

**CONTROL DE VERSIONES Y  
MIGRACIONES DE BDD CON EVOLVE Y  
.NET CORE**

## Guia de Contenido

Requerimientos .....	3
Configurando base de datos POSTGRES-SQL con docker .....	3
Configurando el proyecto usando .net core 3.1 y Rider .....	5
Instalando y Configurando Evolve.....	10
Evaluando cambios en la base de datos .....	13
Observaciones importantes.....	14

## Requerimientos

1. Docker +v.18
2. Docker-Compose + 1.18
3. IDE (Rider , Visual Code, Visual Studio)
4. DataBase Managment (PostgreSQL, DbBeaver, Data Grip)
5. Repositorio Github
  - o <https://github.com/vicentesuc/techcommunityday>
6. Terminal: (Cmd, Terminal (Linux), Windows Terminal)

## Configurando base de datos POSTGRES-SQL con docker

1. Ir al directorio **./docker/postgres**
  - a. docker-compose.yml

```
version: '3'

services:
  postgres-test:
    build: .
    container_name: postgres-test
    restart: always
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
    ports:
      - "5432:5432"
```

- i.
- b. .Dockerfile

```
FROM postgres:11-alpine

LABEL maintainer vicentex360@gmail.com

EXPOSE 4848 8080 8181
```

- i.
2. Usando docker-compose run
    - a. **docker-compose up**
  3. Construcción de la imagen
    - a. Se comenzará a construir la imagen con una serie de pasos

```
Building techday
Step 1/3 : FROM postgres:11-alpine
--> 78b21f6420c0
Step 2/3 : LABEL maintener vicentex360@gmail.com
--> Running in c1caaf0d1e9f
Removing intermediate container c1caaf0d1e9f
--> 127450bfb34c
Step 3/3 : EXPOSE 4848 8080 8181
--> Running in fc03e52964f1
Removing intermediate container fc03e52964f1
--> 633453589f62
Successfully built 633453589f62
Successfully tagged postgres_techday:latest
WARNING: Image for service techday was built because it did not
Creating database_test ... done
Attaching to database_test
```

- i.
- b. Seguido se mostrará un log para mostrar la construcción del usuario y base de datos.

```
database_test | Data page checksums are disabled.
database_test |
database_test | fixing permissions on existing directory /var/lib/postgresql/data ... ok
database_test | creating subdirectories ... ok
database_test | selecting default max_connections ... 100
database_test | selecting default shared_buffers ... 128MB
database_test | selecting default timezone ... UTC
database_test | selecting dynamic shared memory implementation ... posix
database_test | creating configuration files ... ok
database_test | running bootstrap script ... ok
database_test | performing post-bootstrap initialization ... sh: locale: not found
database_test | 2020-06-11 03:10:09.480 UTC [26] WARNING: no usable system locales were found
database_test | ok
database_test | syncing data to disk ... ok
database_test |
database_test | Success. You can now start the database server using:
database_test |
database_test | pg_ctl -D /var/lib/postgresql/data -l logfile start
database_test |
database_test | WARNING: enabling "trust" authentication for local connections
database_test | You can change this by editing pg_hba.conf or using the option -A, or
database_test | --auth-local and --auth-host, the next time you run initdb.
database_test | waiting for server to start...2020-06-11 03:10:13.233 UTC [30] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
database_test | 2020-06-11 03:10:13.532 UTC [31] LOG: database system was shut down at 2020-06-11 03:10:09 UTC
database_test | 2020-06-11 03:10:13.622 UTC [30] LOG: database system is ready to accept connections
database_test |
database_test | done
database_test | server started
database_test | CREATE DATABASE
database_test |
database_test | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
database_test |
database_test | waiting for server to shut down...2020-06-11 03:10:15.794 UTC [30] LOG: received fast shutdown request
database_test | 2020-06-11 03:10:15.877 UTC [30] LOG: aborting any active transactions
database_test | 2020-06-11 03:10:15.879 UTC [30] LOG: background worker "logical replication launcher" (PID 37) exited with exit code 1
database_test | 2020-06-11 03:10:15.879 UTC [32] LOG: shutting down
```

