

Documentación del Proyecto OLAP - Transporte

1. Selección del Negocio

Negocio Elegido: Transporte

2. Diseño de la Base de Datos Transaccional

Se han creado las siguientes tablas para almacenar los datos del negocio de transporte. Los scripts utilizados para la creación de las tablas y otros pasos relacionados se encuentran en el repositorio de GitHub.

Tablas Creadas:

1. Clientes:

- Campos: ClienteID, Nombre, Direccion, Telefono, Email.
- Llave primaria: ClienteID.

2. Vehículos:

- Campos: VehiculoID, Tipo, Matricula, Capacidad, Estado.
- Llave primaria: VehiculoID.

3. Rutas:

- Campos: RutaID, Origen, Destino, DistanciaKM.
- Llave primaria: RutaID.

4. Servicios de Transporte:

- Campos: ServicioID, ClienteID, VehiculoID, RutaID, FechaServicio, Costo, Observaciones.
- Llaves Foráneas: ClienteID, VehiculoID, RutaID.

Script utilizado:

- **Archivo:** [SQLQuery1CreacionTablas.sql](#)
- **Descripción:** Este script crea las tablas Clientes, Vehiculos, Rutas y ServiciosTransporte en la base de datos **TranspoDB**.
- **Ubicación:** Repositorio GitHub

3. Comprobación de la Creación de Tablas

Se ejecutó un script para verificar que todas las tablas se crearon correctamente. Se utilizó una consulta que lista todas las tablas en la base de datos **TranspoDB**.

Script utilizado:

- **Archivo:** [SQLQuery2ComprobaciónTablas.sql](#)
- **Descripción:** Este script verifica la existencia de las tablas creadas en la base de datos **TranspoDB**.
- **Ubicación:** Repositorio GitHub

4. Inserción de Datos de Prueba

Se insertaron datos de prueba en las tablas de **Clientes**, **Vehículos**, **Rutas** y **ServiciosTransporte** para garantizar que las tablas y sus relaciones funcionan correctamente.

Script utilizado:

- **Archivo:** [SQLQuery3InsertarDatos.sql](#)
- **Descripción:** Este script inserta datos de prueba en las tablas **Cientes**, **Vehiculos**, **Rutas** y **ServiciosTransporte**.
- **Ubicación:** Repositorio GitHub

5. Configuración de Perfiles de Usuario

Se configuraron tres perfiles de usuario con diferentes roles:

5. Administrador:

- Tiene acceso completo a la base de datos.
- Usuario: admin_user.

6. Gerente:

- Puede consultar la base de datos, pero no modificarla.
- Usuario: gerente_user.

7. Analista de Datos:

- Puede consultar y generar reportes con permisos limitados de escritura.
- Usuario: analista_user.

Script utilizado:

- **Archivo:** [SQLQuery4Usuarios.sql](#)

- **Descripción:** Este script crea los usuarios `admin_user`, `gerente_user` y `analista_user` con los permisos correspondientes en la base de datos **TranspoDB**.
- **Ubicación:** Repositorio GitHub

6. Configuración de Respaldos

Se configuró un script de respaldo y se programó su ejecución automática mediante el **Programador de Tareas de Windows**. El respaldo se guarda en la ruta `C:\Respaldos\TranspoDB.bak`.

Script utilizado:

- **Archivo:** [SQLQuery5ScriptParaRespaldo.sql](#)
- **Descripción:** Este script realiza un respaldo completo de la base de datos **TranspoDB**.
- **Ubicación:** Repositorio GitHub

7. Configuración de TCP/IP y Prueba de Conexión

Para permitir el acceso remoto a la base de datos **TranspoDB** en SQL Server, se configuró el protocolo **TCP/IP** y se realizó una prueba de conexión desde otro dispositivo utilizando **Telnet** en el puerto **1810**.

Configuración de TCP/IP

1. Habilitar TCP/IP:

- Se accedió a **SQL Server Configuration Manager** y en la sección **SQL Server Network Configuration**, se habilitó el protocolo **TCP/IP** para la instancia de SQL Server.

2. Configuración del Puerto:

- En las propiedades del protocolo **TCP/IP**, bajo la pestaña **IP Addresses**, se configuró el puerto **1810** en el campo **IPAll**.

3. Reinicio del Servicio de SQL Server:

- Después de aplicar los cambios en el protocolo TCP/IP, se reinició el servicio de SQL Server desde el **SQL Server Configuration Manager** para que los cambios tomaran efecto.

4. Configuración del Firewall:

- Se creó una nueva regla de entrada en el **Firewall de Windows** para permitir las conexiones TCP entrantes en el puerto **1810**.
- La regla se aplicó para los perfiles de red **Dominio, Privado y Público**.

Prueba de Conexión desde otro Dispositivo

1. Dispositivo Remoto:

- Se utilizó un segundo ordenador conectado a la misma red para realizar la prueba de conexión. Se utilizó Telnet.

