



FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS  
INGENIERIA DE SISTEMAS

ARQUITECTURA DE SOFTWARE

LABORATORIO 1

Martin Chafloque Mesia  
Laura Sofía Rozo Rodriguez  
María Alejandra Suarez Reyes

30 Abril 2023

## Tabla de Contenidos

<b>Marco Conceptual.....</b>	<b>3</b>
<b>.Net6 .....</b>	<b>3</b>
<b>Componentes de la Arquitectura .NET .....</b>	<b>3</b>
1. Implementaciones .....	3
2. .NET Standard Library .....	4
3. Entorno en Tiempo de Ejecución .....	4
4. Infraestructura Común .....	4
5. Herramientas de Desarrollo.....	5
<b>MS SQL Server 2019 .....</b>	<b>5</b>
1. Clústeres de big data .....	5
2. Compatibilidad con UTF-8 .....	6
3. Creación de índices en línea reanudable (CTP 2.0).....	7
<b>REST.....</b>	<b>7</b>
<b>Swagger 3.....</b>	<b>8</b>
<b>Diseño.....</b>	<b>9</b>
Modelo de datos .....	9
Diagrama de Clases .....	10
Arquitectura de Alto Nivel .....	11
<b>Procedimiento .....</b>	<b>11</b>
<b>Conclusiones.....</b>	<b>20</b>
<b>Lecciones Aprendidas.....</b>	<b>20</b>
<b>Referencias .....</b>	<b>21</b>

## Marco Conceptual

### .Net6

.NET es una plataforma de aplicaciones que permite la creación y ejecución de servicios web y aplicaciones de Internet. En la plataforma de desarrollo se pueden utilizar una serie de lenguajes, implementaciones, herramientas y bibliotecas para el desarrollo de las aplicaciones.

En definitiva, es hoy en día la plataforma de desarrollo de software más usada para nuevos proyectos de desarrollo de software además de Java.

Microsoft .NET es una colección de diferentes plataformas de software de Microsoft. El framework original fue desarrollado como una competencia directa a la plataforma Java. Así pues, los entornos de aplicación pueden ser desarrollados y ejecutados en base a .NET.

Hasta aproximadamente 2003, el término .NET sirvió a Microsoft como término de marketing y como palabra de moda para productos nuevos, pero muy diferentes, como sistemas operativos, servidores y software de oficina. Más tarde, el término se concentró en el desarrollo de software.

Inicialmente no tuvo mucho éxito, pero el entorno .NET ha cambiado significativamente a lo largo de los años y ha ganado en importancia. Hoy en día, el framework .NET se ha vuelto indispensable en la práctica diaria.

### Componentes de la Arquitectura .NET

Los componentes de la arquitectura .NET juegan un papel importante en el desarrollo de aplicaciones. Los podemos clasificar en: el clásico .NET Framework, que es un framework monolítico, el más actual .NET Core framework, que es modular, la plataforma Xamarin y la específica de Windows UWP. A continuación, se explican con mayor detenimiento para tu mayor comprensión.

#### 1. Implementaciones

.NET Framework está dividido en diferentes subcategorías y categorías de programas y, por lo tanto, contiene diferentes modelos de ejecución entre los que el usuario debe elegir al desarrollar el software. La base del desarrollo es la biblioteca de clases, que ha estado disponible en general como fuente compartida desde 2014. La llamada biblioteca de clases base permite el desarrollo de aplicaciones no sólo para entornos Windows, sino también para plataformas como Android o MacOS.

Esta plataforma de desarrollo se utiliza normalmente para crear aplicaciones de windows, windows mobile, windows server, etc con Asp.net, WPF y Windows Forms

Por otro lado, .NET Core es una nueva alternativa que se separó por primera vez del .NET Framework en 2015. Debido a la mejora de la modularidad y a la portabilidad aún más sencilla del

software a plataformas que no sean de Microsoft, .NET Core es particularmente apreciado por muchos desarrolladores.

Generalmente, desarrolla aplicaciones para la plataforma universal de Windows UWP y Asp.NetCore. Además, se utiliza para desarrollar aplicaciones en la nube. La biblioteca de clases Core Ex es compatible con Windows, MacOS y Linux.

Otra implementación por destacar es la plataforma Xamarin en la que se pueden desarrollar aplicaciones para Android, iOS, tvOS, watchOS, macOS y Windows. Ésta, disponía de herramientas y bibliotecas específicas.

Por último, incluir UWP, la plataforma universal de Windows. Aunque algunos desarrolladores la sitúan dentro de la plataforma .NET Core al compartir algunas bibliotecas de este, a Microsoft le gusta que se la considere una implementación más.

## 2. .NET Standard Library

Otro de los componentes que formaban parte de la arquitectura era la biblioteca de clases portable PCL. Con ella se podía compartir el código entre varios proyectos específicos de la plataforma, tanto en IOS, Android, Windows y Windows Phone.

Pero, las PCL presentaban muchas desventajas de compatibilidad entre implementaciones. Para ello, los desarrolladores crearon la API .NET Standard Library. Se trata de una fusión de las bibliotecas base y PCL compatible con todas las implementaciones.

En la actualidad, con la última versión de Visual Studio en 2017, las PCL quedaron obsoletas y borradas del sistema, así como las bibliotecas base de cada implementación. En su lugar fueron reemplazadas por .NET Standard Library.

También, existen otras API suplementarias que son específicas de los sistemas operativos en los que se ejecuta.

## 3. Entorno en Tiempo de Ejecución

Una de las partes relevantes de la arquitectura es el Entorno en Tiempo de Ejecución, que como su nombre indica es donde se ejecuta el programa administrado o el intervalo de tiempo en el que un software se ejecuta en un sistema operativo. Según la implementación utilizada:

.NET framework: CLR (Common Language Runtime).

.NET Core: Core CLR (CoreCommon Language Runtime).

Xamarin: entorno de implementación Mono.

UWP: .NET Native.

## 4. Infraestructura Común

Siguiendo con los componentes de la arquitectura, encontramos la Infraestructura Común donde están los lenguajes de programación: C#, F#, VB y el motor de compilación Ms Build para compilar los proyectos.

## 5. Herramientas de Desarrollo

Finalmente, otro componente son las Herramientas de Desarrollo para la creación de aplicaciones web o móviles en los diferentes sistemas operativos mencionados:

Administrador de paquetes para microsoft: Nuget.

Entorno de desarrollo integrado (IDE): Visual Studio, Xamarin Studio, Visual Studio para Mac, JetBrains Rider.

