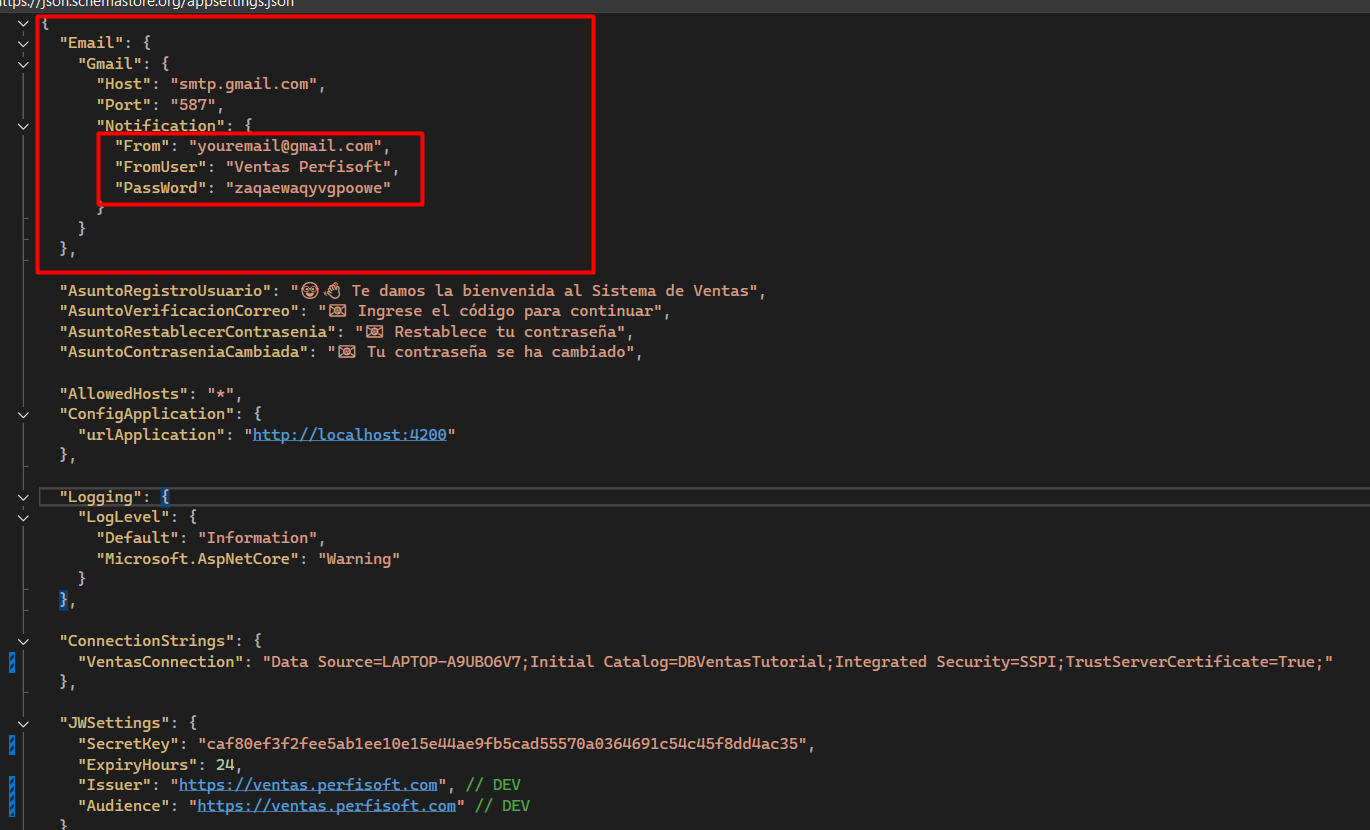


La siguiente imagen muestra el valor que deberíamos copiar y pegar en nuestro “**ApiSecret**” del

******appsettings.json**

1. El valor de la etiqueta **"urlApplication"** se utilizará como URL en funcionalidades como **"Olvidé mi contraseña"** y **"Bienvenida al nuevo usuario"**. Este valor debe actualizarse cada vez que el dominio cambie desde el frontend.
2. Configuración de la sección de Email:

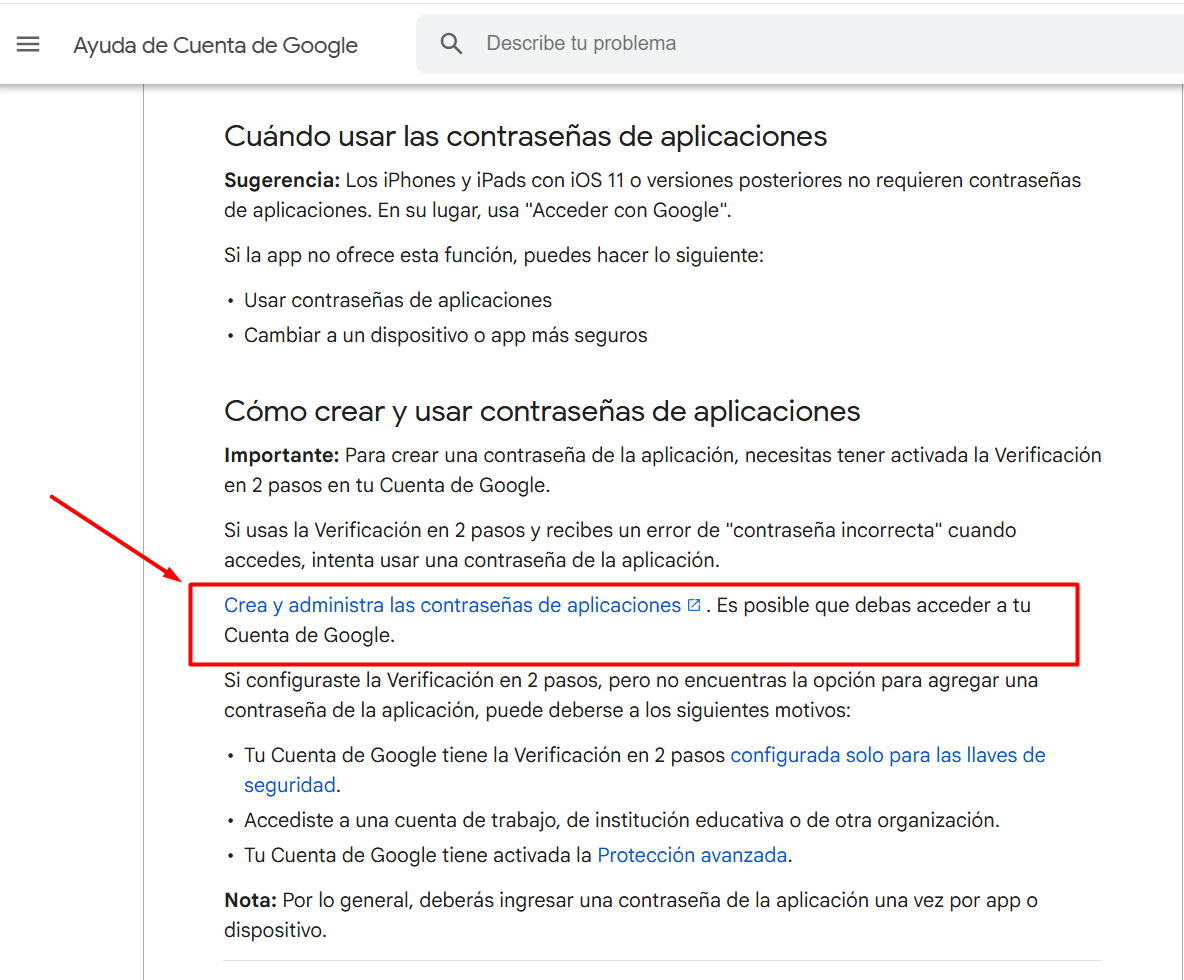
Esta configuración permitirá el envío de correos desde funcionalidades como:

* **Registro de usuario:** Se envía un correo automático al nuevo usuario desde esta dirección de correo.
* **Olvidé mi contraseña:** Cuando el usuario desea recuperar su contraseña, se le envía un enlace con un token temporal. El correo remitente será el que asignemos aquí, y la URL corresponderá al valor de **"urlApplication"** mencionado en el punto anterior.

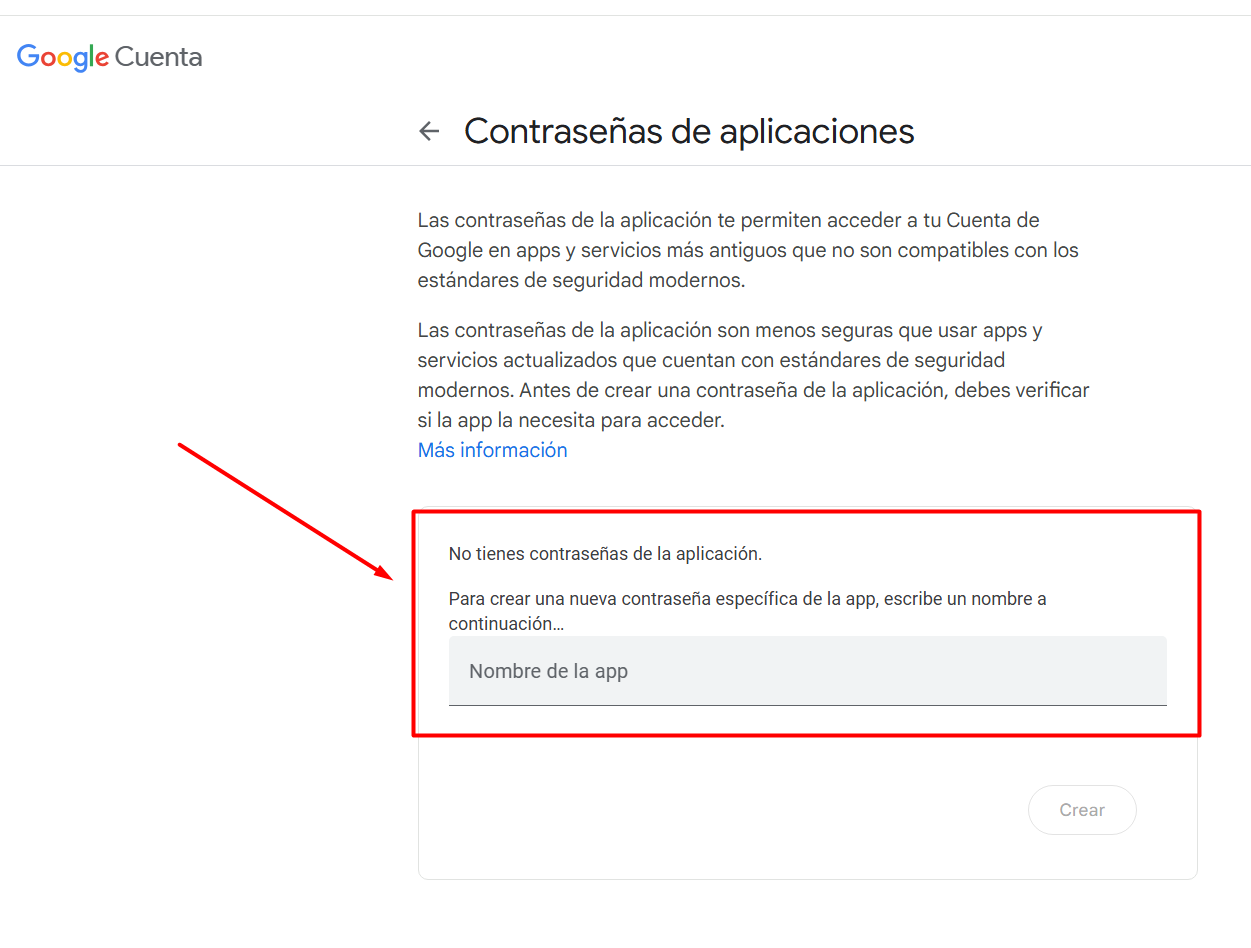
1. Para crear una contraseña específica para nuestra aplicación y enviar correos sin exponer tu contraseña principal, abre tu navegador, busca **"Contraseña de aplicaciones Google"** y selecciona la opción indicada en la imagen.

