

Universidad Tecnológica Metropolitana

Aplicaciones web para 14.0
MSTRO. May Tuz Diego

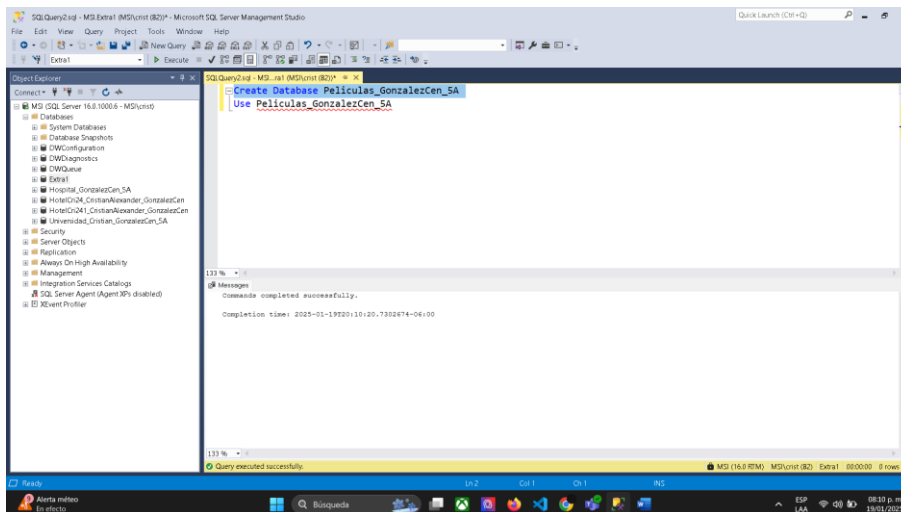
Cuatrimestre 5A

Parcial 1

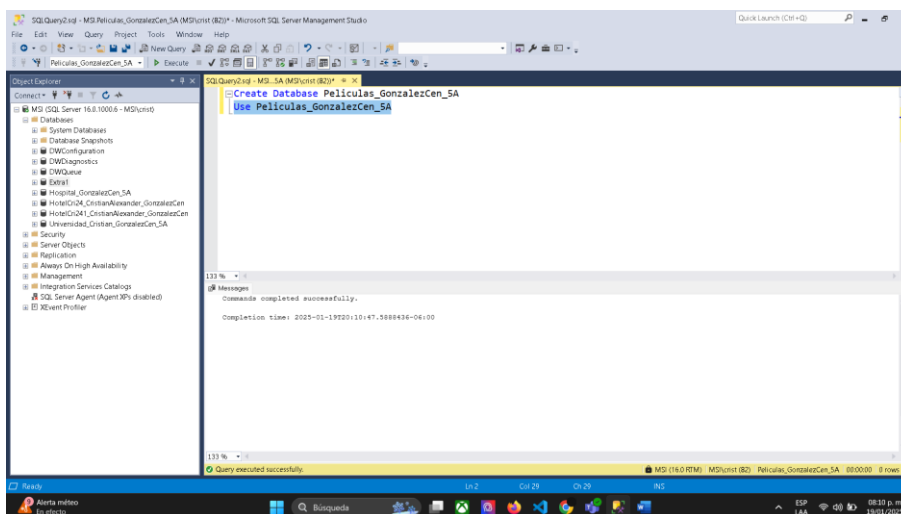
Alumno. Cristian Alexander González Cen

Act 3
API-Peliculas

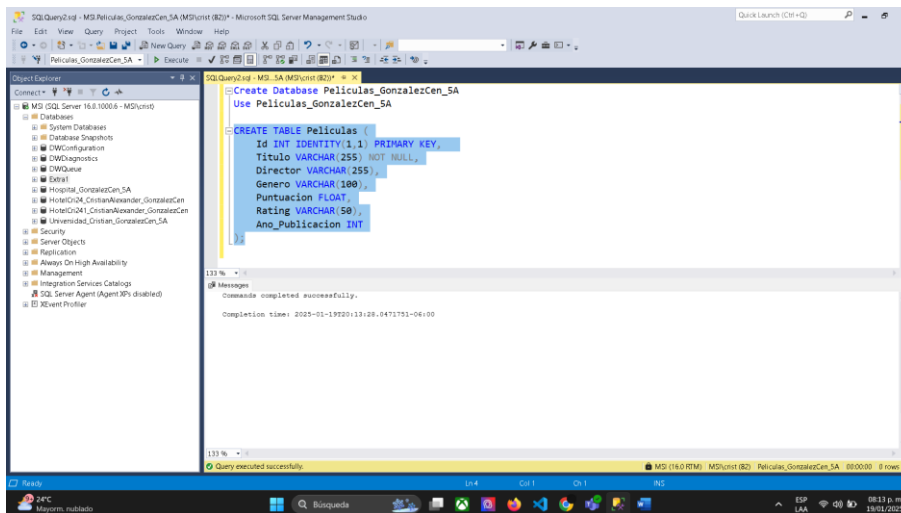
Primero vamos a crear una base de datos con una tabla de películas y algunos datos que le vamos a inyectar. Para crear una BD utilizamos el comando Create Database y ponemos el nombre de la BD y le damos en Execute.



