

Documentación del Proyecto

1. Arquitectura de la Solución

La solución está basada en una arquitectura modular y extensible que implementa buenas prácticas de diseño y desarrollo:

- **Patrón CQRS:** Utiliza **Command Query Responsibility Segregation** para separar las operaciones de escritura (comandos) y lectura (consultas). Esto mejora la organización del código y la escalabilidad.
 - **API RESTful:** Implementada con **ASP.NET Core 6** para manejar todas las operaciones.
 - **Middleware Personalizado:**
 - Manejador global de excepciones.
 - Endpoint de HealthCheck para verificar el estado de salud del sistema y la conectividad con la base de datos.
 - **Base de Datos SQL Server:** Con procedimientos almacenados para manejar operaciones críticas como la inserción de transacciones y el cálculo del estado de cuenta.
 - **Dependencias:**
 - **MediatR:** Para implementar CQRS.
 - **FluentValidation:** Para validar los datos de entrada en los comandos y DTOs.
 - **AutoMapper:** Para mapear entidades de la base de datos a DTOs utilizados por la API.
-

2. Endpoints de la API

A continuación, se describen los principales endpoints disponibles en la API:

2.1 Obtener Estado de Cuenta

- **Método:** GET
- **URL:** /api/CreditCard/{cardHolderId}/statement

- **Descripción:** Devuelve el estado de cuenta del titular de la tarjeta, incluyendo cálculos como el saldo disponible, interés bonificable y pagos mínimos.
- **Respuesta de Ejemplo:**

```
{
  "cardHolderName": "Jose Bonilla",
  "cardNumber": "9643929065058129",
  "creditLimit": 1000.00,
  "currentBalance": 35.00,
  "availableBalance": 965.00,
  "totalPurchasesThisMonth": 160.00,
  "totalPurchasesLastMonth": 0,
  "interestAmount": 8.7500,
  "minimumPayment": 1.7500,
  "totalWithInterest": 43.7500
}
```

2.2 Agregar Transacción

- **Método:** POST
- **URL:** /api/CreditCard/transactions
- **Descripción:** Agrega una nueva transacción (compra o pago) al historial de un titular.
- **Cuerpo de Solicitud:**

```
{
  "cardHolderID": 1,
  "transactionDate": "2024-11-16",
  "description": "Compra Comida",
  "transactionType": "Compra",
  "amount": 5
}
```

Respuesta de Ejemplo:

```
{
  "message": "Transaction added successfully."
}
```

2.3 Obtener Historial de Transacciones

- **Método:** GET
- **URL:** /api/CreditCard/{cardHolderId}/transactions

- **Descripción:** Devuelve el historial de transacciones realizadas por el titular de la tarjeta, incluyendo compras y pagos.
- **Respuesta de Ejemplo:**

```
[
  {
    "TransactionID": 1,
    "CardHolderID": 1,
    "TransactionDate": "2024-11-15",
    "Description": "Compra en tienda",
    "TransactionType": "Compra",
    "Amount": 150.0
  },
  {
    "TransactionID": 2,
    "CardHolderID": 1,
    "TransactionDate": "2024-11-14",
    "Description": "Pago mensual",
    "TransactionType": "Pago",
    "Amount": 50.0
  }
]
```

2.4 HealthCheck

- **Método:** GET
- **URL:** /health
- **Descripción:** Verifica el estado de salud del sistema y la conectividad con la base de datos.
- **Respuesta de Ejemplo**

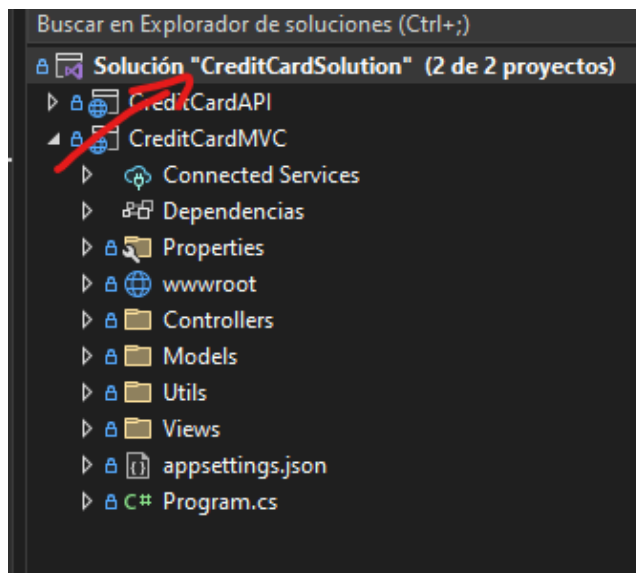
```
{
  "status": "Healthy",
  "results": [
    {
      "component": "SQL Server",
      "status": "Healthy",
      "description": null
    }
  ]
}
```

