**Desafío de Integración Multilenguaje y Base de Datos**

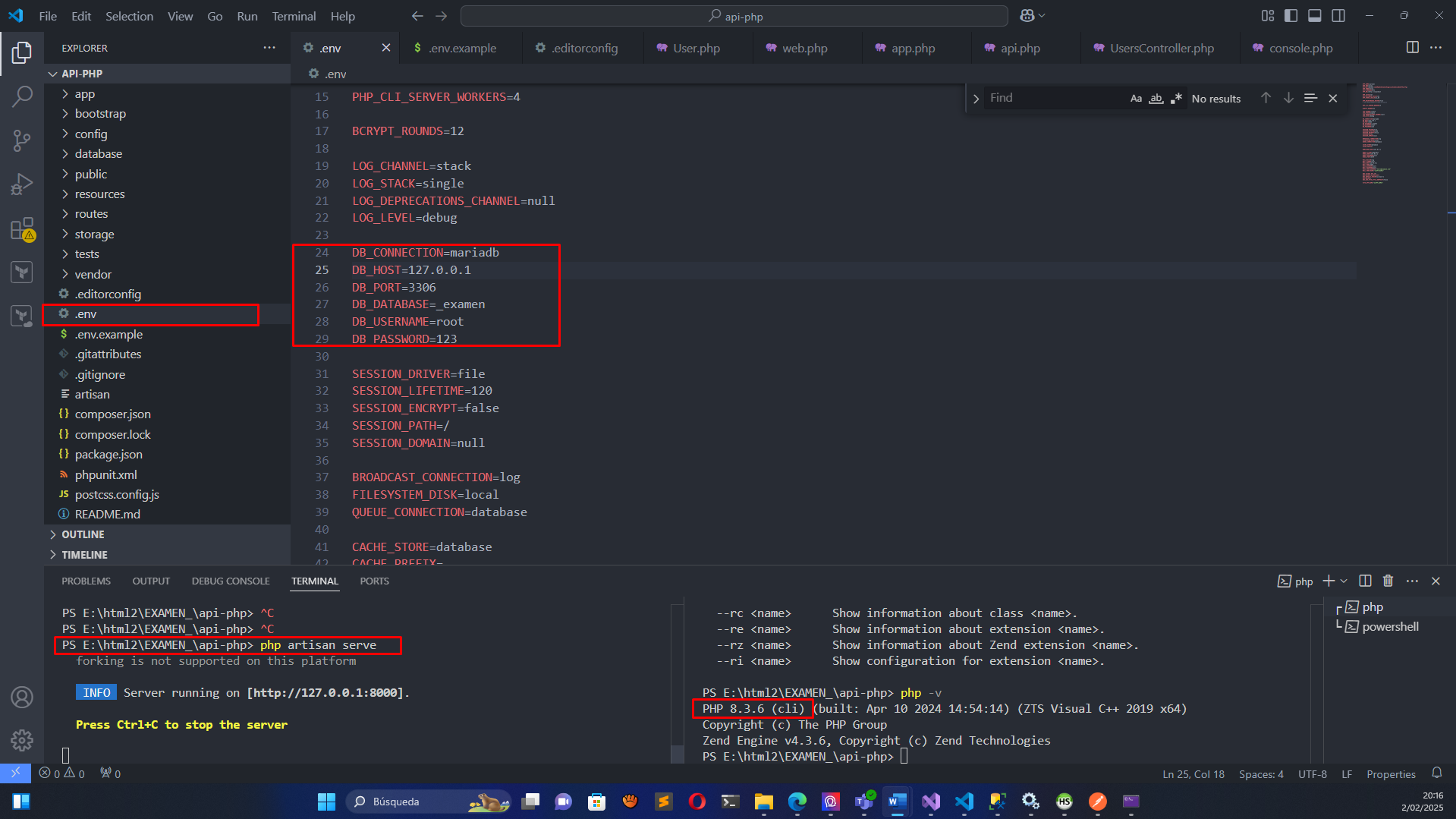
**Uso de API – PHP (Laravel):**

1 - En la carpeta “api-php” debemos abrir el archivo .env y reemplazar las variables de la línea 24 a la 29 con los datos del servidor MySQL.

2 - En la línea de comandos ejecutamos el comando “composer install” para instalar los paquetes necesarios de Laravel.