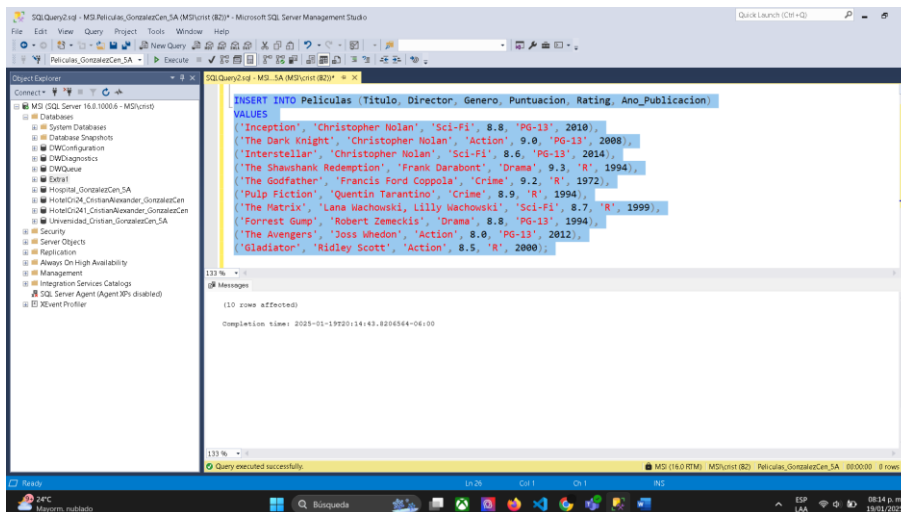
Después de que se cree la BD, vamos a usar el comando use para ponerla en uso y no confundirse de BD.



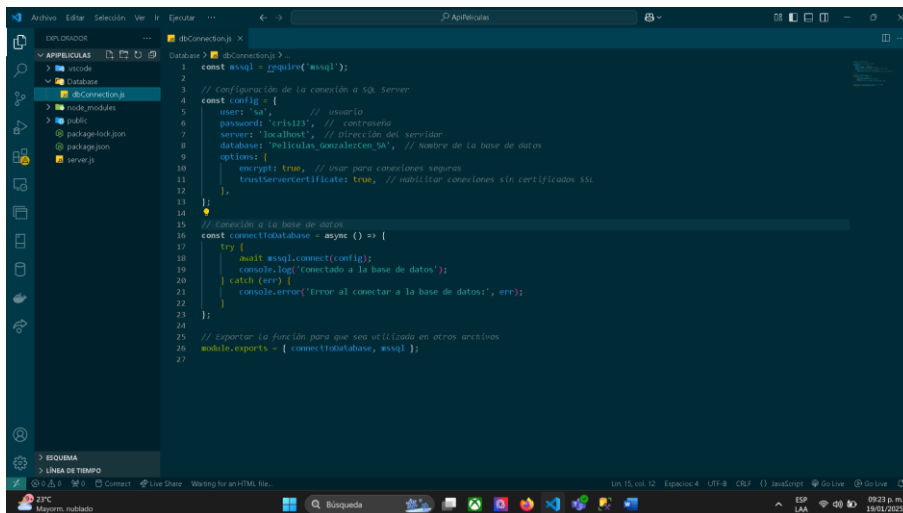
Luego vamos a crear una tabla con los atributos que nos piden, pero le vamos a agregar uno más que es el de id que va a servir como primary key para una película para la tabla Peliculas



Luego de que la tabla se cree vamos a ingresarle datos para que no esté vacía y podamos hacer pruebas más adelante.