- Cuando clones el repositorio ahí vendrá la colección de Postman para probar la API
-

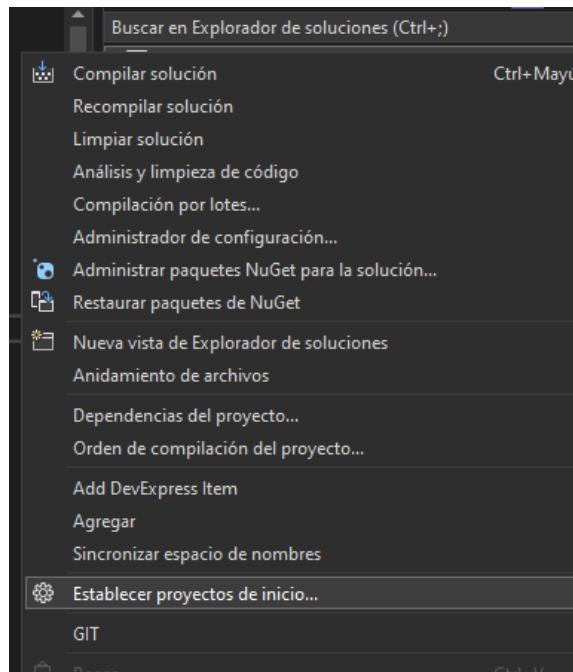
3. Cómo Probar la Aplicación

3.1 Requisitos Previos

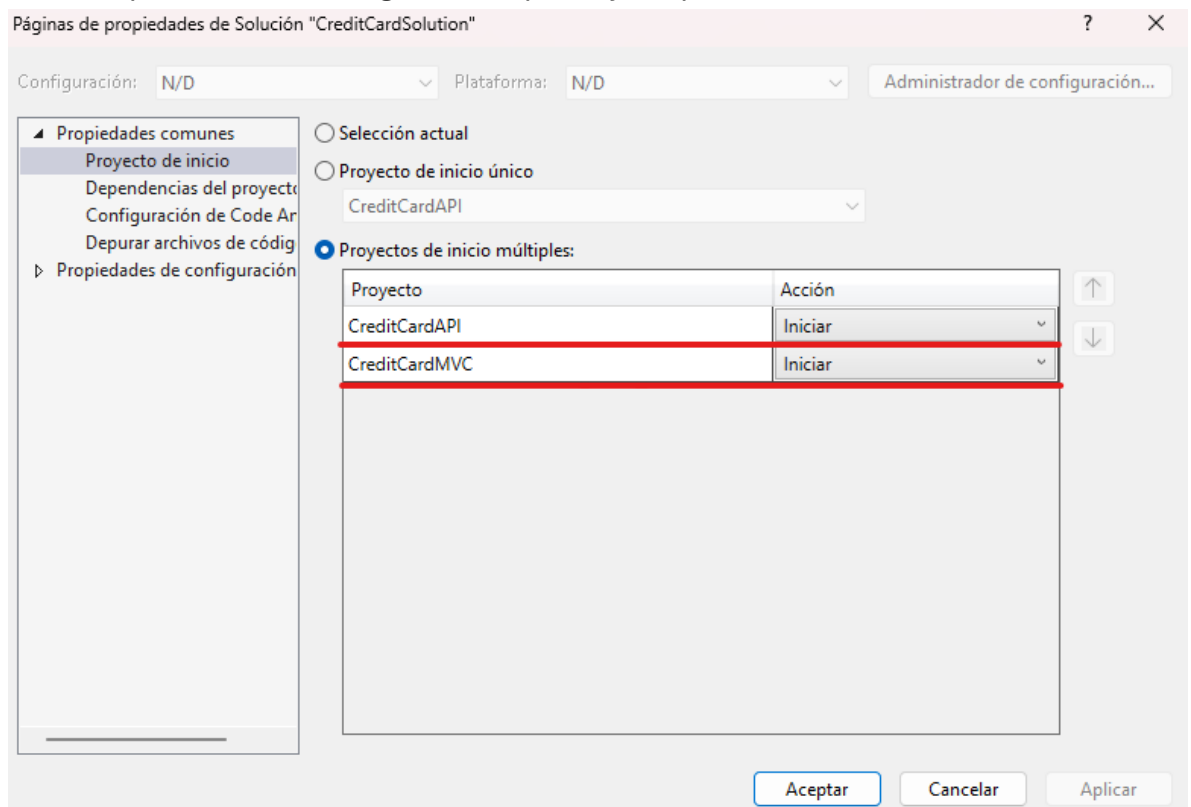
- Tener SQL Server instalado y configurado con la base de datos y procedimientos almacenados.
- Desde Visual Studio clonar el repositorio siguiente:
<https://github.com/Jr6445/Credit-Card-Management-System.git>
- Abrir Sql Server y ejecutar los queries que vienen en el archivo:
scriptDatabase.sql
- Una vez clonado el repositorio dar click derecho en la solución de visual studio



