

# Manual de Implementación, Eventify

Departamento de Ingeniería e Investigaciones Tecnológicas

Tecnicatura en Desarrollo Web

Taller Práctico Integrador / Trabajo Final



UNLaM

38841439 Alexis Javier Verba

40460303 Carla Nahir Galeano

94257410 Scarlet Cortes Valdes

39098059 Fabrizio Franco

38615943 Diego Manuel López

## I - ARQUITECTURA DE LA APLICACIÓN

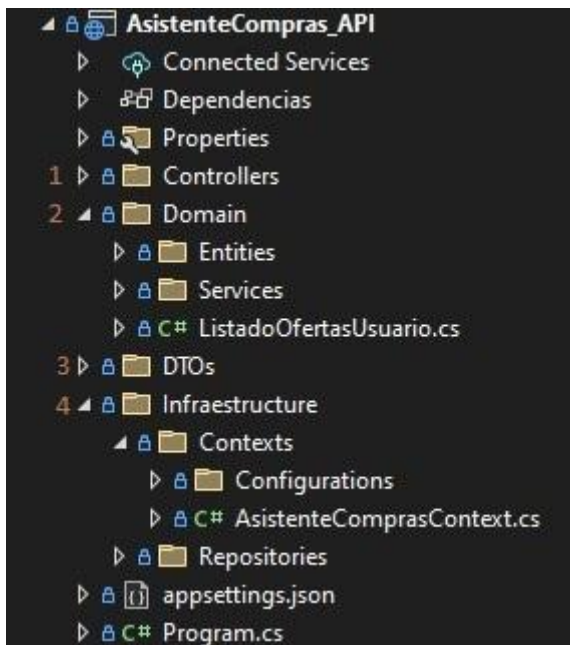
---

La página eventify es una aplicación web (SPA) que está desarrollada en microservicios que consta de un servicio en front-end desarrollado en Angular y el servicio del back-end desarrollado con .NET 6.

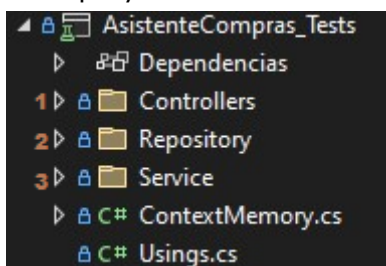
En el servicio del back-end vamos a encontrar dos proyectos un proyecto llamado AsistenteCompras\_API que es donde se encuentra los controladores y servicios que realiza nuestra api y por otro lado tenemos otro proyecto AsistenteCompras\_Tests que se encuentran los test de nuestra aplicación.

Como se puede observar en la siguiente imagen en AsistenteCompras\_API manejamos la siguiente estructura.

1. Aca se encuentra los controladores de nuestra aplicación
2. Aca se encuentra nuestra carpeta de dominio en la cual contiene otra dos carpetas una para entidades que son el modelo de nuestro negocio y otra carpeta para nuestros servicios
3. Aca se encuentra los DTO de nuestra aplicación
4. Aca se encuentra todo lo que se refiere a infraestructura en este caso el contexto de nuestra base de datos, sus configuraciones y los repositorios en donde realizamos nuestras consultas.



En el proyecto de test nos encontramos con la siguiente estructura:



1. Aca se encuentra los test de nuestros controladores
2. Aca se encuentra los test de nuestros repositorios
3. Aca se encuentra los test de nuestros servicios

## II - IMPLEMENTACIÓN PARA DESARROLLADORES

---

### a. Tecnologías instaladas necesarias.

Para cualquier desarrollador que quiera contribuir al desarrollo de la aplicación, deberá clonarse el servicio del back-end ([https://github.com/Sacory/AsistenteCompras\\_BE](https://github.com/Sacory/AsistenteCompras_BE)) y el servicio del front-end ([https://github.com/alexisjv/Eventify\\_FE](https://github.com/alexisjv/Eventify_FE)). Aclaración en ambos repositorios se debe situar en la rama develop para poder trabajar de forma local.