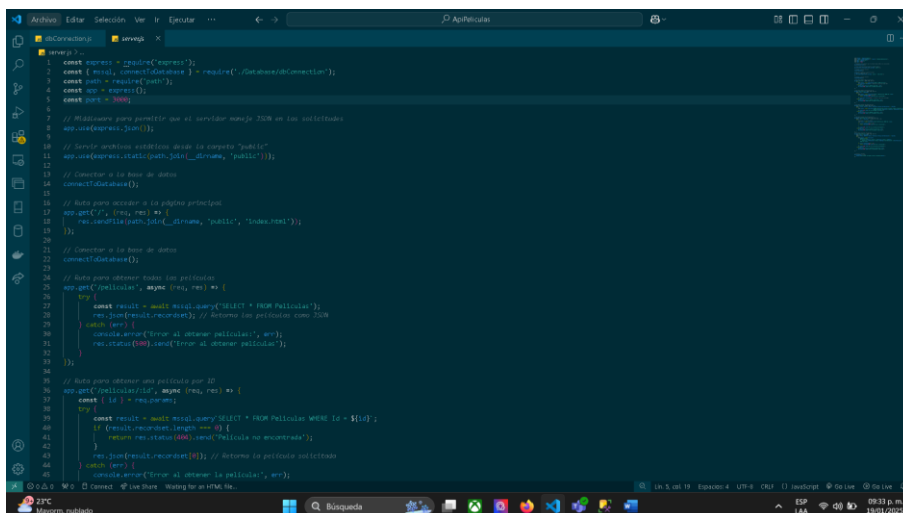


Ahora vamos con la parte de los archivos para crear la api, la api como tiene que extraer información de algún lado, primero hacemos la conexión a la base de datos ya que la creamos y ya le pusimos datos.



```
1 const mysql = require('mysql');
2
3 // Configuración de la conexión a SQL Server
4 const config = {
5   user: 'sa', // usuario
6   password: 'P@ssw0rd!', // contraseña
7   server: 'localhost', // dirección del servidor
8   database: 'Peliculas_GonzalezSA', // nombre de la base de datos
9   options: {
10     encrypt: true, // usar para conexiones seguras
11     trustServerCertificate: true, // inhabilitar conexiones sin certificados SSL
12   },
13 };
14
15 // conexión a la base de datos
16 const connectDatabase = async () => {
17   try {
18     await mysql.connect(config);
19     console.log('Conectado a la base de datos');
20   } catch (err) {
21     console.error('Error al conectar a la base de datos', err);
22   }
23 };
24
25 // Exportar la función para que sea utilizada en otros archivos
26 module.exports = { connectDatabase, mysql };
27
```

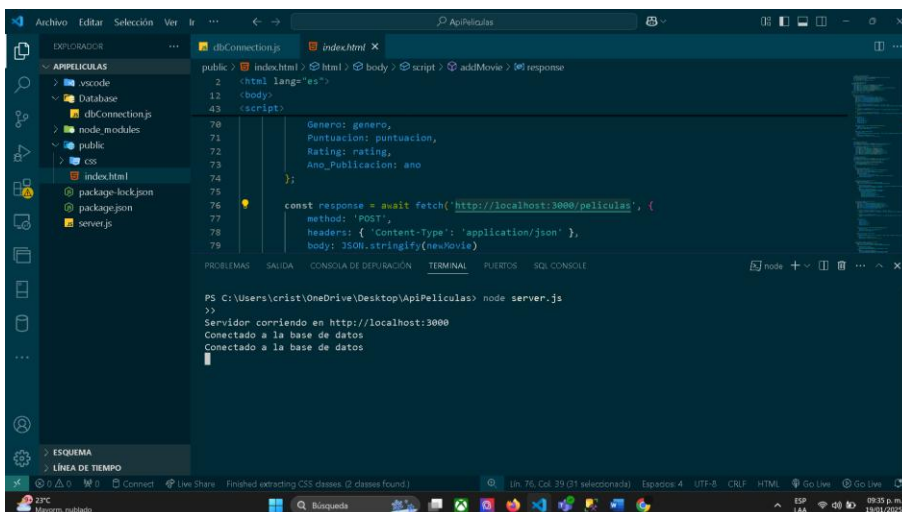
Después de hacer la conexión a la BD, ahora necesitamos hacer métodos para que reciba la BD y haga lo que se le pide, para eso lo hacemos en forma de query junto con JavaScript, para eso creamos un archivo llamado server.js en el que vamos a agregar métodos para agregar, ver, actualizar y borrar películas y también nos va a servir para que tengamos un servidor de manera local para poder acceder a la base de datos siempre y cuando esté corriendo el server de manera correcta.



```
1 const express = require('express');
2 const { mysql, connectDatabase } = require('./dbConnection');
3 const app = express();
4 const port = 3000;
5
6 // Middleware para permitir que el servidor escuche 2000 en las solicitudes
7 app.use(express.json());
8
9 // Servir archivos estáticos desde la carpeta 'public'
10 app.use(express.static(path.join(__dirname, 'public')));
11
12 // Conectar a la base de datos
13 connectDatabase();
14
15 // Ruta para acceder a la página principal
16 app.get('/', req, res => {
17   res.sendFile(path.join(__dirname, 'public', 'index.html'));
18 });
19
20 // Conectar a la base de datos
21 connectDatabase();
22
23 // Ruta para obtener todas las películas
24 app.get('/peliculas', async (req, res) => {
25   try {
26     const result = await mysql.query('SELECT * FROM Peliculas');
27     res.json(result.records); // Retorno las películas como JSON
28   } catch (err) {
29     console.error('Error al obtener películas', err);
30     res.status(500).send('Error al obtener películas');
31   }
32 });
33
34 // Ruta para obtener una película por ID
35 app.get('/peliculas/:id', async (req, res) => {
36   const { id } = req.params;
37   try {
38     const result = await mysql.query('SELECT * FROM Peliculas WHERE id = ?[id]');
39     if (result.records.length === 0) {
40       res.status(404).send('Película no encontrada');
41     }
42     res.json(result.records[0]); // Retorno la película solicitada
43   } catch (err) {
44     console.error('Error al obtener la película', err);
45   }
46 });
47
```

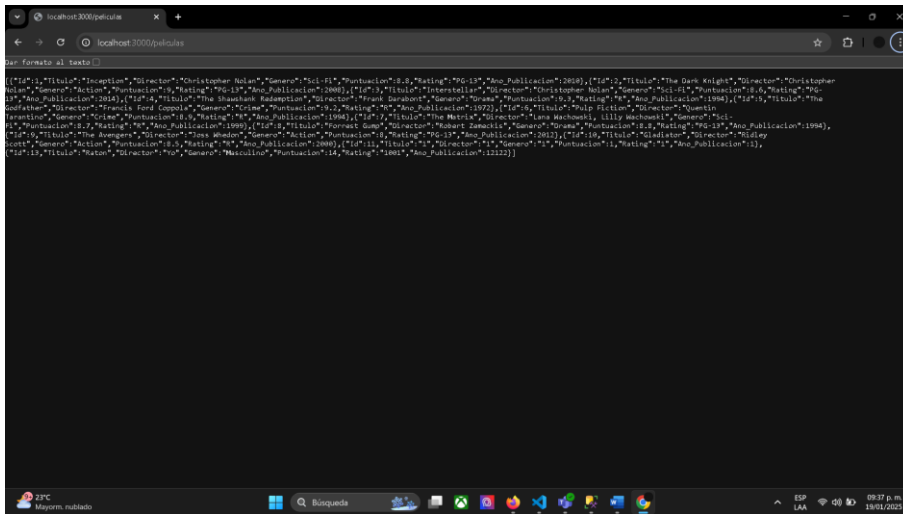
```
42 // Solo para eliminar una película por ID
43
44 app.delete('/peliculas/:id', async (req, res) => {
45   const { id } = req.params; // Obtener el ID de la ID
46   try {
47     // Eliminar la película DELETE
48     const result = await mysql.query('DELETE FROM Peliculas WHERE id = ?')(id);
49     if (result.affectedRows === 0) {
50       console.error('Error al eliminar la película', err);
51       return res.status(404).send('Película no encontrada');
52     }
53     res.status(200).send('Película eliminada exitosamente');
54     console.log('');
55     console.error('Error al eliminar la película', err);
56     res.status(500).send('Error al eliminar la película');
57   }
58 }
59
60 // Solo para actualizar una película
61 app.put('/peliculas/:id', async (req, res) => {
62   const { id } = req.params;
63   const { titulo, director, genero, puntuacion, rating, ano_publicacion } = req.body;
64   try {
65     // Actualizar la película UPDATE
66     const result = await mysql.query(
67       'UPDATE Peliculas
68       SET titulo = ?, director = ?, genero = ?, puntuacion = ?, rating = ?, ano_publicacion = ?
69       WHERE id = ?')(
70       titulo, director, genero, puntuacion, rating, ano_publicacion, id);
71     if (result.affectedRows === 0) {
72       console.error('Error al actualizar la película', err);
73       return res.status(404).send('Película no encontrada');
74     }
75     res.status(200).send('Película actualizada exitosamente');
76     console.log('');
77     console.error('Error al actualizar la película', err);
78     res.status(500).send('Error al actualizar la película');
79   }
80 }
81
82 // Iniciar el servidor
83 app.listen(3000, () => {
84   console.log('Servidor corriendo en http://localhost:3000');
85 });
86
87 4 de 6 27°C Mayoría nublado
```

