# Introduction

The purpose of this assignment is to evaluate the candidate’s familiarity with .NET technologies, programming patterns and to showcase a sample of what clean and reusable code means to the candidate. This is also helps in understanding the candidate’s basic design knowledge.

# Requirements

You need to create an ASP.NET web api application to expose the information provided by this api <http://www.tvmaze.com/api>

## Requirement 1

Implement a job to store in a database the information provided by the endpoint “Shows main information”.

Se implementa FetchShowsJob que obtiene todos los shows y los guarda en base de datos. Este job se ejecuta cada 24h que esta configurado dentro del código. En este cas

## Requirement 2

Expose and endpoint to execute the job when will be need and securize it using an api key.

Para esta parte se crea un middelware para verificar la clave de API en cada solicitud HTTP entrante.

Y se registra en el Program.cs para que se ejecute antes de cualquier autorización o endpoint.

## Requirement 3

Expose a public endpoint to query the information stored in the database.

Se crean 2 endpoints para obtener la información de la base de datos. Una para obtener todos los shows y otra para obtener un show en concreto.

* <https://localhost:7031/shows/1>
* https://localhost:7031/shows

# Architecture

There is no limit for this. Design your application as you wish, but make sure you will focus on clean code, reusability and .NET best practices. Show us that you know how to produce top notch applications.

# Notes

* Complete this assignment using .NET Framework or .NET Core and the C# language version binded to .NET versión used.
* Try to keep the leverage/use of scaffolding tools, boilerplate templates, plugins etc. to a minimum. We don’t expect you to create things from scratch but, we also would like to see your own creation.
* Create a readme file that explains your architectural decisions and make sure you include instructions on how to run your solution.
* It’s mandatory to include unit tests (use the framework that you want).
* Feel free to include more features other than the few mentioned above.

**Arquitectura**

La API sigue el patrón de diseño DDD para separar la lógica de negocio, de la persistencia y las operaciones de negocio.

También se utiliza principios SOLID para tener el código más flexible y de mejor calidad.

La base de datos si implementa con CodeFirst usando E.F.

* Para los logs se utiliza **log4net** y su archivo de configuraciónlog4net.config que guardo los logs en unidad c.
* Para test unitarios se usa **XUNIT** con Mock
* Para mapear el json deserealizado se usa **Automapper**
* La capa Application y Infrastructure tiene 2 ficheros para registrar las dependencias
* La capa Presentación tiene **ApiKeyMiddelware** para realizar la autenticación y autorización de forma segura basado en claves API (Requeriments 2).
* appsettings.json contiene la conexión con DB y la APIKEY.

La **estructura** es la siguiente:

Application

Services

ShowService.cs

Interface

IShowService.cs

Mapping

MappingProfile.cs

ApplicationServiceRegistration.cs

Domain

Entities

Show.cs y demás

Interface

IRepository.cs

Infrastructure

Persistence

AppDbContext.cs

Repositories

Repository.cs

ShowRepository.cs y mas repositorios por cada entidad que me haga falta

Migrations

Se genera con add-migration XXXX + update-database

Job

FetchShowsJob.cs

InfrastructureServiceRegistration.cs

Presentation (API)