### Herramientas back-end

Se debe instalar:

- Visual Studio 2022 (la versión gratuita para la comunidad es suficiente)
  - [Descarga Visual Studio Community](#)
    - En la instalación seleccione los componentes: *ASPNET y Desarrollo Web, .NET desktop desarrollo, .NET Core cross-platform desarrollo.*
    - En la sección de extensiones seleccione: Git para Windows, extensiones de GitHub para Visual Studio.
- Microsoft Sql Server Management Studio 18 (edición gratuita para desarrolladores)
  - [Descarga Sql Server Management Studio 18](#)

### Herramientas front-end

Se debe instalar:

1. Visual Studio Code 2023
  - [Descargar Visual Studio Code](#)
2. Node JS versión 18.16.10
  - [Descarga Node JS 18.16.0](#)
3. Angular CLI
  - En una terminal de Windows se debe ejecutar el siguiente comando:  
**npm install -g @angular/cli**

### **Importante!**

En los equipos cliente de Windows, la ejecución de scripts de PowerShell está deshabilitada de forma predeterminada. Para permitir la ejecución de scripts de PowerShell, que son necesarios para los archivos binarios globales de npm, debe configurar la siguiente política de ejecución en la terminal:

**Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned**

Una vez instaladas todas las herramientas y clonados los repositorios procedemos a levantar el proyecto.

### Back-end

Primero para levantar nuestro proyecto nos vamos a dirigir al **appsettings.json** y en la sección donde dice "CONEXION BASE DE DATOS" vamos a colocar nuestra llave un ejemplo sería:










```
"Server=Nombre del servidor ;Database=Nombre de la base de
datos;Trusted_Connection=True;Encrypt=True;"
```

Además de tener una base de datos para nuestro proyecto debemos tener una cadena de conexión para el uso de los blob storage de azure y ahí almacenar nuestro archivo .csv el cual lo usamos para el endpoint verificar comercio por medio del registro nacional de sociedades. Luego de crear una cuenta de almacenamiento (blob storage) tendrá que crear dentro de ese recurso un contenedor con nombre **csv** y ahí es donde guardará los archivos obtenidos del registro nacional de sociedades.









```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "",
    "BlobStorage": ""
  }
}
```

También debe verificar que en el proyecto se encuentre los siguientes paquetes instalados para el funcionamiento correcto del proyecto.

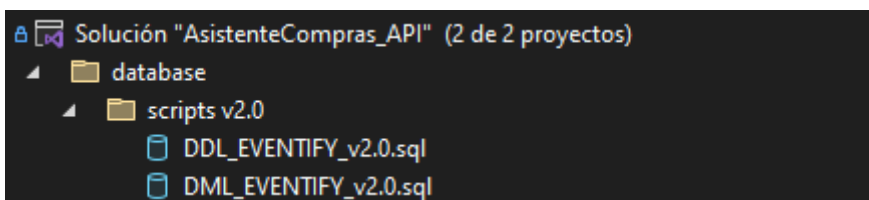
- Para la api

Top-level packages (9)		
	<b>Azure.Storage.Blobs</b> por Microsoft This client library enables working with the Microsoft Azure Storage Blob service for storing binary and text data. For this release see notes - <a href="https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/storage/Azure.Storage.Blobs/README.md">https://github.com/Azure/azure-sdk-for-net/blob/main/sdk/storage/Azure.Storage.Blobs/README.md</a> and <a href="https://github.com/Azure/azure-sdk-for-net/blob/main/s...">https://github.com/Azure/azure-sdk-for-net/blob/main/s...</a>	12.16.0
	<b>Microsoft.EntityFrameworkCore</b> por Microsoft Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.	7.0.5 7.0.8
	<b>Microsoft.EntityFrameworkCore.Design</b> por Microsoft Shared design-time components for Entity Framework Core tools.	7.0.5 7.0.8
	<b>Microsoft.EntityFrameworkCore.SqlServer</b> por Microsoft Microsoft SQL Server database provider for Entity Framework Core.	7.0.5 7.0.8
	<b>Microsoft.EntityFrameworkCore.Tools</b> por Microsoft Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	7.0.5 7.0.8
	<b>Microsoft.Extensions.Configuration.Json</b> por Microsoft JSON configuration provider implementation for Microsoft.Extensions.Configuration.	7.0.0
	<b>Microsoft.Extensions.Configuration.UserSecrets</b> por Microsoft User secrets configuration provider implementation for Microsoft.Extensions.Configuration.	7.0.0
	<b>Microsoft.IdentityModel.Tokens</b> por Microsoft Includes types that provide support for SecurityTokens, Cryptographic operations: Signing, Verifying Signatures, Encryption.	6.31.0
	<b>Swashbuckle.AspNetCore</b> por Swashbuckle.AspNetCore Swagger tools for documenting APIs built on ASP.NET Core	6.2.3 6.5.0