Después de hacer eso hacemos una prueba para ver si funciona lo que hemos hecho. Primero corremos el servidor con el comando `node server.js` en el cual pusimos como mensaje de que ya se levanto la ruta del server, en el cual vamos a darle click para ver los datos en forma de Json, si todo funcionó es que si van a salir los datos y si no funcionó no vamos a poder ver nada

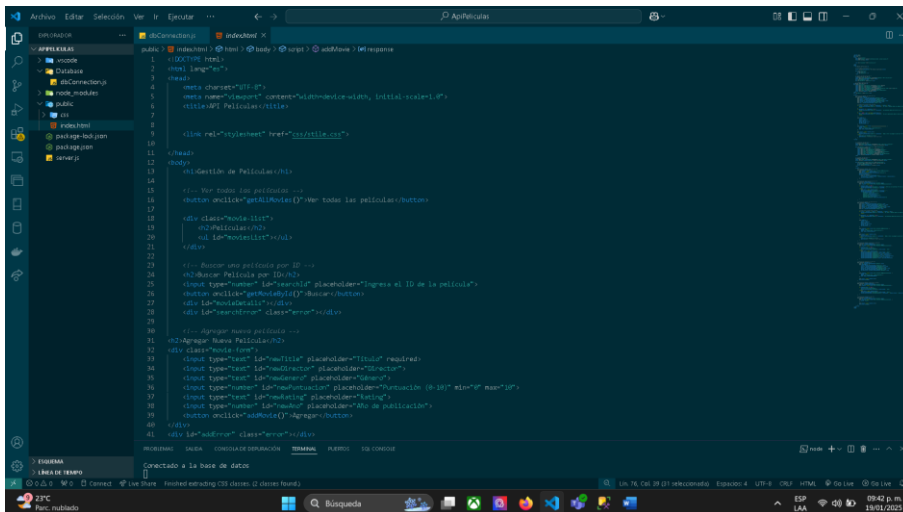


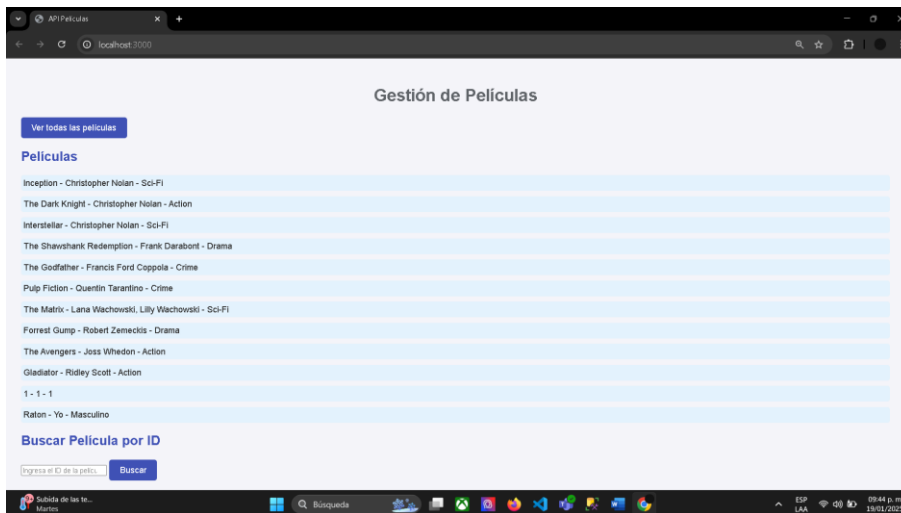
The screenshot shows a VS Code editor with a project named 'ApiPelículas'. The file explorer on the left shows the project structure, including 'dbConnections.js', 'index.html', 'package-lock.json', 'package.json', and 'server.js'. The main editor area shows the 'index.html' file, which contains a REST client request to 'http://localhost:3000/peliculas'. The request is a POST method with a JSON body: `{ "titulo": "Nuevo Pelicula", "director": "Nuevo Director", "genero": "Nuevo Genero", "puntuacion": "Nuevo Puntuacion", "rating": "Nuevo Rating", "ano_publicacion": "Nuevo Ano" }`. The response is a JSON object: `{ "titulo": "Nuevo Pelicula", "director": "Nuevo Director", "genero": "Nuevo Genero", "puntuacion": "Nuevo Puntuacion", "rating": "Nuevo Rating", "ano_publicacion": "Nuevo Ano" }`. The terminal window at the bottom shows the command `node server.js` being executed, and the output is: `Server corriendo en http://localhost:3000`, `Conectado a la base de datos`, and `Conectado a la base de datos`.

