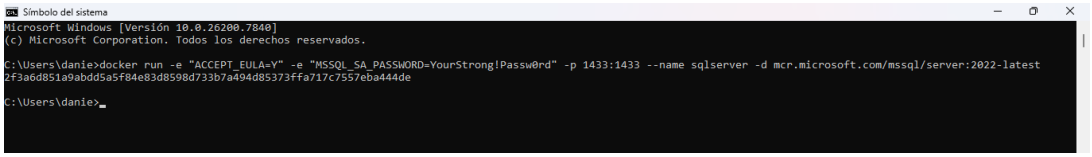
Estructura de Evidencias: Prueba Técnica Nuxiba

Información General

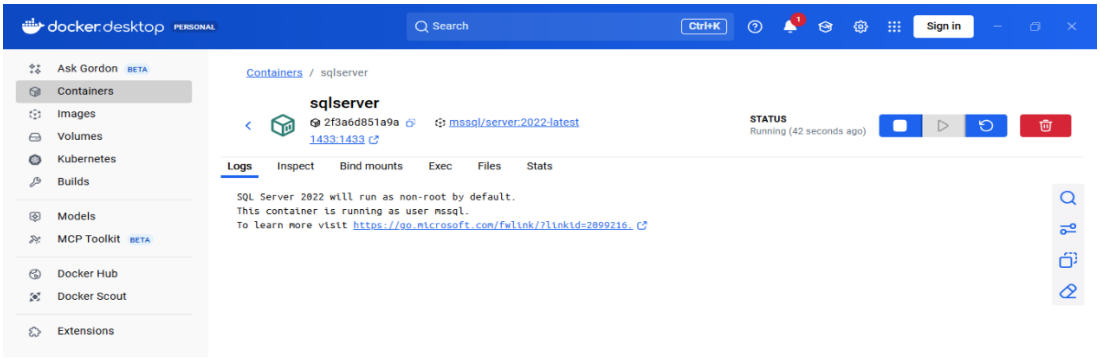
- **Desarrollador:** Daniel Sebastian Calzada Guerrero
- **Fecha:** Febrero 2026
- **Stack Tecnológico:** .NET 8, SQL Server (Docker), Entity Framework Core, xUnit, Postman.

Orquestación de Infraestructura y Automatización

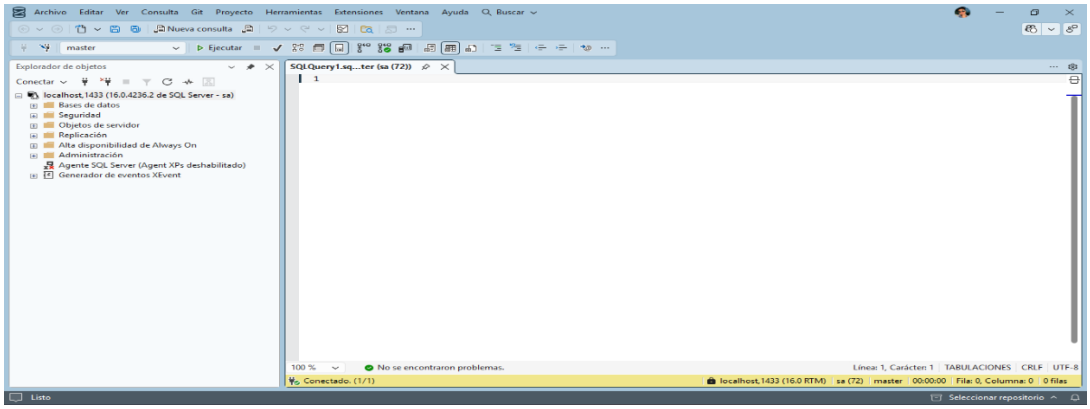
Se implementó una arquitectura basada en contenedores utilizando **Docker** para el despliegue de una instancia de **SQL Server 2022**. Este enfoque garantiza la portabilidad de la solución y un entorno de base de datos aislado. La comunicación se estableció mediante cadenas de conexión seguras dentro del archivo appsettings.json, permitiendo que la API tome el control total del servidor de base de datos desde el primer arranque.