3 - Ejecutamos el proyecto con el comando “php artisan serve”.

Nota: se está usando la versión 8.3.6 de PHP.



¿Por qué Laravel?, sucede que este framework crea en automático toda una serie de carpetas de tal modo que al levantar el proyecto, ya tenemos una configuración de rutas, namespaces, etc; resaltando el uso del ORM, para poder hacer los querys rápidamente sobre la BD.

Además, las rutas de api de Laravel agregan un prefijo /api en cada ruta, por lo que por un tema de configuración se optó por mantener ello y actualizar las rutas propuestas (ejemplo: se propuso /users, y se creó api/users)

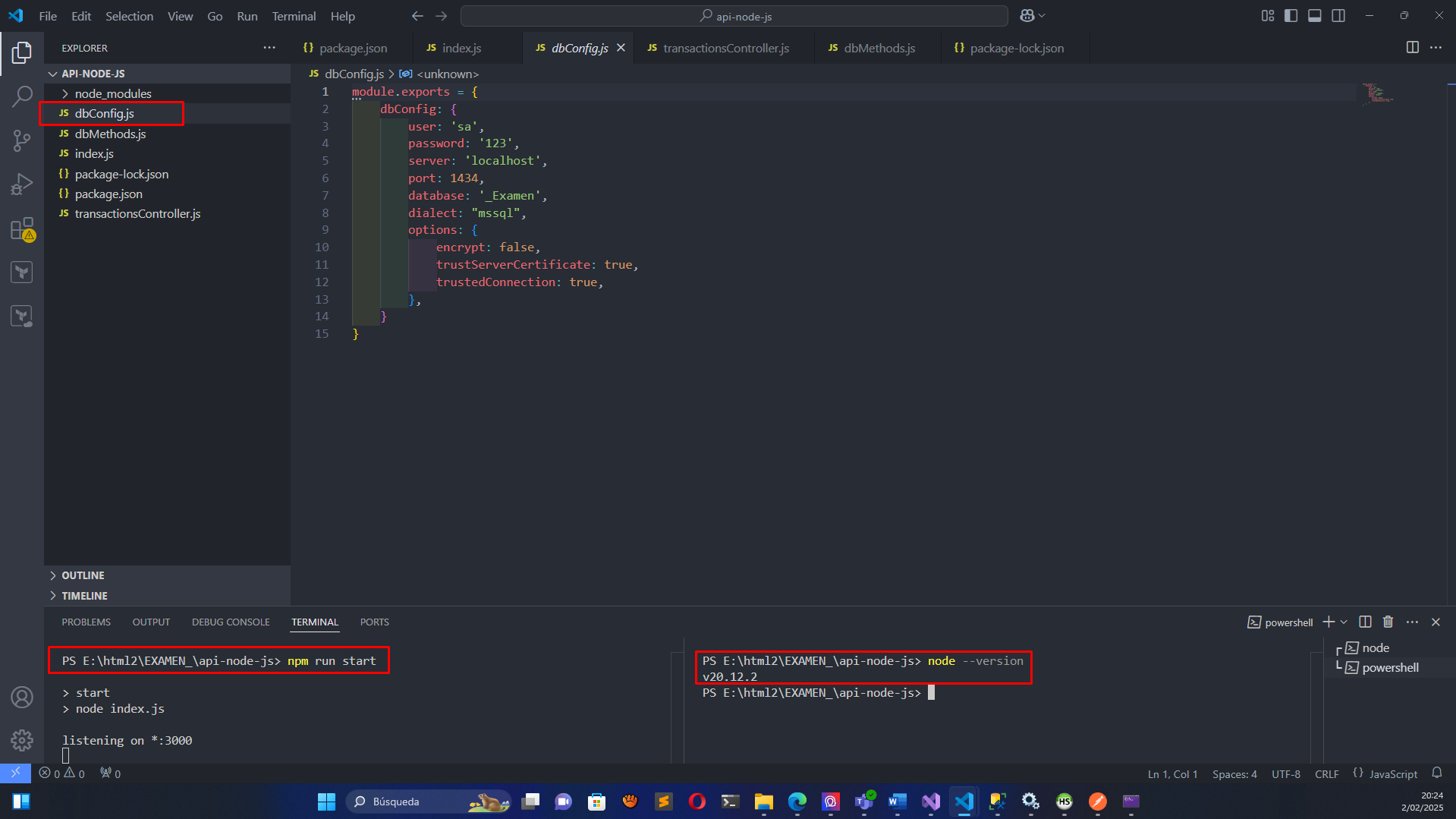
**Uso de API – Nodejs (Express):**

1 - En la carpeta “api-nodejs” debemos abrir el archivo dbConfig.js y reemplazar las variables con la información de nuestra bd SQL Server.