2. Prueba con Telnet:

- En el dispositivo remoto, se habilitó el cliente Telnet desde el Panel de Control.
- En el Símbolo del Sistema del dispositivo remoto, se ejecutó el siguiente comando para probar la conectividad al servidor SQL en el puerto 1810:
 - telnet (192.168...) 1810

3. Conclusión de la Prueba:

- La pantalla de Telnet se mostró en blanco, indicando que el puerto estaba accesible desde el dispositivo remoto.

8. Cubo OLAP

El cubo OLAP fue creado para el análisis de los datos de la base de datos TranspoBD en el servidor EMILIO. Este cubo permite realizar consultas y análisis avanzados sobre las operaciones del negocio de transporte, utilizando medidas como la cantidad de rutas, capacidad de vehículos, y costo de los servicios de transporte. A continuación se describe el proceso paso a paso.

1. Entorno de trabajo

- **Servidor:** EMILIO
- **Base de Datos:** TranspoBD
- **Usuario Utilizado:** sa
- **Herramienta:** SQL Server Data Tools (SSDT) en Visual Studio 2022
- **Base de Datos OLAP:** Analysis Services en SQL Server

2. Proceso de Creación del Cubo OLAP

- **Instalación de SQL Server Analysis Services (SSAS):** El servidor **EMILIO** tiene instalada la instancia de SQL Server Analysis Services en modo multidimensional, lo que permite el uso de cubos OLAP.
- **Conexión a la Base de Datos:** Se utilizó la base de datos **TranspoBD**, la cual contiene las tablas necesarias para almacenar la información de

clientes, rutas, vehículos, y servicios de transporte. La conexión se realizó utilizando el usuario **sa** con los privilegios correspondientes.

3. Creación del proyecto en Visual Studio 2022

1. Creación del Data Source (Fuente de Datos):
 - En Visual Studio, se añadió una nueva fuente de datos que conectaba al servidor EMILIO y a la base de datos TranspoBD utilizando las credenciales del usuario sa.
2. Creación de la Vista de Origen de Datos (Data Source View):
 - Se crearon vistas de origen de datos que incluían todas las tablas relevantes de la base de datos TranspoBD, como Clientes, Rutas, Vehículos, y Servicios Transporte.
3. Diseño del cubo OLAP
 - Se utilizó el asistente de Visual Studio para crear el cubo. En este paso, se seleccionaron las siguientes medidas clave:
 - Clientes: Recuento de clientes.
 - Rutas: Distancia en kilómetros y recuento de rutas.
 - Servicios Transporte: Costo y recuento de servicios de transporte.
 - Vehículos: Capacidad de vehículos y recuento de vehículos.
4. Definición de las dimensiones
 - Se crearon las dimensiones del cubo basadas en las tablas de la base de datos, las cuales incluían:
 - Clientes
 - Rutas
 - Vehículos
 - Servicios Transporte
5. Procesamiento del cubo
 - Una vez configuradas las medidas y dimensiones, el cubo fue procesado con éxito en el servidor EMILIO. El procesamiento permitió cargar todos los datos de la base de datos transaccional TranspoBD en el cubo OLAP para su posterior análisis.
 - Las medidas y dimensiones fueron verificadas para asegurar que los datos estuvieran representados correctamente en el cubo.
6. Roles y seguridad
 - En el cubo, se configuraron roles para restringir el acceso basado en permisos. Los roles definidos incluyen:
 - Administrador: Acceso total al cubo.

- Gerente: Capacidad de consultar reportes y generar gráficos.
- Analista de Datos: Capacidad de crear nuevos reportes y análisis.

7. Asignación de Roles:

- Los roles se asignaron a los usuarios correspondientes, definiendo los niveles de acceso en función de sus responsabilidades dentro de la organización.

Cap2. Documentación Formal: Creación del Proyecto PortalWebASP_Proyecto

1. Introducción

El proyecto **PortalWebASP_Proyecto** está basado en ASP.NET Core MVC y tiene como objetivo el desarrollo de un portal web que se conecte a una base de datos SQL Server. La aplicación utiliza **Entity Framework Core** para la comunicación con la base de datos, y este documento detallará los pasos realizados hasta ahora en el proceso de creación y configuración del proyecto.

2. Configuración Inicial del Proyecto

Paso 1: Creación del Proyecto en Visual Studio

- Abrimos Visual Studio y seleccionamos la plantilla **Aplicación Web de ASP.NET Core (Modelo-Vista-Controlador)**.
- En la ventana de configuración del proyecto, se eligieron las siguientes opciones:
 - **Nombre del proyecto:** PortalWebASP_Proyecto.
 - **Ubicación:** Se especificó la ubicación en la que se guardará el proyecto en el equipo.

- **Framework de .NET:** Se seleccionó .NET 6.0 como versión de framework.
- **Autenticación:** Se eligió "Sin autenticación" (esto puede configurarse más adelante).
- **HTTPS:** Se activó la opción "Configurar para HTTPS".