- i.
- c. En cuanto finalice deberá mostrar el siguiente mensaje, y la base de datos está lista para aceptar conexiones

```
2020-06-11 03:10:16.819 UTC [1] LOG: database system is ready to accept connections
```

#### 4. Revisión de conexión

- a. Para esto pueden usar cualquier cliente de base de datos con soporte para postgres en mi caso estoy usando Data Grip

Name: videogames@localhost Reset

Comment:

General Options SSH/SSL Schemas Advanced

Connection type: default Driver: PostgreSQL

Host: localhost Port: 5432

User: admin

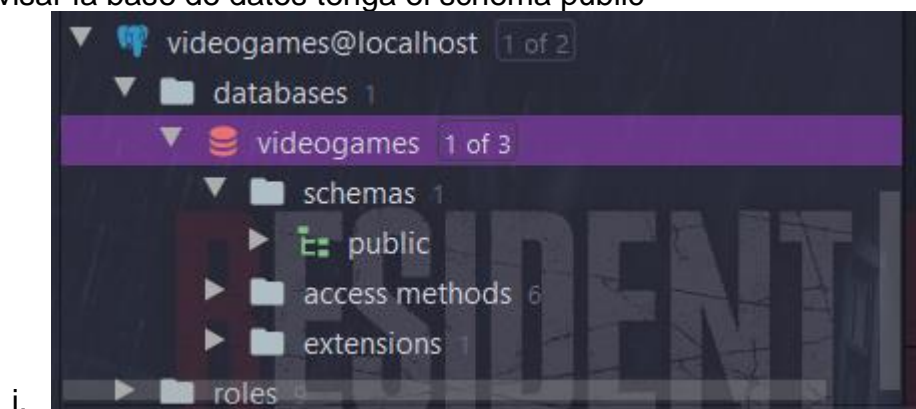
Password: <hidden> Save: Forever

Database: videogames

URL: jdbc:postgresql://localhost:5432/videogames  
Overrides settings above

✓ DBMS: PostgreSQL (ver. 11.5)  
Case sensitivity: plain=lower, delimited=exact  
Driver: PostgreSQL JDBC Driver (ver. 42.2.5, JDBC4.2)  
Ping: 18 ms  
SSL: no

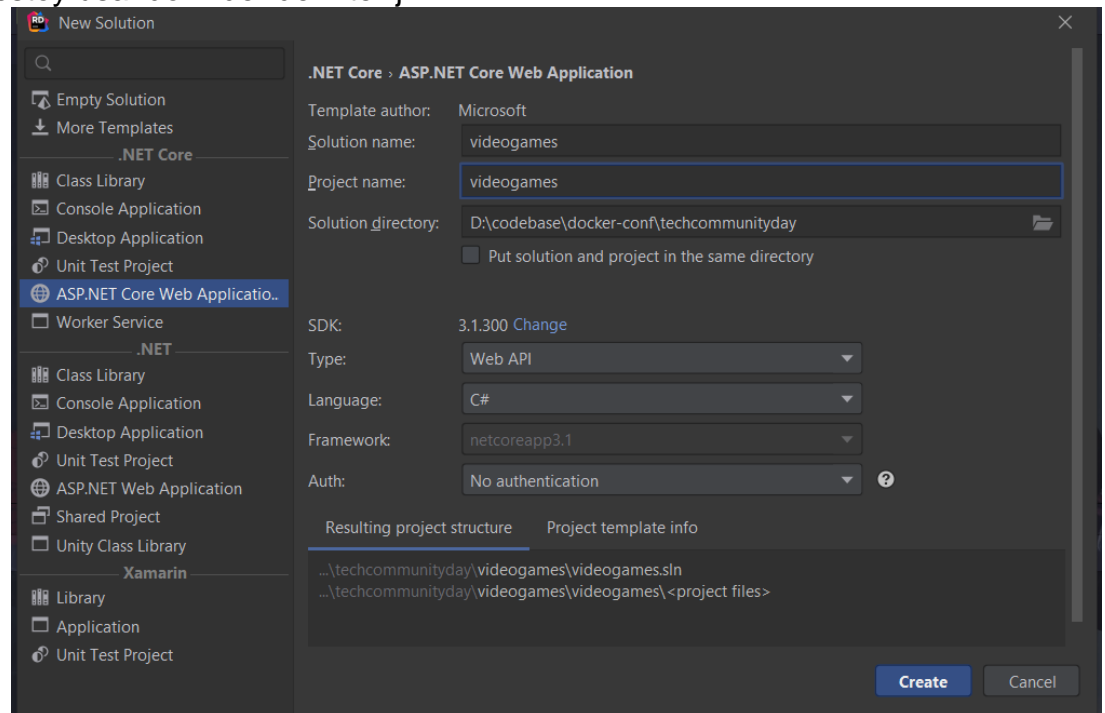
- i.
- b. Revisar la base de datos tenga el schema public



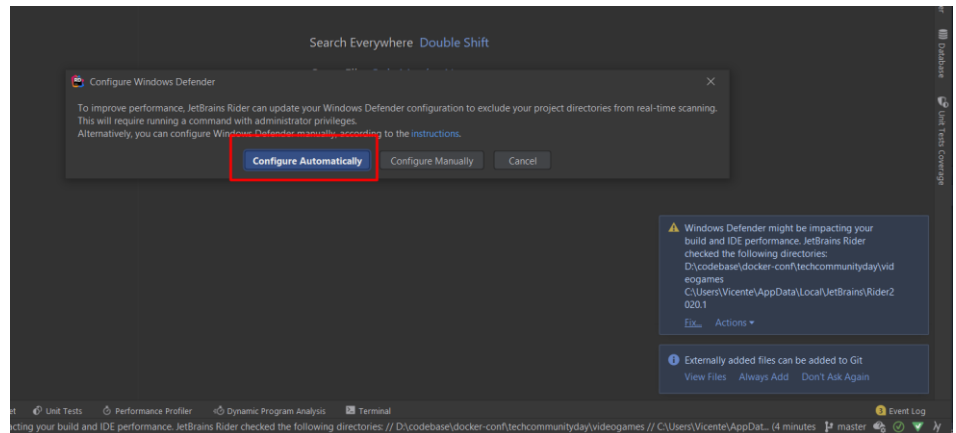
## Configurando el proyecto usando .net core 3.1 y Rider

- 1. Para instalar .net core pueden usar el siguiente enlace
  - a. <https://dotnet.microsoft.com/download>

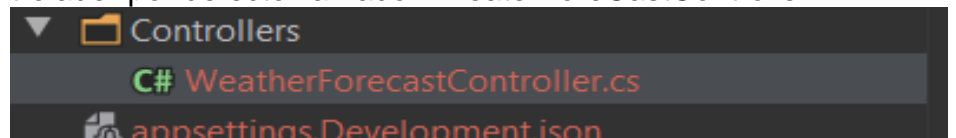
2. Para iniciar un proyecto pueden usar Visual Code, Visual Studio , em mi caso estoy usando Rider de IntelliJ.



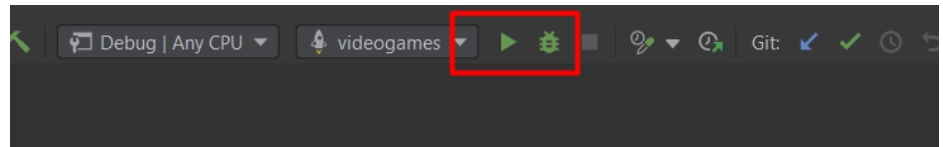
- a.
  - i. SDK : 3.1
  - ii. Type : Web Api
  - iii. C#
  - iv. Sin Autenticación (opcional)
- b. Se usará DDD para control de archivos
  - i. Crear carpeta model
    1. Almacenara las clases modelo
  - ii. Crear carpeta repositorio
    1. Almacenara las interfaces
    2. Almacenara los repositorios
  - iii. Crear Carpeta controller
    1. Almacenera los controller de nuestras API
  - iv. Crear carpeta útil
    1. Almacenara funciones, DbContext
- c. Es muy probable que si usas Windows 10 y tengas activo Windows defender muestre un error de permisos, pero se resuelve presionando en fix , luego configurar automáticamente y por último aceptar.



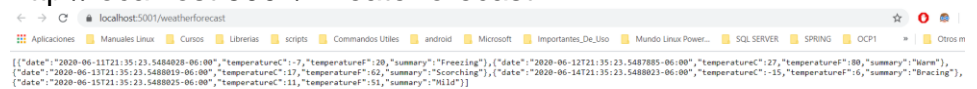
- i.
- d. Si iniciamos nuestro proyecto por primera vez el proyecto mostrara un controlador por defecto llamado WheaterForeCastController.



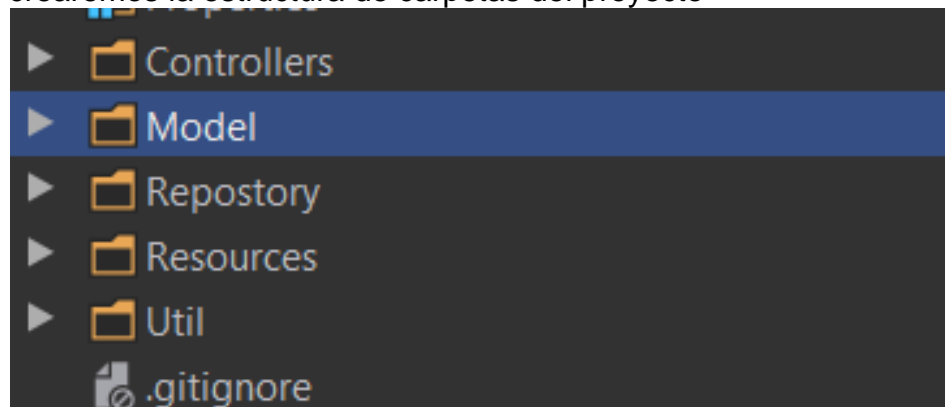
- i.
- e. Para comprobar que nuestro proyecto funciona correctamente hasta el momento , presionamos play en la parte superior del proyecto.



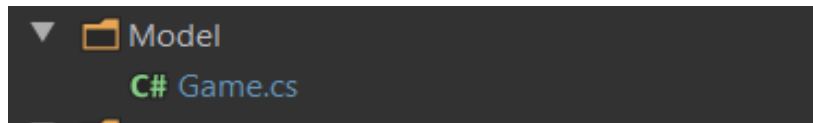
- i.
- f. Y mostrara una lista por defecto de temperaturas y climas
- i. <http://localhost:5001/wheaterforecast>



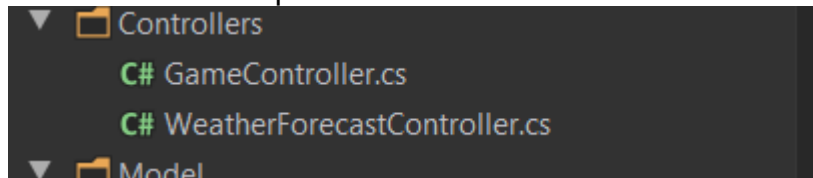
- ii.
- g. Ahora crearemos la estructura de carpetas del proyecto