En nuestro caso todo funcionó ya que podemos ver los datos de la base de datos

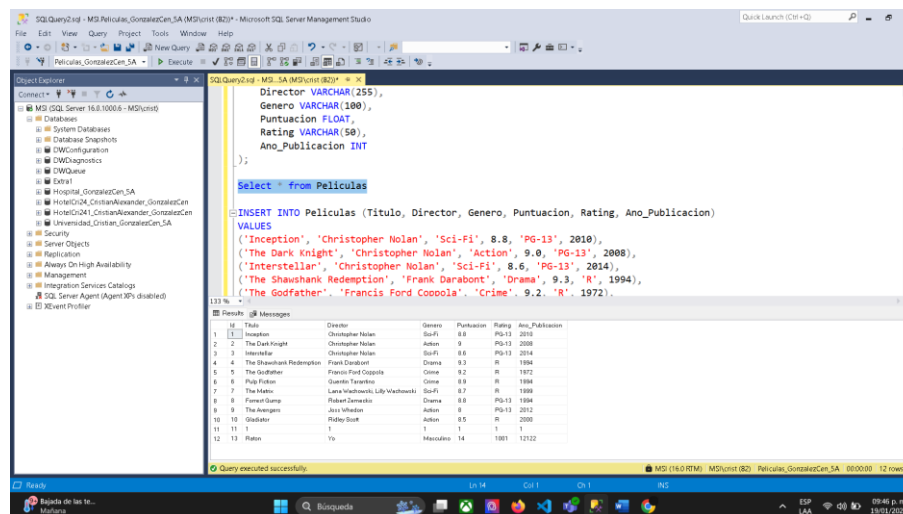


Como no se entiende nada de los datos, vamos a crear un archivo html para que se vea mejor la información y luego le agregamos diseño. Para eso cree una carpeta llamada public y dentro de esa está un archivo index.html que es el que tiene la forma en que se va a ver la información