Con estos parámetros configurados, se hizo clic en **Crear** para generar la estructura inicial del proyecto.

3. Configuración de Entity Framework Core

Paso1: Instalación de Entity Framework Core

Para poder interactuar con la base de datos **SQL Server**, instalamos **Entity Framework Core**. Utilizamos la **Consola del Administrador de Paquetes NuGet** para ejecutar los siguientes comandos:

- `Install-Package Microsoft.EntityFrameworkCore.SqlServer`
- `Install-Package Microsoft.EntityFrameworkCore.Tools`

Esto instaló los paquetes necesarios para utilizar **Entity Framework Core** en nuestro proyecto

4. Creación del Modelo Ruta

Paso 3: Definición del Modelo

Se creó un modelo llamado **Ruta** que representa una entidad en la base de datos. El modelo fue definido dentro de la carpeta **Models** en el archivo `Ruta.cs`. La estructura de la clase es la siguiente:

```
using System.Collections.Generic;

namespace PortalWebASP_Proyecto.Models
{
    1 referencia
    public class Ruta
    {
        0 referencias
        public int RutaId { get; set; }
        0 referencias
        public string Origen { get; set; }
        0 referencias
        public string Destino { get; set; }
        0 referencias
        public DateTime FechaCreacion { get; set; }
    }
}
```

El modelo Ruta tiene los siguientes campos:

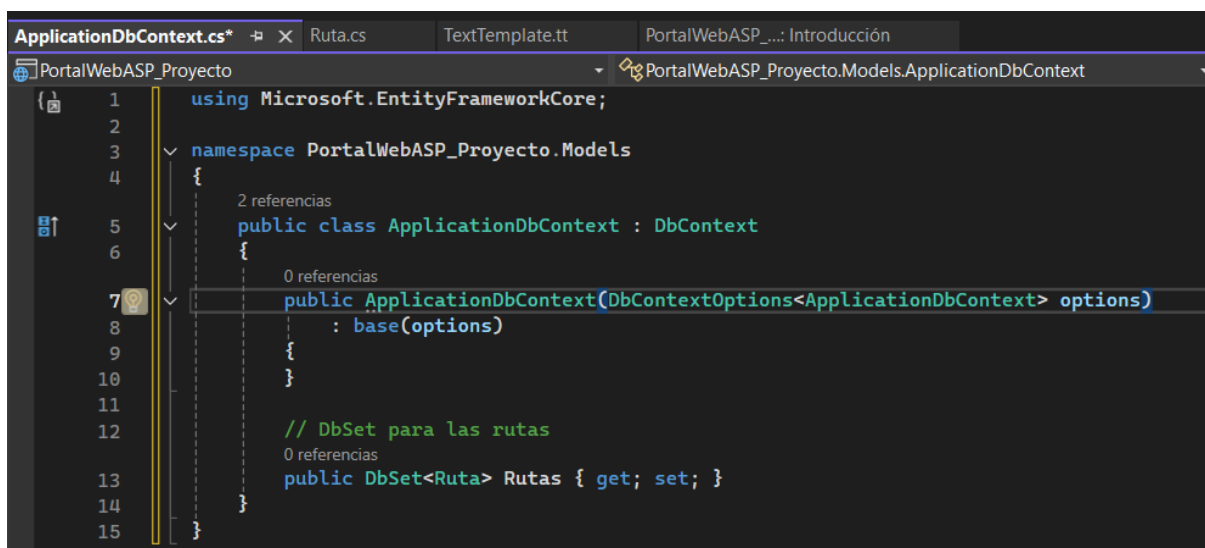
- **RutaId:** Identificador único de la ruta.

- **Origen:** El lugar de origen de la ruta.
- **Destino:** El lugar de destino de la ruta.
- **FechaCreacion:** Fecha en que la ruta fue creada.

5. Configuración del Contexto de Base de Datos

Paso 1: Creación del ApplicationDbContext

Para que **Entity Framework Core** pueda gestionar el acceso a la base de datos, se creó la clase **ApplicationDbContext.cs** en la carpeta **Models**. La clase fue configurada de la siguiente manera:



```

1  using Microsoft.EntityFrameworkCore;
2
3  namespace PortalWebASP_Proyecto.Models
4  {
5      public class ApplicationDbContext : DbContext
6      {
7          public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
8              : base(options)
9          {
10          }
11
12          // DbSet para las rutas
13          public DbSet<Ruta> Rutas { get; set; }
14      }
15  
```

Esta clase define el **DbSet** para el modelo **Ruta**, lo que permite que Entity Framework lo trate como una tabla en la base de datos.

6. Creación y Aplicación de Migraciones

Paso 5: Creación de la Migración

Se generó una migración para crear la tabla **Rutas** en la base de datos SQL Server. Para esto, se utilizó la consola de NuGet con el siguiente comando:

- Add-Migration CrearTablaRutas

Se ejecutó en