Cadena de conexión desde CMD



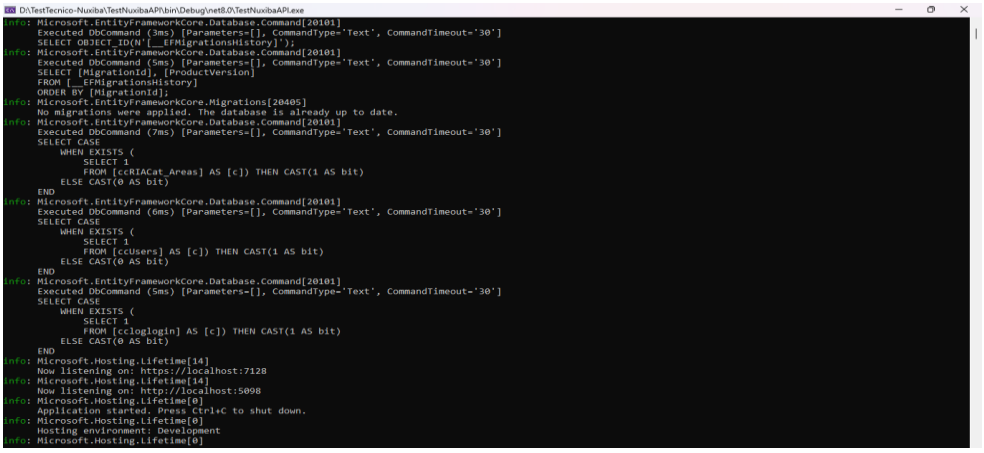
Contenedor en Dockers exitoso



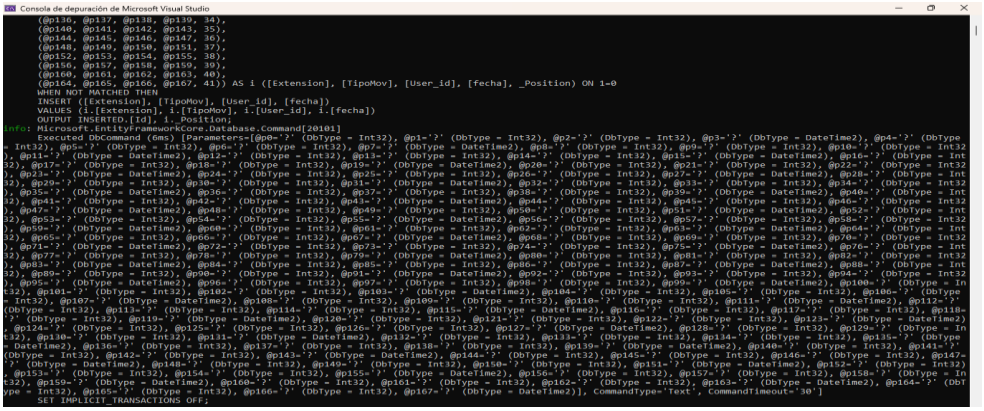
Conexión con SSMS exitosa

Ciclo de Vida de Datos: Migraciones y Poblado Automático

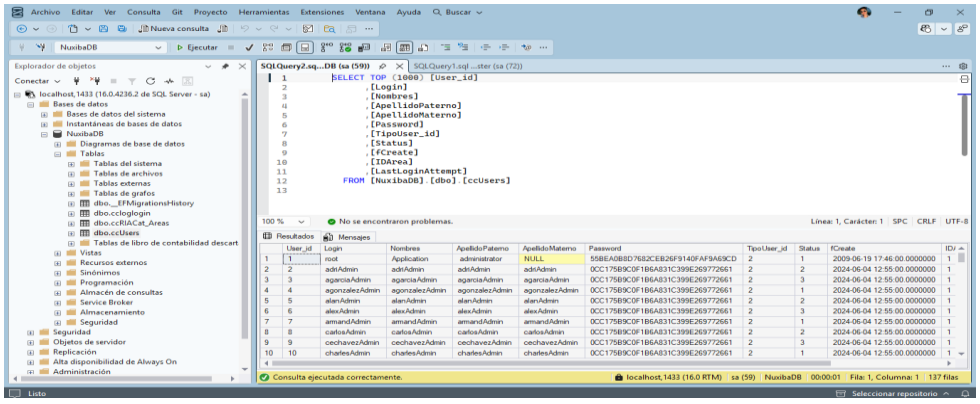
Utilizando el enfoque de **Code First**, se gestionó la creación de las tablas (ccUsers, ccRIACat_Areas, ccloglogin) mediante **Entity Framework Migrations**, asegurando la integridad referencial y el tipado de datos. Adicionalmente, se desarrolló un **DbInitializer** personalizado que automatiza la lectura de datos históricos desde un archivo Excel (usando la librería **MiniExcel**). Los resultados de este proceso se reflejan en las capturas de **SQL Server Management Studio (SSMS)**, confirmando que la base de datos se puebla correctamente sin intervención manual.