Controllers

ShowController.cs

Middleware

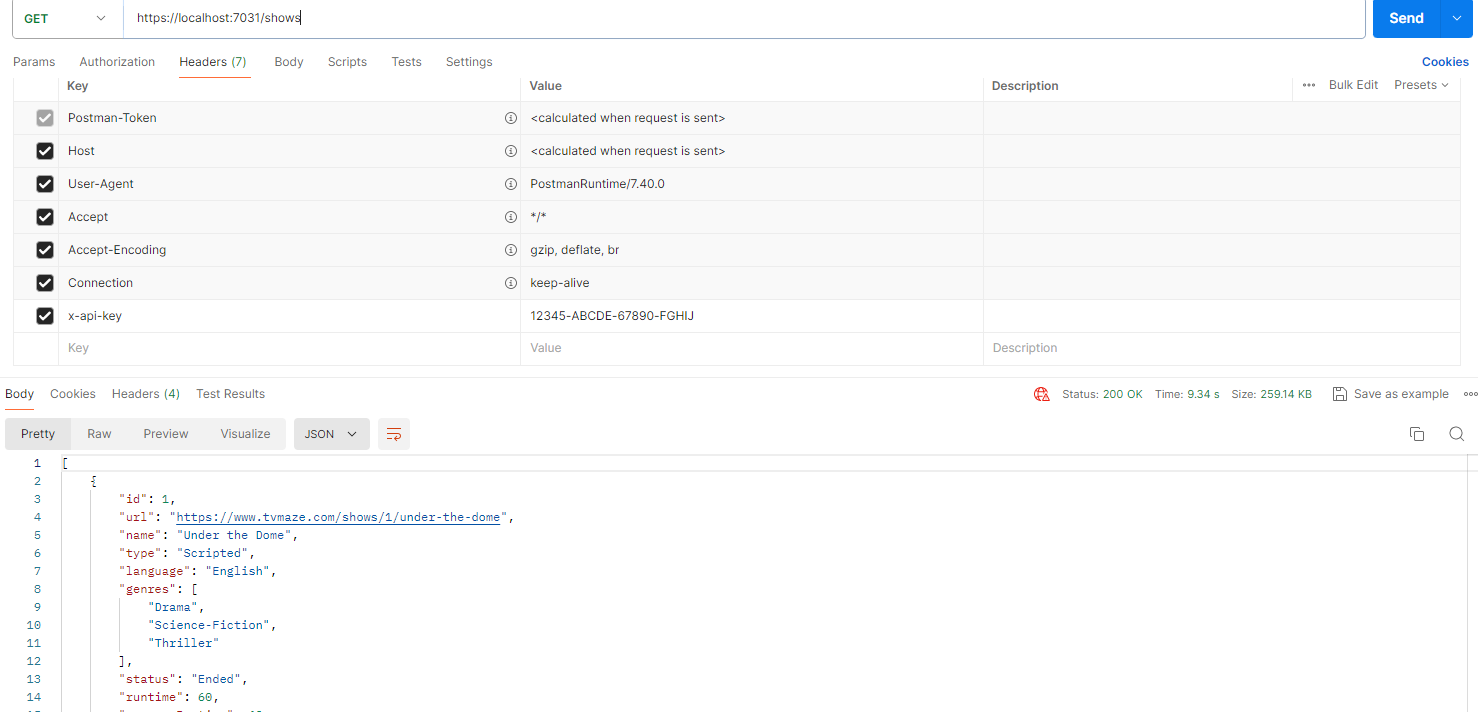
ApiKeyMiddleware.cs

log4net.config

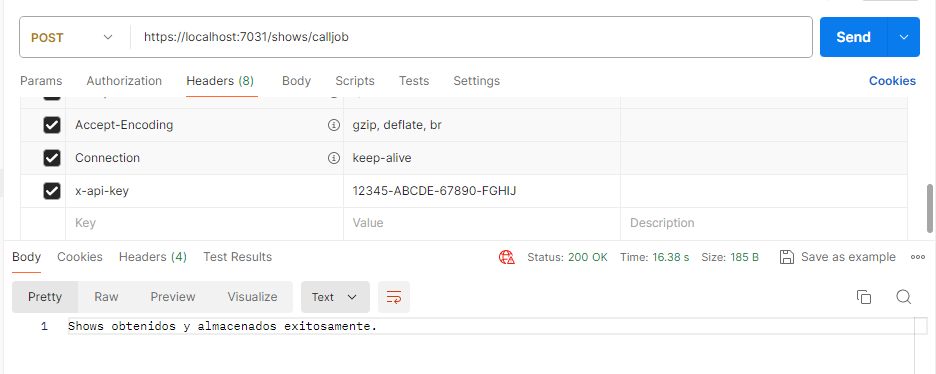
**Test**

Para testear los endpoints se usa Postman usando la APIKEY:

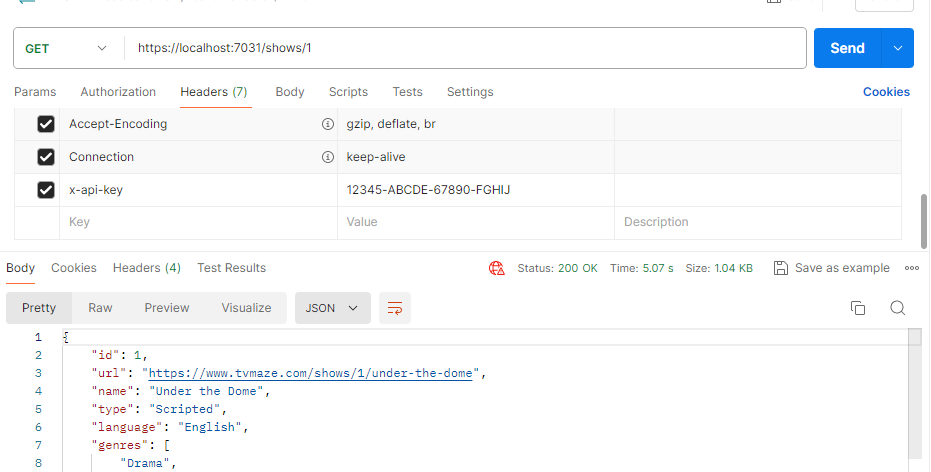
Llamada para obtener todo los shows: https://localhost:7031/shows



Llamada para ejecutar el job manualmente: https://localhost:7031/shows/calljob



Llamada para obtener un show: <https://localhost:7031/shows/1>



NOTA

* El código aún se podría mejorar creando otros servicios para Network, Webchannel, etc y no tenerlo todo en ShowService.cs.