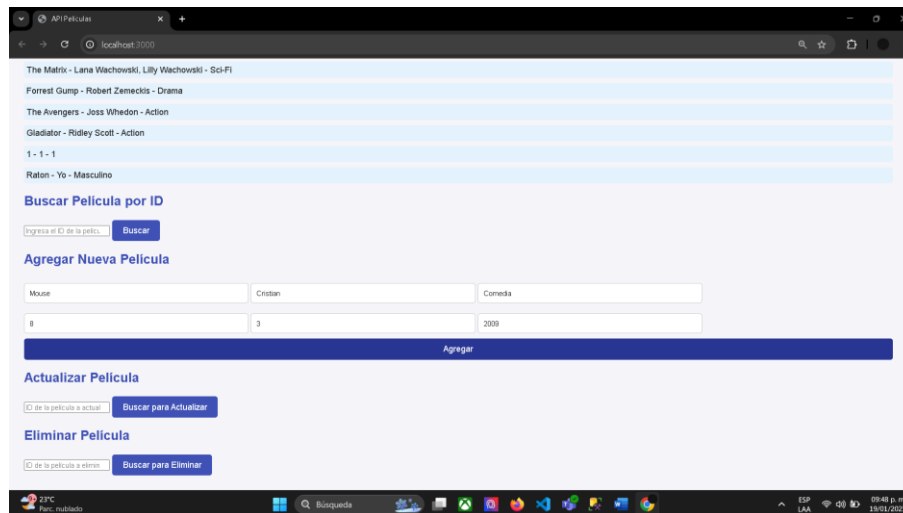


Para corroborar que es correcto vamos a la BD y hacemos un select * from

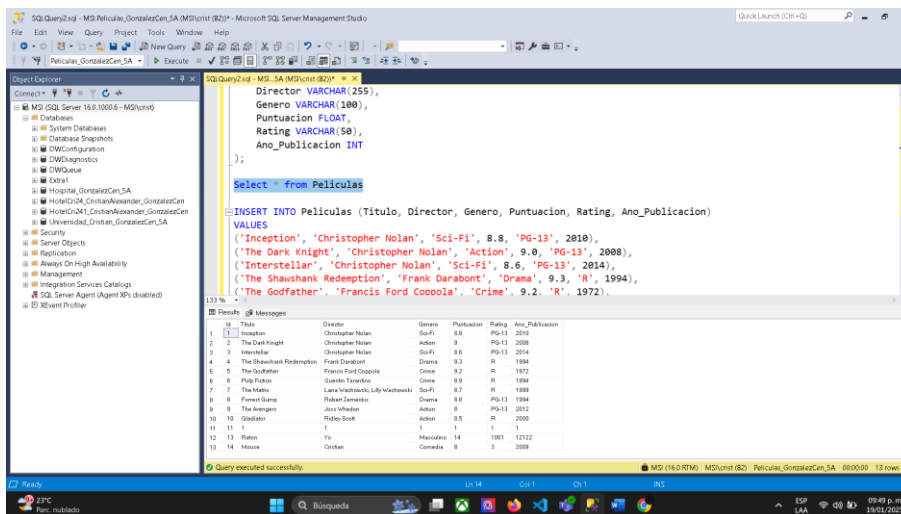


Como se alcanza a ver es igual a lo que nos arrojó desde el html, eso quiere decir que todo está bien.

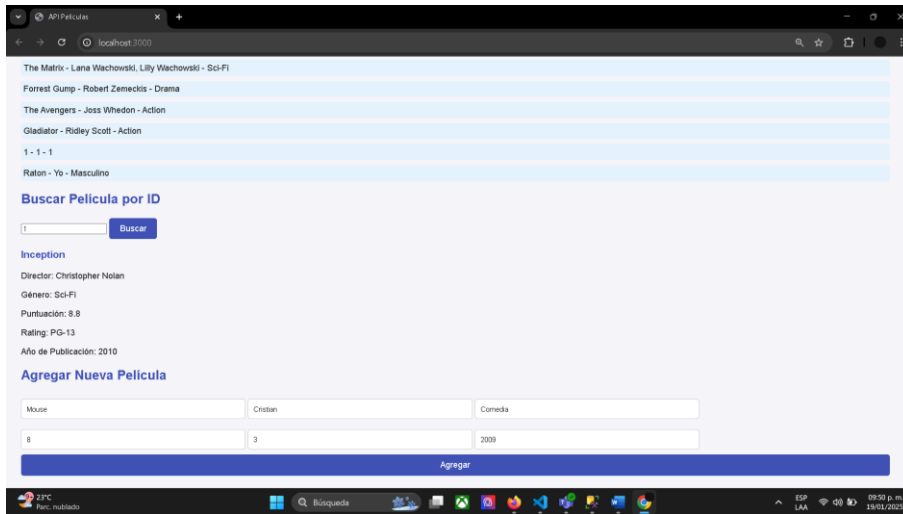
Ahora vamos a hacer la prueba agregar una película, para eso llenamos el formulario para agregar una película y luego le damos en agregar



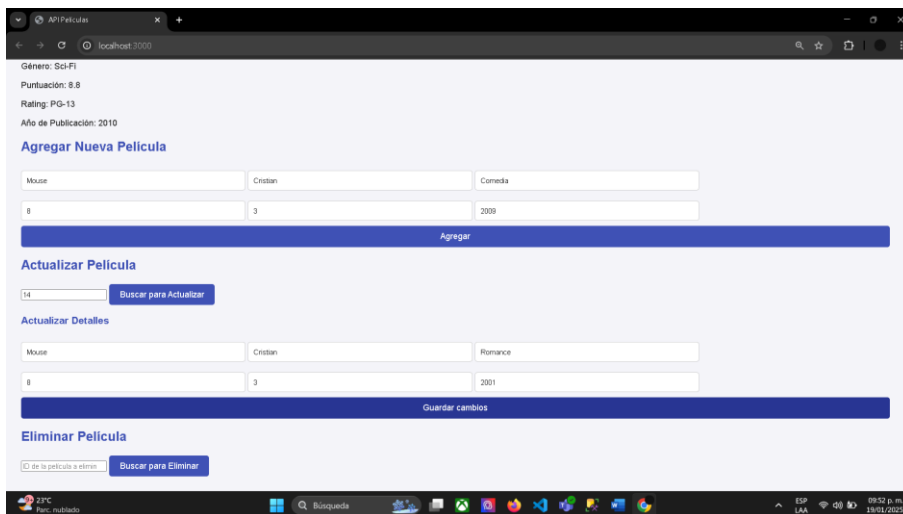
Una vez que ya hemos agregado la película vamos a comprobarlo en la BD haciendo de nuevo un select * from y como se ve en la imagen acaba de agregarse la película