2 - En la línea de comandos ejecutamos el comando “npm i” para instalar los paquetes necesarios de Express.

3 - Ejecutamos el proyecto con el comando “npm run start”.

Nota: se está usando la versión 20.12.2 de node.



¿Por qué se tiene esa estructura? Sucede que al ser un API con pocos endpoints, y sin mucha configuración, se optó por hacer algunas cosas dentro de un archivo, como los endpoints, uso de consultas SQL directas, etc; lo que hace ver algunas oportunidades de mejora, como el uso de ORM o procedimientos almacenados, aplicar más a detalle los principios SOLID, entre otras mejoras que se podrían hacer en una segunda versión(a pedido del equipo de desarrollo).

**Uso de API – C# (.NET 8):**

1 - En la carpeta “api-c-sharp” debemos abrir el archivo api-c-sharp.sln y esperar a que visual studio levanta la solución. (ver imagen 1)

2 – Una vez abierto el visual studio, actualizamos las variables de conexión a los API’s (ver imagen 2) y luego presionamos la tecla F5 para levantar el API, o caso contrario presionamos el botón ejecutar. (ver imagen 3)

Nota: se está usando la versión de .net 8

Imagen 1:

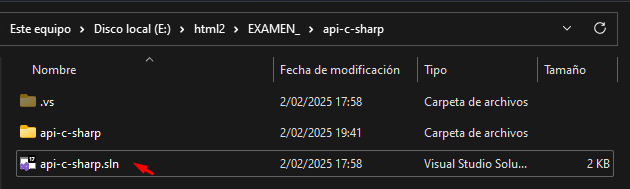


Imagen 2:

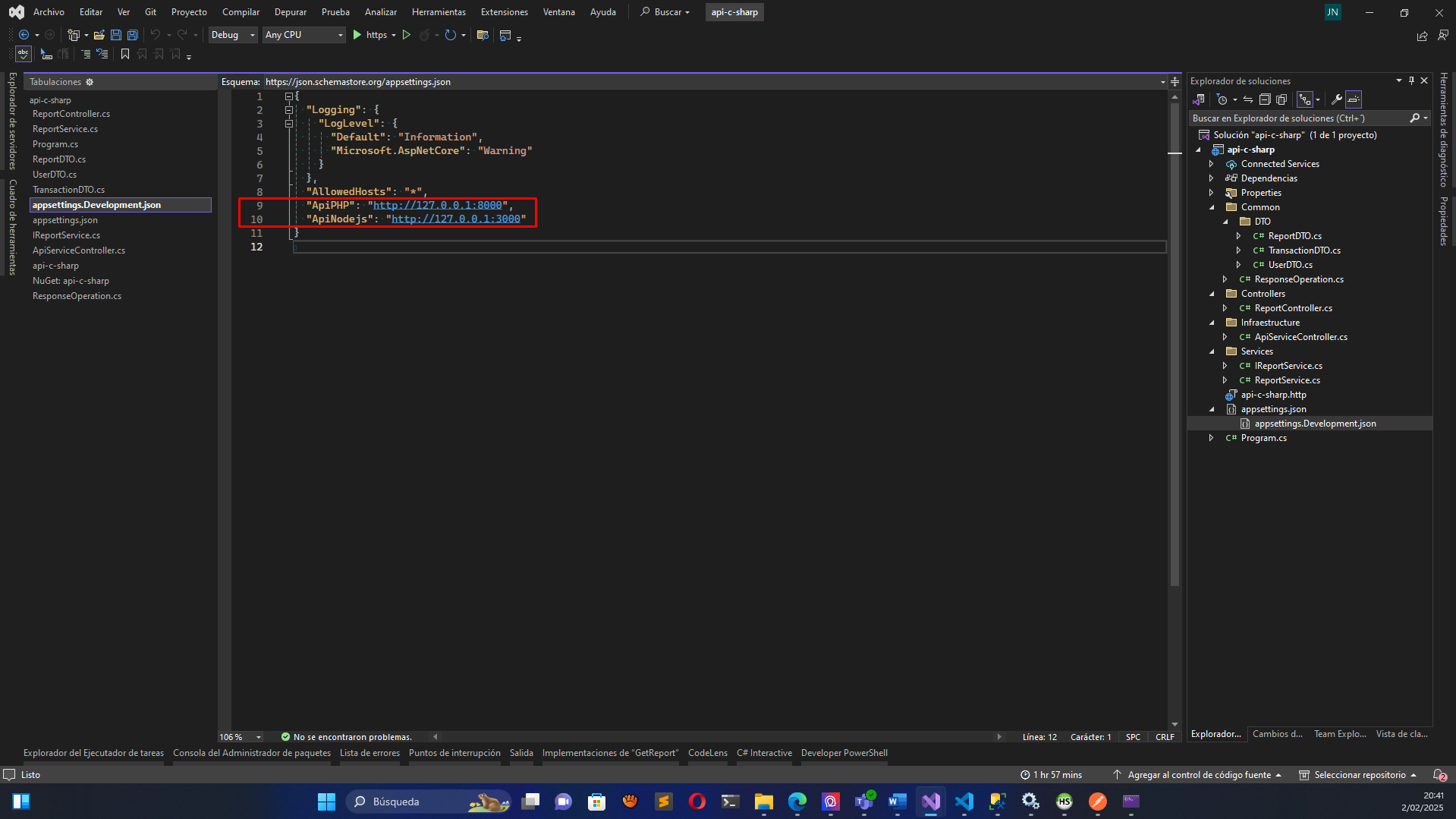
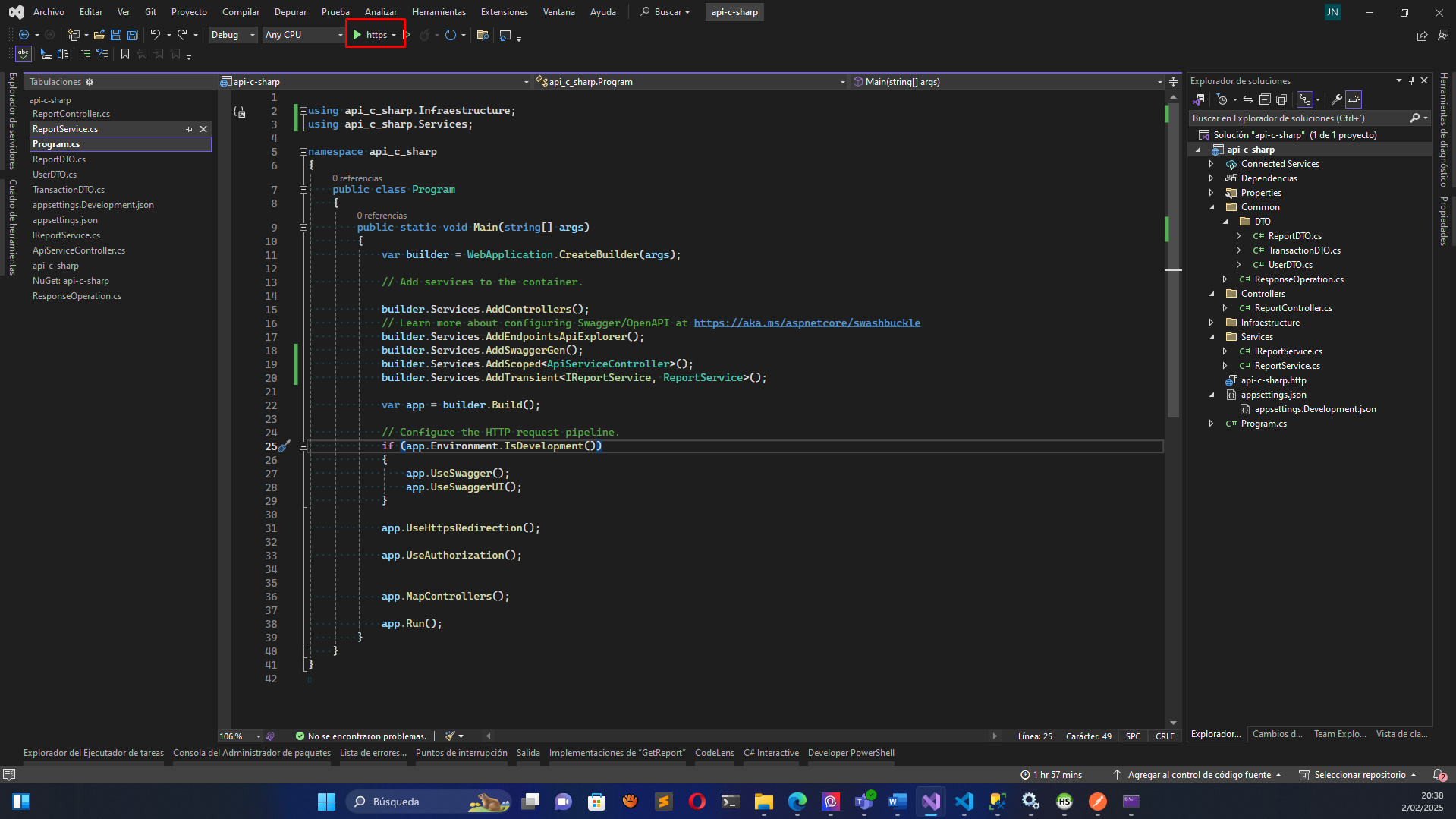