Creacion de base de datos y tablas desde la terminal .NET



Población de datos en las tablas User, Area y Login

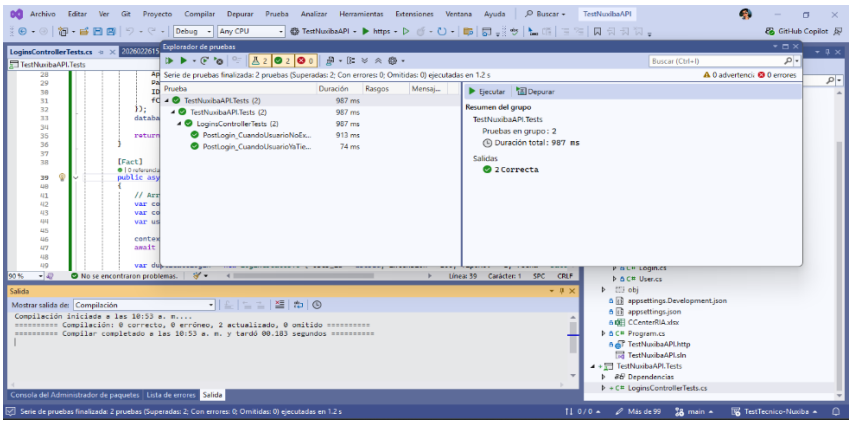


Verificación de tablas y datos en SSMS

Aseguramiento de Calidad: Pruebas Unitarias

Para garantizar la robustez de los controladores, se integró un proyecto de pruebas unitarias con **xUnit**. Se utilizó el proveedor **InMemory Database** de Entity Framework para simular el comportamiento de la base de datos en un entorno controlado y de alta velocidad.

Las pruebas validan escenarios críticos como la **secuencialidad de los movimientos** (evitar logins duplicados sin logout previo) y la validación de existencia de registros, logrando una cobertura que asegura la estabilidad del sistema ante cambios futuros.