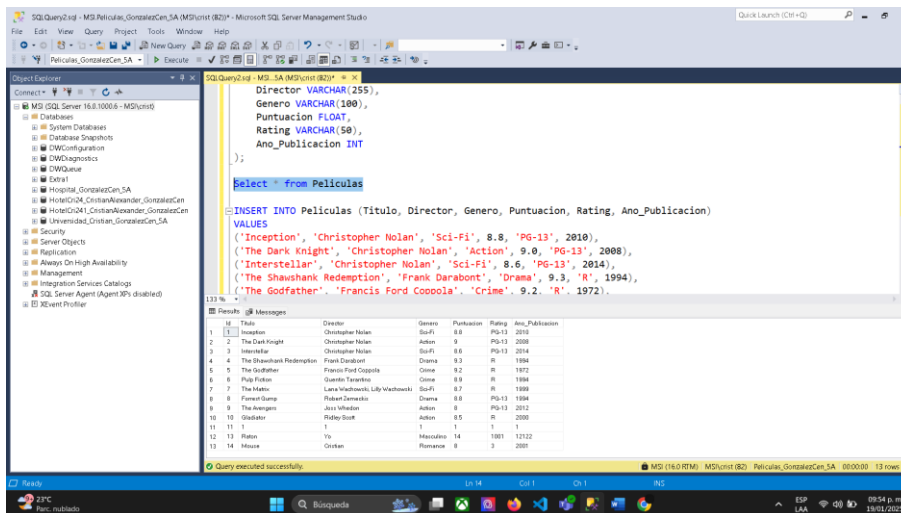
Ahora vamos a buscar una película por medio de la ID y como se ve en la imagen esto es correcto



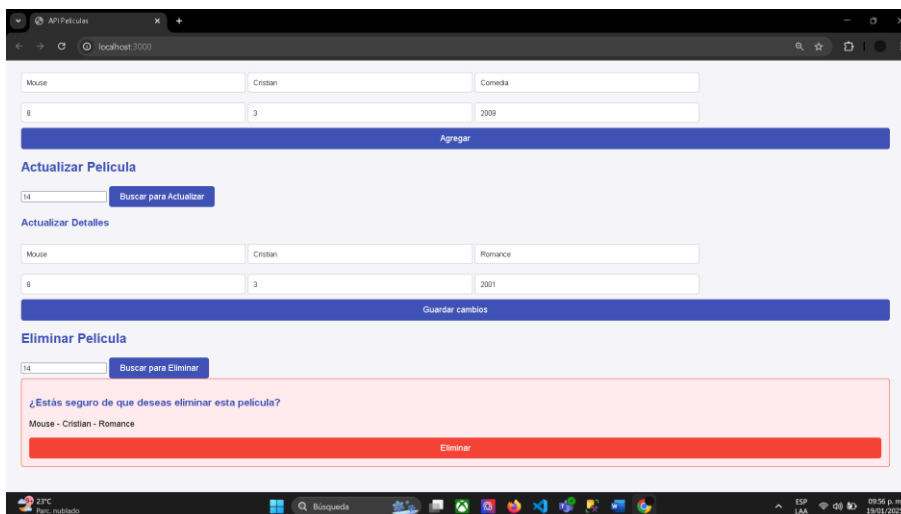
Ahora vamos a actualizar alguna película y para eso vamos a elegir la película que agregamos, para eso primero se busca la película por medio de la ID y luego aparecerá los datos de la película el cual cambiamos el año de 2009 por 2001, el genero de comedia a romance y luego le damos en agregar



Para saber que salió bien lo comprobamos con la BD y como se ve en la imagen todo salió bien



Ahora vamos con la de eliminar, para eso vamos a eliminar la película que actualizamos, pero primero se tiene que buscar por medio de la ID que es 14 y luego nos aparecerá el botón para eliminar la película. Cuando aparece el botón para eliminar aparece el título y el género de la película. Ahora le damos en eliminar y luego vamos a comprobarlo con la BD



Y como se ve en la imagen fue eliminada con éxito

