

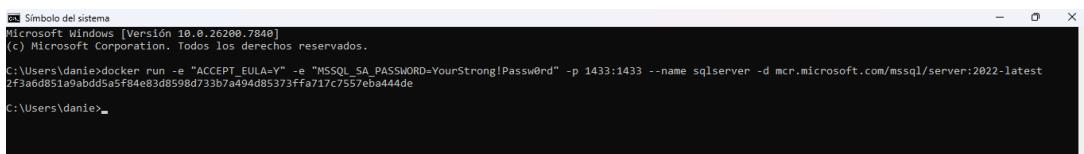
Estructura de Evidencias: Prueba Técnica Nuxiba

Información General

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

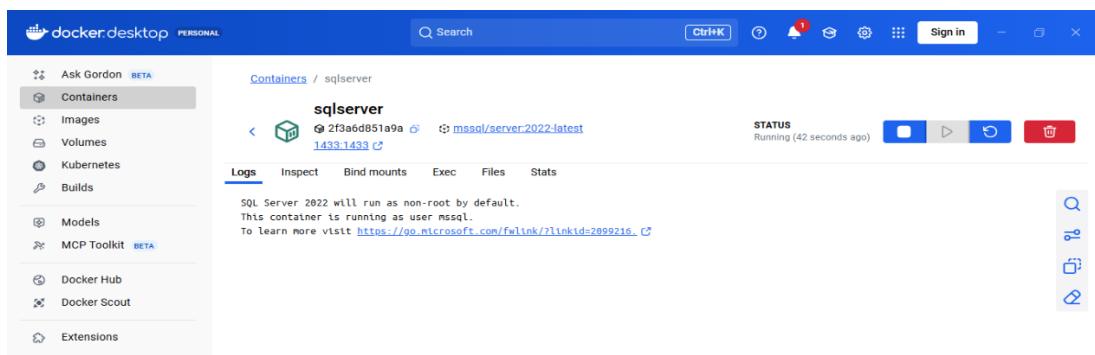
Orquestación de Infraestructura y Automatización

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.

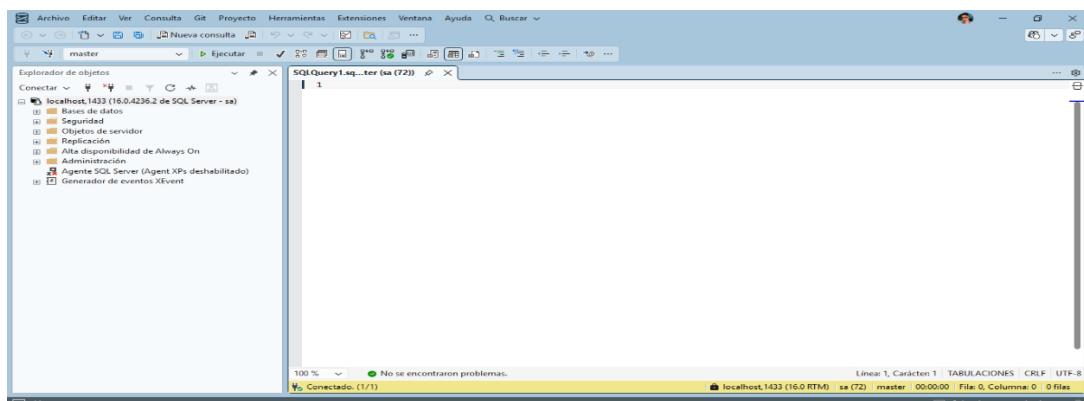


```
C:\Users\danie>docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=YourStrong!Passw0rd" -p 1433:1433 --name sqlserver -d mcr.microsoft.com/mssql/server:2022-latest
Zf3a6d851a0bdd5a5f84e83d8598d733b7a494d85373ffa717c7557eba444de
C:\Users\danie>
```

Cadena de conexión desde CMD



Contenedor en Docker exitoso



Conexión con SSMS exitosa

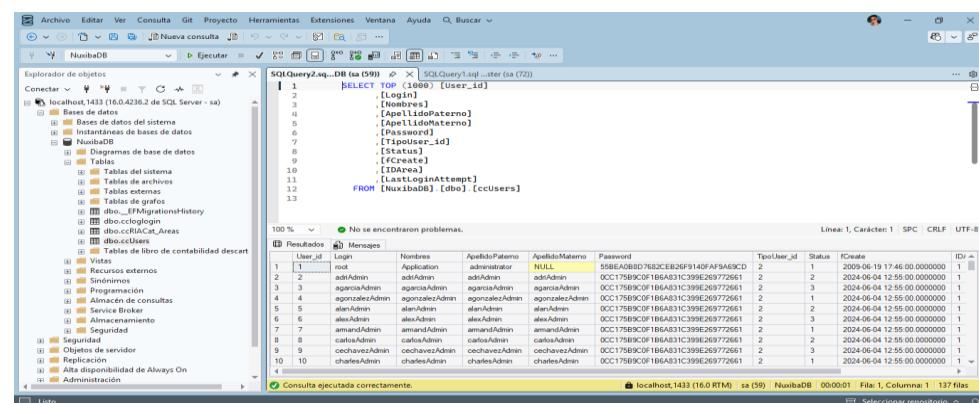
Ciclo de Vida de Datos: Migraciones y Poblado Automático