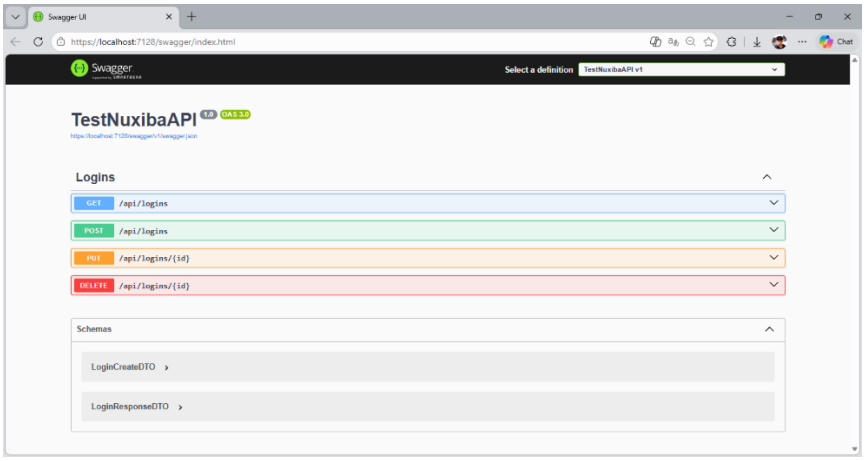


Pruebas de test xUnit exitosas

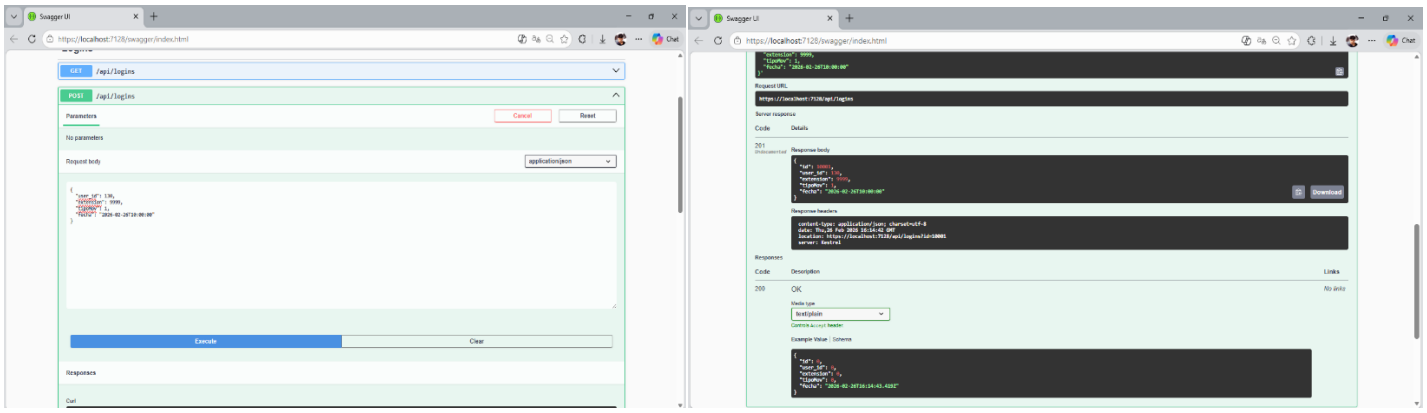
Interfaz de Consumo y Reporteo Analítico

Se documentaron los contratos de la API a través de **Swagger/OpenAPI**, permitiendo una interacción fluida con los recursos del sistema. Se incluyen evidencias de:

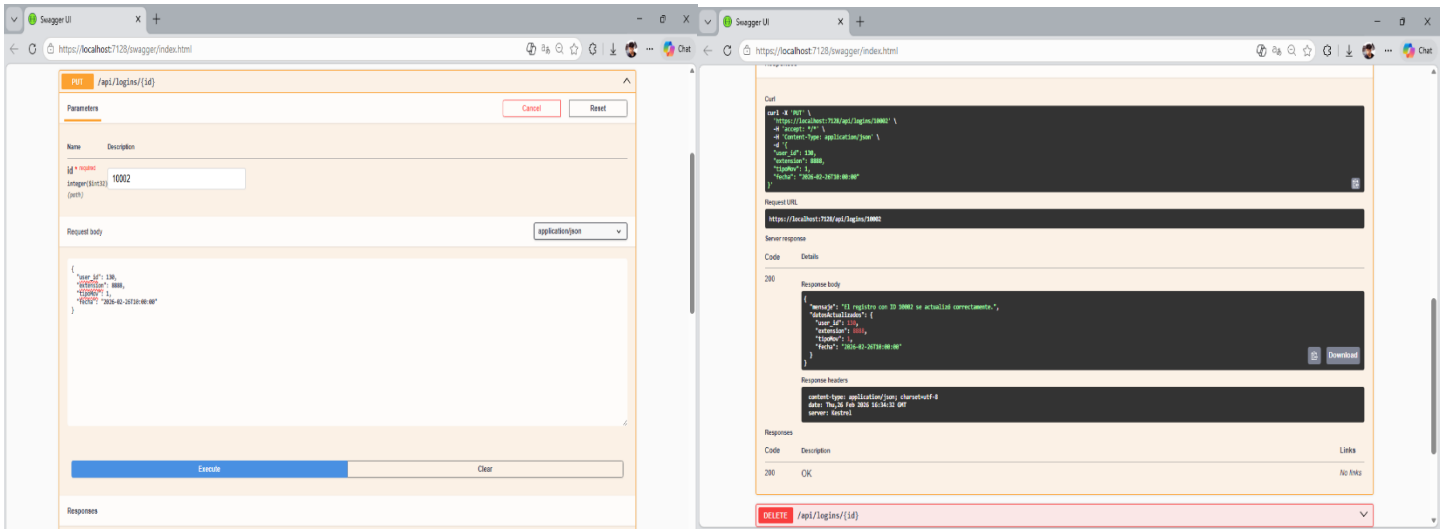
- **Operaciones CRUD:** Pruebas de creación, actualización y borrado de registros con respuestas JSON claras.
- **Módulo de Reportes:** Generación dinámica de archivos **CSV** que calculan el total de horas trabajadas por usuario, validando el procesamiento de intervalos de tiempo.
- **Consumo Externo:** Pruebas de estrés y descarga de archivos realizadas desde **Postman**, confirmando la interoperabilidad de la API con herramientas estándar de la industria."