Editores de Código: Visual Studio Code y Plugin OmniSharp.

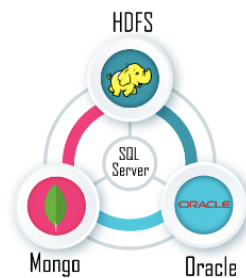
## MS SQL Server 2019

### 1. Clústeres de big data

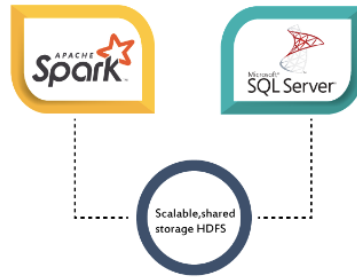
Los clústeres de big data son nuevas incorporaciones a la versión SQL server 2019. Esta característica le permite implementar múltiples clústeres escalables de contenedores SQL Server, Spark y HDFS que se ejecutan en Kubernetes, a la vez. El Big data Cluster, como infraestructura, permite que estos clústeres se ejecuten en paralelo, donde se puede leer, escribir y procesar Big Data desde Transact-SQL hasta Spark. Nos permite combinar y analizar fácilmente los datos relacionales de alto valor con Big Data de gran volumen.

Características:

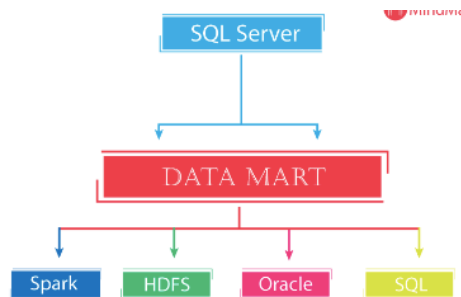
- Virtualización de datos: SQL Server PolyBase ha facilitado la tarea de consultar las fuentes de datos externas para los clústeres de big data de SQL Server, reduciendo el esfuerzo de mover o copiar los datos para realizar una consulta. La vista previa de SQL Server 2019 ha introducido nuevos conectores a fuentes de datos.



- Data Lake: El clúster de big data permite un pool de almacenamiento HDFS escalable. Esto aumenta potencialmente la eficiencia del almacenamiento de big data de fuentes externas.



- Scale-out data mart: Big data cluster provides scale-out compute and storage to improve the data analysis. The data can be ingested and stored across multiple data pool nodes as cache, for further analysis.



- IA y aprendizaje automático integrados: El clúster de big data permite IA y ML en los datos almacenados a través de múltiples pools de almacenamiento HDFS y pools de datos. SQL Server proporciona muchas herramientas de IA integradas como R, Python, Scala o Java.
- Gestión y monitorización: El portal del administrador del clúster es el sitio web que proporciona el estado y la salud de los pods en el clúster. También proporciona enlaces a otros cuadros de mando para el análisis de registros y la supervisión.
- La gestión y la supervisión se realizarán mediante una combinación de herramientas de línea de comandos, API, portal del administrador y vistas de gestión dinámicas.

## 2. Compatibilidad con UTF-8

El nuevo SQL Server 2019 admite el muy popular sistema de codificación de datos UTF-8. La codificación de caracteres UTF-8 se emplea en la exportación de datos, la importación, el cotejo de datos a nivel de base de datos y a nivel de columna. Se habilita al crear o cambiar el tipo de cotejo de objetos a cotejo de objetos con UTF-8. Es compatible con los tipos de datos char y varchar.

La razón por la que hay que codificar los datos al almacenarlos y recuperarlos se debe principalmente a 2 motivos.

- Para reducir la ocupación de memoria o el espacio de almacenamiento.
- Para proporcionar seguridad a los datos sensibles.

Nota: A partir de Microsoft SQL Server 2016, UTF-8 es compatible con BCP, BULK\_INSERT y OPENROWSET.

Las versiones anteriores de SQL Server tenían la codificación realizada en diferentes formatos como UCS-2 y no soportaban el formato UTF-8. Sin embargo, la introducción de la codificación Unicode se hizo sólo a partir de SQL Server 7.0.

### 3. Creación de índices en línea reanudable (CTP 2.0)

Esta es la característica que permite pausar una operación de creación de índice y reanudarla más tarde, justo desde el punto en el que la operación falló o se detuvo, en lugar de volver a iniciar el proceso.

El índice es una de las herramientas más potentes para la gestión de bases de datos. Con más operaciones en las bases de datos como insertar, actualizar y eliminar, el índice se vuelve más fragmentado y, por lo tanto, menos eficiente. Para evitarlo, los administradores de bases de datos recurren cada vez más a las operaciones de reconstrucción de índices.

Resumable Online Index Rebuilding (ROIR) se adoptó a partir de SQL Server 2017 como una característica importante para mejorar el rendimiento de la base de datos.

Sin embargo, en la versión SQL Server 2019, se incorpora una versión más nueva de la característica, que es "Resumable Online Index Create"

Características de la creación de índices en línea reanudable:

Puede reanudar la operación de creación de índice después de que un índice crea fallo en caso de uso excesivo de espacio en disco o durante la pérdida de base de datos.

La pausa de la operación de creación de índices en curso en caso de bloqueos dará lugar a la liberación temporal de los recursos, para reanudar las tareas bloqueadas.

La generación de registros pesados debido a la engorrosa operación de creación de índices puede ser manejada pausando la operación de creación de índices, truncando o tomando la copia de seguridad del registro y luego reanudando la misma.

### REST

REST (Representational State Transfer) es un estilo arquitectónico para diseñar aplicaciones en red propuesto por Roy Fielding en su tesis doctoral en el año 2000. REST establece una serie de principios y restricciones para construir sistemas distribuidos basados en la web que son escalables, mantenibles y fáciles de comprender.

Las características principales de REST incluyen:

- Recursos identificables: En REST, cada recurso (como un objeto de datos o una entidad) es identificable mediante una URL única. La identificación de recursos permite simplificar la interacción entre el cliente y el servidor.
- Comunicación sin estado: Cada solicitud del cliente al servidor debe contener toda la información necesaria para que el servidor comprenda y procese la solicitud. Esto significa que el servidor no debe almacenar información sobre el estado de las interacciones anteriores con un cliente específico.

- **Cacheabilidad:** Las respuestas del servidor pueden ser almacenadas en caché por el cliente. Esto permite mejorar la eficiencia y el rendimiento de la comunicación, ya que las respuestas almacenadas en caché pueden ser reutilizadas para futuras solicitudes similares.
- **Sistema en capas:** REST permite la organización de la arquitectura en capas, lo que facilita la separación de preocupaciones y la evolución independiente de cada capa.
- **Interfaz uniforme:** REST define un conjunto limitado y estandarizado de métodos y convenciones para interactuar con los recursos. Los métodos HTTP comunes incluyen GET (para recuperar datos), POST (para crear nuevos recursos), PUT (para actualizar recursos existentes) y DELETE (para eliminar recursos).
- **Uso de HATEOAS (Hypermedia as the Engine of Application State):** Los recursos en una API REST pueden contener enlaces a otros recursos relacionados, lo que facilita la navegación y el descubrimiento de funcionalidades en la API. Esto promueve el desacoplamiento entre el cliente y el servidor.

### Swagger 3

Swagger, también conocido como OpenAPI, es una especificación para describir, documentar y consumir APIs RESTful. OpenAPI 3.0 es la tercera versión de esta especificación, lanzada en 2017, que introduce varias mejoras y características nuevas en comparación con la versión anterior (Swagger 2.0).

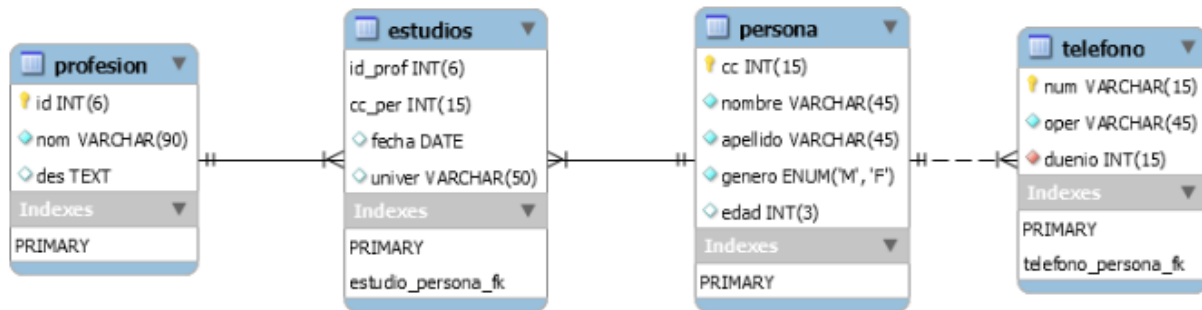
Características principales de Swagger 3.0 (OpenAPI 3.0):

- **Descripción estructurada:** OpenAPI 3.0 proporciona un formato estructurado (en JSON o YAML) para describir todos los aspectos de una API REST, como las rutas, los métodos HTTP, los parámetros, las respuestas, los esquemas de datos y la autenticación.
- **Mejoras en la reutilización y modularidad:** OpenAPI 3.0 introduce la posibilidad de utilizar componentes reutilizables en la especificación, lo que facilita la creación y mantenimiento de descripciones de API complejas y extensas. Los componentes pueden incluir esquemas, respuestas, parámetros, encabezados y otros elementos comunes.
- **Compatibilidad con más formatos de datos:** OpenAPI 3.0 mejora el soporte para diferentes formatos de datos y medios, permitiendo describir con mayor precisión cómo se deben transmitir los datos entre el cliente y el servidor.
- **Mejoras en la seguridad:** OpenAPI 3.0 ofrece una mejor definición y configuración de esquemas de seguridad y autenticación, como API Key, OAuth 2.0, OpenID Connect y otros métodos personalizados.
- **Ejemplos de respuesta:** OpenAPI 3.0 permite incluir múltiples ejemplos de respuestas para cada operación, lo que facilita la comprensión de cómo se deben interpretar y utilizar las respuestas del servidor.
- **Soporte para servidores:** OpenAPI 3.0 permite describir varios servidores y ambientes (como producción, desarrollo y pruebas), facilitando la configuración y el cambio entre diferentes entornos.
- **Herramientas y ecosistema:** Existe un amplio ecosistema de herramientas y bibliotecas que facilitan la creación, validación, generación de código, documentación y pruebas de APIs basadas en OpenAPI 3.0.



## Diseño

### Modelo de datos

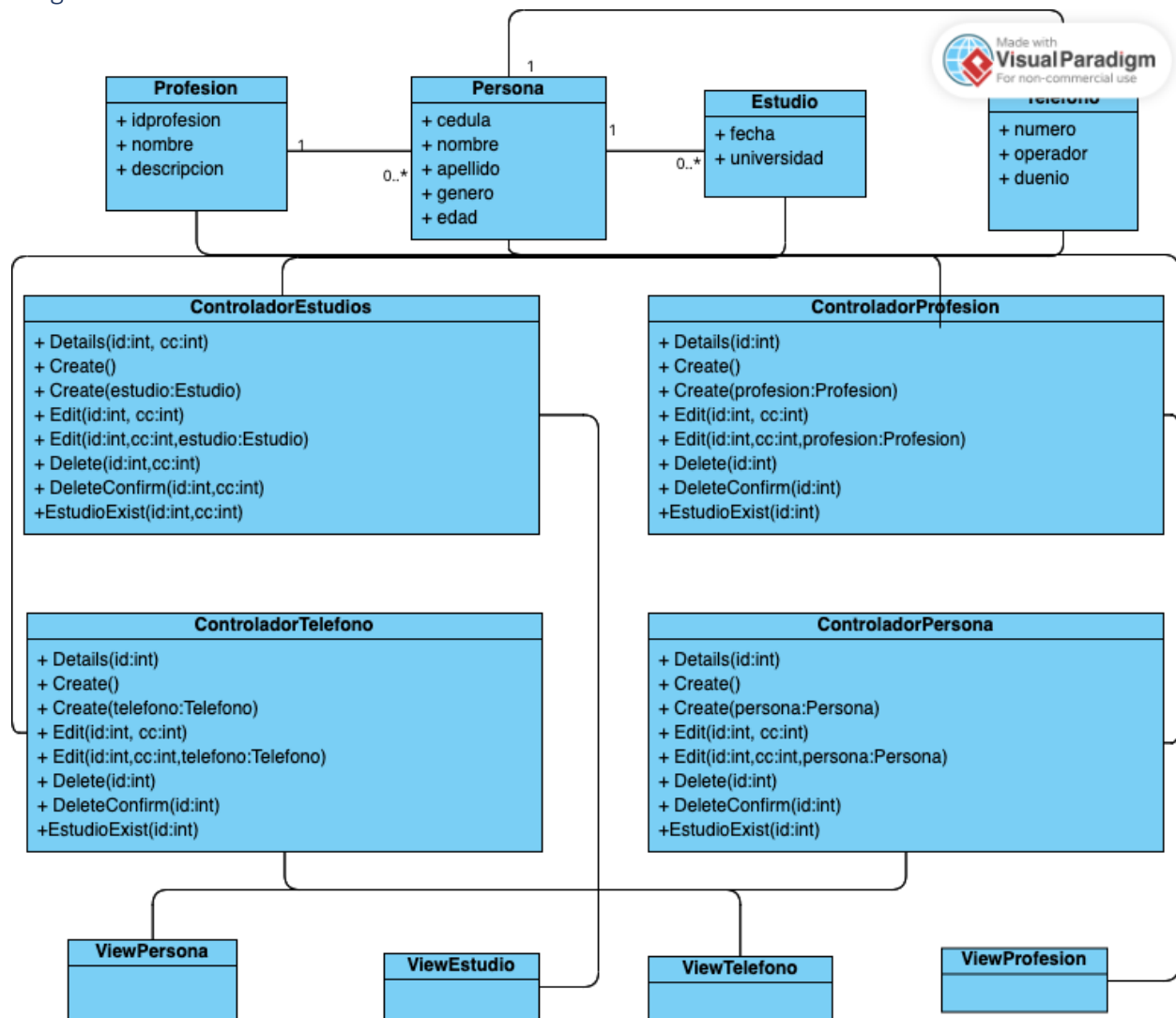


En este modelo de datos, tenemos cuatro entidades interconectadas que representan información relacionada con profesiones, estudios, personas y teléfonos. A continuación, se detalla la estructura y las relaciones entre estas entidades:

1. **Profesión:** La entidad "Profesión" almacena información sobre diferentes profesiones, donde cada profesión se identifica de manera única por su "id" (llave primaria). Además, cada profesión tiene un "nombre" y una "descripción" que describen la profesión en cuestión.
2. **Persona:** La entidad "Persona" representa información sobre las personas. Cada persona se identifica de manera única por su "cc" (llave primaria) y tiene atributos adicionales como "nombre", "apellido", "género" y "edad".
3. **Estudios:** La entidad "Estudios" almacena información sobre los estudios realizados por las personas en diferentes profesiones. Cada registro en "Estudios" se identifica por una combinación de "id profesiones" y "cc persona" (ambas llaves foráneas y primarias en esta entidad). Esto crea una relación entre las entidades "Persona" y "Profesión". Además, esta entidad tiene atributos como "fecha" y "universidad", que indican cuándo y dónde se completaron los estudios.
4. **Teléfono:** La entidad "Teléfono" contiene información sobre los números de teléfono de las personas. Cada registro en "Teléfono" se identifica por un "número" único (llave primaria) y tiene atributos adicionales como "operador" (proveedor de servicios de telecomunicaciones) y "dueño" (la persona a la que está asignado el número de teléfono). Para establecer una relación con la entidad "Persona", es posible utilizar el atributo "dueño", que podría ser una referencia a la "cc" de la entidad "Persona" (esto no se menciona explícitamente en la descripción proporcionada, pero es una suposición razonable).

Este modelo de datos permite almacenar y gestionar información relacionada con personas, sus profesiones, estudios y números de teléfono. Al establecer relaciones entre las entidades, es posible consultar y combinar datos de diferentes entidades, lo que permite responder preguntas como "¿Qué personas tienen estudios en una profesión específica?", "¿Cuál es la información de contacto de las personas con ciertos estudios?" o "¿Cuántas personas han estudiado en una universidad específica?".

## Diagrama de Clases

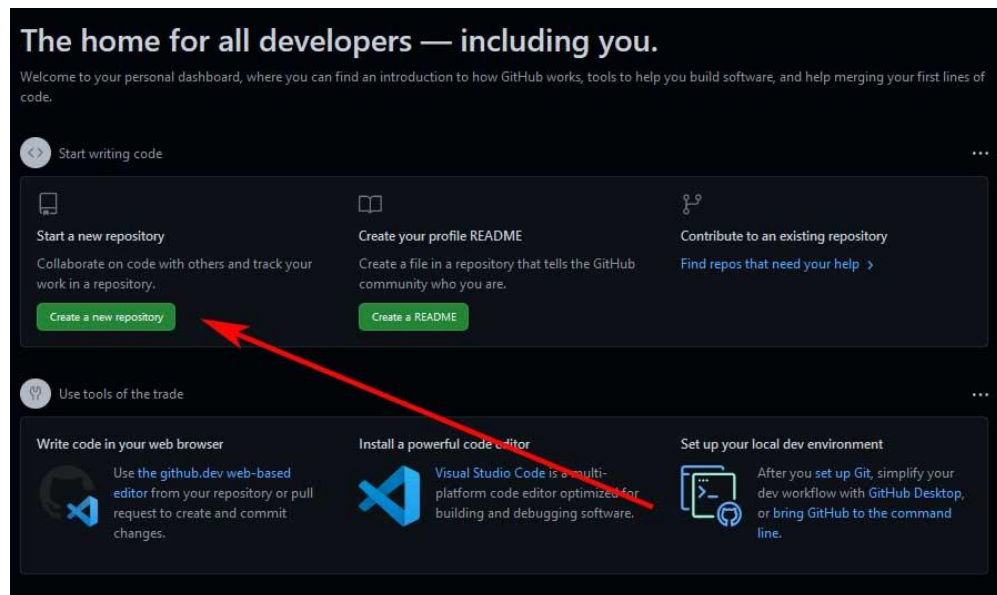


## Arquitectura de Alto Nivel




## Procedimiento

1. Crear el repositorio git publico en github o en cualquier sistema git nombre del repositorio debe ser personapi-dotnet



2. Instalar SQL Server 2019 Express modo Basico

O bien, descarga una edición especializada gratuita



**Desarrollador**

SQL Server 2019 Developer es una edición gratuita con todas las características que se puede usar como base de datos de desarrollo y pruebas en un entorno que no sea de producción.