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.

```
[Info: Microsoft.EntityFrameworkCore.NuGetPackageAppliesOnDbUpgrader.OnDetermineNebulaApExe]
[Info: Executed DbCommand ([0ms] [Parameters=[], CommandType='Text', CommandTimeout='30'])
SELECT OBJECT_ID(N'[__EFMigrationsHistory]')
[Info: Microsoft.EntityFrameworkCore.NuGetPackageAppliesOnDbUpgrader.OnDetermineNebulaApExe]
[Info: Executed DbCommand ([0ms] [Parameters=[], CommandType='Text', CommandTimeout='30'])
SELECT [MigrationId], [ProductVersion]
FROM [__EFMigrationsHistory]
ORDER BY [MigrationId]
[Info: Microsoft.EntityFrameworkCore.Migrations[20405]
No migrations were applied. The database is already up to date.
[Info: Microsoft.EntityFrameworkCore.NuGetPackageAppliesOnDbUpgrader.OnDetermineNebulaApExe]
[Info: Executed DbCommand ([7ms] [Parameters=[], CommandType='Text', CommandTimeout='30'])
SELECT CASE
WHEN EXISTS (
    SELECT 1
    FROM [ccIACat_Areas] AS [c]) THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
END
[Info: Microsoft.EntityFrameworkCore.NuGetPackageAppliesOnDbUpgrader.OnDetermineNebulaApExe]
[Info: Executed DbCommand ([0ms] [Parameters=[], CommandType='Text', CommandTimeout='30'])
SELECT CASE
WHEN EXISTS (
    SELECT 1
    FROM [ccUsers] AS [c]) THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
END
[Info: Microsoft.EntityFrameworkCore.NuGetPackageAppliesOnDbUpgrader.OnDetermineNebulaApExe]
[Info: Executed DbCommand ([0ms] [Parameters=[], CommandType='Text', CommandTimeout='30'])
SELECT CASE
WHEN EXISTS (
    SELECT 1
    FROM [ccLoginLogin] AS [c]) THEN CAST(1 AS bit)
    ELSE CAST(0 AS bit)
END
[Info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:7128
[Info: Microsoft.Hosting.Lifetime[14]
Application started. Press Ctrl+C to shut down.
[Info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
[Info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
[Info: Microsoft.Hosting.Lifetime[0]
```

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

Población de datos en las tablas User, Area y Login

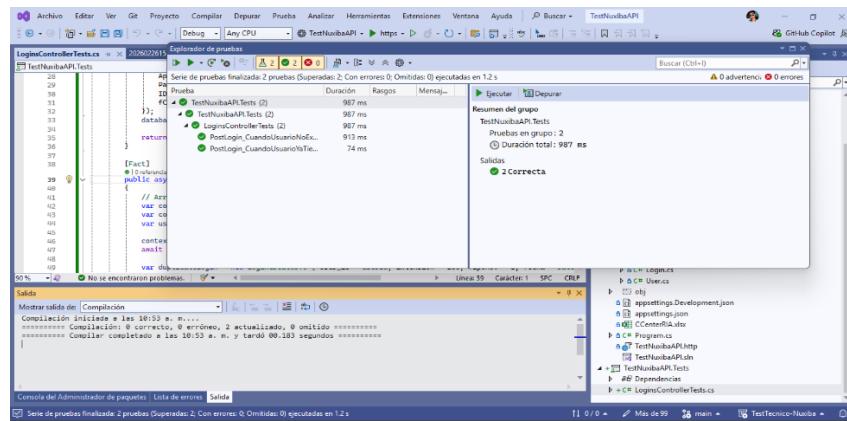


Verificación de tablas y datos en SSMS

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."

A screenshot of the Swagger UI interface, which displays the OpenAPI specification for the 'TestNuxibaAPI'. The main page shows the 'Logins' section with four operations: 'GET /api/logins' (highlighted in blue), 'POST /api/logins' (highlighted in green), 'PUT /api/logins/{id}' (highlighted in orange), and 'DELETE /api/logins/{id}' (highlighted in red). Below the operations, there are sections for 'Schemas' containing 'LoginCreateDTO' and 'LoginResponseDTO'. The URL in the browser is https://localhost:7125/swagger/index.html.

CRUD get, post, put y delete en Swagger

The image shows two side-by-side browser windows, both titled "Swagger UI".

Left Window: Shows the "POST /api/logins" interface. It has a "Parameters" section with "No parameters" and a "Request body" section containing a JSON object:

```
{
  "user_id": 138,
  "username": "mario",
  "password": "123456"
}
{
  "user_id": 138,
  "username": "mario",
  "password": "123456"
}
```

Right Window: Shows the results of a successful POST request. The "Request URL" is <https://localhost:7128/api/logins>. The "Server response" shows a 201 status code with the following JSON body:{
 "id": 10002,
 "username": "mario",
 "password": "123456",
 "user_id": 138,
 "created_at": "2024-02-28T10:00:00"
}

Verificación de método POST

The image shows two side-by-side browser windows, both titled "Swagger UI".

Left Window: Shows the "PUT /api/logins/{id}" interface. It has a "Parameters" section with "id" set to "10002" and a "Request body" section containing a JSON object:{
 "user_id": 138,
 "username": "mario",
 "password": "123456"
}

Right Window: Shows the results of a successful PUT request. The "Request URL" is <https://localhost:7128/api/logins/10002>. The "Server response" shows a 200 status code with the following JSON body:{
 "message": "El registro con ID 10002 se actualizó correctamente."
}

Verificación de método PUT

The image shows a single browser window titled "Swagger UI".

It displays the "DELETE /api/logins/{id}" interface. The "Parameters" section shows "id" set to "10002".

The "Responses" section shows a 200 status code with the following JSON body:{
 "message": "El registro con ID 10002 fue eliminado exitosamente de la base de datos."
}

Verificación de método DELETE

Integración de CSV y verificación de descarga

Verificación de endpoint CSV desde POSTMAN