Imagen 3:

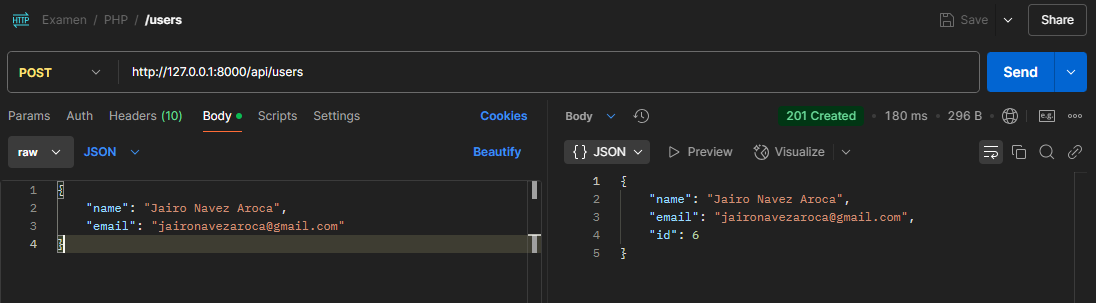


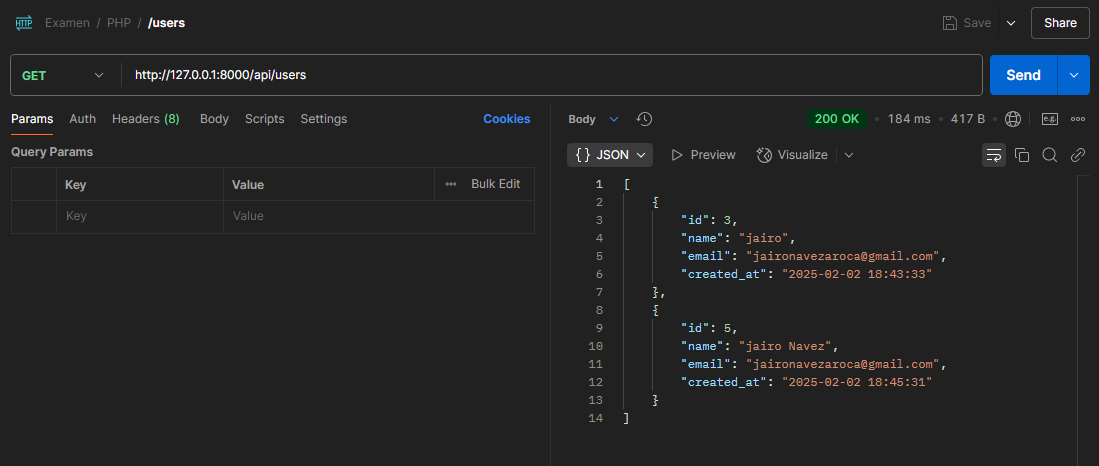
Consideraciones importantes: se asume que la data llega sin inconsistencias, ya que levantar las inconsistencias es parte del proceso de registro de una transacción.