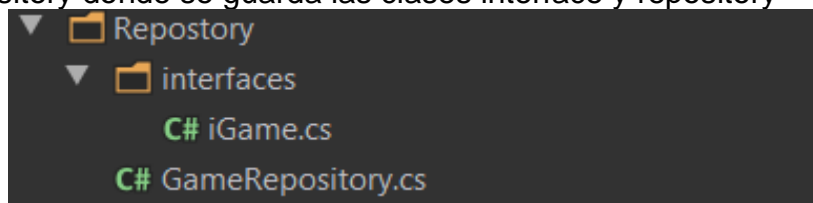
- i.
- ii. Controllers, Model, Repository Resources, Util
- h. Model para guardar la clase modelo que apunta a la tabla [games] en el schema public



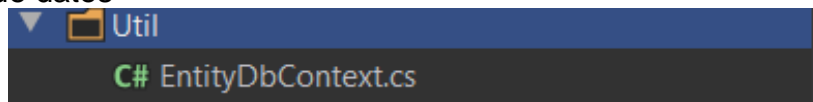
- i.
- i. Controllers donde se guardarán las clases que configuran los accesos o rutas a los endpoints



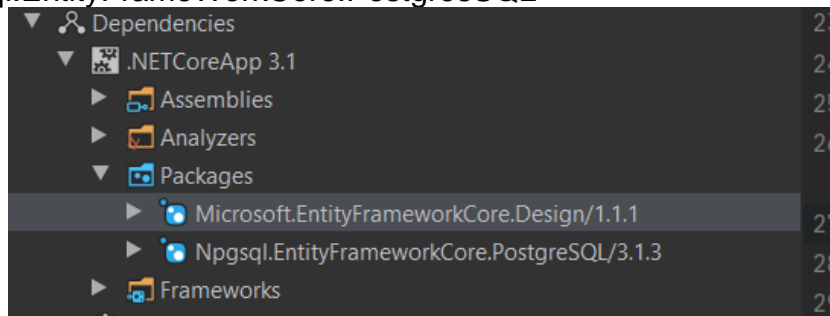
- i.
- j. Repository donde se guarda las clases interface y repository



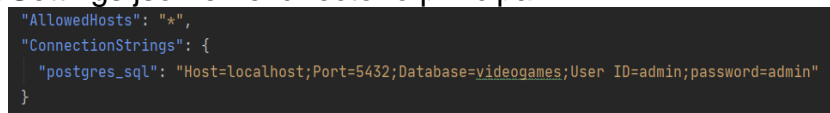
- i.
- k. Útil donde se guarda el Entity DbContext para comunicar a la base de datos



- i.
- 3. Agregar dos dependencias usando manejador de paquetes NUGET
  - a. Microsoft.EntityFrameWorkCore.Design
  - b. Npgsql.EntityFrameWorkCore.PostgresSQL



- i.
- 4. Configurando cadena de conexion
  - a. Agregar propiedad ConnectionString en archivo appSettings.json en el directorio principal

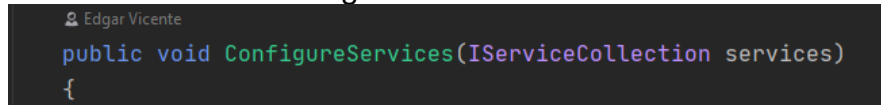


- i.
- 5. Configurando las entidades de acceso
  - a. Agregar el service transient en el proyecto
  - b. Dentro del Archivo Startup.cs

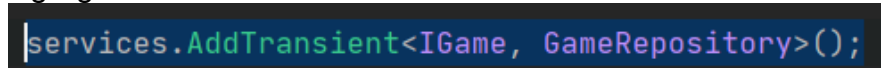




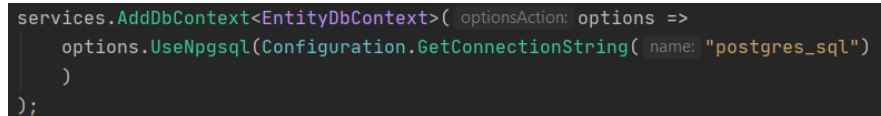
- i. matches and Consoles
- ii. Buscar el método ConfigureServices