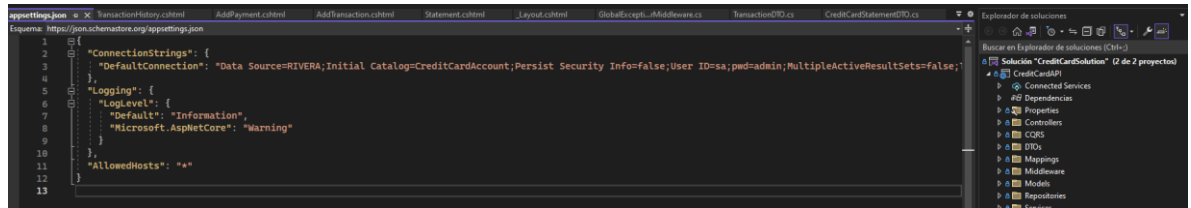
- Click en establecer proyectos de inicio:



- Verifica que este esta configuracion Aplicar y aceptar



- Ahora ve al proyecto CreditCard API y abre el appsettings.json y modifica la cadena de conexión



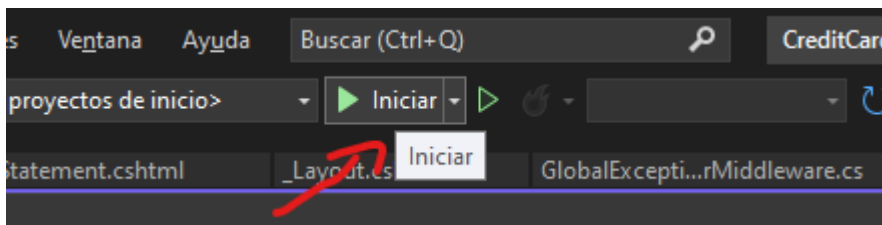
```

1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Data Source=RIVERA;Initial Catalog=CreditCardAccount;Persist Security Info=false;User ID=sa;pwd=admin;MultipleActiveResultSets=false;"
4   },
5   "Logging": {
6     "LogLevel": {
7       "default": "Information",
8       "Microsoft.AspNetCore": "Warning"
9     }
10  },
11  "AllowedHosts": "*"
12 }
13

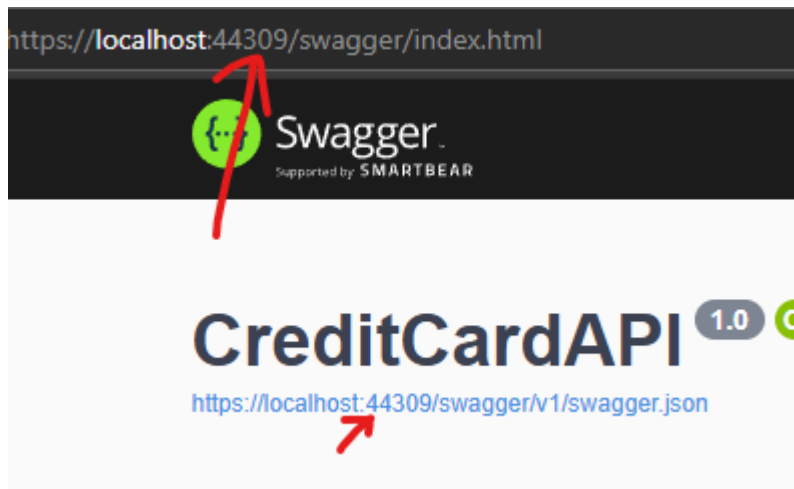
```

"ConnectionStrings": { "DefaultConnection": "Data Source=TU_SERVIDOR;Initial Catalog=CreditCardAccount; Persist Security Info=false;User ID=TU_USUARIO; pwd =TU_PASSWORD; MultipleActiveResultSets=false;TrustServerCertificate=true" }

- Ejecuta la solución y listo



- Puede que en tu pc cambie el puerto de la api, si llega a fallar el proyecto verifica eso, ejecuta la solución y ve al swagger de la api y veremos que puerto usa



- Si tu api usa otro puerto diferente a este debes ir al proyecto CreditCardMVC y abrir el archivo appsettings.json y cambiar el puerto de la ApiBaseUrl, Vuelves a compilar y ya debería funcionar.

```
os://json.schemastore.org/appsettings.json
{
  "ApiBaseUrl": "https://localhost:44309/api",
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

