“UNIVERSIDAD PRIVADA DE TACNA”

Facultad de Ingeniería

“Escuela Profesional de ingeniería de Sistemas”

Informe de Proyecto de Unidad-I

Creación de una Web API ASP .Net Core con Docker

Curso: “Base de Datos-II”

Docente: Ing. Patrick Cuadros Quiroga

Integrantes:

- Cano Sucso, Anthony Alexander
- Chambilla Zuñiga, Josue Abraham
- Jarro Cachi, Jose Luis
- Rivera Mendoza, Jhonny
- Valverde Zamora, Jean Pier Elias

TACNA-PERÚ 2023

Resumen

Este informe detalla el proceso integral de desarrollo de una API web utilizando tecnologías clave en el panorama de desarrollo moderno. Comenzando con la configuración del entorno de desarrollo y la creación de un proyecto ASP .NET Core, avanzamos hacia el diseño de modelos de datos, la implementación de controladores y rutas para gestionar solicitudes HTTP, y la configuración de relaciones en la base de datos utilizando Entity Framework.

La migración y creación de la base de datos SQL Server se abordan mediante migraciones de Entity Framework, seguidas de la implementación de lógica de negocio en los controladores. Luego, exploramos la dockerización de la aplicación, lo que permite empaquetarla en un contenedor Docker para un despliegue eficiente y portátil.

La conexión a SQL Server se configura para interactuar con la base de datos y realizar operaciones CRUD. Se realizan pruebas exhaustivas y se resuelven problemas mediante pruebas locales y depuración. Finalmente, se describe la implementación en un entorno de producción y la documentación del uso de la API.

Este enfoque completo proporciona a los desarrolladores una guía sólida para crear aplicaciones web modernas, escalables y portátiles, aprovechando ASP .NET Core, Docker y SQL Server con Entity Framework, lo que facilita el despliegue y la gestión eficiente de aplicaciones en diversos escenarios.

Abstract

This white paper details the end-to-end process of developing a web API using key technologies in the modern development landscape. Beginning with setting up the development environment and creating an ASP .NET Core project, we move on to designing data models, implementing controllers and routes to handle HTTP requests, and configuring database relationships using the Entity Framework.

SQL Server database migration and creation are addressed using Entity Framework migrations, followed by implementing business logic in controllers. Next, we explore the dockerization of the application, which allows it to be packaged in a Docker container for efficient and portable deployment.

The connection to SQL Server is configured to interact with the database and perform CRUD operations. Extensive testing is performed and issues are resolved through local testing and debugging. Finally, implementation in a production environment and documentation of API usage is described.

This comprehensive approach provides developers with a solid guide to building modern, scalable and portable web applications, leveraging ASP .NET Core, Docker and SQL Server with Entity Framework, which facilitates the efficient deployment and management of applications in a variety of scenarios.

Índice

1) Antecedentes o introducción:

En el contexto actual de las tecnologías de la información, el desarrollo de aplicaciones web ha alcanzado un papel central en el funcionamiento y crecimiento de las organizaciones. La creación de interfaces de programación de aplicaciones (API) se ha convertido en una parte integral de la exposición de funcionalidades y datos en la web, y ASP .NET Core se ha convertido en una plataforma de desarrollo poderosa y confiable para este propósito.

Por otro lado, la tecnología de contenedores, liderada por Docker, ha revolucionado el empaquetado y despliegue de aplicaciones, proporcionando portabilidad y eficiencia tanto en entornos de desarrollo como de producción. Además, la persistencia de los datos es un requisito crítico para muchas aplicaciones y Microsoft SQL Server es una solución ampliamente utilizada para la gestión de bases de datos relacionales.

En este contexto, surge la necesidad de comprender y gestionar la creación de una API ASP .NET Core que utilice Docker para empaquetado y distribución, así como la capacidad de comunicarse con una base de datos SQL Server utilizando Entity Framework. ORM (mapeo relacional de objetos) ampliamente utilizado.

Esta guía cubre estas importantes tecnologías y proporciona una guía paso a paso para crear una API funcional con todas estas características. Esta es una guía esencial para quienes quieran profundizar en el desarrollo de aplicaciones web modernas y potentes.