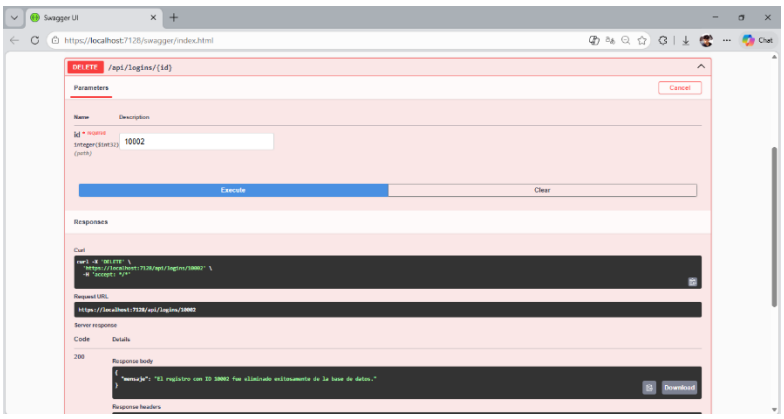
CRUD get, post, put y delete en Swagger



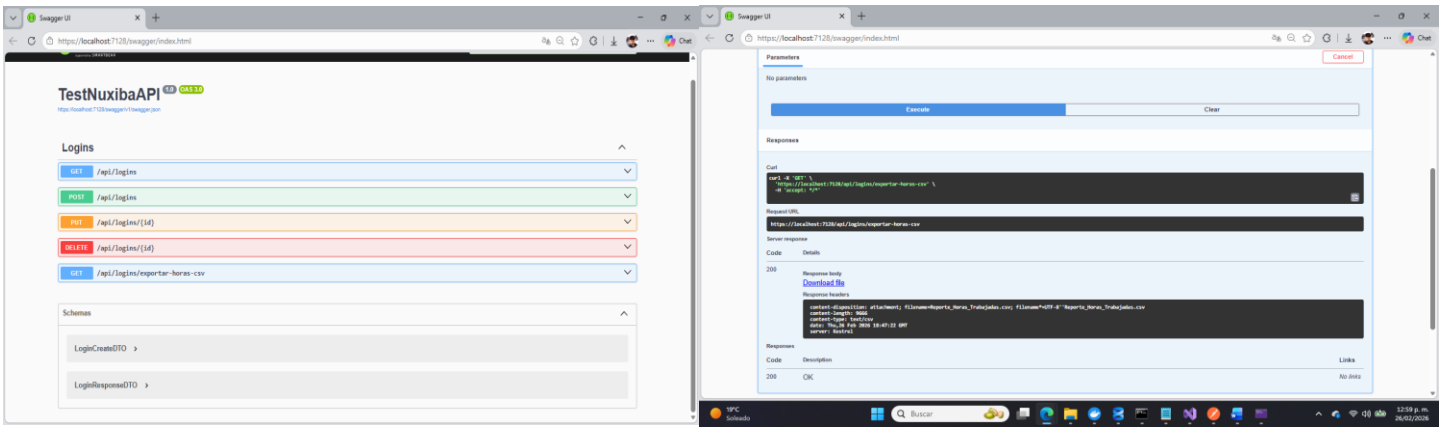
Verificación de metodo POST



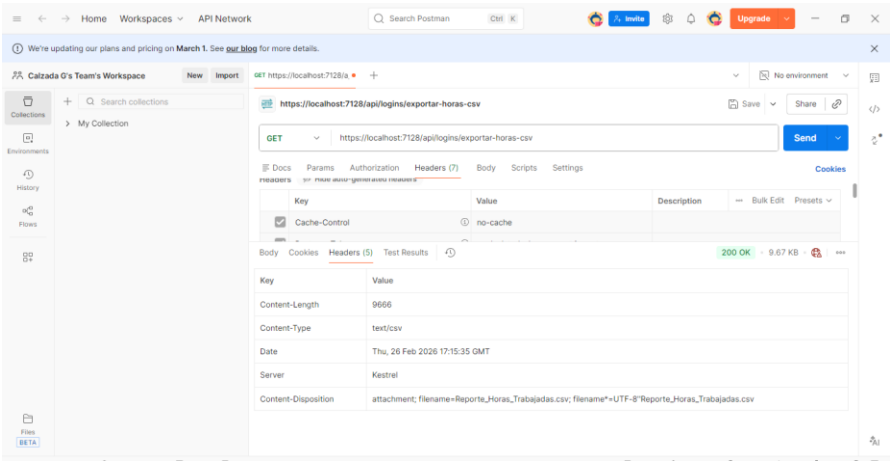
Verificación de metodo PUT



Verificación de metodo DELETE



Integración de CSV y verificación de descarga



Verificación de endpoint CSV desde POSTMAN