- Para los test

Top-level packages (8)		
	<b>coverlet.collector</b> por tonerdo Coverlet is a cross platform code coverage library for .NET, with support for line, branch and method coverage.	3.1.2 6.0.0
	<b>Microsoft.AspNetCore.Authentication.JwtBearer</b> por Microsoft ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.	6.0.4
	<b>Microsoft.EntityFrameworkCore.InMemory</b> por Microsoft In-memory database provider for Entity Framework Core (to be used for testing purposes).	7.0.5
	<b>Microsoft.IdentityModel.Tokens</b> por Microsoft Includes types that provide support for SecurityTokens, Cryptographic operations: Signing, Verifying Signatures, Encryption.	6.31.0
	<b>Microsoft.NET.Test.Sdk</b> por Microsoft The MSbuild targets and properties for building .NET test projects.	17.1.0
	<b>Moq</b> por Daniel Cazzulino, kzu Moq is the most popular and friendly mocking framework for .NET.	4.18.4
	<b>xunit</b> por James Newkirk, Brad Wilson xUnit.net is a developer testing framework, built to support Test Driven Development.	2.4.1 2.5.0
	<b>xunit.runner.visualstudio</b> por .NET Foundation and Contributors Visual Studio 2017 15.9+ Test Explorer runner for the xUnit.net framework. Capable of running xUnit.net v1.9.2 and v2.0+ tests. Supports .NET 2.0 or later, .NET Core 2.1 or later, and Universal Windows 10.0.16299 or later.	2.4.3 2.5.0

Una vez hecho esto tenemos que levantar nuestra base de datos de forma local con los scripts que se encuentran en el proyecto en la carpeta database es de suma importancia que lo haga ya que el proyecto fue desarrollado con entity framework **DatabaseFirst**.