- iii. Agregar la cadena services.AddTransient
- iv. Y la instancia DbContext

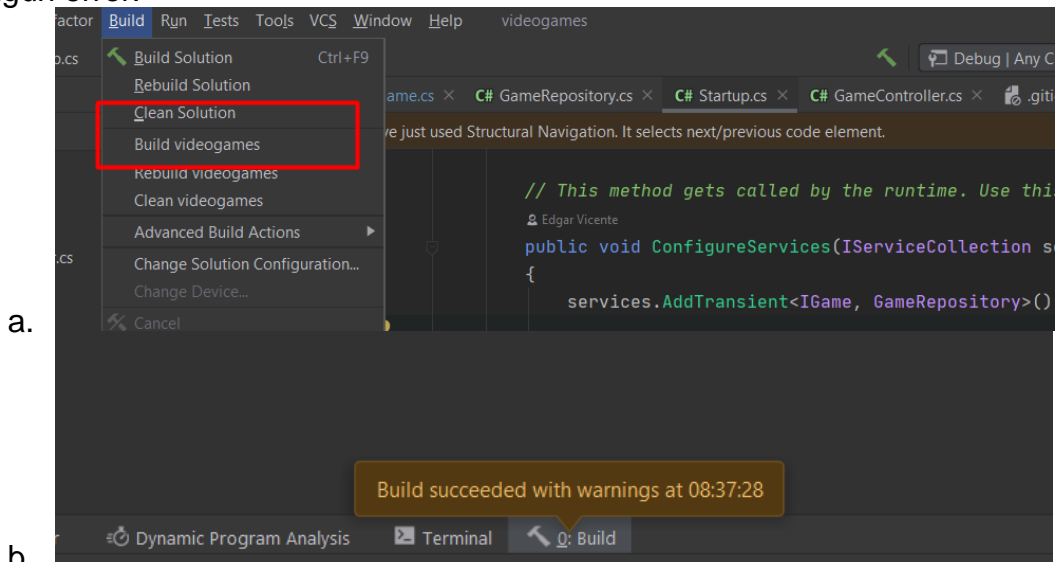


- v. Y la instancia DbContext
- vi.



- vii.

6. Hasta este punto si hacen un build de su proyecto no deberían tener ningún error.

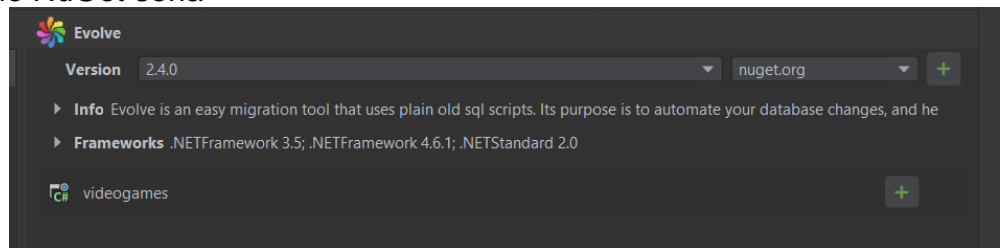


a.

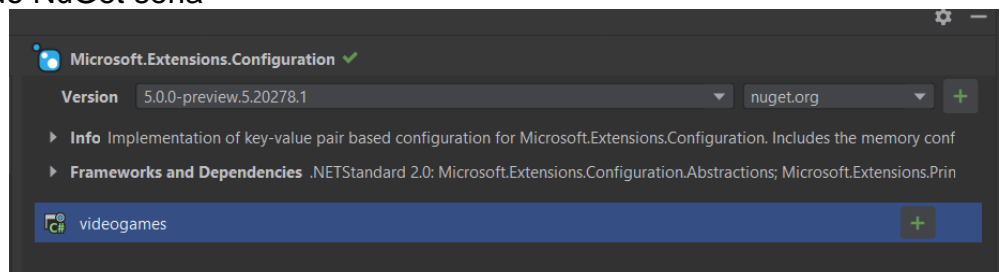
b.