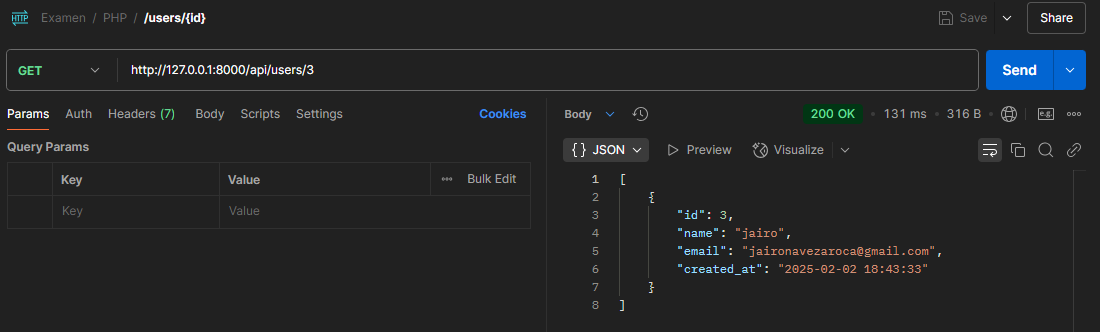
Se optó por seguir una estructura de carpetas basadas en servicios(que tiene la lógica de negocio), infraestructura (conexión a servicios externos como API) y common (clases comunes por toda la aplicación).

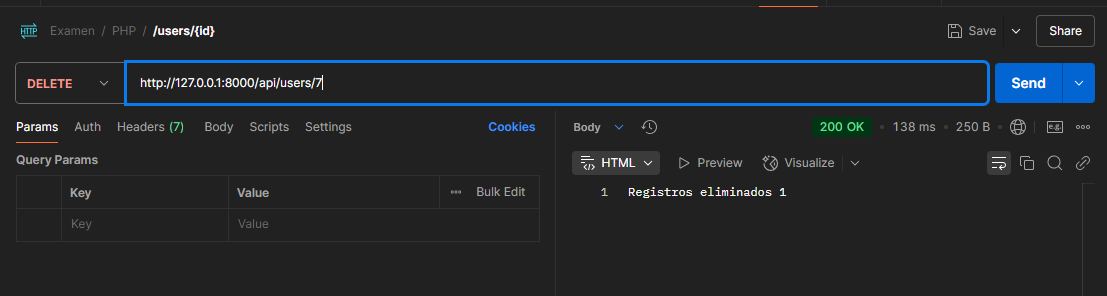
Además se está usando inyección de dependencias + validación sobre indisponibilidad de los API hechos con php y/o nodejs.