2) Titulo: “Creacion de web API para la gestión de clientes”

3) Autores:

- Cano Sucso, Anthony Alexander
- Chambilla Zuñiga, Josue Abraham
- Jarro Cachi, Jose Luis
- Rivera Mendoza, Jhonny
- Valverde Zamora, Jean Pier Elias

4) Planteamiento del problema:

a) Problema:

En un entorno tecnológico en constante evolución, las organizaciones se enfrentan al desafío de desarrollar aplicaciones web eficientes y escalables que puedan satisfacer las demandas cambiantes de los usuarios y clientes. La creación de APIs es un componente crítico en este proceso, ya que permite la exposición de servicios y datos a través de la web, facilitando la interacción con aplicaciones móviles, clientes web y otros sistemas.

Sin embargo, el proceso de desarrollo de una API puede ser complejo, con desafíos que incluyen la elección de la tecnología adecuada, la gestión de datos, la escalabilidad y el despliegue eficiente en entornos variados. Además, la gestión de bases de datos relacionales, como Microsoft SQL Server, puede ser un obstáculo adicional en el camino hacia la implementación exitosa de una API.

b) Justificación:

Para abordar este problema radica en la necesidad de proporcionar a los desarrolladores y profesionales de TI una guía completa y práctica para la creación de APIs ASP .NET Core que aborden los desafíos mencionados anteriormente. ASP .NET Core es una plataforma sólida y versátil para el desarrollo de aplicaciones web, mientras que Docker ofrece una solución eficaz para el empaquetamiento y despliegue de aplicaciones. Además, Microsoft SQL Server es una opción popular para la gestión de bases de datos, y Entity Framework simplifica la interacción con bases de datos relacionales.

La justificación de este informe radica en la necesidad de proporcionar una guía paso a paso que permita a los desarrolladores dominar estas tecnologías esenciales y crear APIs que sean eficientes, escalables y que puedan interactuar de manera efectiva con bases de datos SQL Server.

c) Alcance:

Abarca la configuración inicial del entorno de desarrollo, el diseño de modelos de datos y lógica de negocio, la administración de bases de datos SQL Server con Entity Framework, la dockerización de la aplicación para facilitar el despliegue y la portabilidad, así como técnicas de pruebas y depuración. También se incluyen directrices para documentar la API y consideraciones para su despliegue en entornos de producción. Este informe proporcionará a los desarrolladores una base sólida para comprender y aplicar estas tecnologías esenciales en sus proyectos, con un enfoque en la adquisición de habilidades prácticas y conocimientos que les permitirán crear aplicaciones web modernas y eficientes.

5) Objetivos:

a) General:

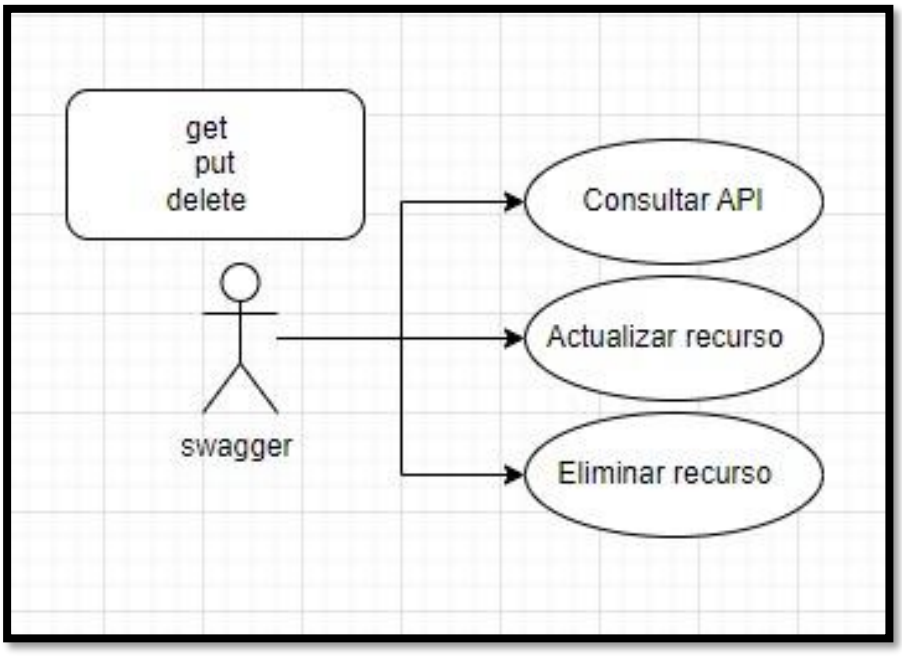
- Creación de una API web utilizando ASP .NET Core, la implementación de contenedores Docker y la integración con Microsoft SQL Server a través de Entity Framework.