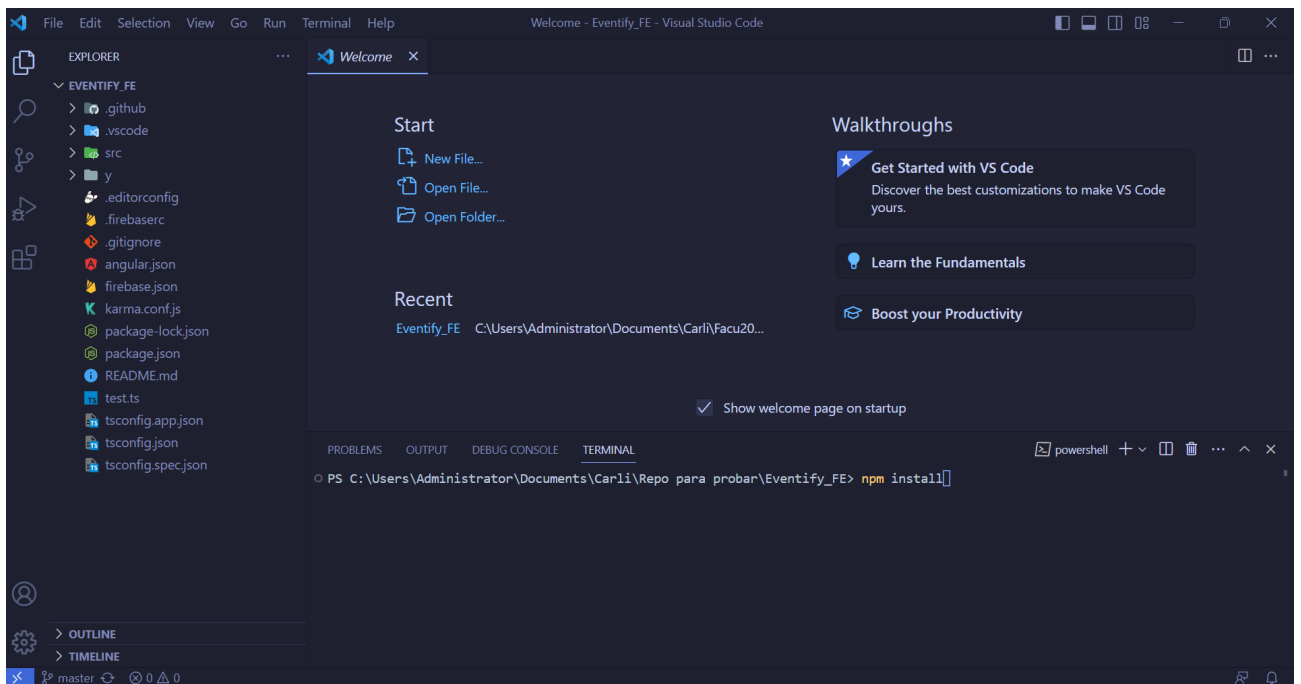


Realizado todos los pasos anteriores podrá levantar el proyecto haciendo click en el botón AsistenteCompras\_API luego de realizar esto se abrirá por defecto el siguiente enlace <https://localhost:7292/swagger/index.html> en el cual podrá probar los endpoint o si lo prefiere también puede usar otra herramienta.

### Front-end

Para levantar nuestro proyecto de front-end vamos abrirlo con Visual Studio Code de ahí abrimos la terminal y ejecutaremos el siguiente comando:

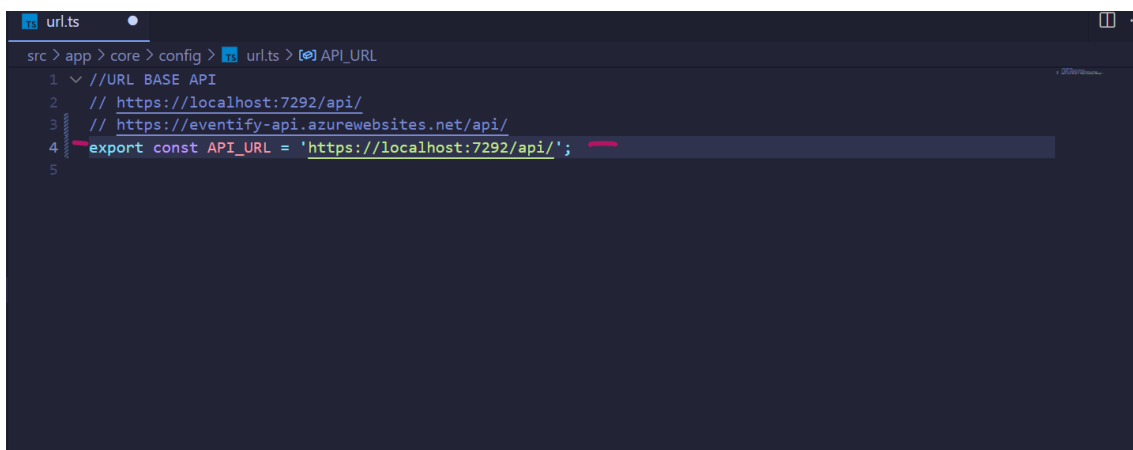
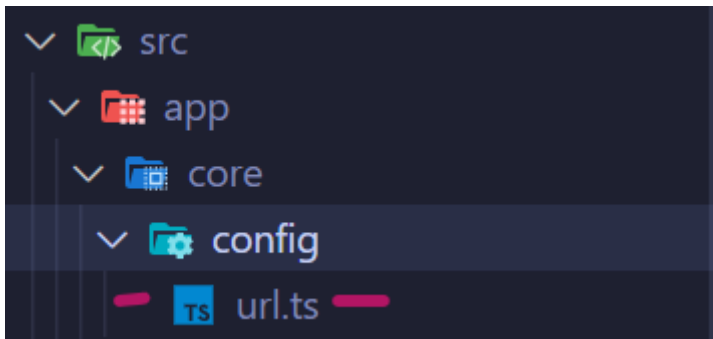
***npm install***



Esto hará que se instalen los paquetes necesarios para nuestra aplicación lo cual tardará unos minutos.

Una vez terminado el proceso de instalación debemos corroborar en el archivo *url.ts* que la aplicación consuma el servicio de forma local:

***'https://localhost:7292/api/'***



Una vez finalizados todos estos pasos ya podemos levantar el proyecto con el comando:

***ng serve***

```
Initial Chunk Files | Names | Raw Size
runtime.js          | runtime | 12.62 kB |
7 unchanged chunks

Build at: 2023-06-24T16:46:54.298Z - Hash: 0307f1ade8052213 - Time: 6688ms

✓ Compiled successfully.
```

Compiled successfully nos indica que el proyecto ya está levantado correctamente y podemos ingresar a la url <http://localhost:4200/> para ver nuestro proyecto:

