

LABORATORIO #1

ANDRES ARMANDO SANCHEZ MARTIN

ANDRES FELIPE DUARTE LEAL
JUAN MANUEL AGUIAR OROZCO
HUMBERTO RUEDA CATAÑO



Pontificia Universidad
JAVERIANA
Colombia

FACULTAD DE INGENIERÍA

ARQUITECTURA DE SOFTWARE

PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTÁ D.C.

2024

1. Marco Conceptual

1.1 .NET Framework

Acorde a la documentación de Microsoft, .NET es un marco de desarrollo de software para compilar y ejecutar aplicaciones en Windows. Existen varias implementaciones de .NET dependiendo de cada uno de los sistemas operativos, ya sea Linux, macOS, entre otros más.

- **.NET Framework:** Esta es la implementación que se considera original, sirve únicamente para Windows y permite la ejecución de sitios web, servicios, ETC
- **.NET:** Esta es la implementación multiplataforma, cumple con las mismas funciones de .NET Framework solo que esta implementación sirve para los demás sistemas operativos aparte de Windows.

.NET de manera genérica, maneja en su arquitectura dos componentes principales, uno llamado Common Language Runtime y la biblioteca de clases del mismo .NET. Common Language Runtime (CLR) es el motor de ejecución de las aplicaciones y las bibliotecas de clases son las que proporcionan API's y tipos para funciones comunes.

1.2 Patrón de diseño MVC

El patrón de modelo, vista, controlador (MVC) es un modelo de diseño el cual se utiliza para dividir la aplicación en 3 grupos principales que son los mencionados anteriormente, esto se hace con el objetivo de separar intereses. A continuación, se presentan cada una de las responsabilidades que tiene cada uno de estos componentes:

- **Modelo:** La parte del modelo representa el estado de la aplicación, o para ser mas específicos, se encarga del moldeado de los datos de la aplicación.
- **Vista:** Las vistas son las que se encargan de representar los datos mediante una interfaz al usuario.

- **Controlador:** Los controladores son los que controlan la interacción del usuario, trabajan con el modelo y, en última instancia, seleccionan una vista para representarla.

1.3 Data Access Object

El patrón DAO es un patrón de diseño que se utiliza para separar la lógica de acceso a datos de la lógica de negocio en una aplicación. Proporciona una capa de abstracción entre la aplicación y la base de datos, lo que permite cambiar la implementación de la base de datos sin afectar otras partes del sistema.

1.4 MS SQL Server

Microsoft SQL Server es un sistema de administración de bases de datos relacionales, esta herramienta se encuentra disponible para Windows y Linux. Dentro de la siguiente tabla extraída de la documentación de Microsoft se muestra las tecnologías que este sistema implementa.

Componente	Descripción
Motor de base de datos	El Motor de base de datos es el servicio principal para almacenar, procesar y proteger datos. El motor de base de datos proporciona acceso controlado y procesamiento de transacciones para cumplir los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa. El motor de base de datos también proporciona compatibilidad enriquecida para mantener la continuidad empresarial a través de la continuidad empresarial y la recuperación de bases de datos - SQL Server.
Machine Learning Services (MLS)	SQL Server Machine Learning Services admite la integración del aprendizaje automático a través de los lenguajes populares R y Python en los flujos de trabajo empresariales. Machine Learning Services (en base de

	<p>datos) integra R y Python con SQL Server, lo que simplifica la compilación, el reciclaje y los modelos de puntuación al llamar a procedimientos almacenados. Machine Learning Server proporciona una compatibilidad de escala empresarial con R y Python sin necesidad de utilizar SQL Server.</p>
Integration Services (SSIS)	<p>SQL Server Integration Services es una plataforma para generar soluciones de integración de datos de alto rendimiento, lo que incluye paquetes que proporcionan procesamiento de extracción, transformación y carga (ETL) para almacenamiento de datos.</p>
Analysis Services (SSAS)	<p>SQL Server Analysis Services es una plataforma y un conjunto de herramientas de datos analíticos para Business Intelligence en un entorno personal, de equipo o empresa. Los servidores y los diseñadores de cliente admiten soluciones OLAP tradicionales, nuevas soluciones de modelado tabular, y análisis y colaboración de autoservicio mediante Power Pivot, Excel y un entorno de SharePoint Server. Analysis Services también incluye minería de datos para permitir descubrir las relaciones y los patrones ocultos en grandes volúmenes de datos.</p>
Reporting Services (SSRS)	<p>SQL Server Reporting Services ofrece funcionalidad empresarial de informes habilitados para web. Puede crear informes que extraigan contenido a partir de diversos orígenes de datos, publicar informes con distintos formatos y administrar la seguridad y las suscripciones de forma centralizada.</p>
Replicación	<p>SQL Server Replication consiste en un conjunto de tecnologías para copiar y distribuir datos y objetos de base de datos de una base de datos a otra y, a continuación, sincronizar las bases de datos para mantener la coherencia. La replicación permite distribuir datos a diferentes ubicaciones y a usuarios remotos o móviles mediante redes de área local y de área extensa, conexiones</p>

	de acceso telefónico, conexiones inalámbricas e Internet.
Data Quality Services (DQS)	Data Quality Services proporciona una solución de limpieza de datos controlada por conocimiento. DQS permite generar una base de conocimiento y usarla para realizar tareas de corrección de datos y eliminación de datos duplicados, usando medios asistidos por ordenador e interactivos. Puede usar servicios de consulta de datos basados en la nube y puede generar una solución de administración de datos que integra DQS con SQL Server Integration Services y Master Data Services.
Master Data Services (MDS)	Master Data Services es la solución de SQL Server para la administración de datos maestros. Una solución basada en Master Data Services ayuda a asegurarse de que los informes y los análisis se basan en la información correcta. Con Master Data Services se crea un repositorio central de los datos maestros y se mantiene un registro auditable y protegible de los mismos a medida que van cambiando con el tiempo.

1.5 REST

Representational State Transfer (REST) es un estilo de arquitectura para sistemas distribuidos basado en comunicación mediante solicitudes HTTP. A continuación, se presentan las características principales de REST.

- **Recursos:** En REST, todo es considerado como un recurso, que puede ser cualquier cosa que se pueda nombrar, como un documento, una imagen, un servicio web, etc.
- **Identificación de recursos:** Cada recurso en un sistema RESTful se identifica de manera única mediante un URI (Identificador de Recursos Uniforme).

- **Operaciones sobre recursos:** Las operaciones comunes que se realizan sobre recursos son GET (obtener), POST (crear), PUT (actualizar) y DELETE (eliminar).
- **Interfaz uniforme:** REST utiliza una interfaz uniforme para acceder y manipular recursos, lo que facilita la comunicación entre sistemas.
- **Sin estado:** Las comunicaciones en una arquitectura REST son sin estado, lo que significa que cada solicitud del cliente al servidor debe contener toda la información necesaria para comprender la solicitud. Esto simplifica la gestión del servidor y permite una mejor escalabilidad.

1.6 Swagger

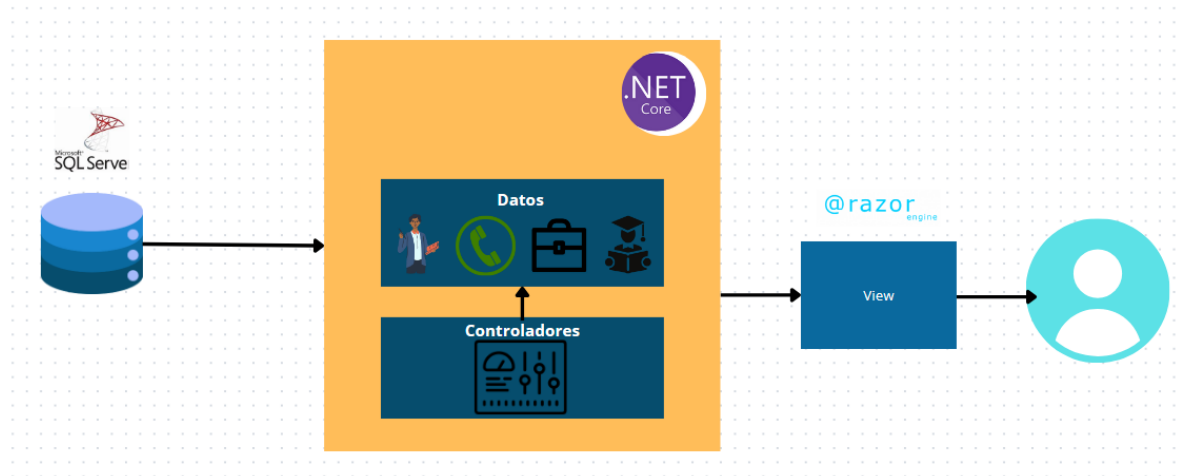
Swagger es un framework de código abierto y respaldado por un gran ecosistema de herramientas, que como desarrolladores nos ayudan a diseñar, construir, documentar y consumir un servicio RESTful.

Una de las más utilizadas es Swagger UI Tool, que permite tener una interfaz de usuario que nos muestre para una API las peticiones y la documentación de las mismas.

2. Diseño

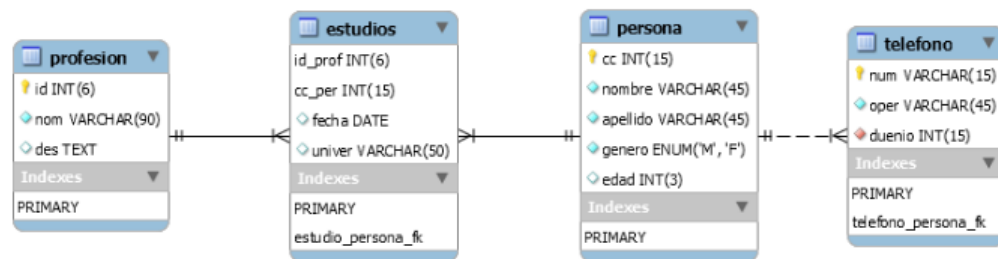
2.1 Diagrama de alto nivel

El diagrama de alto nivel muestra un monolítico simple que utiliza tecnología .NET Core y SQL Server para almacenar y recuperar datos. Sus componentes principales incluyen una base de datos Microsoft SQL Server para el almacenamiento, .NET Core como plataforma de desarrollo, Razor Pages para la creación de páginas web dinámicas, controladores para la lógica de negocio, y vistas para la interfaz de usuario. El flujo de datos comienza con el usuario accediendo a la página web, que se renderiza utilizando Razor Pages y envía una solicitud al servidor. Luego, el servidor web invoca el controlador correspondiente, consulta la base de datos para obtener los datos necesarios, y finalmente, los devuelve a la vista que los renderiza en la página.



2.3 Modelo de datos

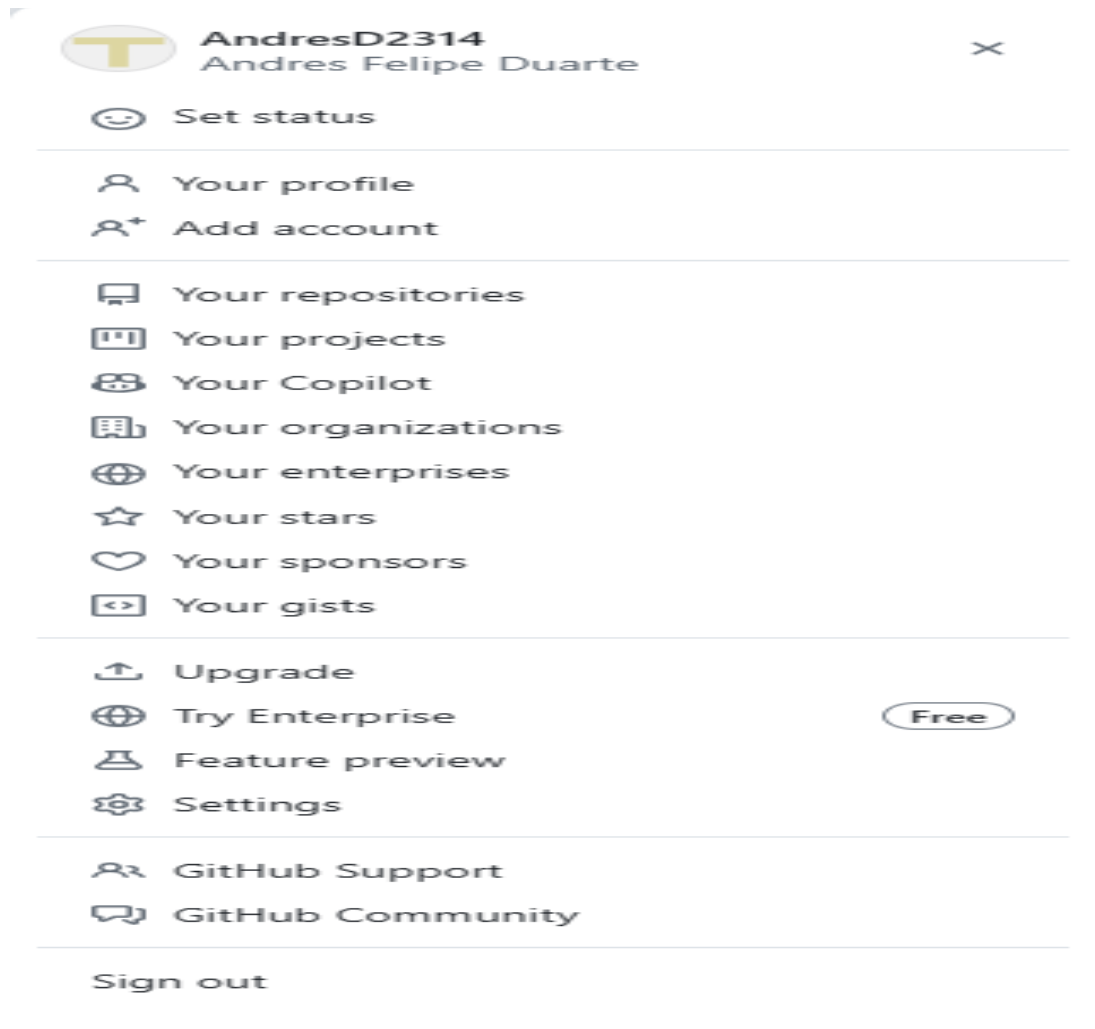
El modelo de datos muestra relaciones donde una persona puede haber realizado varios estudios y una persona puede tener uno o varios teléfonos. Estas relaciones se establecen entre las entidades Estudios y Personas, y entre Personas y Telefonos, respectivamente.



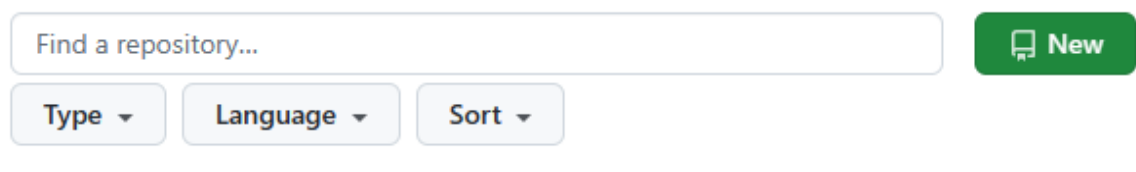
3. Procedimiento

3.1 Creación de repositorio

Para la creación de este repositorio es necesario estar registrado con una cuenta en GitHub. Una vez tengamos la cuenta creada, nos encontraremos con la página principal de GitHub. Nos dirigimos a nuestro perfil y vamos a la parte de nuestros repositorios.



Una vez nos encontremos en la parte de nuestros repositorios, seleccionamos la opción new



Cuando le damos clic a new, nos redirigirá a una nueva ventana en la cual pondremos el nombre del repositorio que en este caso es personaapi-dotnet

Owner *

AndresD2314

Repository name *

personaapi-dotnet

✓ personaapi-dotnet is available.

Great repository names are short and memorable. Need inspiration? How about [refactored-octo-chainsaw](#) ?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Una vez configuremos la creación de nuestro repositorio le daremos a la opción Create Repository y se creara el repositorio en Github.

3.2 Instalación de SQL Server 2019 Express

Para la instalación de SQL Server, nos dirigimos a la pagina oficial de Microsoft en el navegador y descargamos la versión del 2019. Una vez que nos descargue el instalador, le damos doble clic y aparecerá la siguiente ventana emergente.

SQL Server 2019



Express Edition

Seleccione un tipo de instalación:

Básica

Seleccione el tipo de instalación Básica para instalar la funcionalidad de motor de base de datos de SQL Server con la configuración predeterminada.

Personalizado

Seleccione el tipo de instalación Personalizada para ejecutar paso a paso el asistente para instalación de SQL Server y elija los elementos que quiera instalar. Este tipo de instalación es detallado y lleva más tiempo que la instalación Básica.

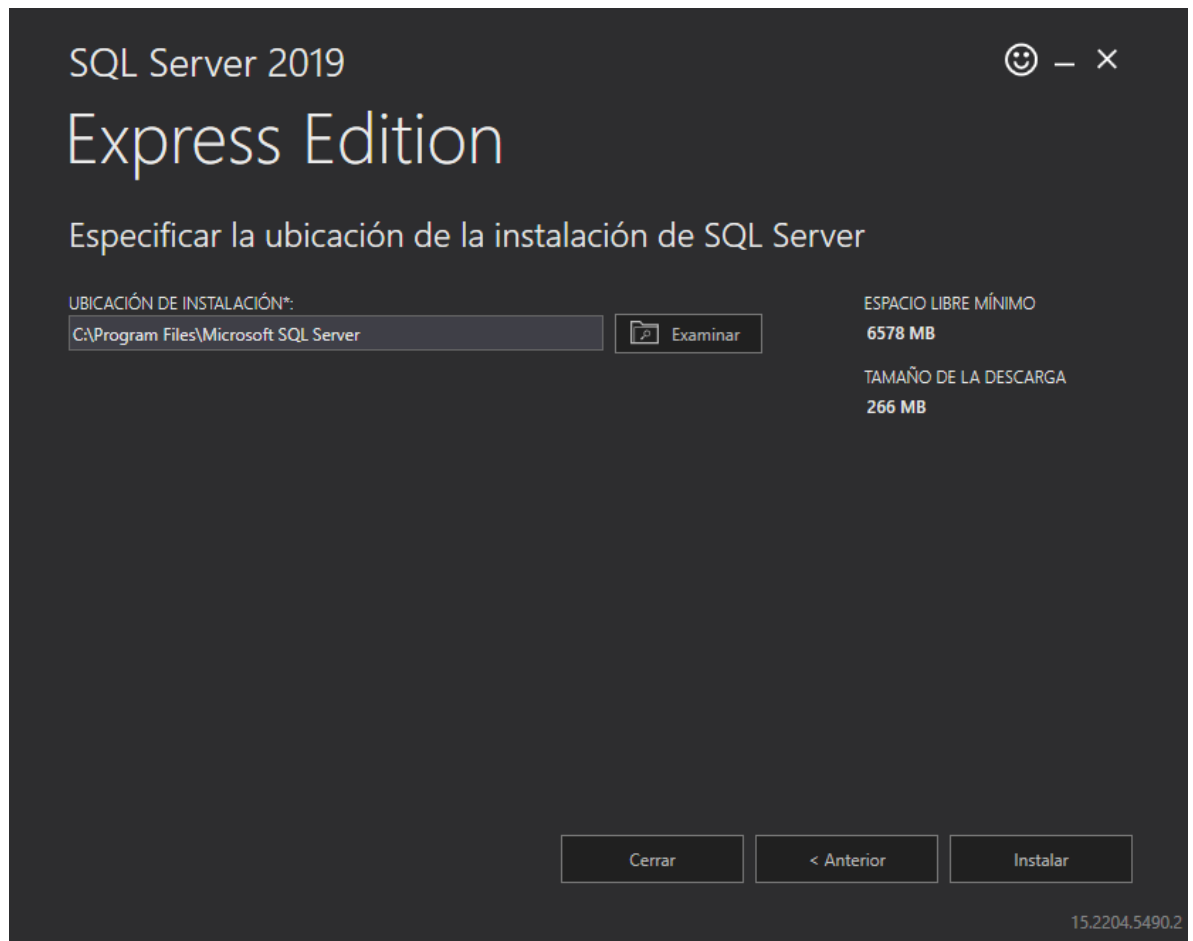
Descargar medios

Descargue los archivos de instalación de SQL Server ahora e instálelos más tarde en una máquina de su elección.

SQL Server transmite a Microsoft información sobre su experiencia de instalación, así como otros datos de uso y rendimiento, con el fin de mejorar el producto. Para obtener más información sobre el procesamiento de datos y los controles de privacidad, y para desactivar la recopilación de esta información después de la instalación, vea [documentación](#)

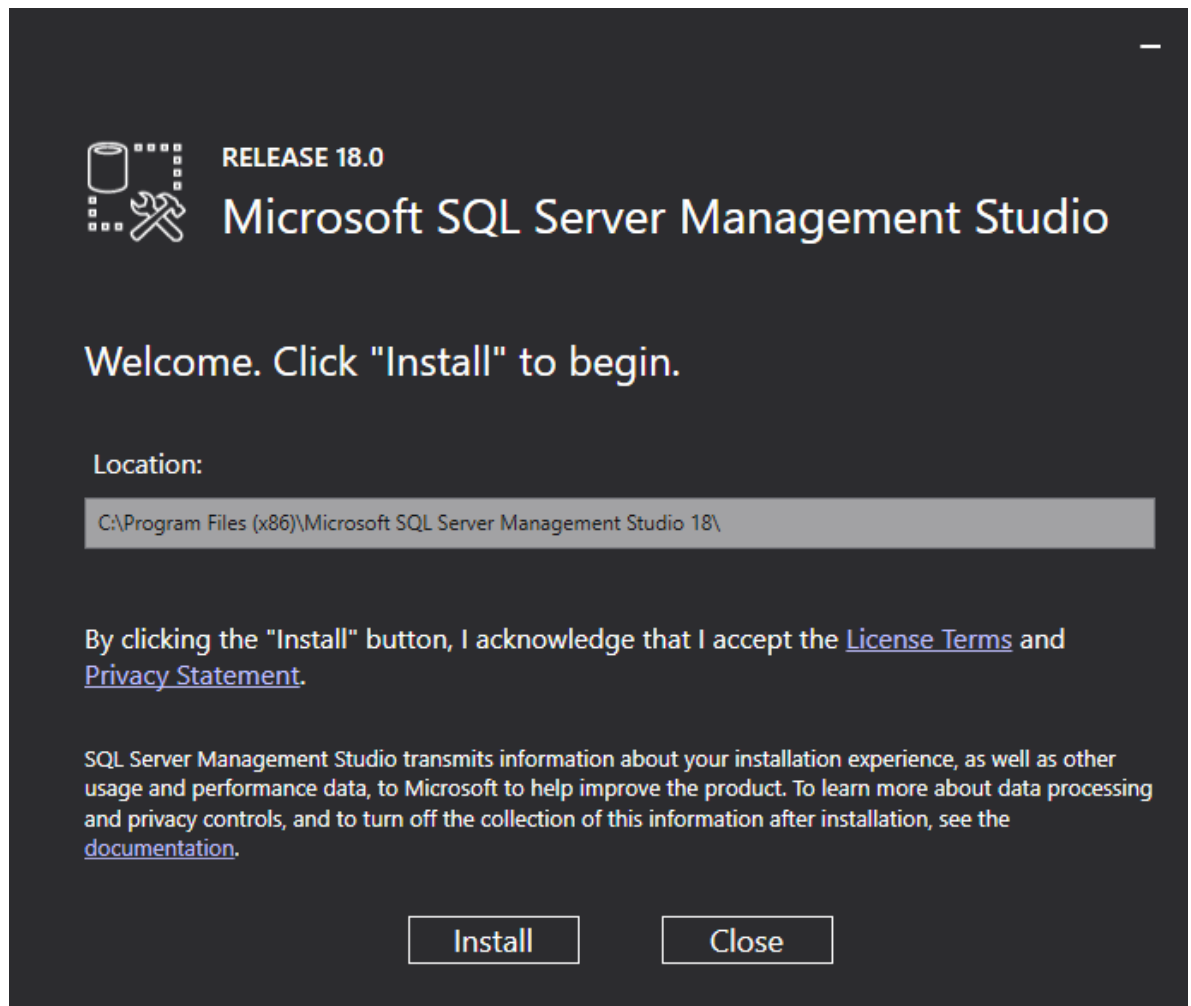
15.2204.5490.2

Una vez acá, seleccionamos la opción básica y configuramos donde queremos que se guarde la instalación, una vez realizado eso le damos a instalar.



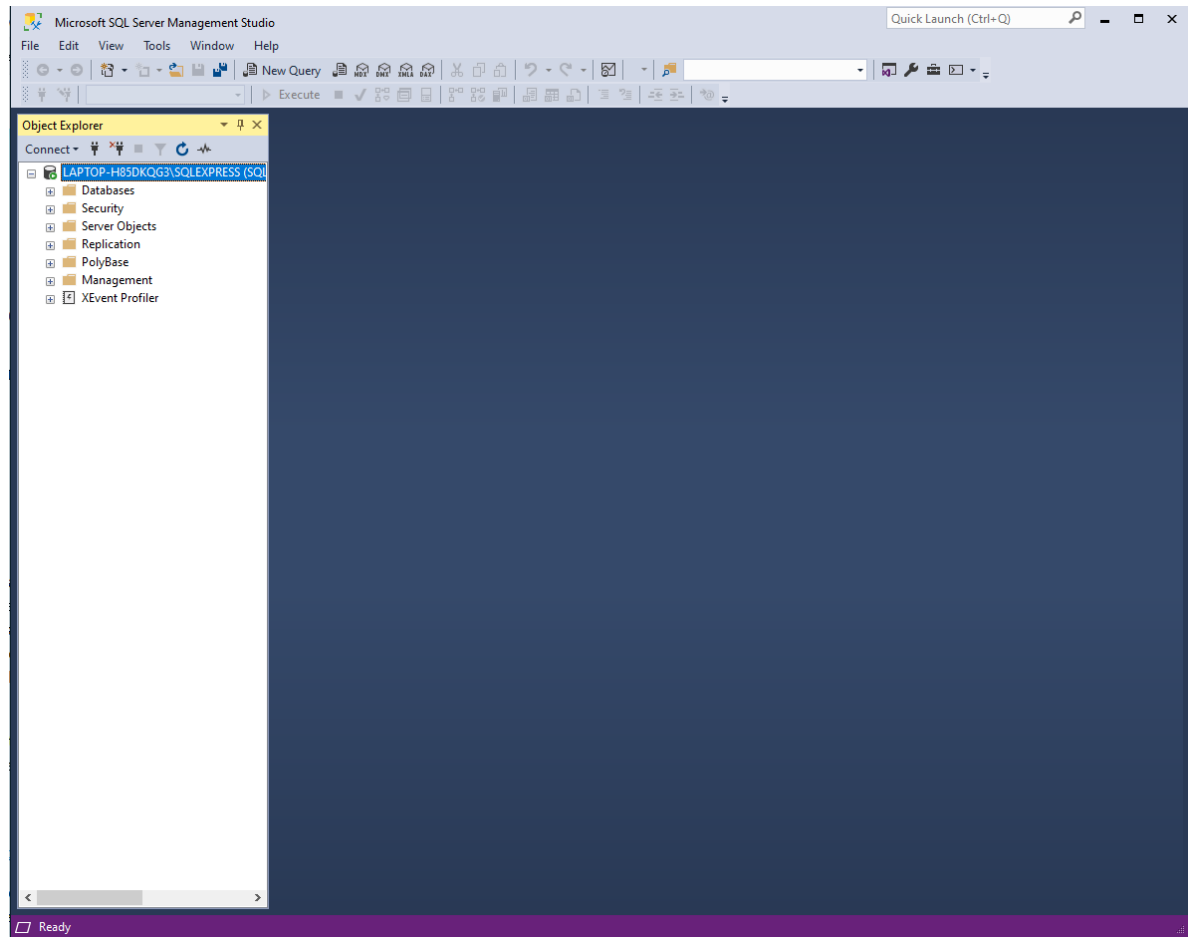
3.3 Instalación de SQL Server Management Studio 2018

Para la instalación de SQL Server, nos dirigimos a la página oficial de Microsoft en el navegador y descargamos la versión del 2018. Una vez que nos descargue el instalador, le damos doble clic y aparecerá la siguiente ventana emergente y sencillamente lo instalamos.

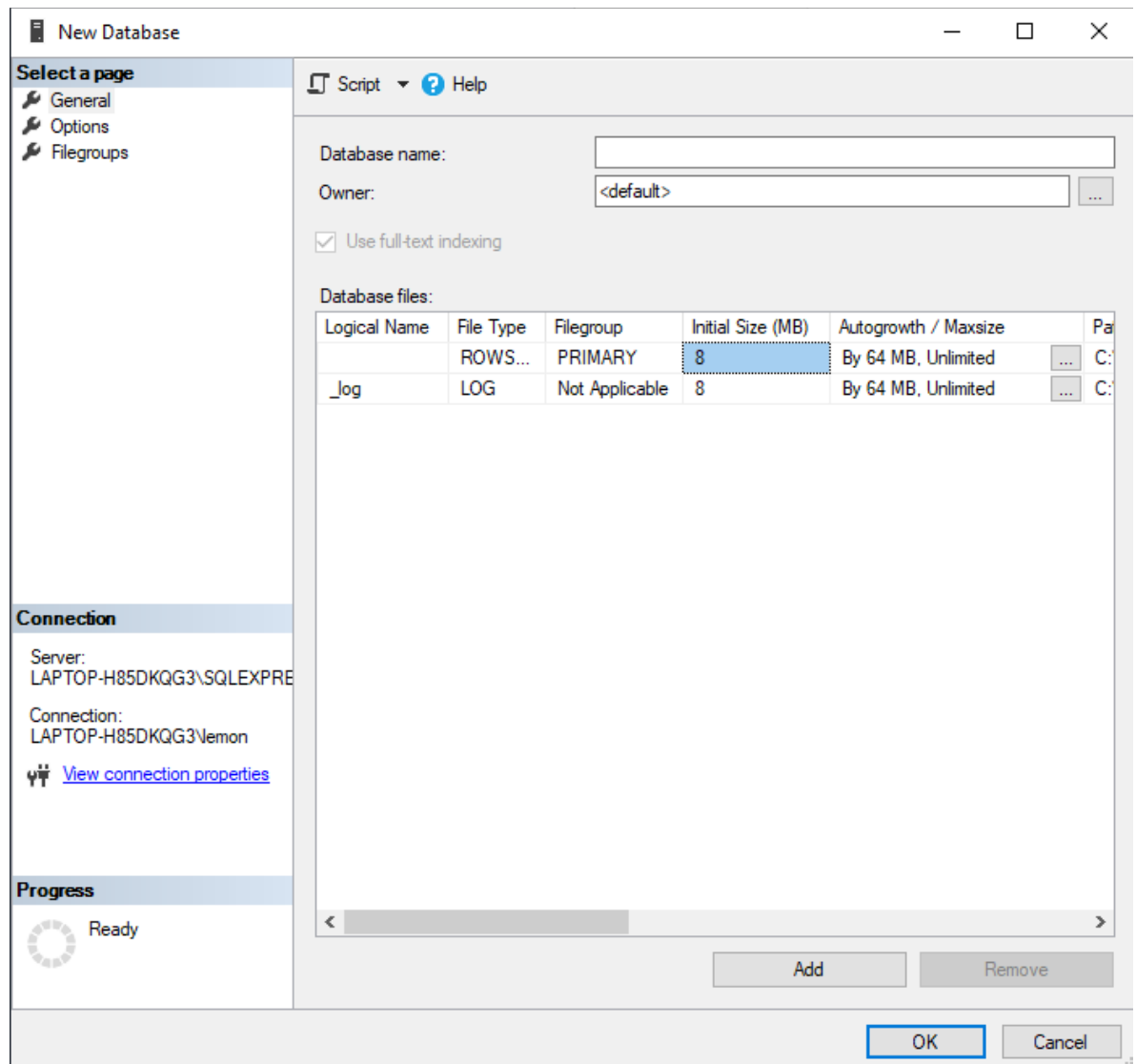


3.4 Creación de la base de datos persona_db

Una vez que se tenga instalado el SQL Server manager aparecerá la siguiente ventana de la aplicación.



En la parte que esta en azul la seleccionamos la parte donde dice Databases y creamos una nueva base de datos.



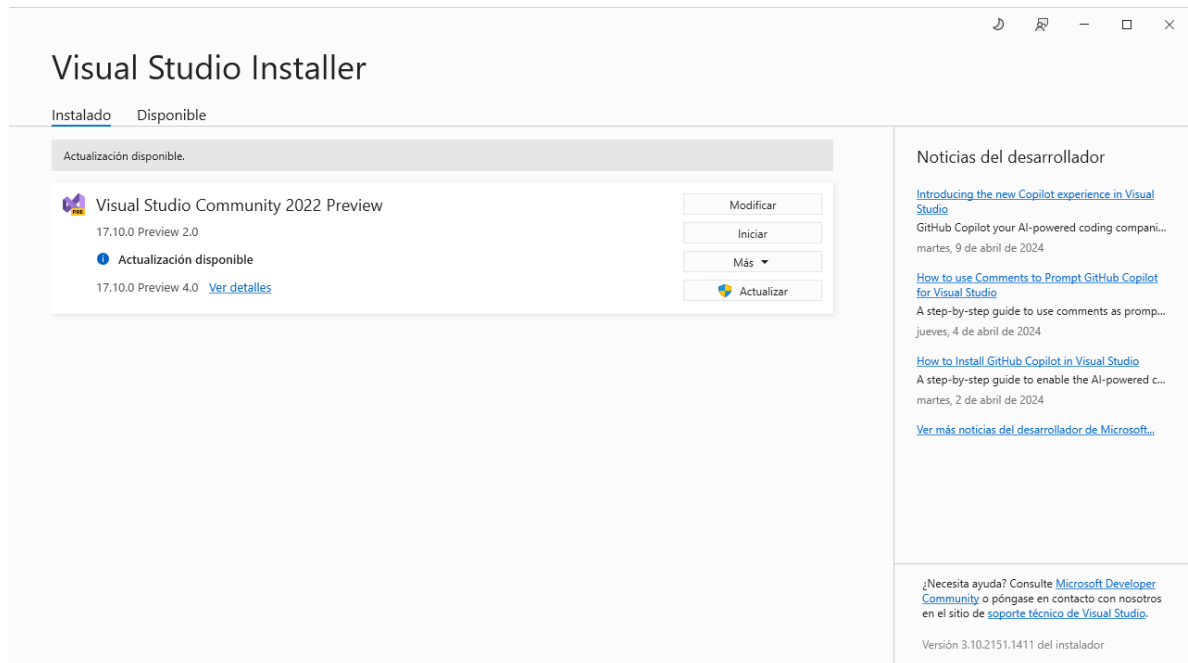
Una vez estemos en esta ventana, ponemos el nombre de `people_db` y en la parte owner le asignamos el `sa`. Una vez eso listo, le damos OK para crear la base de datos.

3.5 Creación de las tablas

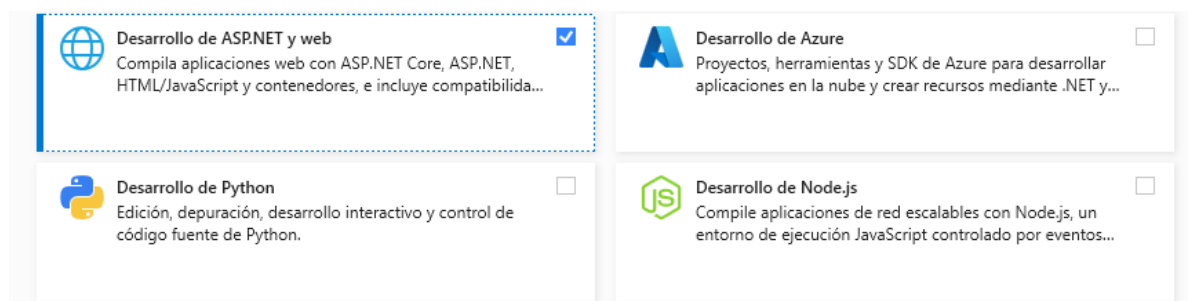
Para la creación de tablas, únicamente se selecciona la base de datos creada anteriormente, y mediante un script que se encuentra en el repositorio de Git, creamos las tablas dentro de la base de datos.

3.6 Instalación de Visual Studio Community 2022

Para este caso, como se tenía previamente instalado el Visual Studio, solo fue necesario agregarle los complementos necesarios. Para hacer esto, junto a la instalación de Visual Studio se tiene un instalador, se abre esa aplicación y directamente desde ahí se instalan estos complementos en la parte de modificar.



Una vez ahí, en la parte de cargas de trabajo, seleccionábamos en el apartado de Web y Nube el complemento de desarrollo de ASP.NET y web y en la parte de otros conjuntos de herramientas, seleccionábamos el complemento de Almacenamiento y procesamiento de datos.



Otros conjuntos de herramientas (5)



Almacenamiento y procesamiento de datos

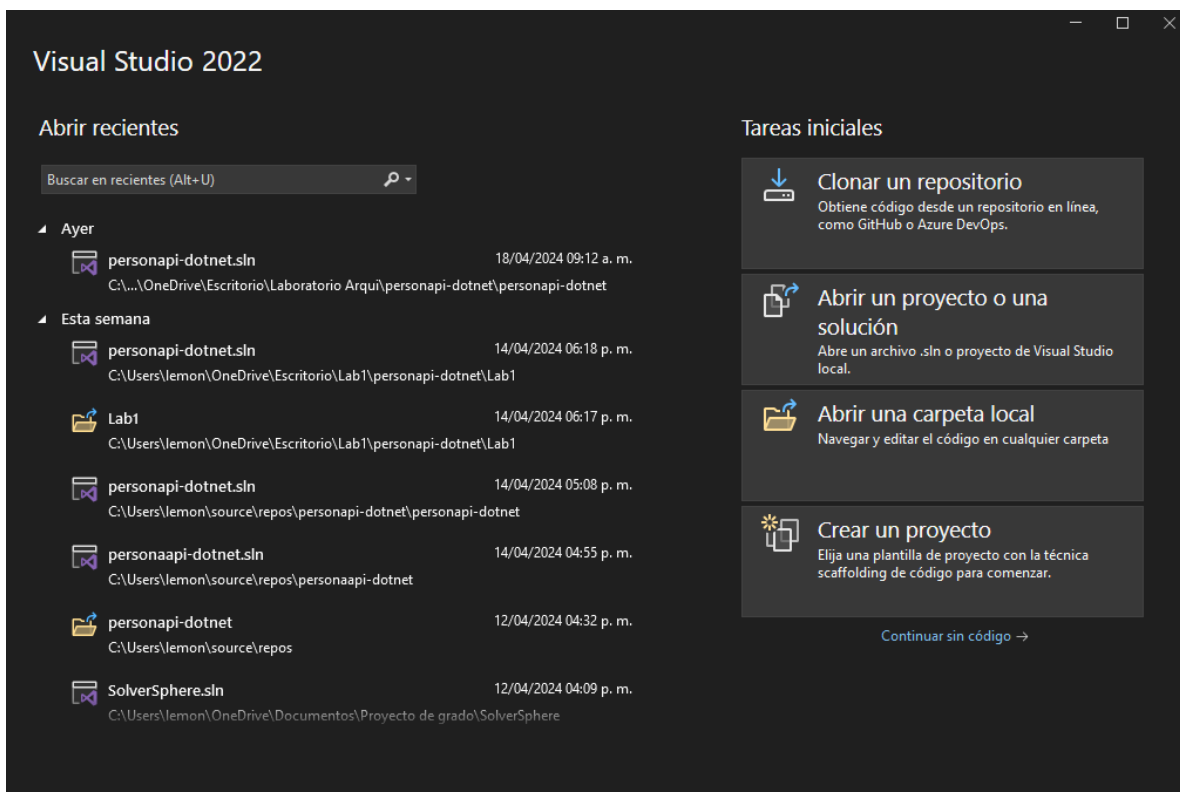
Conecte, desarrolle y pruebe soluciones de datos con SQL Server, Azure Data Lake o Hadoop.



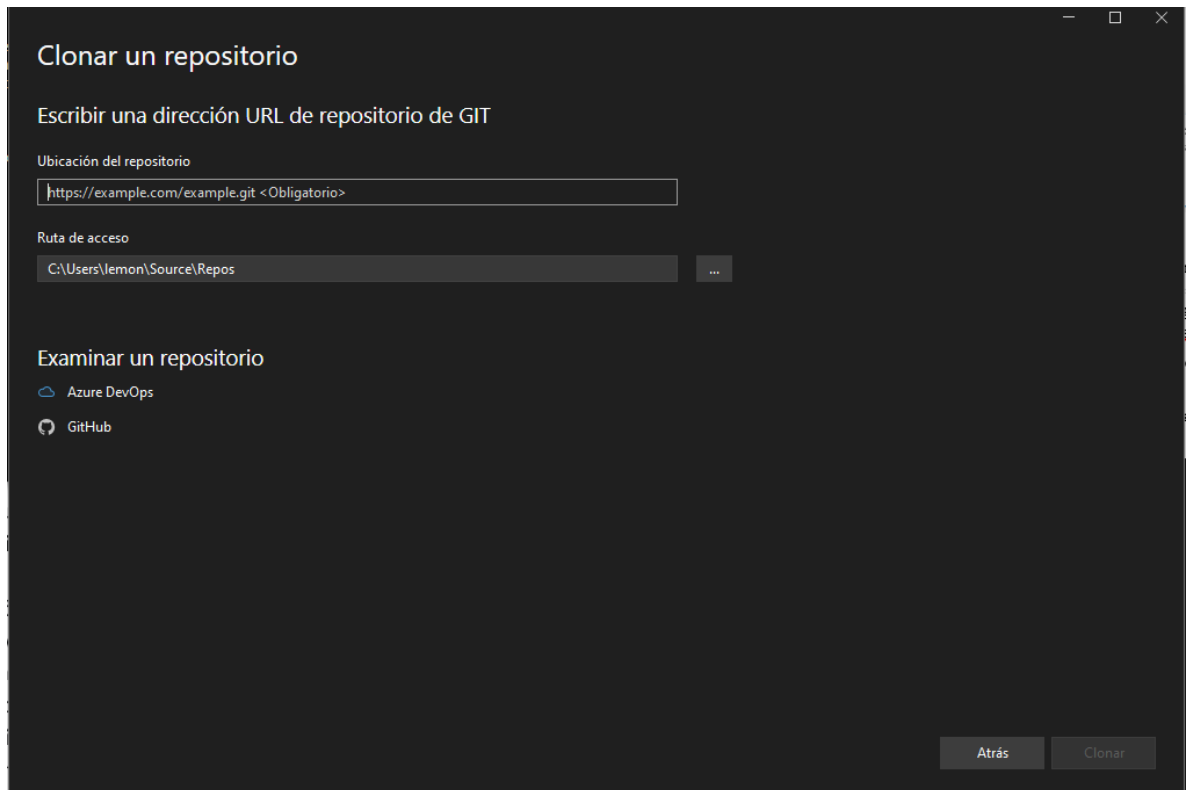
El resto de los complementos, se encontraron en la parte de componentes individuales en donde aparece una lista de todos los componentes disponibles para Visual Studio. Únicamente se descargaron Plantillas de proyecto y elementos de .Net Framework y Características avanzadas de ASP.NET.

3.7 Clonación del repositorio creado y configuración de la base de datos dentro de Visual Studio

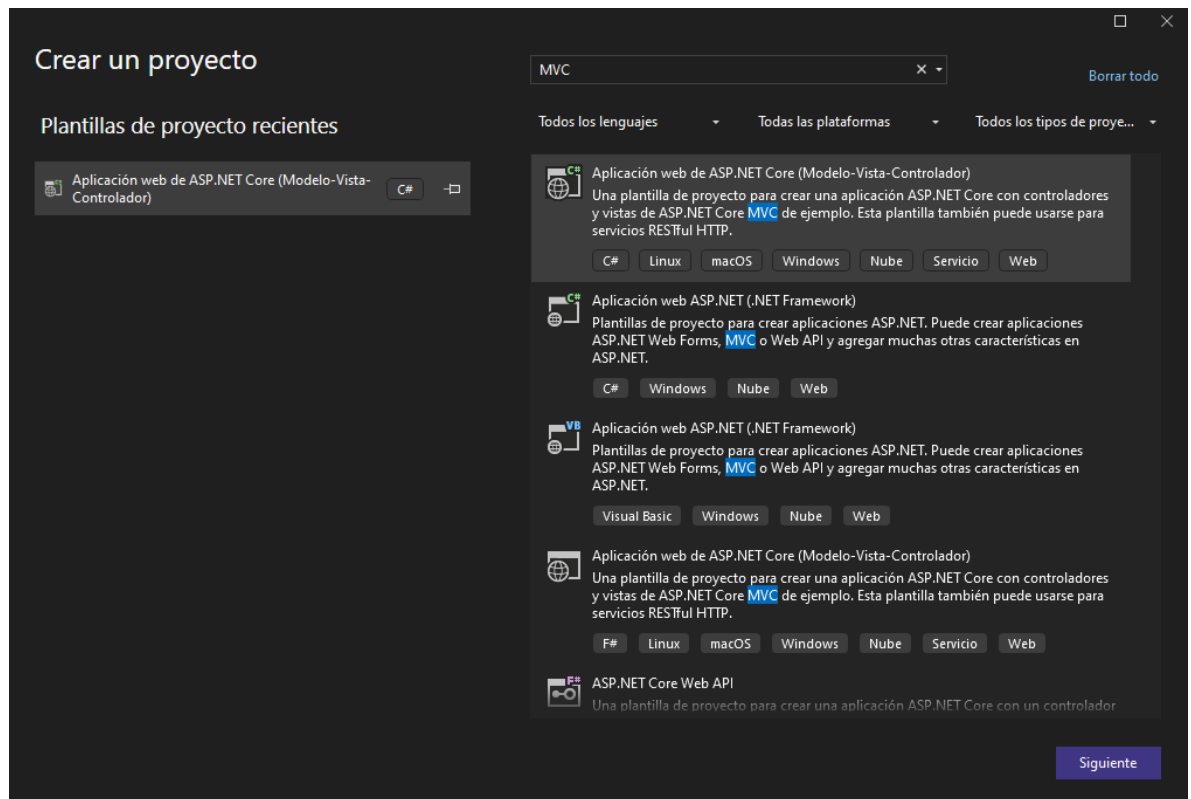
Una vez tengamos todo instalado, abrimos el Visual Studio y nos aparecerá la siguiente ventana



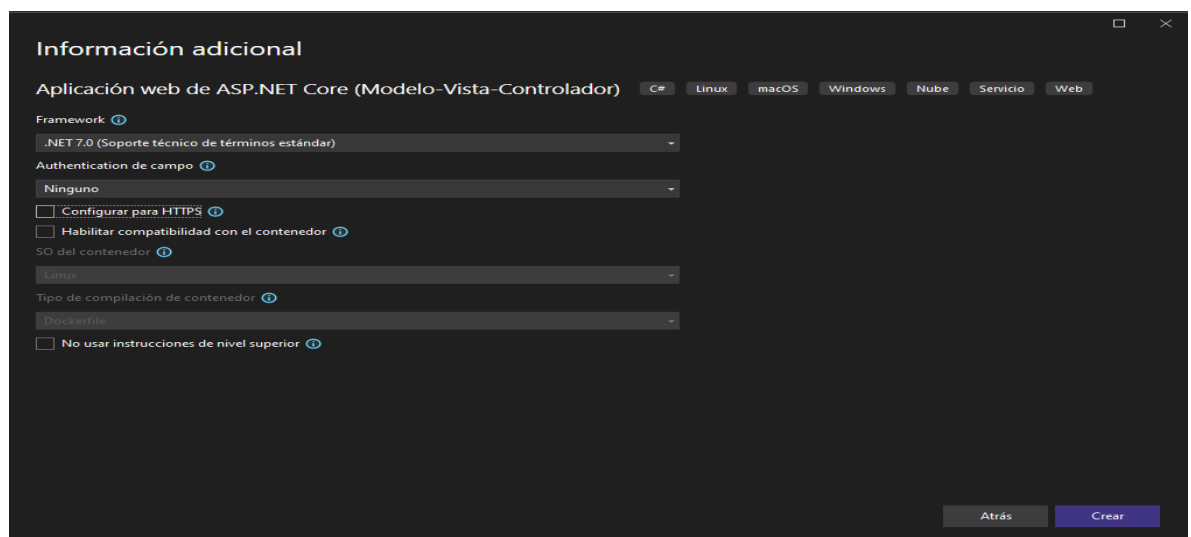
Una vez acá, seleccionamos la opción de clonar un repositorio, una vez ahí dentro, copiamos la URL de nuestro repositorio y la pegamos y clonamos el repositorio



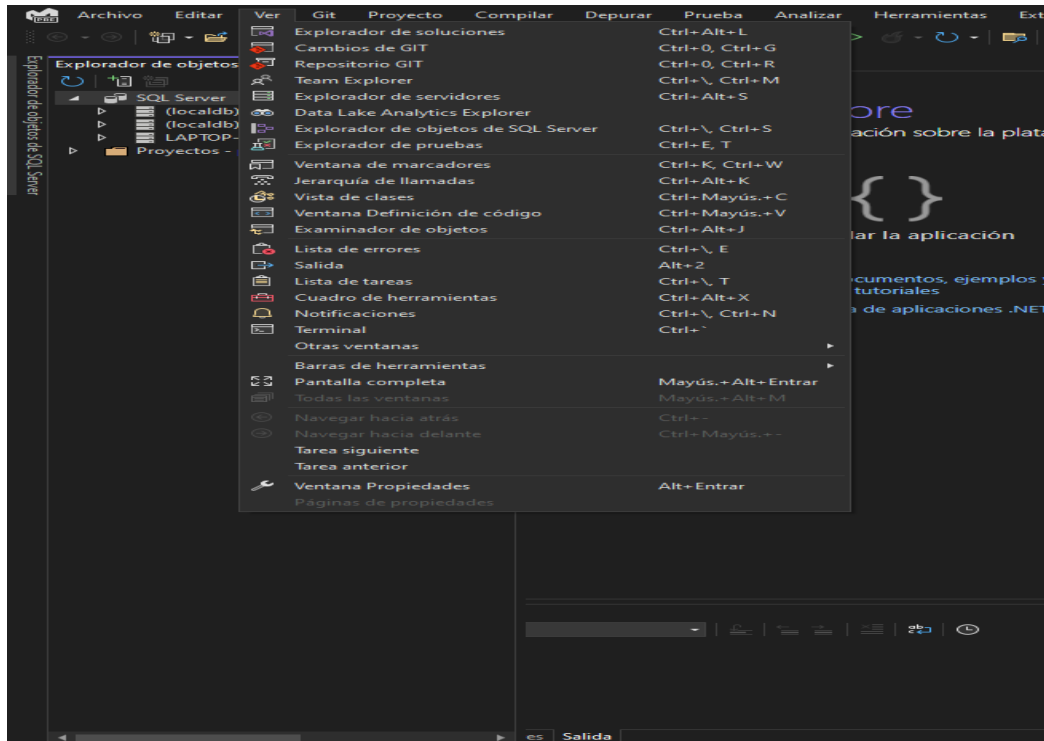
Una vez se cree, nos dirigiremos a la parte de archivo y crearemos un nuevo proyecto y seleccionamos la plantilla del modelo vista y controlador. Una vez seleccionamos esa opción, le damos a siguiente



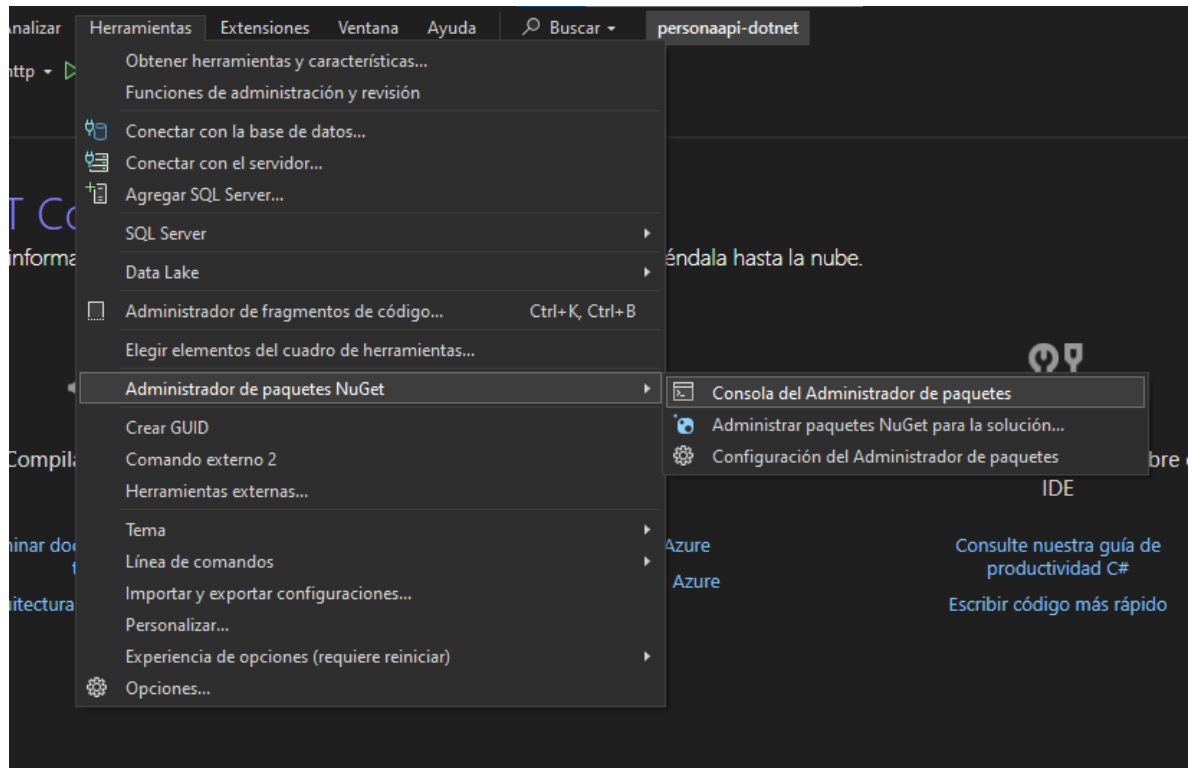
En la nueva ventana, al nombre del proyecto lo llamamos exactamente igual que el repositorio y le damos otra vez a siguiente. Por ultimo, en la ventana final, seleccionamos el framework de .NET 7.0 y desmarcamos la opción de configurar HTTPS y creamos el proyecto.



Una vez ya creamos el proyecto, este se abre de manera automática. Ahora para configurar la base de datos, nos dirigimos a la opción de Ver y seleccionamos la opción de vista de Explorador de objetos de SQL Server.

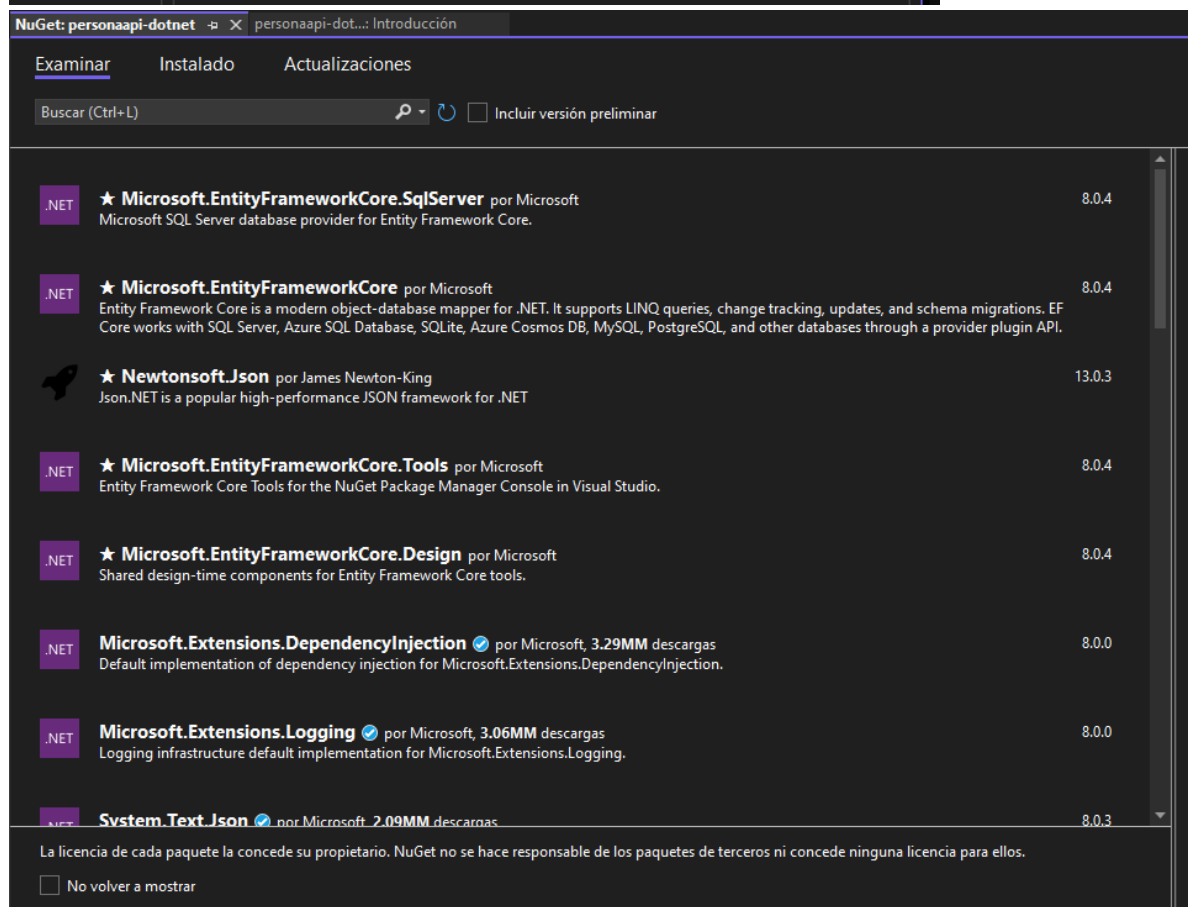
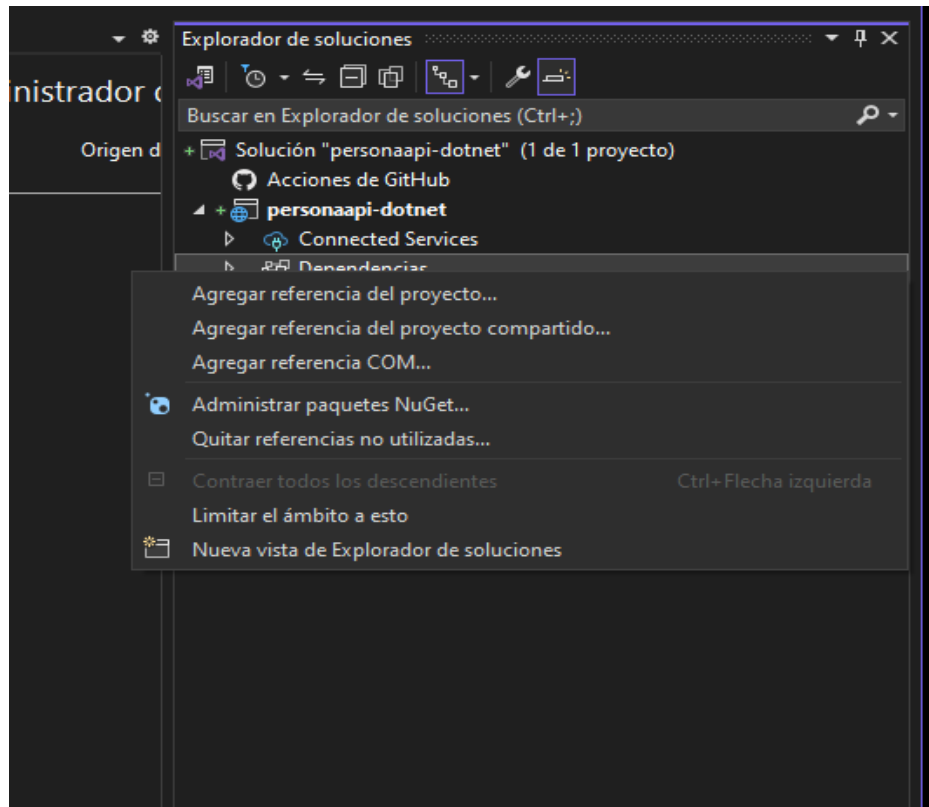


Una vez seleccionemos esa opción, se nos abrirá una pestaña en la parte izquierda de las conexiones de bases de datos que tenemos, para agregar la que se creo desde SQL Server Management, le damos clic derecho a SQL Server y agregamos la conexión de tipo local express, generalmente esta automáticamente se agrega sola, pero en caso de que no se hace el procedimiento anteriormente especificado. Una vez realizada la conexión, nos dirigimos a la parte Herramientas>Administrador de paquetes NuGet>Consola del Administrador de paquetes.

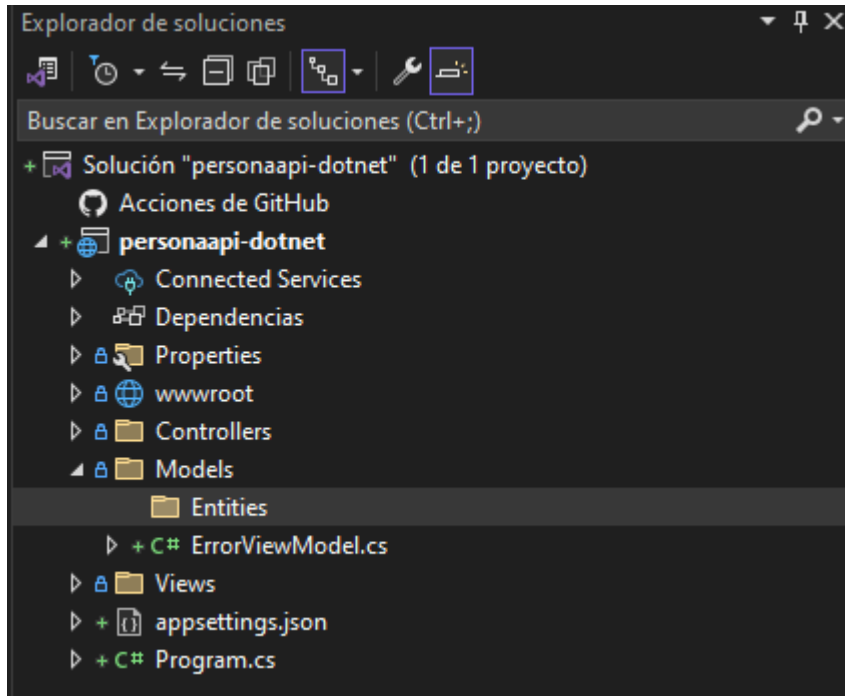


Una vez tengamos abierta la terminal, nos dirigimos al apartado de explorador de soluciones ubicado a mano derecha, le damos click derecho e instalamos los siguientes paquetes acorde a la versión del .NET utilizado.

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

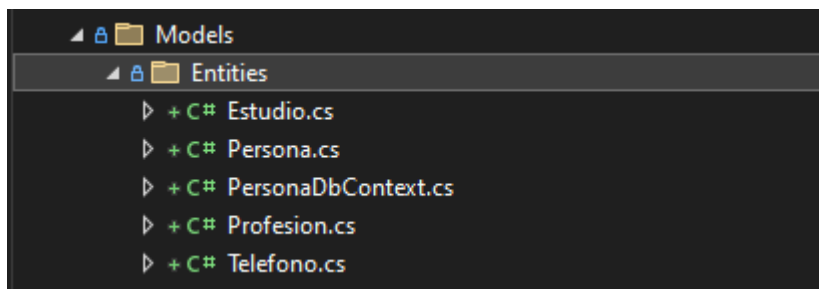


Una vez instalado los paquetes, se procede a crear una nueva carpeta dentro de la carpeta Models en el explorador de soluciones.



Una vez se crea esta carpeta, procedemos a ir a la consola de administrador de paquetes y se escribe el siguiente comando `Scaffold-DbContext "Server=localhost\SQLEXPRESS;Database=persona_db;Trusted_Connection=True;TrustServerCertificate=true"`

`Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models/Entities`. Gracias a este comando, se nos generan en código las entidades que se tienen dentro de la base de datos en la carpeta de Entities.



Y por ultimo y no menos importante, nos dirigimos al appsettings.json y agregamos la conexión.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "PubContext": "Server=LAPTOP-H8DKQG3\\SQLEXPRESS;Database=persona_db;Trusted_Connection=True"
  }
}
```

Ya con todos los pasos anteriores listos, lo único que procedería a realizarse es la implementación tanto de los controladores, repositorios e interfaces.

4. Conclusiones y lecciones aprendidas

Durante el desarrollo de este laboratorio, hemos experimentado una amplia gama de tecnologías disponibles para el desarrollo de APIs. A menudo, nos inclinamos hacia lo que conocemos mejor o lo que hemos aprendido durante nuestra formación académica, como Java y Spring Boot. Sin embargo, este proyecto nos presentó el desafío de trabajar con API utilizando el framework .NET, del cual no teníamos experiencia previa. A pesar de la complejidad inicial, nos adaptamos rápidamente gracias a la similitud de sintaxis con Java.

El uso de plantillas predefinidas resultó ser un recurso valioso que simplificó notablemente el acceso a la base de datos y la creación de entidades. Esto fue especialmente útil dado que ya teníamos un modelo de datos establecido; las dependencias de Microsoft SQL Server en Visual Studio automatizaron la generación de las entidades necesarias. Además, nos impresionó la versatilidad de la plantilla MVC en .NET. Gracias al componente de la vista proporcionado por Razor, pudimos realizar operaciones CRUD directamente en el navegador, eliminando la necesidad de utilizar aplicaciones como Postman para hacer llamadas a la API.

Aunque se tenía planeado realizar el despliegue inicialmente con Docker, tuvimos demasiados problemas creando la imagen para la aplicación, ya que siempre que intentábamos crear la imagen esta se demoraba entre 20 a 21 minutos en crearse y siempre se generaban distintos errores después de la compilación. Por lo cual se decidió dejar un .README en el tag mandado del repositorio

Este laboratorio nos permitió reconocer la potencia de las tecnologías utilizadas y ampliar nuestro conocimiento al descubrir nuevos frameworks para el desarrollo de APIs. Nos llevamos una lección invaluable sobre la importancia de explorar y experimentar con diversas herramientas tecnológicas para mejorar nuestras habilidades como desarrolladores.

5. Referencias

1. *Patrón MVC de Asp.net*. (s/f). Microsoft. Recuperado el 16 de abril de 2024, de <https://dotnet.microsoft.com/es-es/apps/aspnet/mvc>
2. *Overview of .NET Framework*. (s/f). Microsoft.com. Recuperado el 16 de abril de 2024, de <https://learn.microsoft.com/enus/dotnet/framework/get-started/overview>
3. Guamán, V. (2022, junio 24). *API con Asp.net MVC 6 y SQL Server mediante Entity Framework core 6 - Code First - Parte 1*. DEV Community. <https://nam10.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdev.to%2Fveronicaguamann%2Fapi-con-aspnet-mvc-6-y-sql-server-mediante-entity-framework-core-6-code-first-parte-2io5&data=05%7Co1%7Casanchezm%40javeriana.edu.co%7C8b47844444d4867aaa308daacb7df55%7Cdaf7990e8a3f409c9b762a5475098000%7Co%7Co%7C638012202991803905%7CUnknown%7CTWFpbGZsb3d8eyJWljiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6IkhhaWwiLCJXVCI6Mno%3D>

%7C3000%7C%7C%7C&sdata=IsfsERq5SoCV%2BwmPB1JiIaTf5qNjEu9w
ox1afors8A%3D&reserved=o

4. Guamán, V. (2022a, junio 24). *API con Asp.net MVC 6 y SQL Server mediante Entity Framework core 6 - Code First - Parte 1*. DEV Community.
<https://nam10.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdev.to%2Fveronicaguamann%2Fapi-con-aspnet-mvc-6-y-sql-server-mediante-entity-framework-core-6-code-first-parte-1-2io5&data=05%7C01%7Casanchezm%40javeriana.edu.co%7C8b4784444444d4867aaa308daacb7df55%7Cdaf7990e8a3f409c9b762a5475098000%7C0%7C0%7C638012202991803905%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ikp1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=IsfsERq5SoCV%2BwmPB1JiIaTf5qNjEu9wox1afors8A%3D&reserved=o>
5. *SQL Server 2019*. (s/f). Microsoft.com. Recuperado el 15 de abril de 2024, de <https://www.microsoft.com/es-es/sql-server/sql-server-2019>