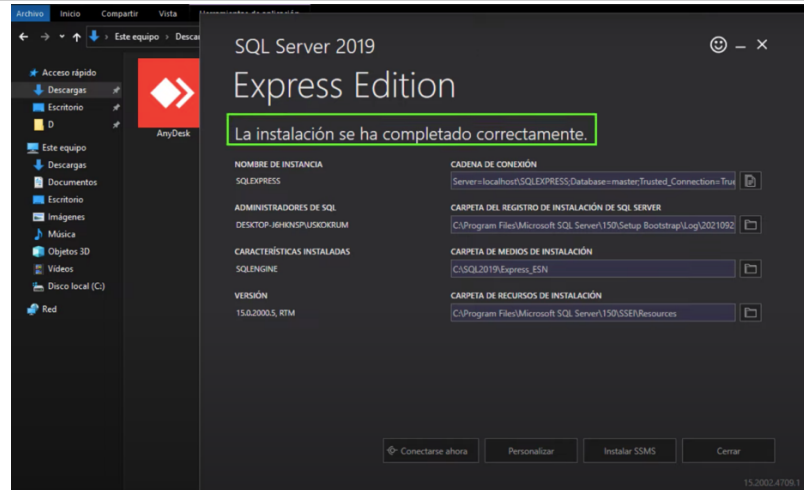
[Descargar ahora >](#)



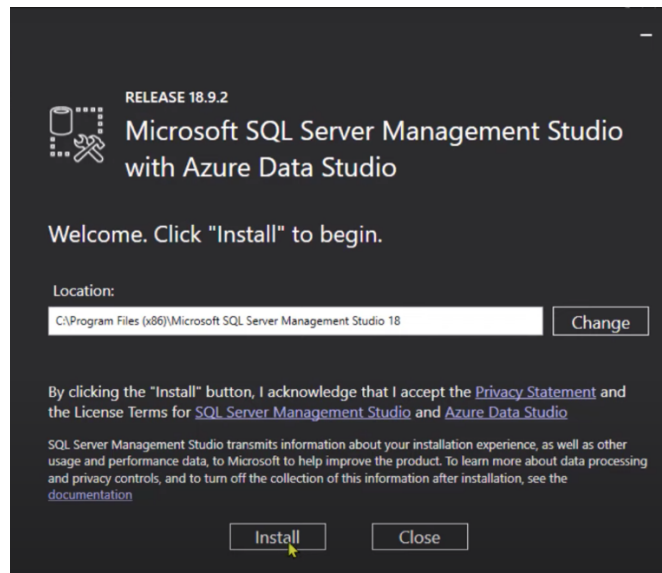
**Express**

SQL Server 2019 Express es una edición gratuita de SQL Server ideal para el desarrollo y la producción de aplicaciones de escritorio, aplicaciones web y pequeñas aplicaciones de servidor.

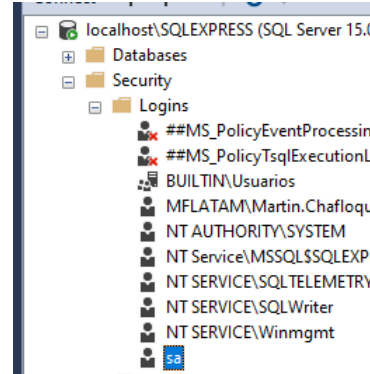
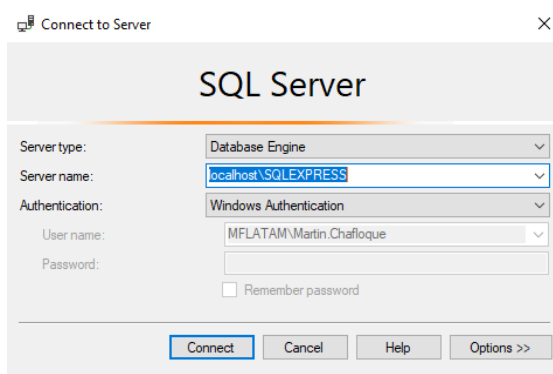
[Descargar ahora >](#)



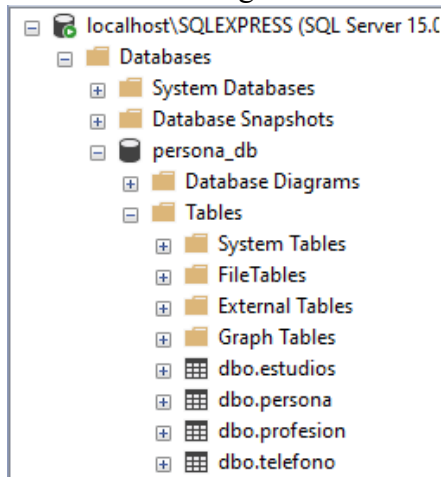
### 3. Instalar SQL Server Management Studio 18



### 4. Crear la base de datos llamada persona\_db y darle la propiedad al usuario sa.



## 5. Crear las tablas segun el modelo.

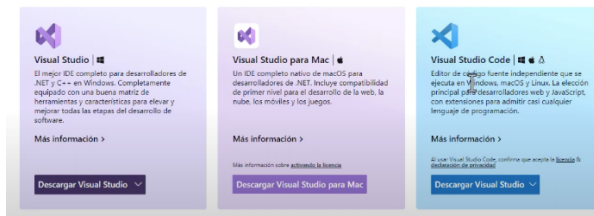


Column Name	Data Type	Allow Nulls
id_prof	int	<input type="checkbox"/>
cc_per	int	<input type="checkbox"/>
fecha	date	<input checked="" type="checkbox"/>
univer	varchar(50)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

## 6. Instalar Visual Studio Community 2022 con los complementos

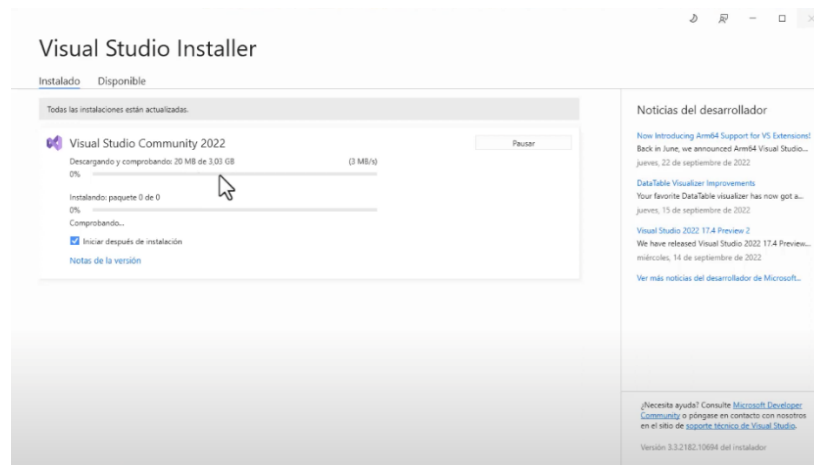
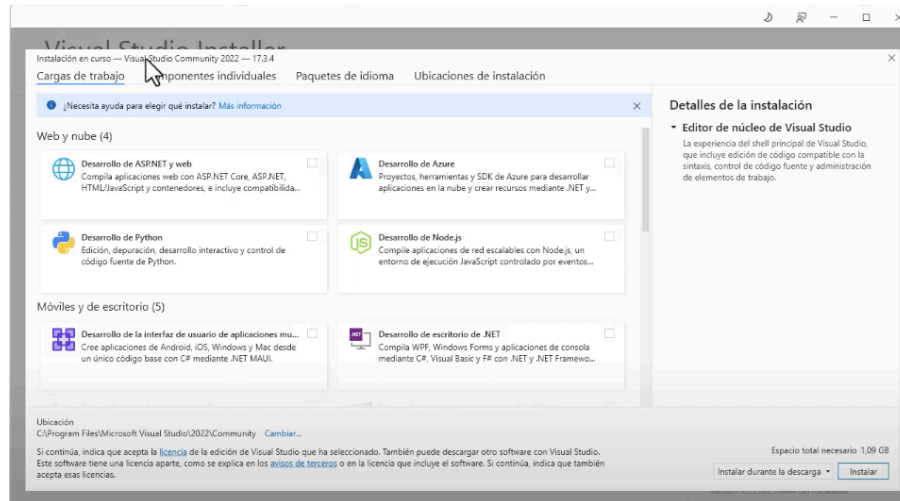
- Desarrollo ASP.NET y web
- Almacenamiento y procesamiento de datos
- Plantillas de proyecto y elementos de .Net Framework
- Características avanzadas de ASP.NET

### Conozca la familia Visual Studio

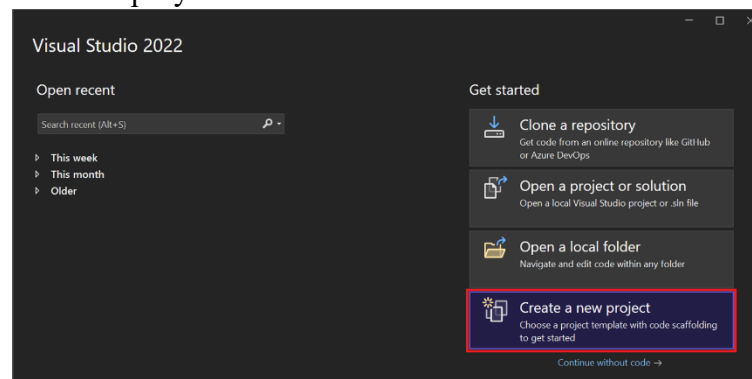


### Descargas

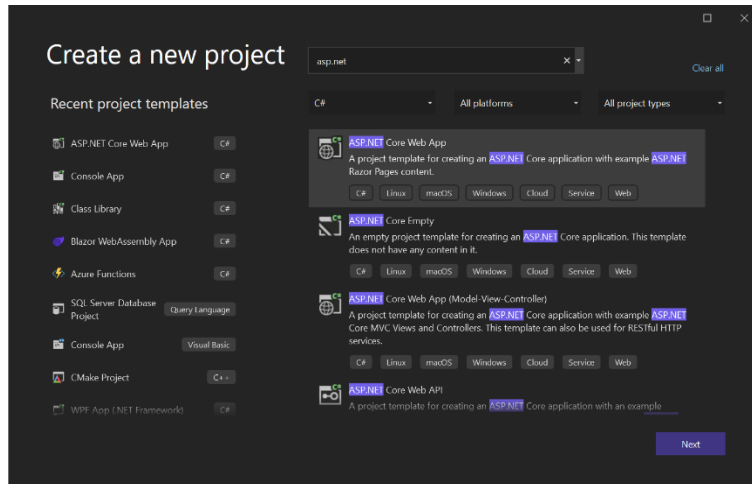




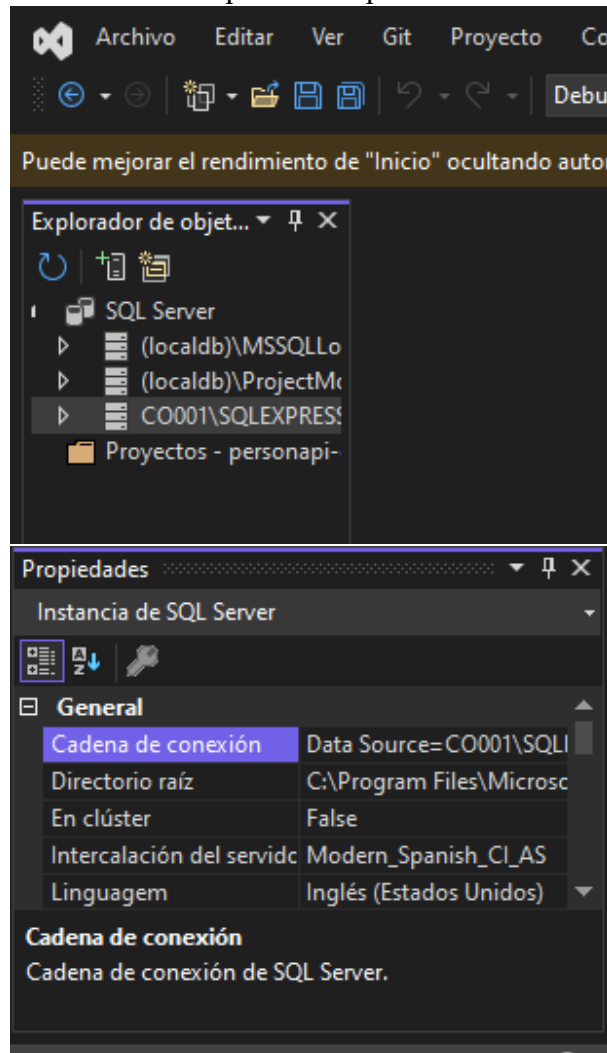
7. Clonar el repositorio local git a partir del remoto creado previamente.
8. En Visual Studio Community 2022
  01. Crear un proyecto.



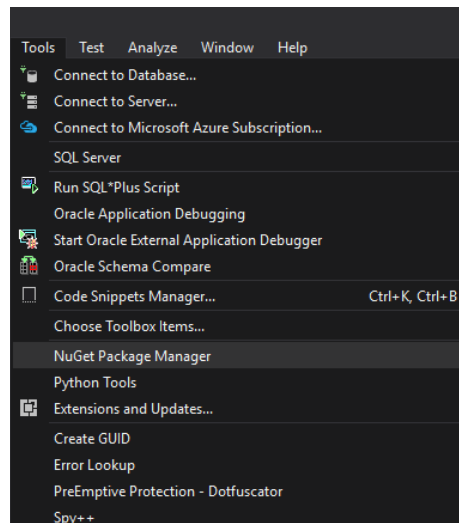
02. Seleccionar la plantilla Aplicación web de ASP.NET core (MVC).



03. El nombre de la aplicación debe ser el mismo del repositorio “personapi-donet”.
04. Framework .NET 6.0 sin autenticación y sin configuración HTTPS.
05. En el menú “Ver” activar la vista de Explorador de objetos de SQL Server.
06. Agregar y probar la conexión de tipo local express

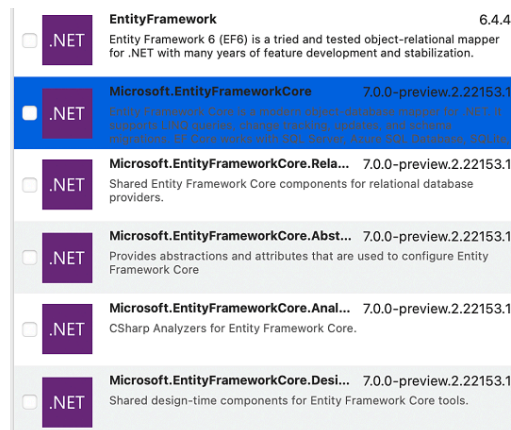


07. Ir al menú Herramientas>Administrador de Paquetes NuGet>Consola del Administrador de paquetes.



08. En el explorador de soluciones, hacer clic derecho en dependencias e ir a Administrar paquetes NuGet e instalar

1. Microsoft.EntityFrameworkCore
2. Microsoft.EntityFrameworkCore.SqlServer
3. Microsoft.EntityFrameworkCore.Tools



09. Crear entidades, en el explorador de soluciones, en la carpeta Models hacer clic derecho y agregar una carpeta llamada Entities

10. En la Consola del Administrador de paquetes escribir:

1. Scaffold-DbContext

"Server=localhost\SQLEXPRESS;Database=persona\_db;Trusted\_Connection=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models/Entities

```
PM> Scaffold-DbContext "Server=localhost\SQLEXPRESS;Database=persona_db;Trusted_Connection=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models/Entities
```

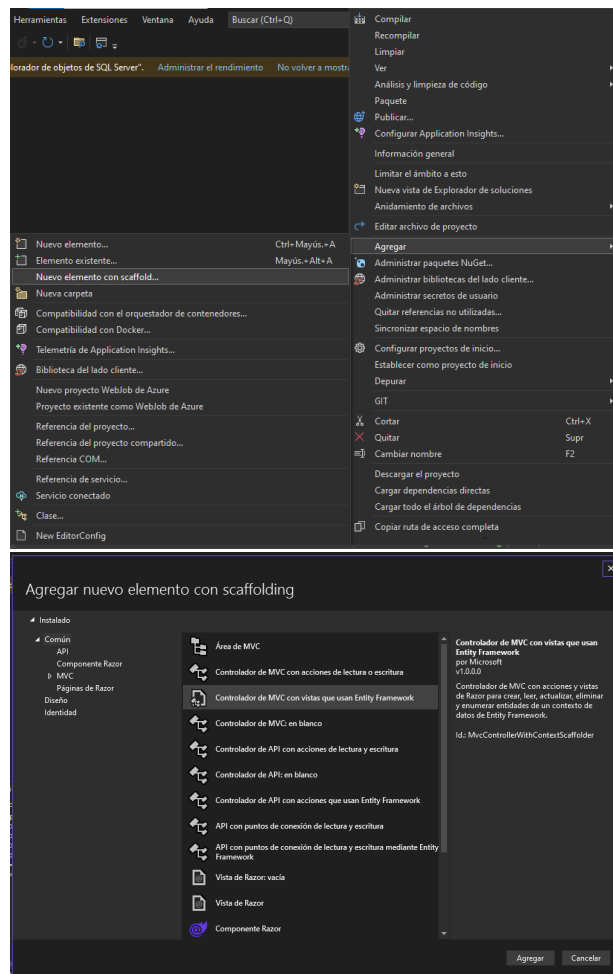


2. se crean las clases entidad a partir de las tablas existentes de la base de datos y el contexto
3. se crean las clases entidad a partir de las tablas existentes de la base de datos y el contexto

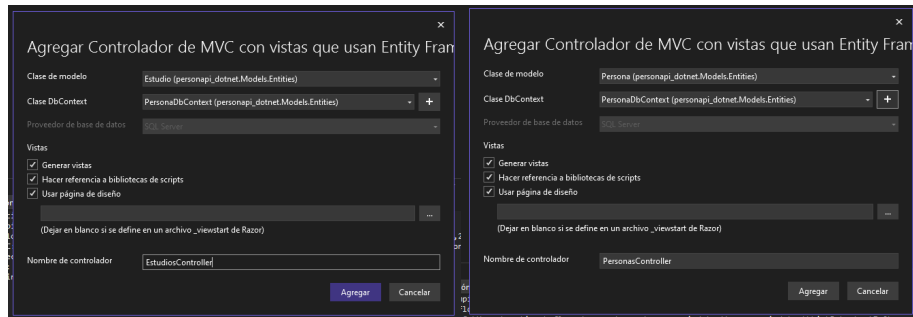
## 11. Crear Interfaces, Crear Repositorios, Crear Controladores.

Nota: Los siguientes pasos se realizan por entidad con el fin de crear cada uno de estos componentes para cada entidad.