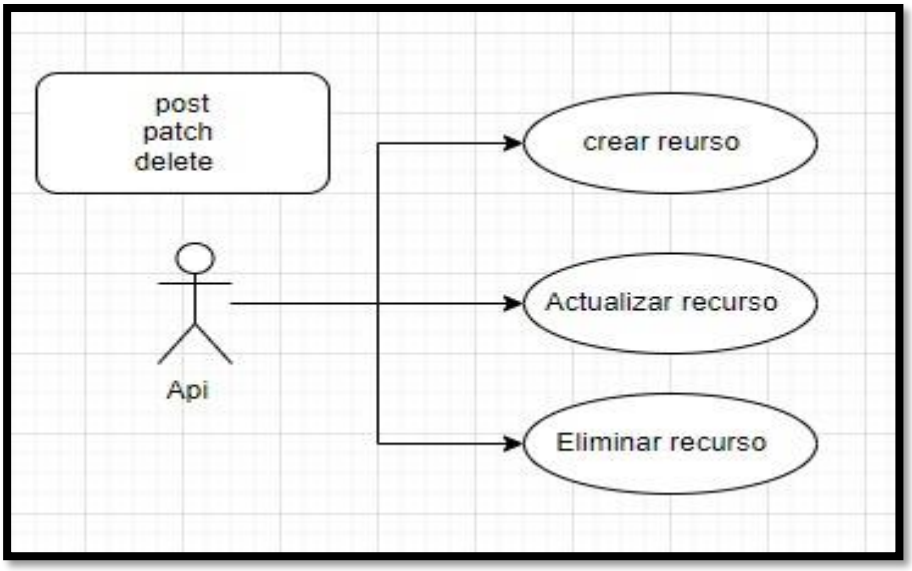
b) Específicos:

- Creación de una API web utilizando ASP .NET Core, la implementación de contenedores Docker y la integración con Microsoft SQL Server a través de Entity Framework
- Dockerización de la Aplicación
- Realizar pruebas y depuración
- Despliegue en un entorno de producción