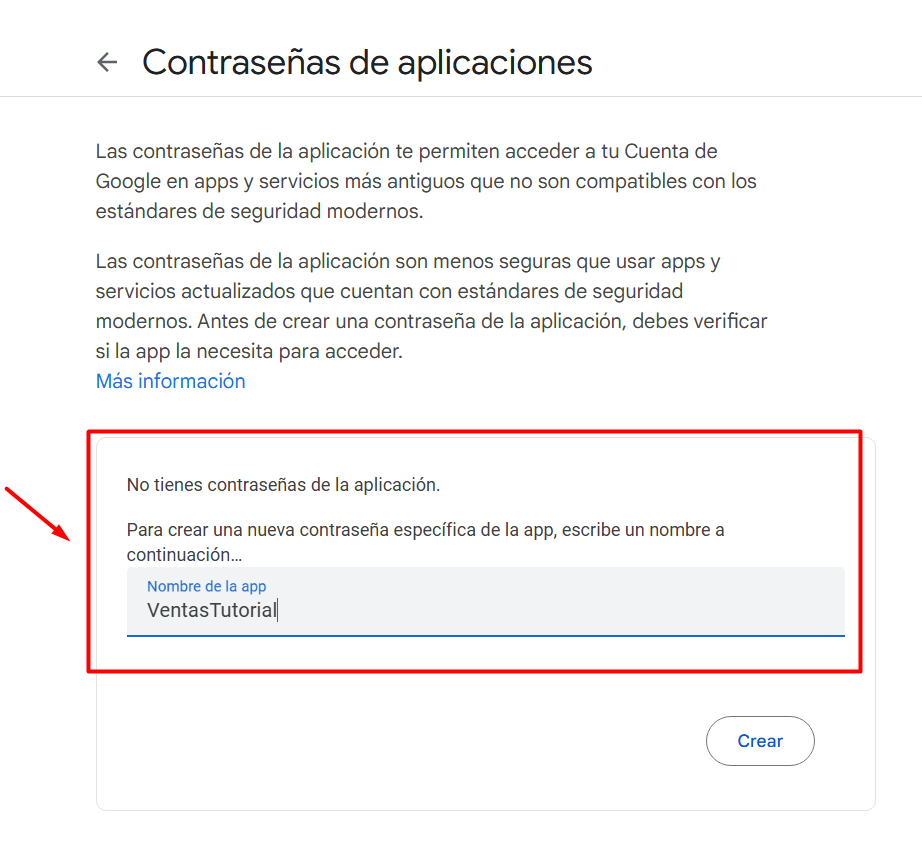
|

1. Nos ubicamos en la opción "**Crea y administra las contraseñas de aplicaciones**" y damos clic. (***Recuerda que debes estar logueado con tu correo Gmail)***



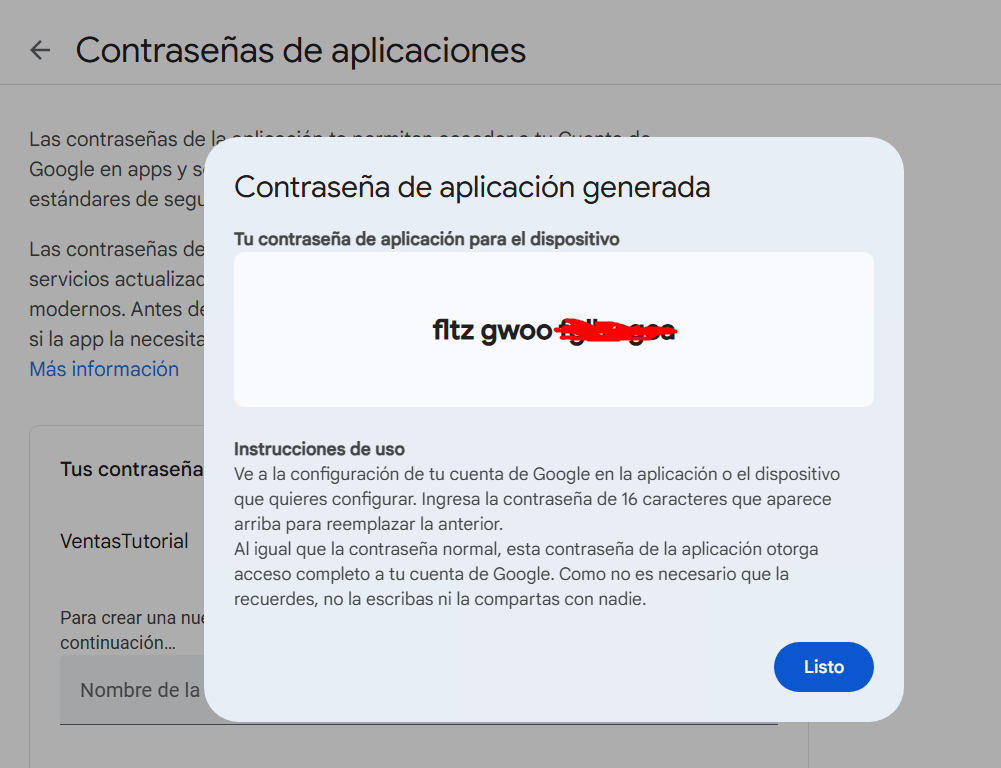
1. Asignamos un nombre de aplicación en el campo “**Nombre de la app**” y presionamos en crear

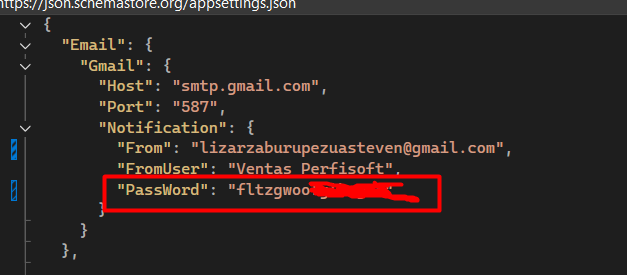


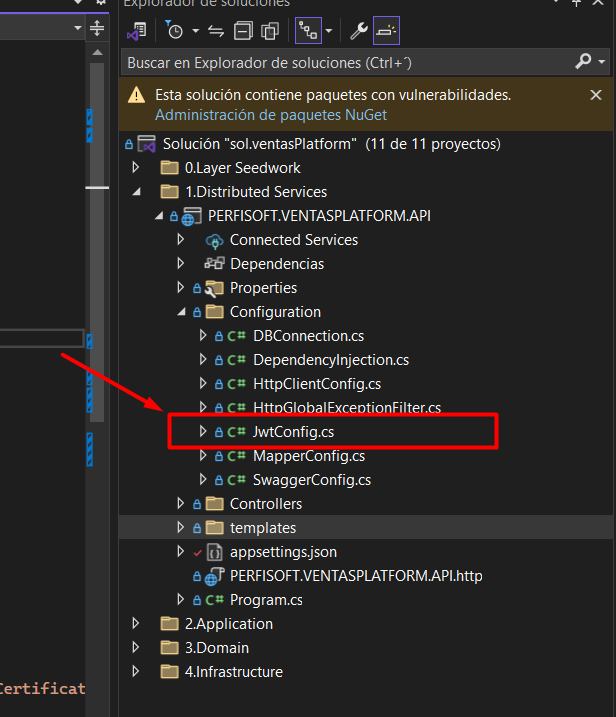


1. Al momento de darle al botón “**Crear**” se abrirá un popup automáticamente para pegar la contraseña generada.

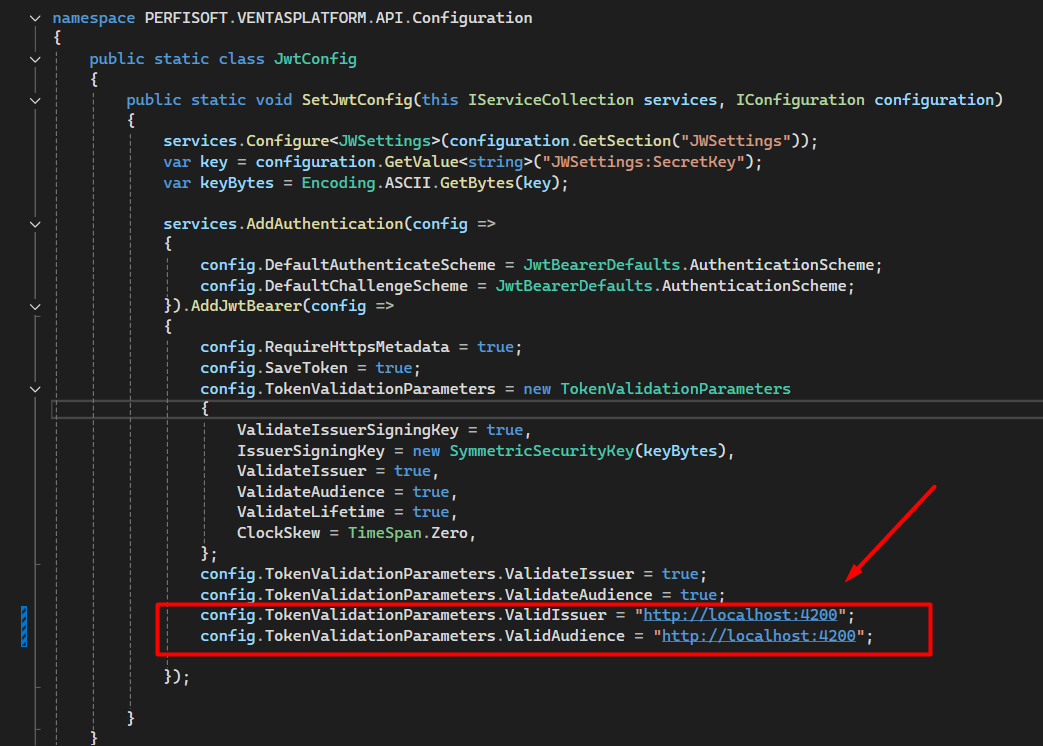
*La siguiente imagen muestra la contraseña generada que debemos copiar y pegar en la etiqueta Password de nuestro Email en* ***appsettings.json***