1. Agregar un nuevo elemento “Controlador de MVC con vistas que usen Ef”



2. Asignar el nombre al controlador, seleccionar la clase del modelo y seleccionar el DbContext.



### 3. Verificar el código generado

```

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
personapi_dotnet.Controllers.EstudiosC
0 referencias
public EstudiosController(PersonaDbContext context)
{
    _context = context;
}
// GET: Estudios
public async Task<IActionResult> Index()
{
    var personaDbContext = _context.Estudios.Include(e =>
    return View(await personaDbContext.ToListAsync());
}
// GET: Estudios/Details/5/5
public async Task<IActionResult> Details(int? idProf, int

```

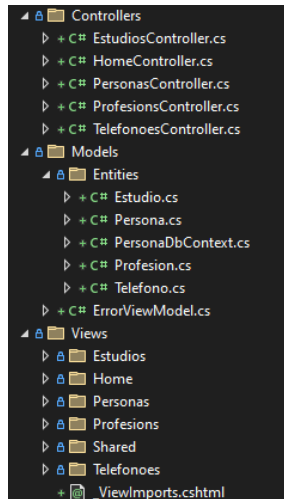
### 4. Revisar la vista

```

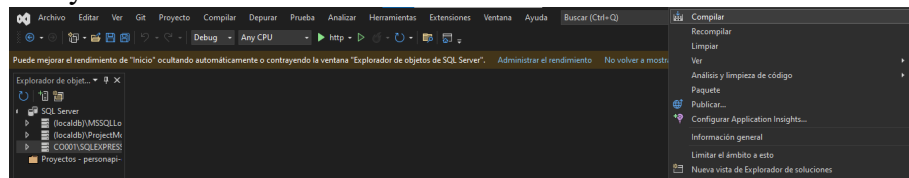
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Index.cshtml
@model IEnumerable<personapi_dotnet.Models.Entities.Estudio>
@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Fecha)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Univer)
            </th>
        </tr>
    </thead>
    <tbody>

```

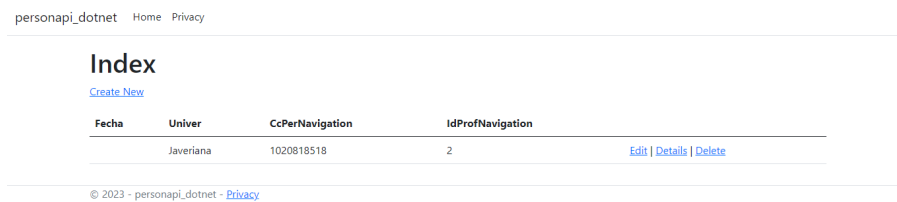
Finalmente se tiene un controlador y una vista para cada entidad del proyecto. De esta manera, se puede interactuar con las entidades y realizar operaciones de creación, lectura, actualización y eliminación de datos a través de la interfaz de usuario.



## 5. Guardar y Generar todo

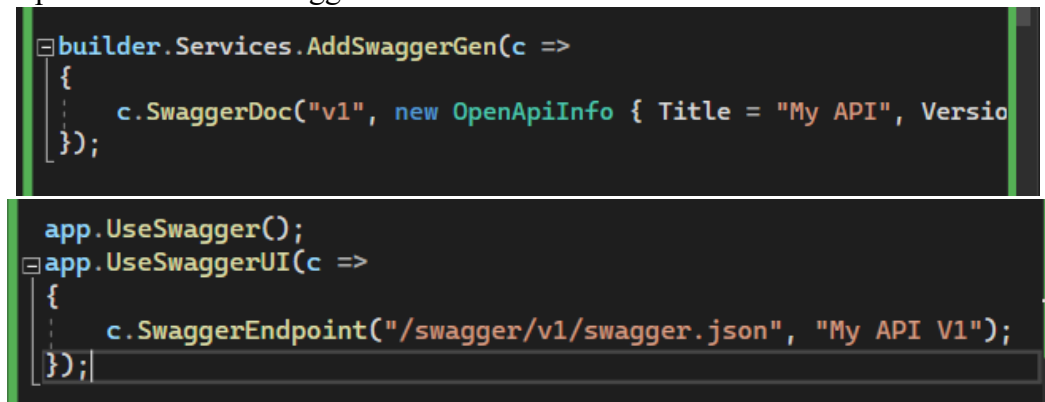


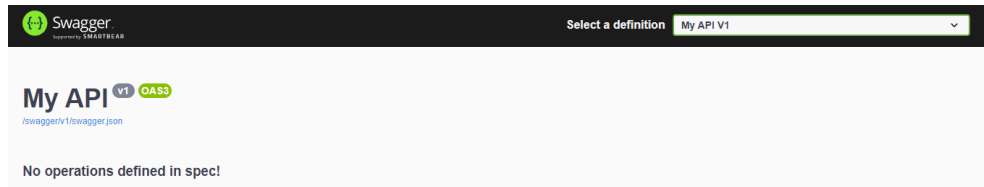
## 6. Ejecutar



## 12. Desplegar

## 13. Implementación de Swagger





9. Hacer push al repositorio
10. Crear TAG

## Conclusiones

La arquitectura monolítica permite un desarrollo y despliegue más sencillos en comparación con enfoques basados en microservicios, lo que facilita la construcción de una aplicación para un escenario con menos complejidad.

El patrón MVC ayuda a separar las responsabilidades de la aplicación, organizando el código en modelos (representaciones de datos), vistas (interfaces de usuario) y controladores (lógica de negocio). Esta separación mejora la mantenibilidad y la comprensión del código.

El patrón DAO proporciona una capa de abstracción entre la aplicación y la base de datos, facilitando el acceso y la manipulación de los datos almacenados en MS SQL Server 2019.

La combinación de REST y Swagger 3 permite crear APIs que son fácilmente comprensibles, consumibles y documentadas. Swagger 3 también facilita la generación de código y la creación de pruebas automatizadas.

## Lecciones Aprendidas

Elegir una arquitectura y patrones de diseño adecuados para el problema a resolver es crucial para el éxito del proyecto. En este caso, el monolito y los patrones MVC y DAO resultaron apropiados para el escenario planteado.

La selección de un stack tecnológico bien integrado y ampliamente adoptado, como .NET 6 y MS SQL Server 2019, puede facilitar el desarrollo y mejorar la productividad al brindar acceso a recursos y herramientas de calidad.

La documentación y especificación de APIs mediante Swagger 3 es una práctica valiosa que facilita la colaboración y la interoperabilidad entre diferentes equipos y sistemas.

La experiencia práctica en la construcción de aplicaciones utilizando tecnologías y patrones modernos es esencial para el crecimiento como desarrollador de software y arquitecto de soluciones. La aplicación de estos conceptos en proyectos futuros puede mejorar la calidad y la eficiencia en el desarrollo de software.

## Referencias

CursosAula21. (s. f.). ¿Qué es .NET? Retrieved from <https://www.cursosaula21.com/que-es-net/>

Mindmajix. (n.d.). SQL Server 2019: What's New, Features, Installation, and Benefits. Retrieved from <https://mindmajix.com/sql-server-2019#features>

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine. Retrieved from <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

OpenAPI Initiative. (2017). OpenAPI Specification 3.0. Retrieved from <https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.0.md>

Swagger. (n.d.). What is Swagger. Retrieved from <https://swagger.io/resources/articles/what-is-swagger/>