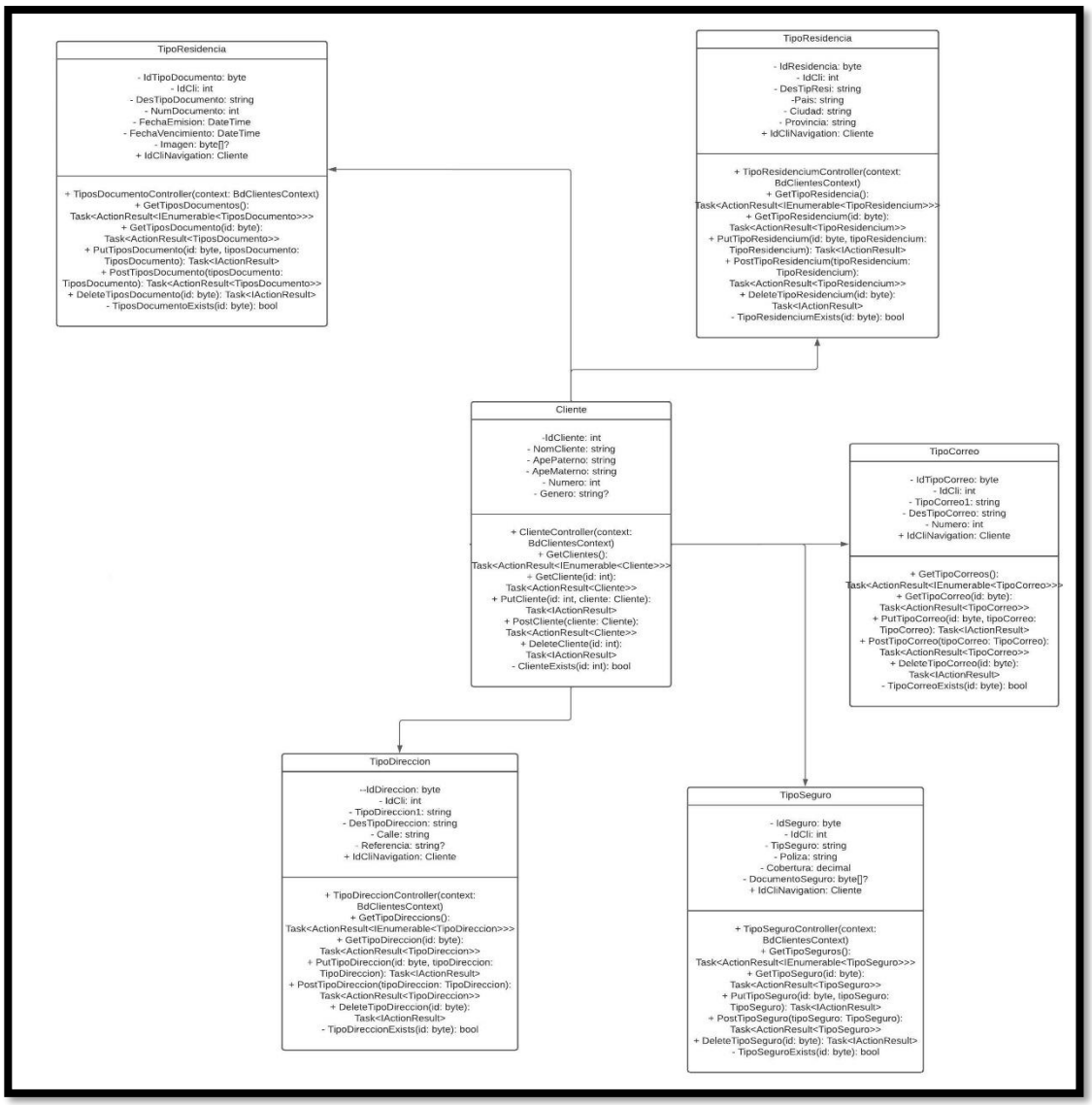
6) Desarrollo de la propuesta :

a) Diagramas de Casos de Uso:

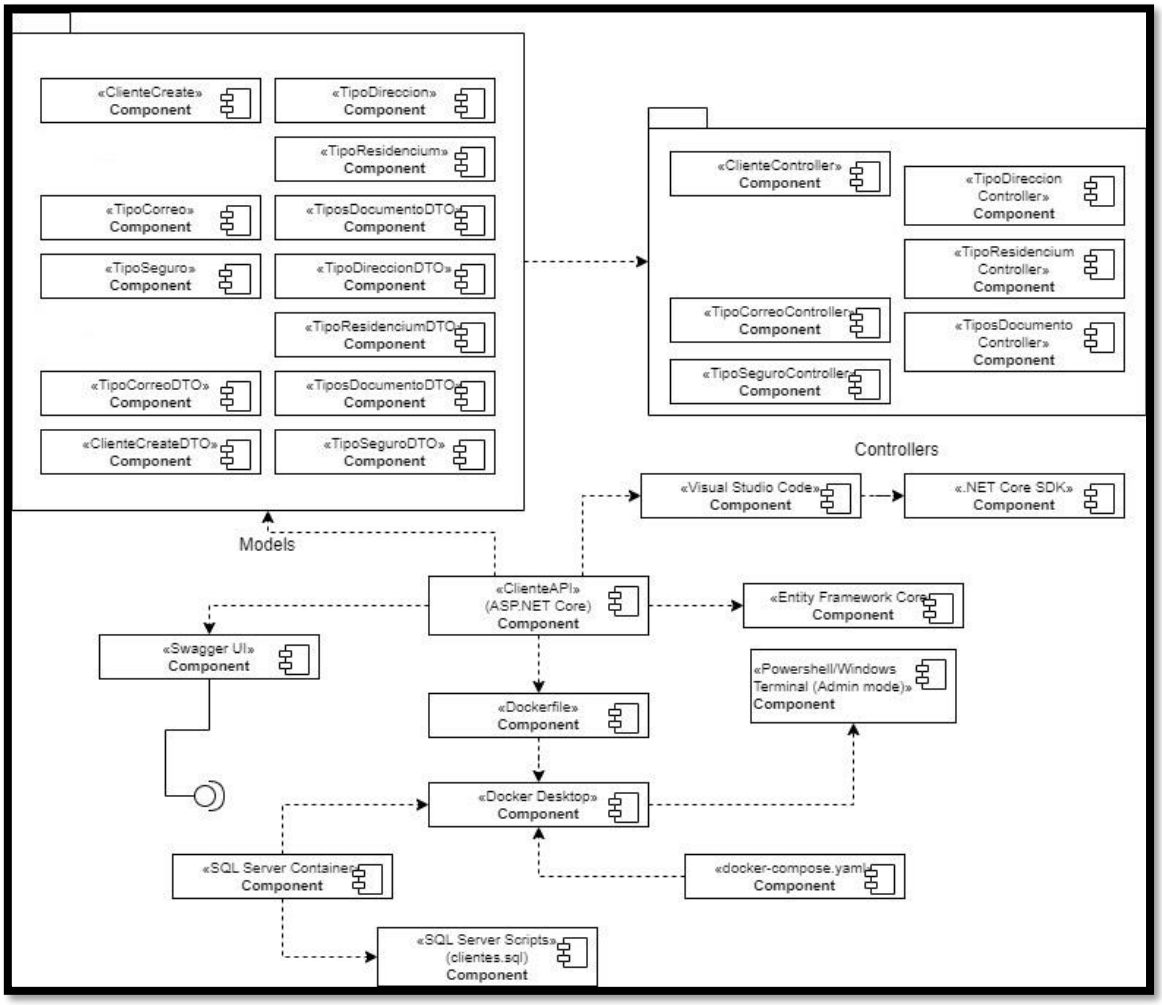




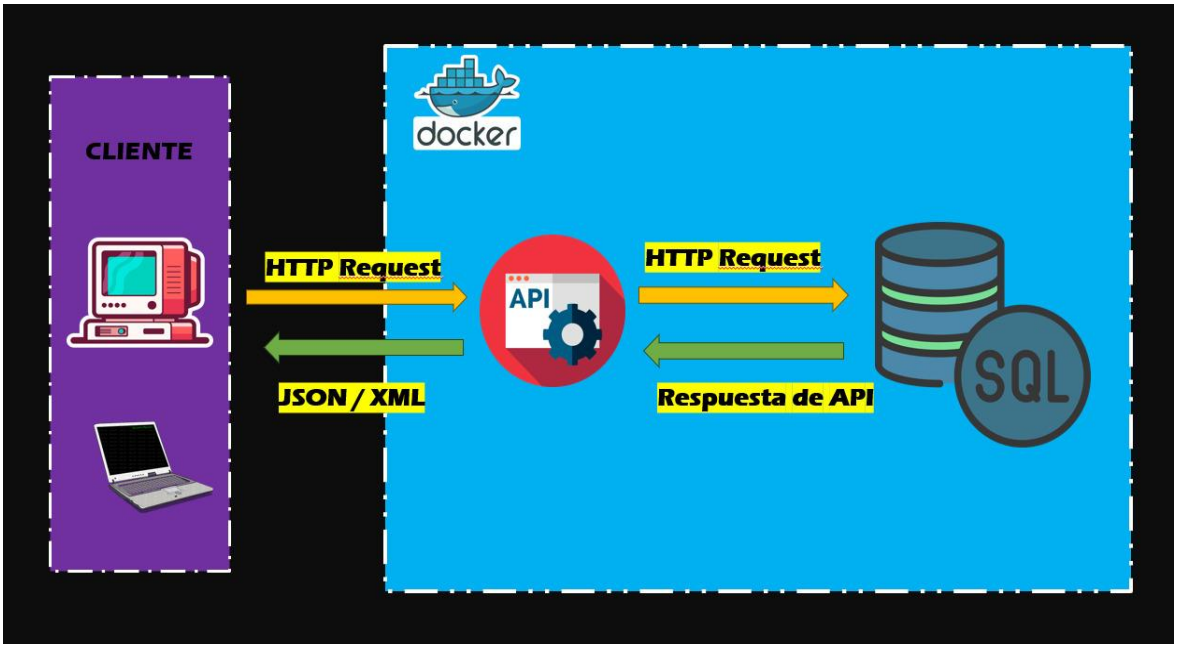
b) Diagrama de Clases:



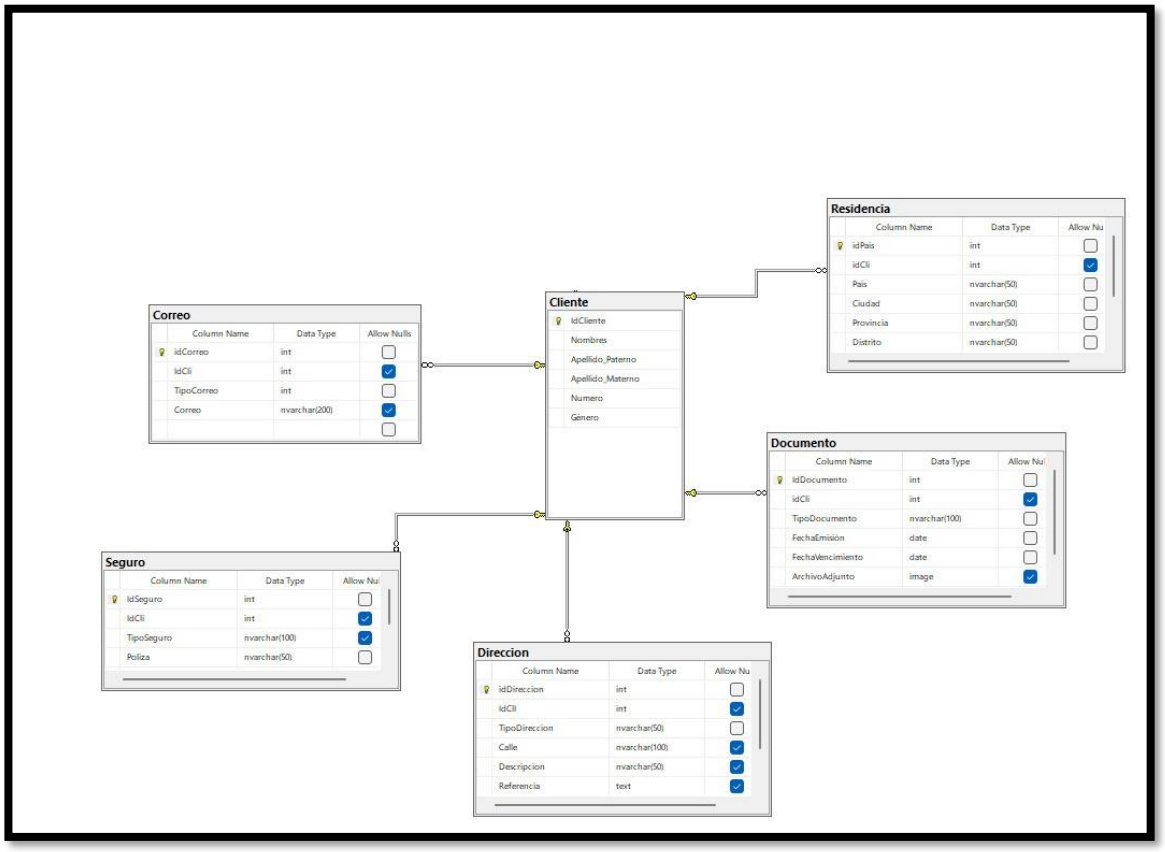
c) Diagrama de Componentes:



d) Arquitectura:



e) Diagrama de Base de Datos:



7) Bibliografía:

Hadsonpar. (2020, February). *ASP.NET Core - Crear Web API con C# y EF*. Blog | Hadsonpar; Blogger. http://blog.hadsonpar.com/2020/02/aspnet-core-crear-web-api-con-c-y-ef.html

IEvangelist. (2023, March 18). *Tutorial sobre cómo incluir una aplicación en un contenedor con Docker - .NET*. Microsoft.com. https://learn.microsoft.com/es-es/dotnet/core/docker/build-container?tabs=windows