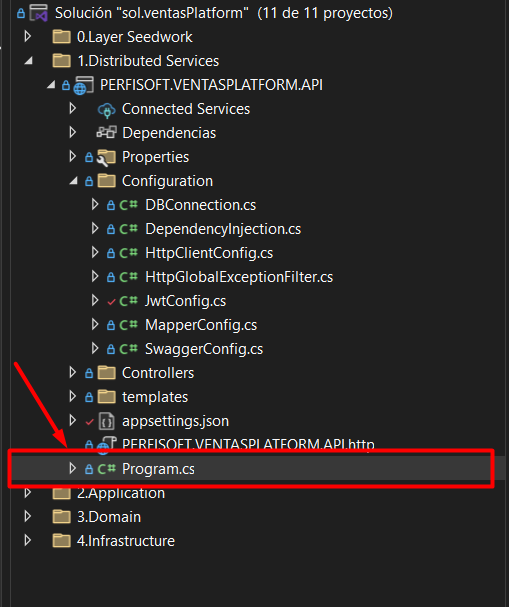
1.  Lo pegamos en la etiqueta Password como muestra la imagen (**Sin Espacios**).
2. Luego nos ubicamos en la clase **JWTConfig** que se encuentra en la carpeta **Configuration**.

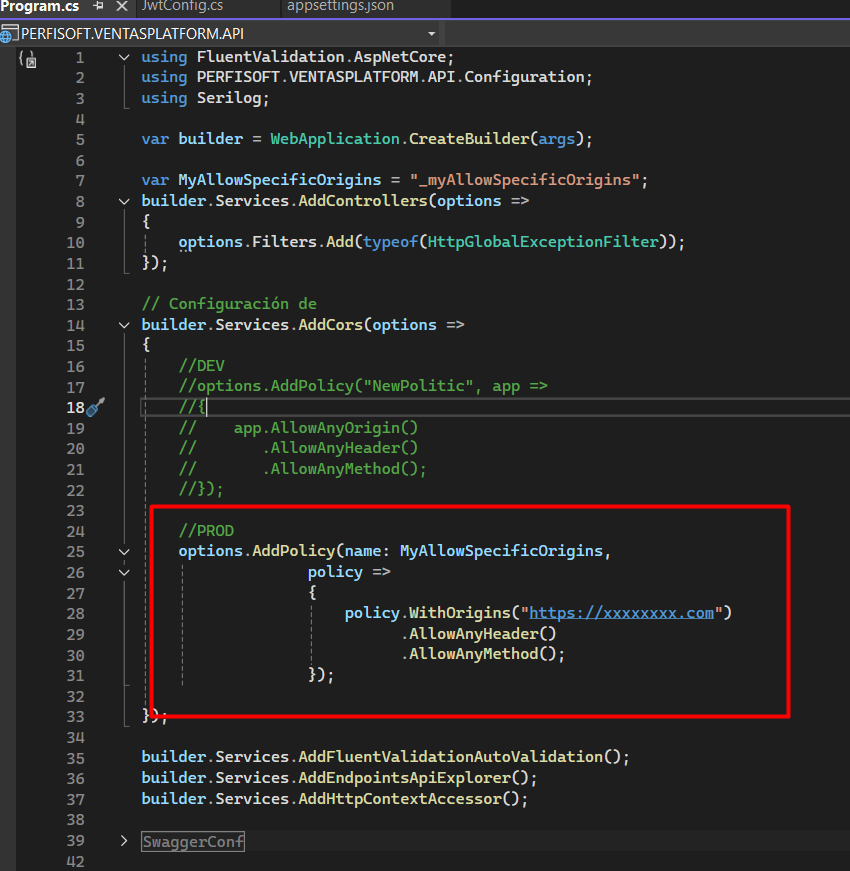
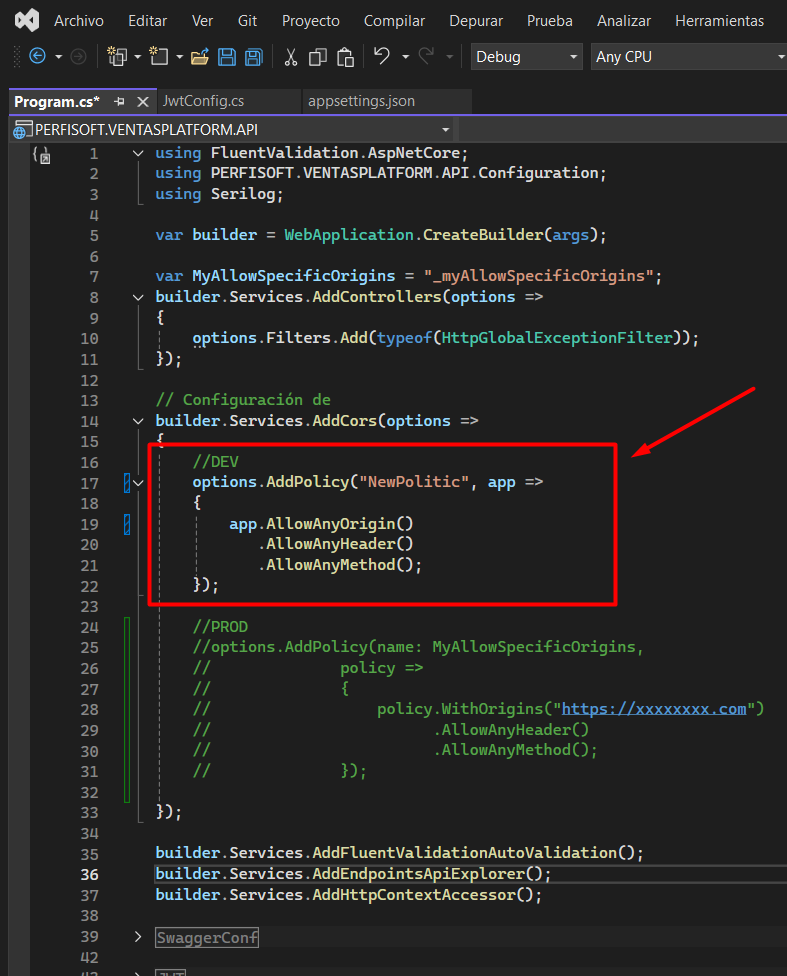


1. Configuramos los valores como se muestra a continuación

*Valor de la imagen:* [***http://localhost:4200***](http://localhost:4200) ***(recuerda que debe ser los valores que asignamos en el appsettings.json PASO 21)***

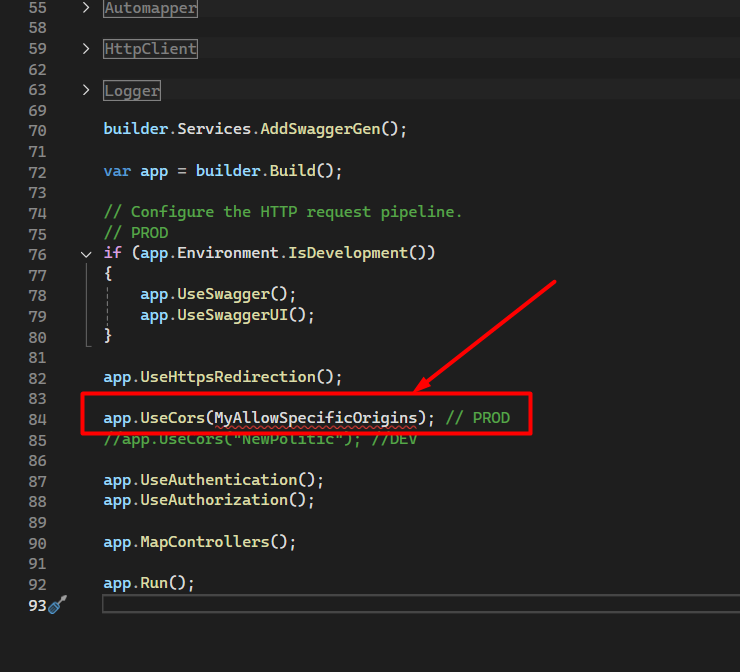
1. Nos dirigimos a la clase **"Program.cs"**, ubicada dentro de la carpeta **"1. Distributed Services"**



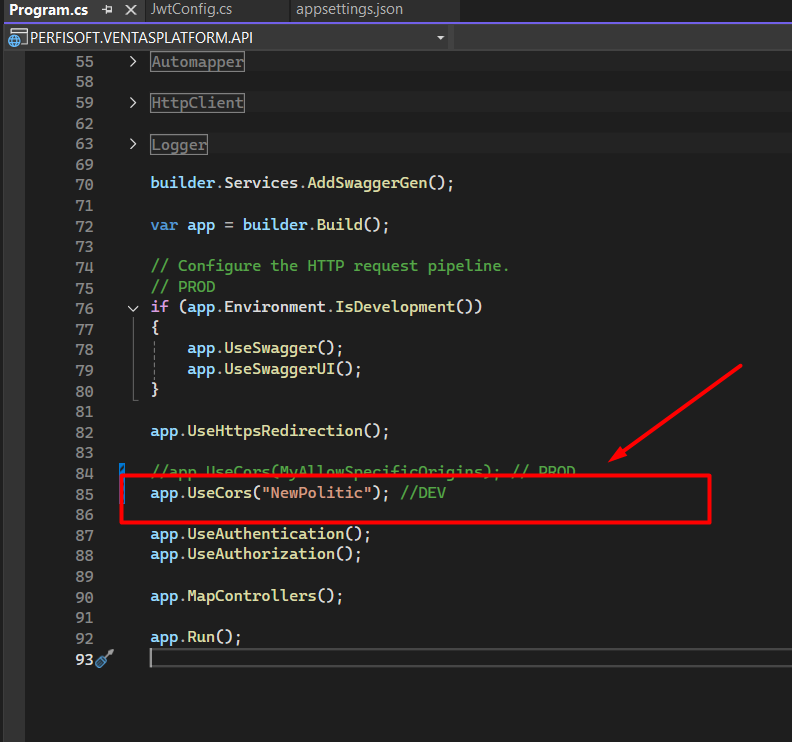
1.  Comentamos toda la sección marcada como **"//PROD"** y descomentamos la sección **"//DEV".**
2. La siguiente imagen muestra cómo debería estar configurada la sección **"//DEV"**.
3. Comentamos la siguiente variable **MyAllowSpecificOrigins**

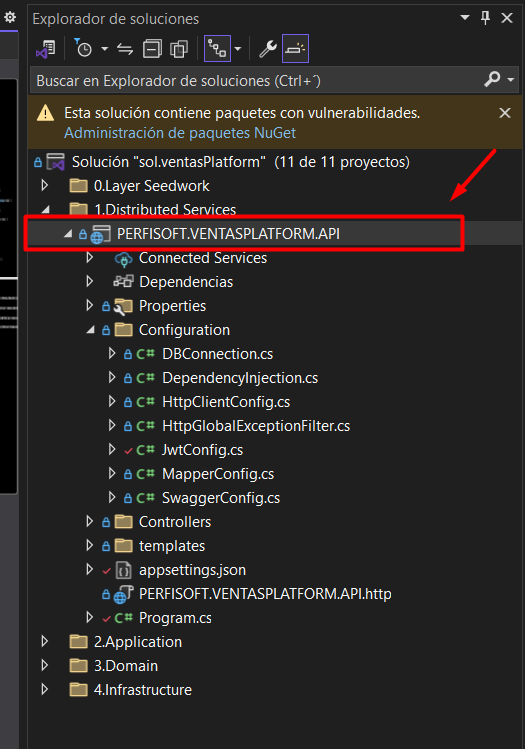
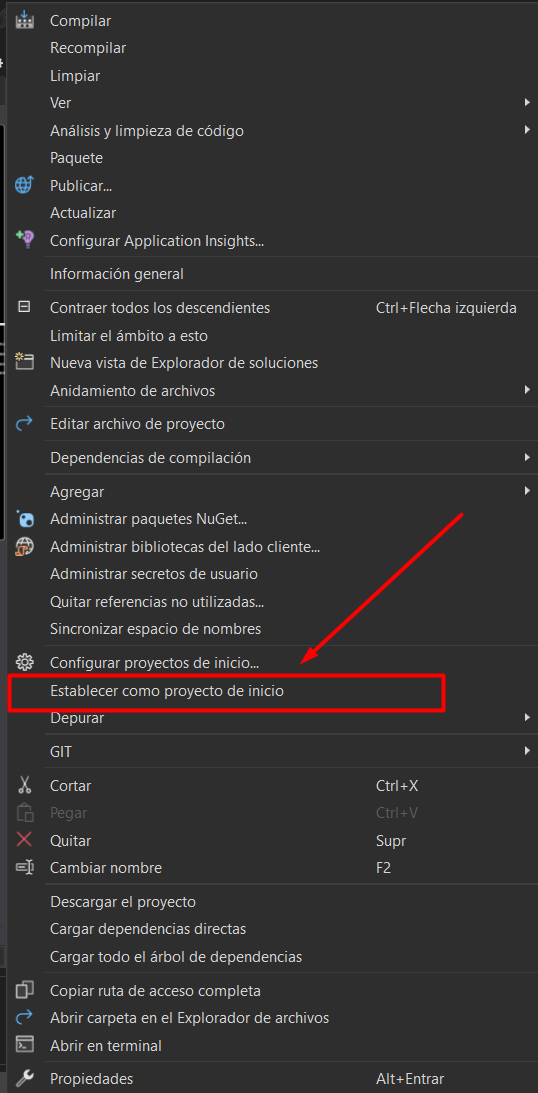


1. Comentamos la línea 84, dado que la variable fue comentada en el punto anterior.



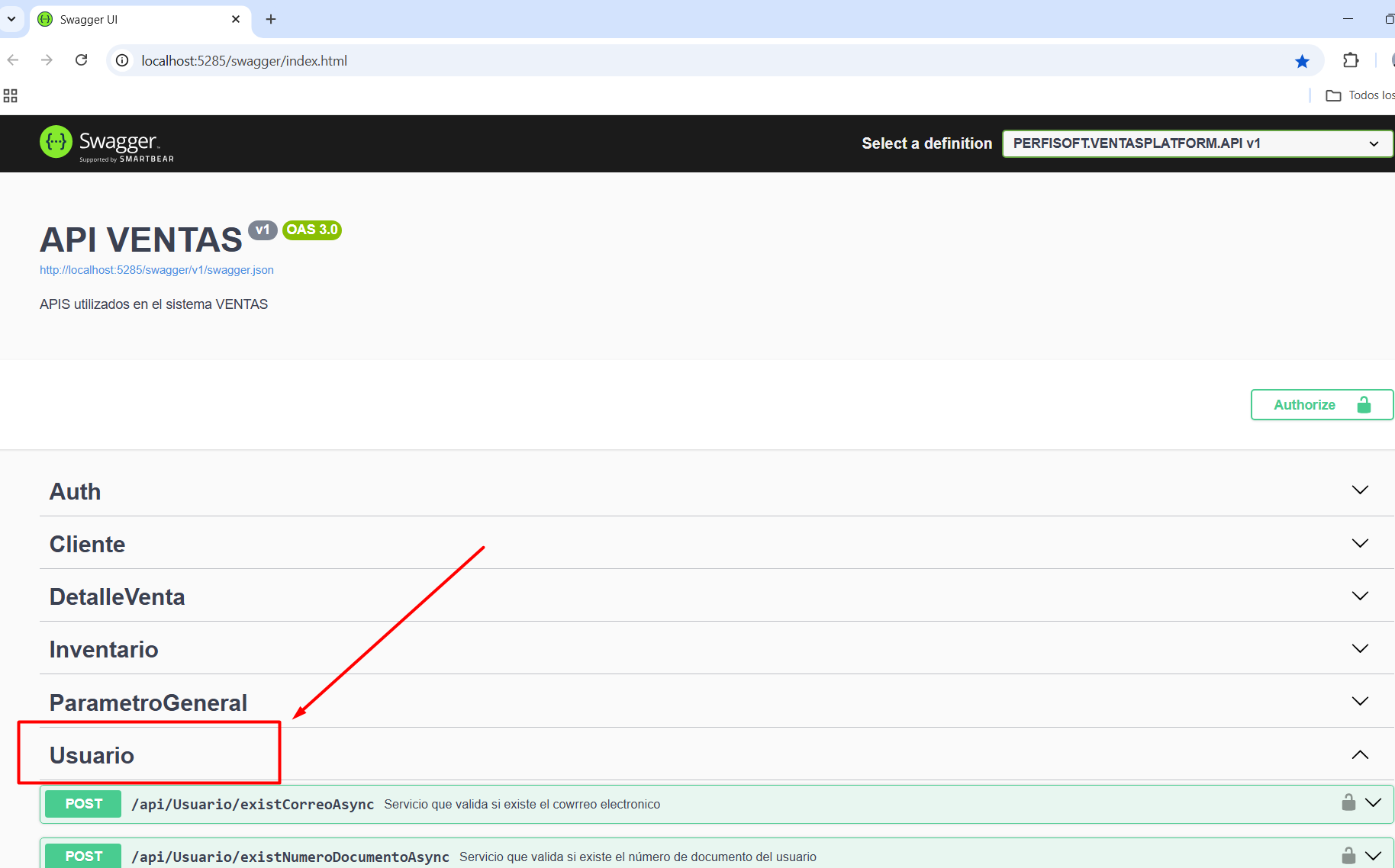
1. **Descomentamos la línea 85** para permitir solicitudes desde cualquier origen, es decir, desde el localhost, tal como se configuró en el punto 44.

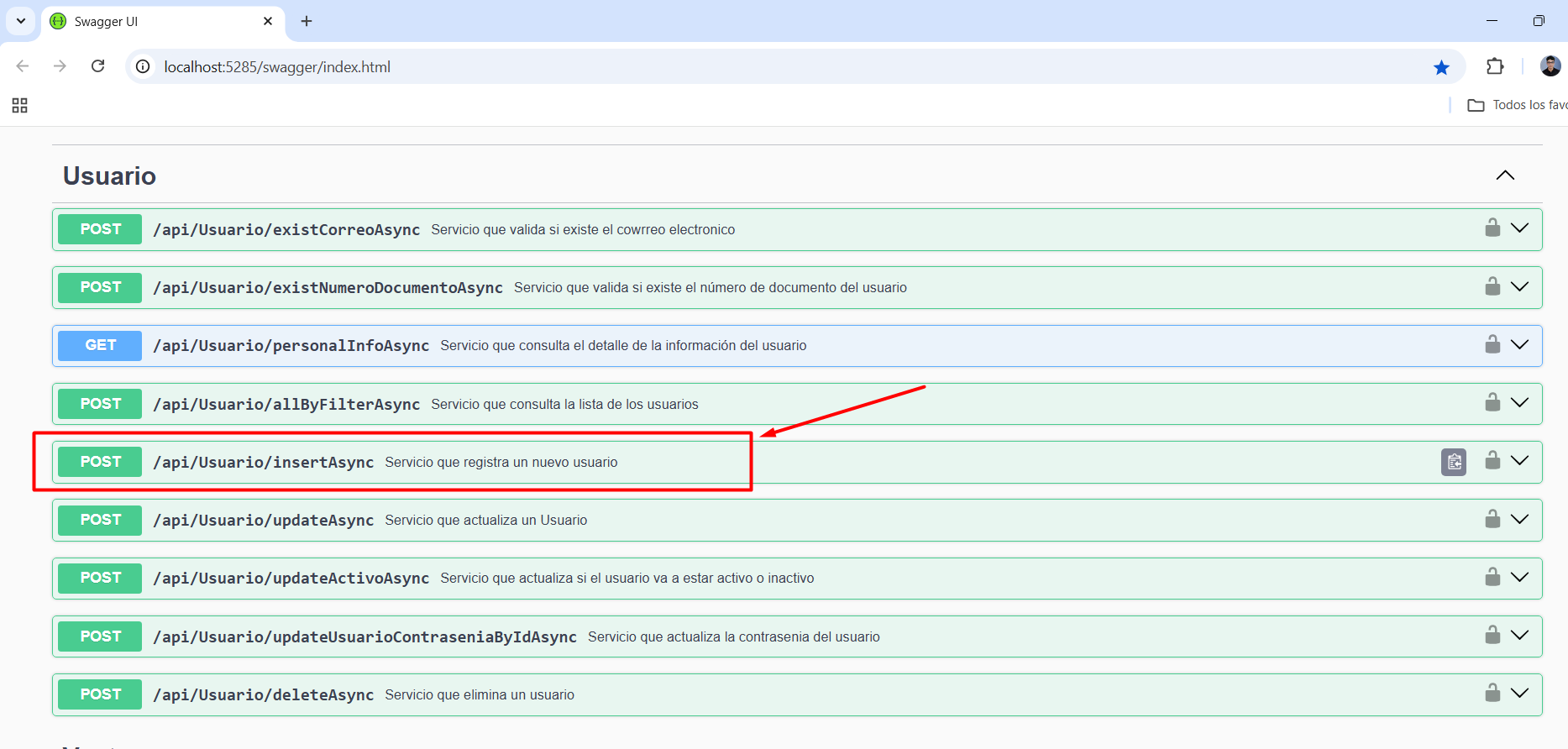


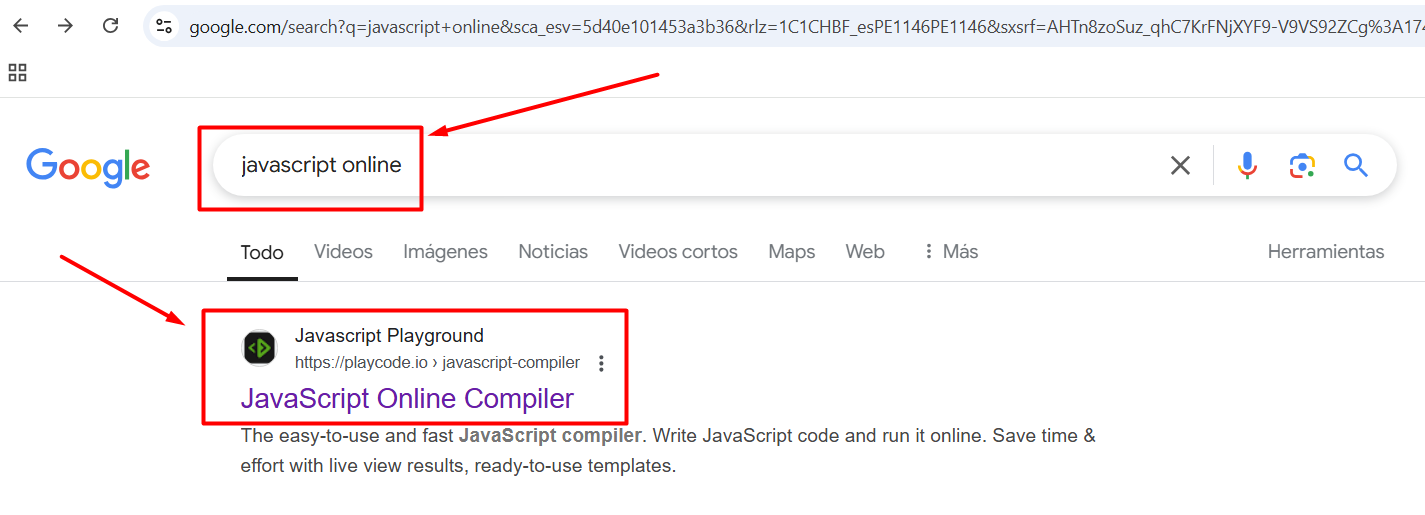
1. Ubicamos nuestra API **PERFISOFT.VENTASPLATFORM.API** y le damos **“clic derecho”.**
2. ****Le damos clic en “**Establecer como proyecto de inicio**”.
3. Iniciamos el proyecto

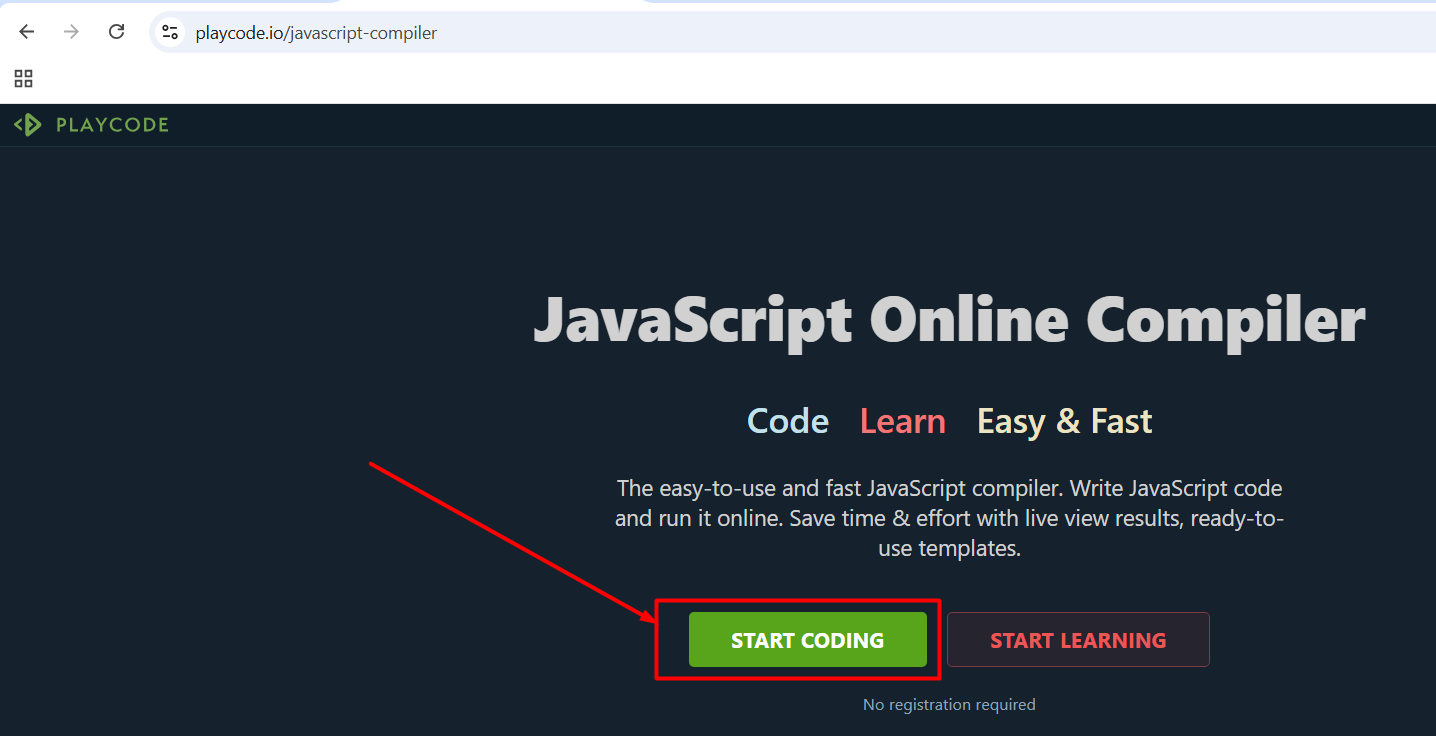
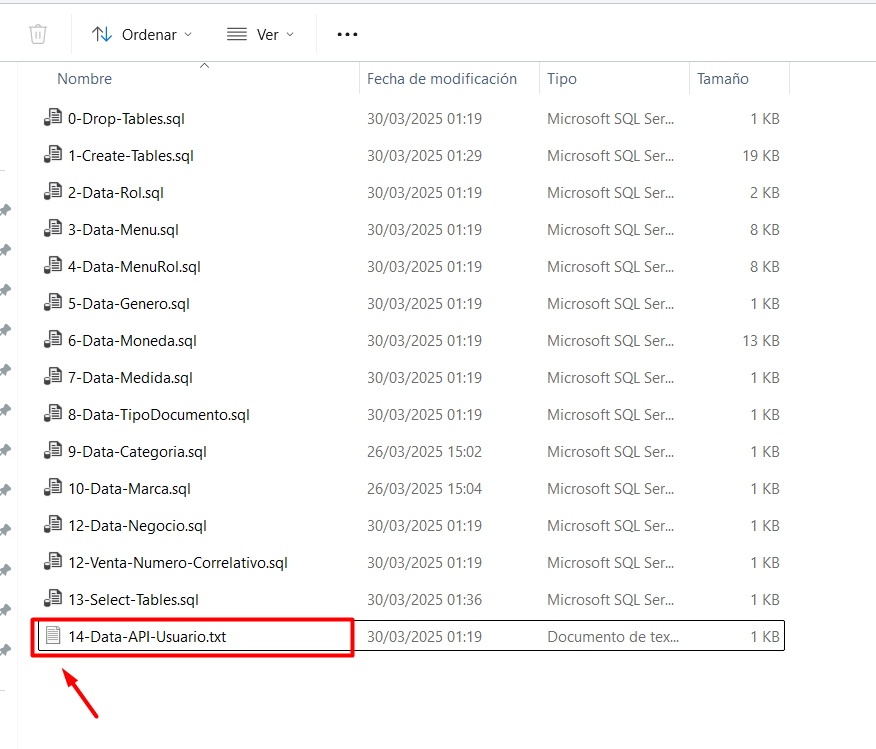
Interfaz de usuario gráfica, Texto, Aplicación

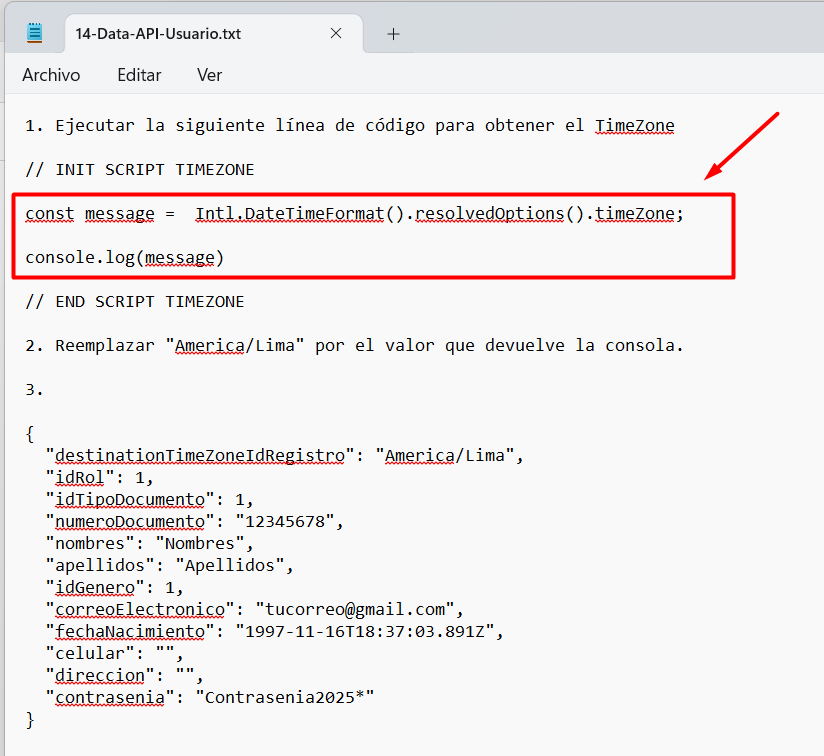
El contenido generado por IA puede ser incorrecto.

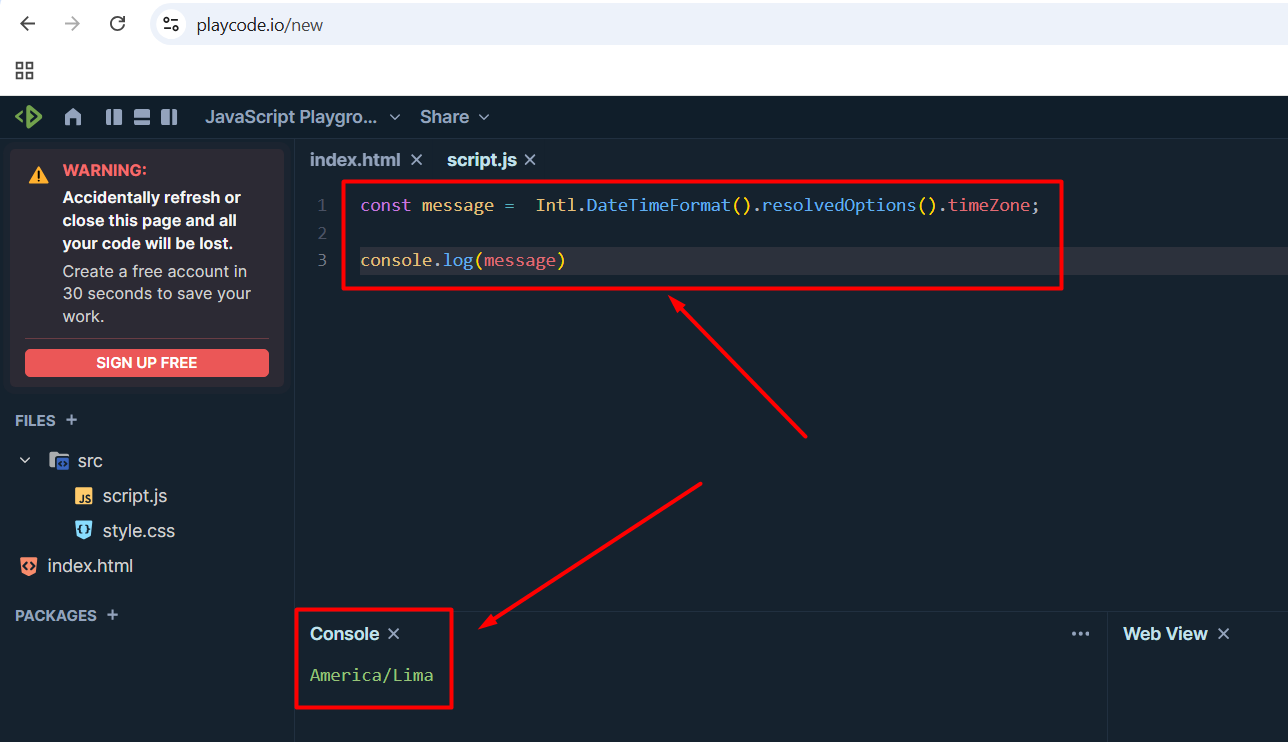
1. Ahora registraremos nuestro primer usuario. Nos ubicamos en la sección “**Usuario**”

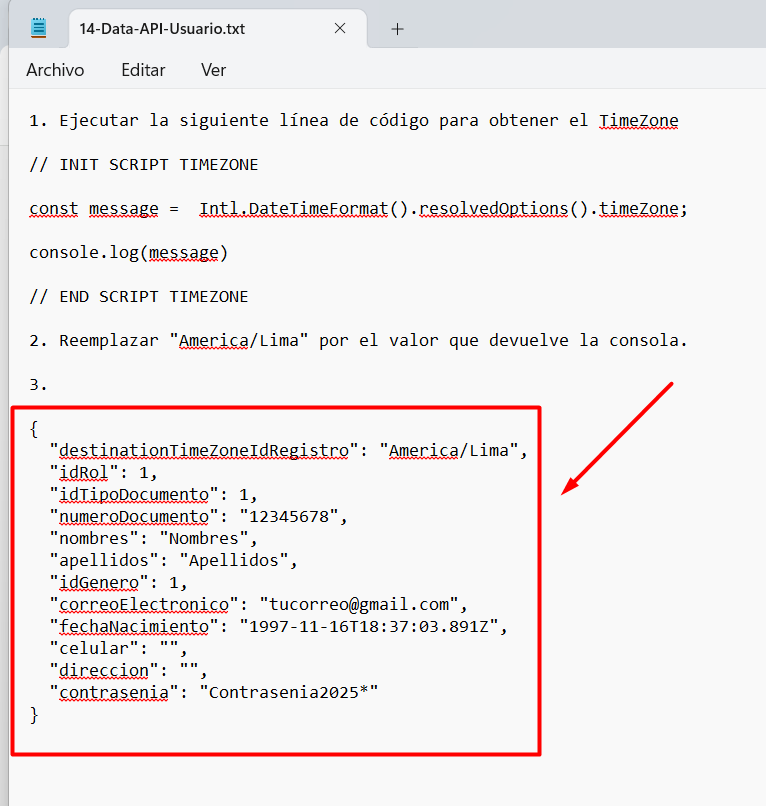
1. ampliamos el endpoint “**/api/Usuario/insertAsync**”
2. Antes de insertar nuestros datos debemos obtener el valor de nuestra ubicación. Para ello, abrimos nuestro navegador y escribimos “**javascript online**” y seleccionamos nuestra primera opción. Enlace directo: <https://playcode.io/>



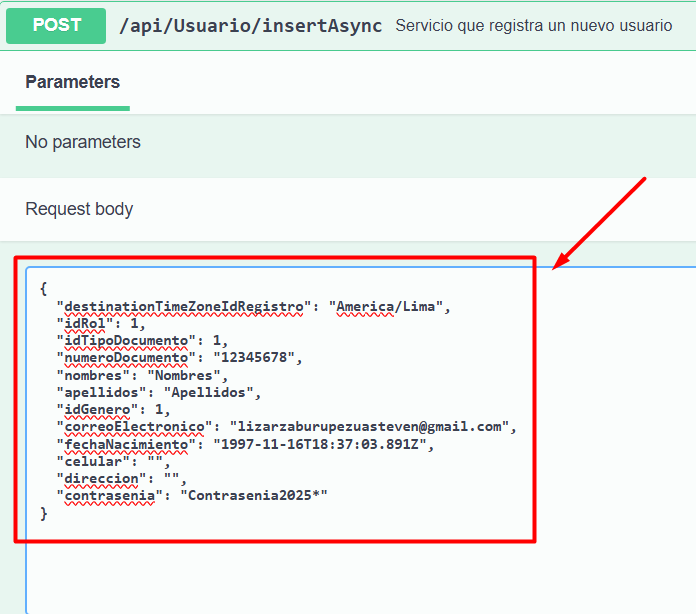
1. Presionamos el botón “**START CODING**”
2. Nos ubicamos en nuestra carpeta de “**1-Scripts**” y abrimos el archivo txt llamado “**14-Data-API-Usuario.txt**”
3. Copiamos el código TypeScript marcada en la sección en rojo



1. Pegamos el valor en la consola de la página abierta anteriormente y copiamos el resultado mostrado en pantalla. En mi caso, aparece "**America/Lima**".
2. Pegamos el valor en la etiqueta “**destinationTimezoneIdRegistro**” y copiamos todo el **json** marcado en rojo



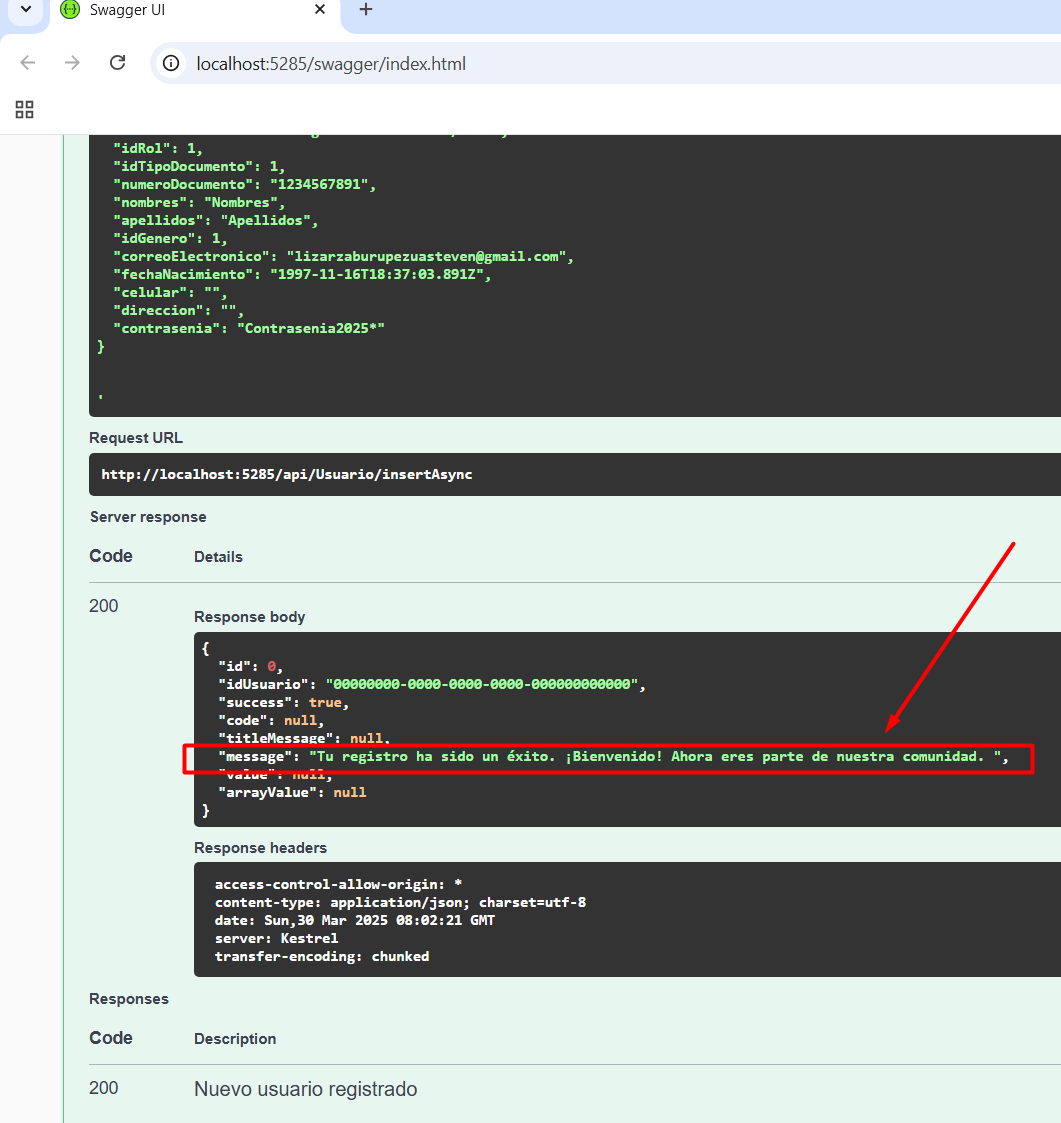
1. Regresamos a la API de Swagger y pegamos todo lo copiado. ***Recuerda que el correo debe ser válido, ya que al completar el registro recibirás un correo de bienvenida****.*

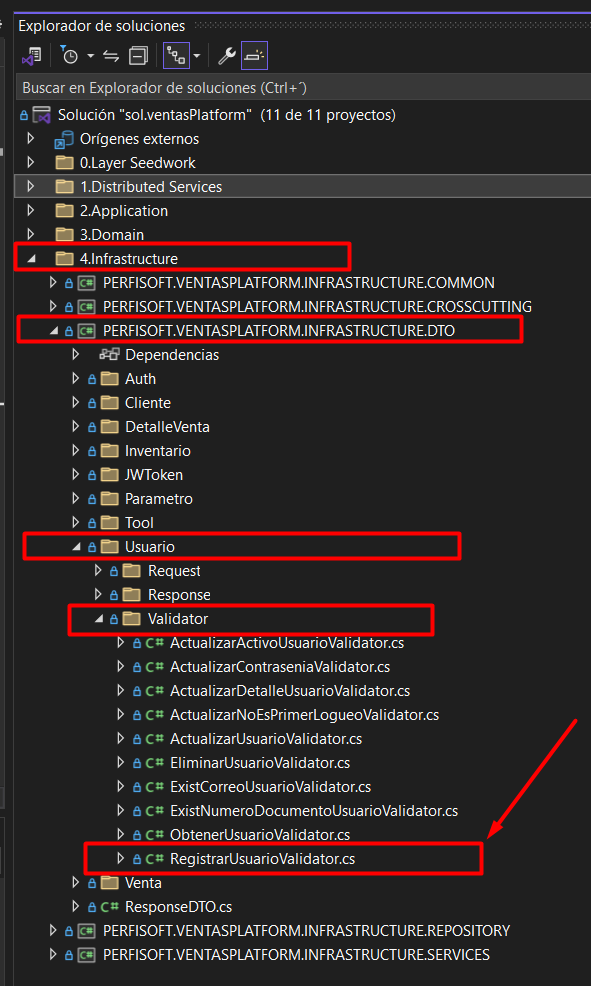


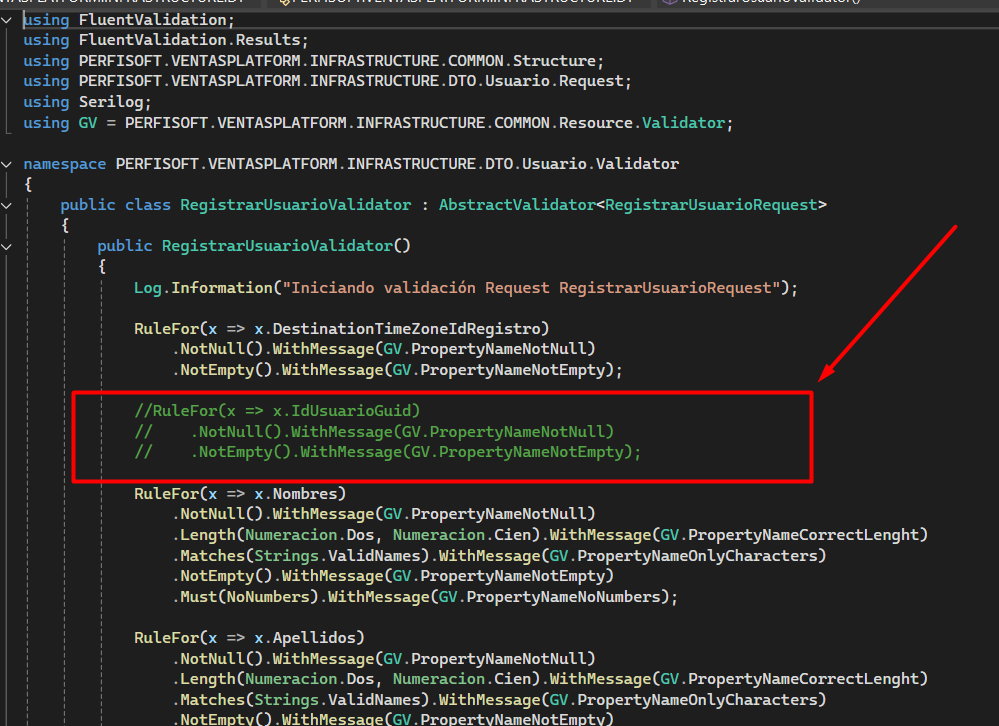
1. Luego presionamos en el botón “**Execute**”.

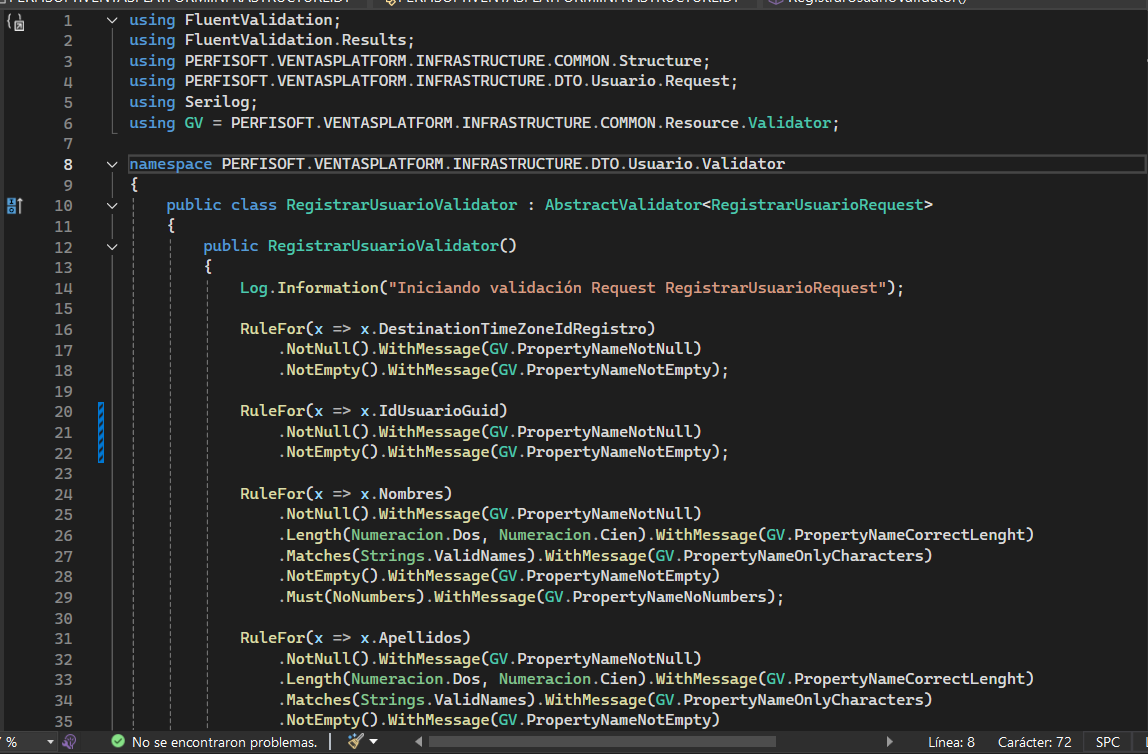


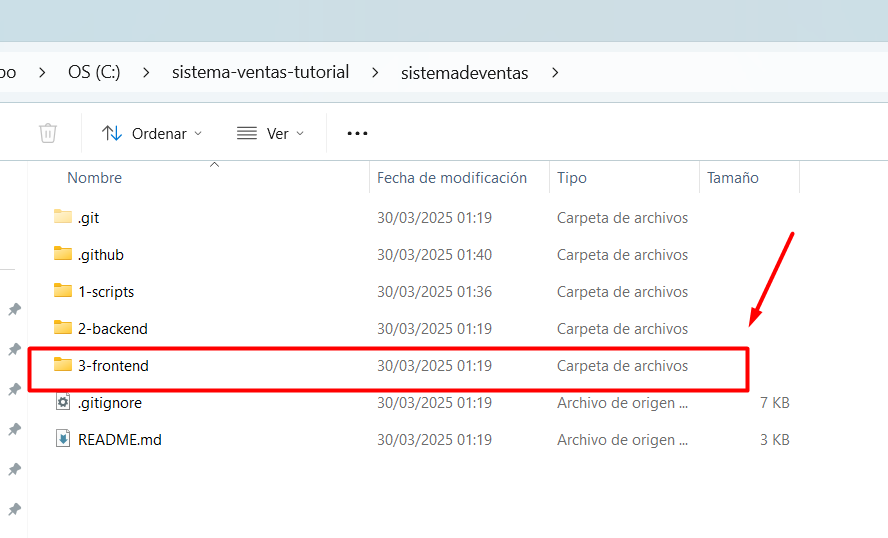
1. La siguiente imagen muestra el resultado esperado luego de haber presionado el botón “**Execute**”



1. Luego por temas de seguridad ingresamos a la clase “**RegistrarUsuarioValidator**”
2. Descomentamos la sección indicada. Esto hará que nuestro backend exija un identificador válido, es decir, que el usuario exista.

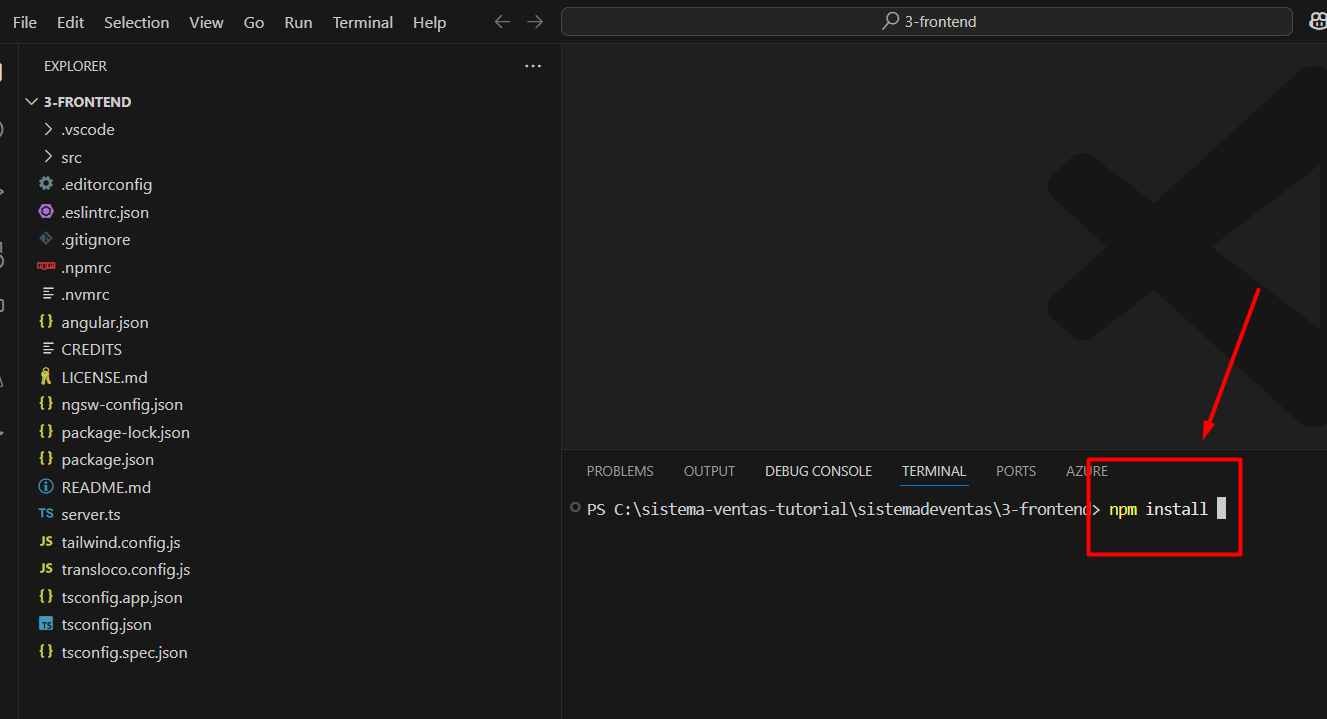


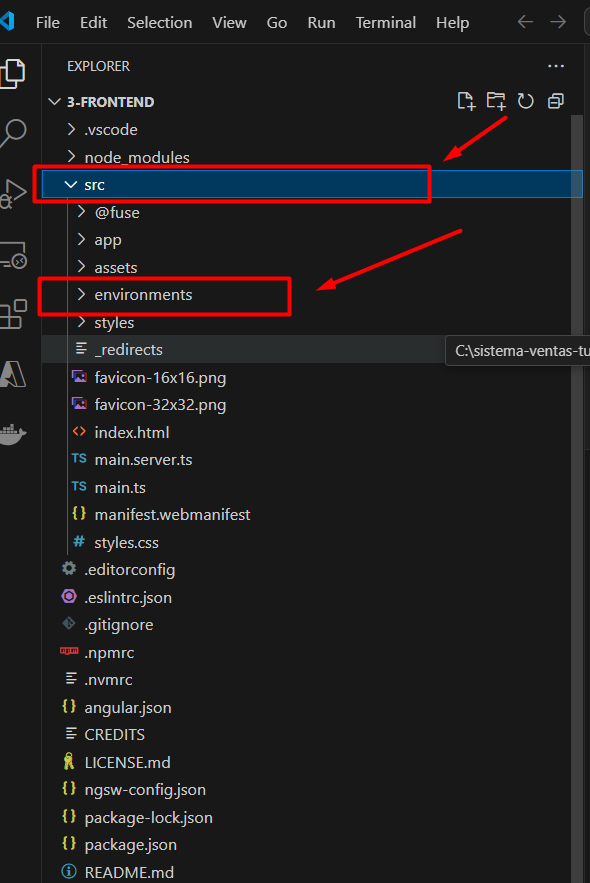
1. La siguiente imagen muestra cómo debería quedar la clase luego de haber descomentado la sección solicitada
2. Es hora de configurar nuestro proyecto Frontend. Abrimos la carpeta “**3-frontend**”.



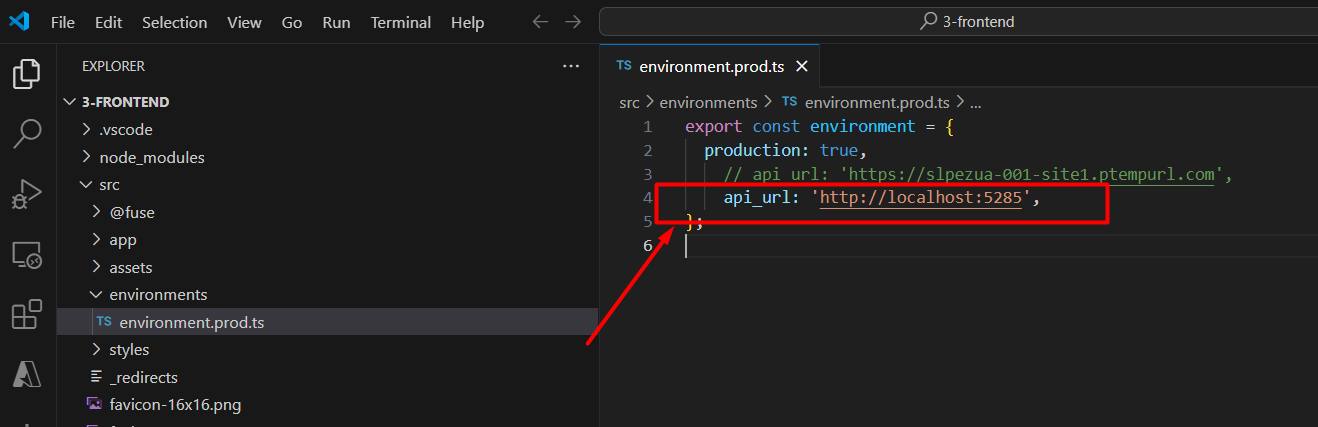
1. Abrimos Visual Studio Code y arrastramos la carpeta completa **"3-frontend"**. En la terminal, escribimos y ejecutamos con “**Enter**” el siguiente comando:

**npm install**

****

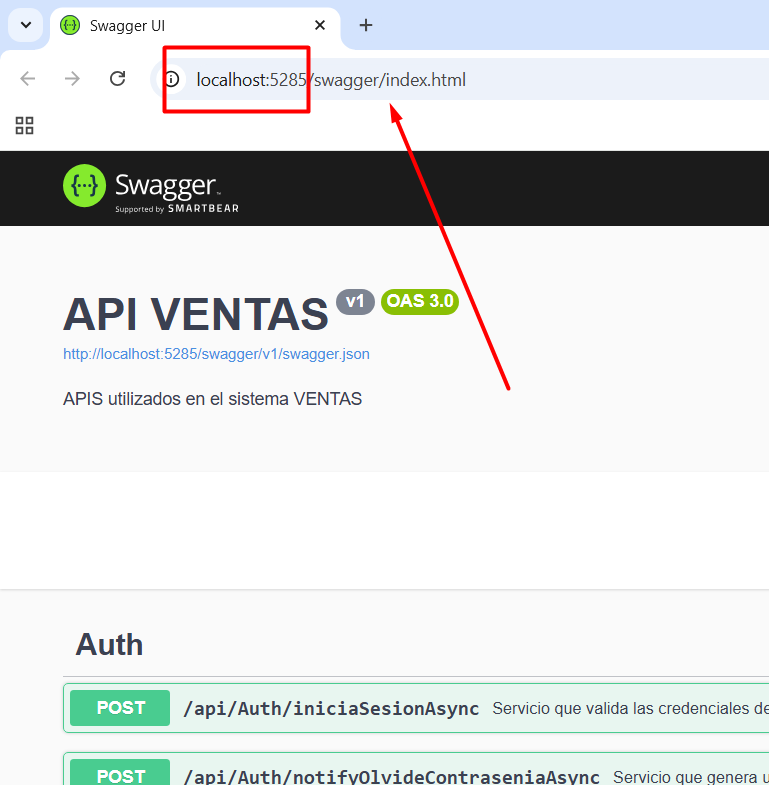
1. Nos dirigimos a la carpeta **"src"**, la expandimos y accedemos a **"environments"**. Luego, abrimos el archivo **"environment.prod.ts"**.

1. Asignamos el valor de nuestro API como se muestra en la imagen

\_

1. En mi caso el valor de mi url es **http://localhost:5285**

Es importante este paso ya que si asignamos mal el valor de nuestro API, el aplicativo no va a funcionar



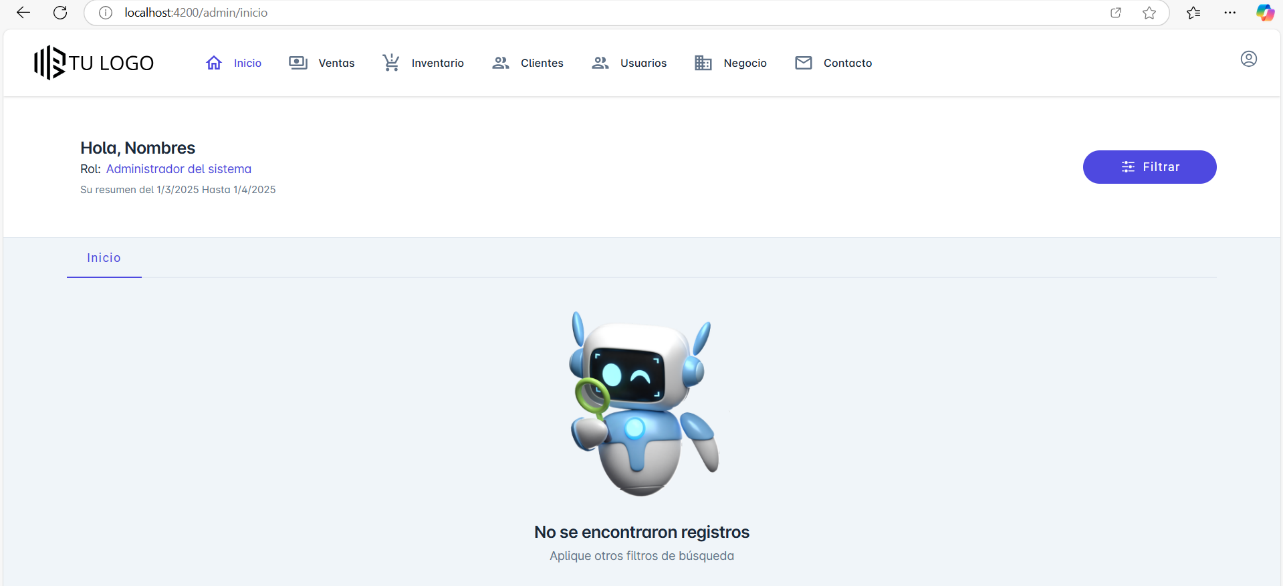
1. Finalmente, iniciamos la aplicación desde la terminal ejecutando el siguiente comando:

**ng serve -o**

1. La pantalla inicial se mostrará como en la siguiente imagen. Ingresamos nuestro correo electrónico y la contraseña configurada en el punto 59.



1. Si todo fue configurado correctamente, la primera pantalla que veremos será la siguiente.



Para cualquier duda, consulta o error, comuníquese a notificaciones@perfisoft.com