## Instalando y Configurando Evolve.

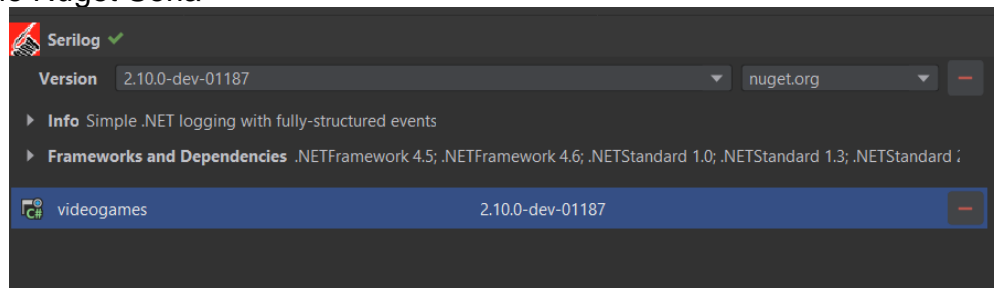
1. Documentacion Oficial
  - a. La documentacion de Evolve la pueden encontrar en el siguiente enlace.
    - i. <https://evolve-db.netlify.app/>
  - b. Los ejemplos de implementacion los pueden encontrar en el siguiente enlace.
    - i. <https://github.com/lecaillon/Evolve/tree/master/samples>
2. Agregamos la dependencia de Evolve
  - a. Usando linea de comandos seria
    - i. Install-Package Evolve -Version 2.4.0
  - b. Usando NuGet seria



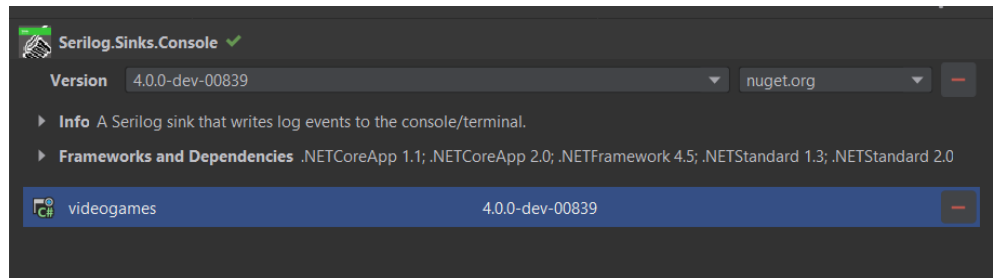
- i.
3. Agregamos la dependencia de Microsoft extensiones
  - a. Usando NuGet seria



- i.
4. Agregamos la dependencia Serilog para mostrar en pantalla el log de
  - a. Usando Nuget Seria



- i.
  - ii. Y agregamos serilog console, para hacer mostrar logs en consola del proyecto.



iii.

5. Crear método inicializador en Program.cs
  - i. Hace una inyección de IConfiguration para leer las propiedades del archivo appSettings.json

```
private static readonly IConfiguration Configuration;
```

ii.

6. Crear método constructor en Program.cs
  - a. La instancia de IConfiguration
  - b. La instancia de Logger para escribir en consola

```
static Program()
{
    Configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile(path: "appsettings.json", optional: false)
        .AddEnvironmentVariables() // IConfigurationBuilder
        .Build(); // IConfigurationRoot

    Log.Logger = new LoggerConfiguration()
        .WriteTo.Console() // LoggerConfiguration
        .CreateLogger(); // Logger
}
```

i.