Ruiz, R. (2019, September 25). Crear una aplicación de ASP.NET Core y contenerizarla con Docker | Medium. Medium; Medium. https://rruizdev.medium.com/how-to-create-a-asp-net-core-application-and-containerize-with-docker-49cf4c017e48

8) Anexos:

a) Diccionario de datos de su base de datos relacional

Tabla CLIENTES:

ID_CLIENTE (INT): Identificador único del cliente (clave primaria).
NOM_CLIENTE (VARCHAR(100)): Nombre del cliente.
APE_PATERNO (VARCHAR(100)): Apellido paterno del cliente.
APE_MATERNO (VARCHAR(100)): Apellido materno del cliente.
NUMERO (INT): Número de cliente.
GENERO (VARCHAR(20)): Género del cliente.

Tabla TIPOS_DOCUMENTOS:

ID_TIPO_DOCUMENTO (TINYINT): Identificador único del tipo de documento (clave primaria).
ID_CLI (INT): Identificador del cliente al que pertenece el tipo de documento (clave foránea).
DES_TIPO_DOCUMENTO (VARCHAR(50)): Descripción del tipo de documento.
NUM_DOCUMENTO (INT): Número del documento.
FECHA_EMISION (DATE): Fecha de emisión del documento.
FECHA_VENCIMIENTO (DATE): Fecha de vencimiento del documento.
IMAGEN (IMAGE): Imagen del documento (puede ser nulo).

Tabla TIPO_CORREO:

ID_TIPO_CORREO (TINYINT): Identificador único del tipo de correo (clave primaria).
ID_CLI (INT): Identificador del cliente al que pertenece el tipo de correo (clave foránea).
TIPO_CORREO (VARCHAR(40)): Tipo de correo.
DES_TIPO_CORREO (NVARCHAR(200)): Descripción del tipo de correo.

Tabla TIPO_DIRECCION:

ID_DIRECCION (TINYINT): Identificador único del tipo de dirección (clave primaria).
ID_CLI (INT): Identificador del cliente al que pertenece el tipo de dirección (clave foránea).
TIPO_DIRECCION (VARCHAR(40)): Tipo de dirección.
DES_TIPO_DIRECCION (NVARCHAR(200)): Descripción del tipo de dirección.

CALLE (NVARCHAR(50)): Calle de la dirección.

REFERENCIA (TEXT): Referencia de la dirección (puede ser nulo).

Tabla TIPO_SEGURO:

ID_SEGURO (TINYINT): Identificador único del tipo de seguro (clave primaria).

ID_CLI (INT): Identificador del cliente al que pertenece el tipo de seguro (clave foránea).

TIP_SEGURO (NVARCHAR(40)): Tipo de seguro.

POLIZA (NVARCHAR(30)): Número de póliza del seguro.

COBERTURA (DECIMAL(18, 2)): Cobertura del seguro (valor decimal).

DOCUMENTO_SEGURO (IMAGE): Documento del seguro (puede ser nulo).

Tabla TIPO_RESIDENCIA:

ID_RESIDENCIA (TINYINT): Identificador único del tipo de residencia (clave primaria).

ID_CLI (INT): Identificador del cliente al que pertenece el tipo de residencia (clave foránea).

DES_TIP_RESI (NVARCHAR(30)): Descripción del tipo de residencia.

PAIS (NVARCHAR(50)): País de residencia.

CIUDAD (NVARCHAR(50)): Ciudad de residencia.

PROVINCIA (NVARCHAR(50)): Provincia de residencia.

Tabla CLIENTE_DETALLE:

ID_CLI_DET (TINYINT): Identificador único del detalle del cliente (clave primaria).

ID_CLI (INT): Identificador del cliente al que pertenece el detalle (clave foránea).

CORREO (NVARCHAR(200)): Correo electrónico del cliente.

SEGURO (VARCHAR(40)): Tipo de seguro del cliente.

DIRECCION (NVARCHAR(200)): Dirección del cliente.

DOCUMENTO (INT): Número de documento del cliente.

RESIDENCIA (NVARCHAR(200)): Residencia del cliente.