**Responses de api – Usuario**

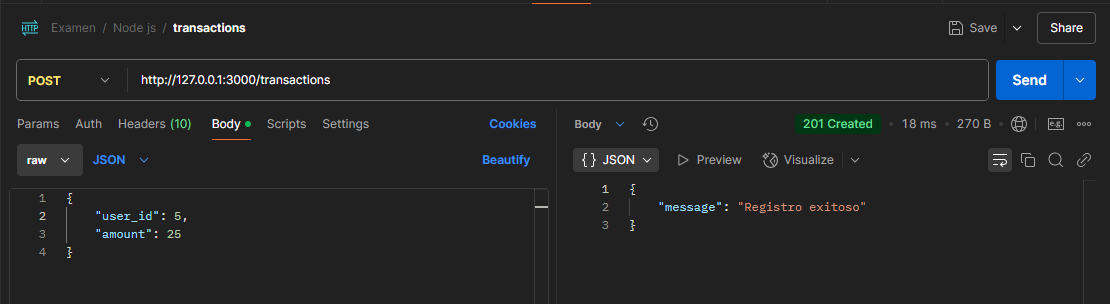


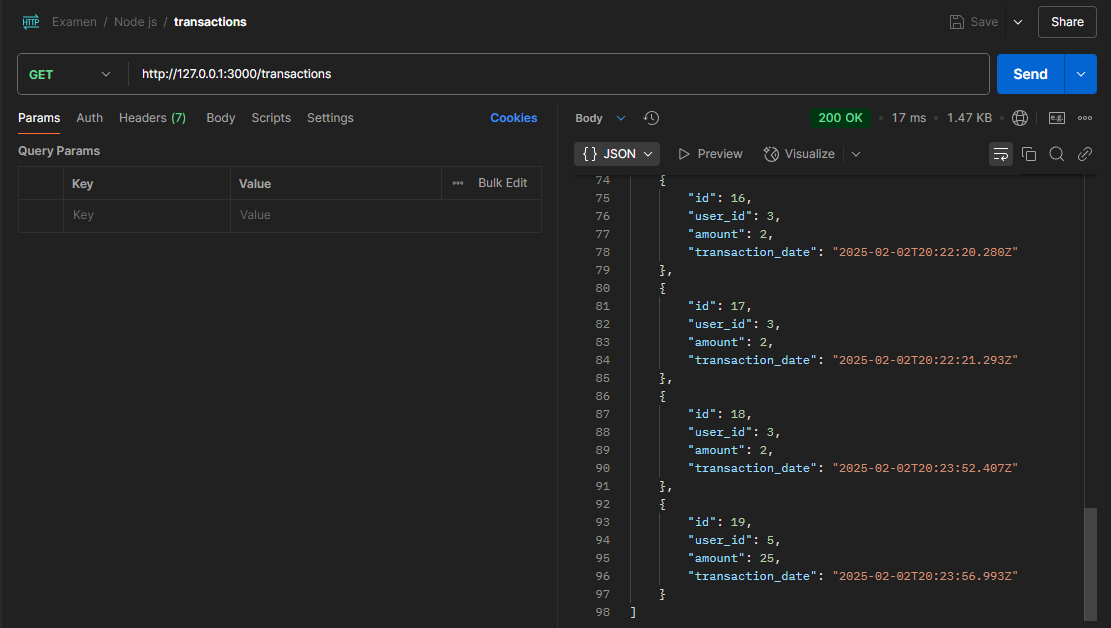


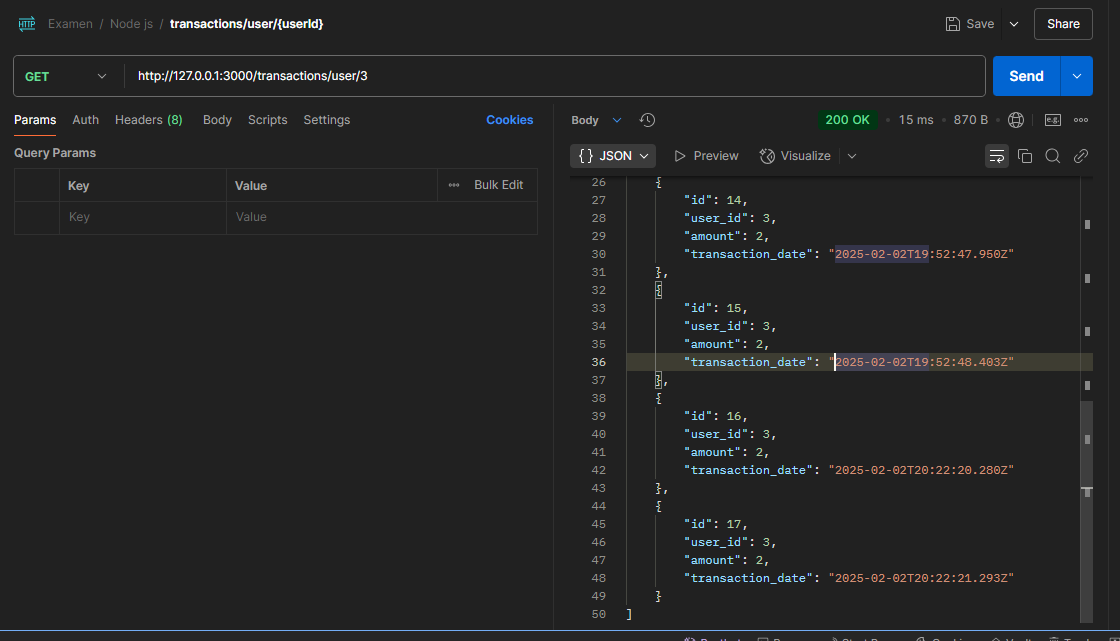




**Responses de api – Transacción**







**Responses de api – Reporte**