7. Crearemos un método llamado MigrateDatabase()
  - a. Configuraremos la instancia de conexión a Nuestra base de datos partiendo del archivo de configuración de appSettings.json
  - b. Configuraremos el directorio donde están nuestros archivos de migraciones
  - c. Configuraremos la instancia de Evolve que se inicializara en cuanto el programa comience.

```
private static void MigrateDatabase()
{
    try
    {
        var cnx = new Npgsql.NpgsqlConnection( Configuration.GetConnectionString( name: "postgres_sql" ));
        var evolve = new Evolve.Evolve(cnx, logDelegate: msg => Log.Information(msg))
        {
            Locations = new[] { "Resources/postgres-sql" },
            IsEraseDisabled = true,
            Placeholders = new Dictionary<string, string>
            {
                [ "${table4}" ] = "table4"
            }
        };

        evolve.Migrate();
    }
    catch (Exception ex)
    {
        Log.Error( messageTemplate: "Database migration failed.", ex);
        throw;
    }
}
```

- d.
8. Modificaremos el método CreateHostBuilder
  - a. Agregaremos la función MigrateDatabase()

```
public static IHostBuilder CreateHostBuilder(string[] args)
{
    MigrateDatabase();
```

- i.
9. Presionamos Run Project
  - a. Debemos ver el log de nuestras migraciones en la consola

```
[11:15:29 INF] Executing Migrate...
[11:15:29 INF] Evolve initialized.
[11:15:29 INF] Mark schema public as empty.
[11:15:29 INF] No metadata found.
[11:15:29 INF] Successfully applied migration V1_1_1__create_tables.sql in 94 ms.
[11:15:29 INF] Database migrated to version 1.1.1. 1 migration(s) applied in 94 ms.
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
```

- b.
- c.

## Evaluando cambios en la base de datos

1. Revisar que se haya creado una tabla changelog en la base de datos



- a.
2. Al revisar la tabla changelog aparecerá el historial de cambios de nuestra base de datos.

A screenshot of a database table named 'changelog'. The table has columns: id, type, version, description, name, checksum, and installed\_by. The data shows two rows of changes.

id	type	version	description	name	checksum	installed_by
1	2	0	Empty schema found: public.	public		admin
2	0	1.1.1	create tables (95 ms)	V1_1_1_create_tables.sql	5028D0A69559EBBEAD89F40DDC4D4F5F	admin

- a.
3. Revisamos que la tabla game haya sido creada con su respectivo secuenciador

A screenshot of a database table named 'game'. The table has columns: id, title, and phrase. The data shows two rows of game entries.

id	title	phrase
1	mass effect 2	Is simply awe-inspiring
2	the last of us	the most beautiful game seen on any console

- a.
- b. Debera aparecer tambien el secuenciador de la tabla changelog



- c.

## Observaciones importantes

1. Si desplegamos nuestro proyecto y no hay migraciones entonces aparecerá un mensaje en logs .

```
iles\dotnet\dotnet.exe" D:/codebase/docker-conf/techcommunityday/videogames/videogames/bin/Debug/netcoreapp3.1/vi
[11:40:18 INF] Executing Migrate...
[11:40:18 INF] Evolve initialized.
[11:40:18 INF] Metadata validated.
[11:40:18 INF] Database is up to date. No migration needed.
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
```

- a.
2. Si queremos eliminar la base de datos y la tabla de histórico usamos el siguiente comando

```
evolve.Erase();
```

- a.
3. Si queremos cambiar el nombre de la tabla donde se guarda el historico de cambios usamos el siguiente comando

```
MetadataTableName = "schema_version"
```

- a.
4. Para configurar evolve con algún base de datos ya construida usar el siguiente comando especificando la migracion a iniciar.

```
StartVersion = new MigrationVersion("2.1"),
```

- a.
5. Para revisar el estado del esquema actual podemos usar el siguiente comando.

```
evolve.Info();
```

```
17:05:24 INF] +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | Version | Category | Description | Installed on | Installed by | Success | Checksum |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | 1.1.1 | Pending | Empty schema found: public. | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 1.1.1 | Pending | create tables | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

b.