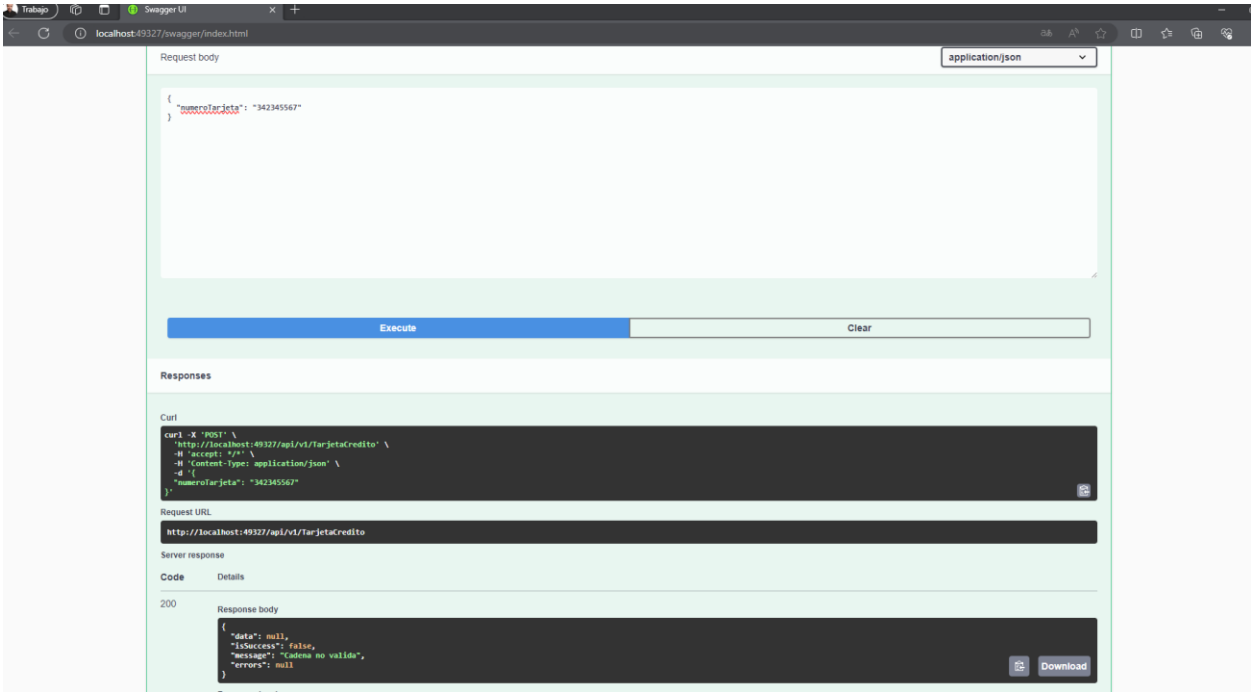
Validando una cadena que no corresponde a una tarjeta de crédito.



La siguiente pantalla muestra.

Número de tarjeta.

Número de tarjeta enmascarado.

Número de tarjeta en sha256hash.

Número de tarjeta encriptado.

Numero de tarjeta desencriptado que sería el número de tarjeta original.

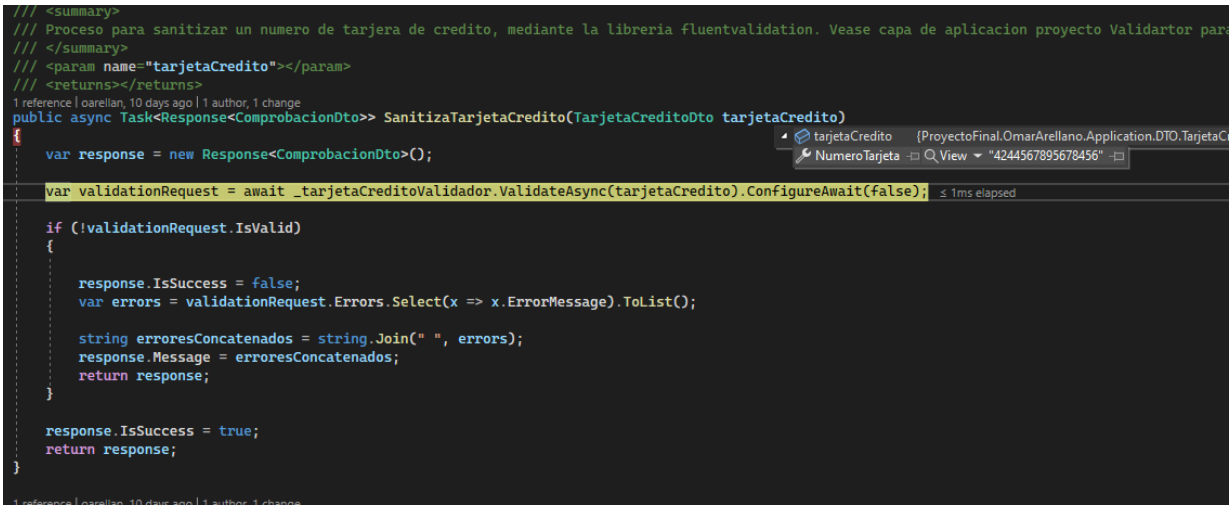
Una bandera que confirma que la cadena descifrada siga resultando en el mismo hash.

Response body

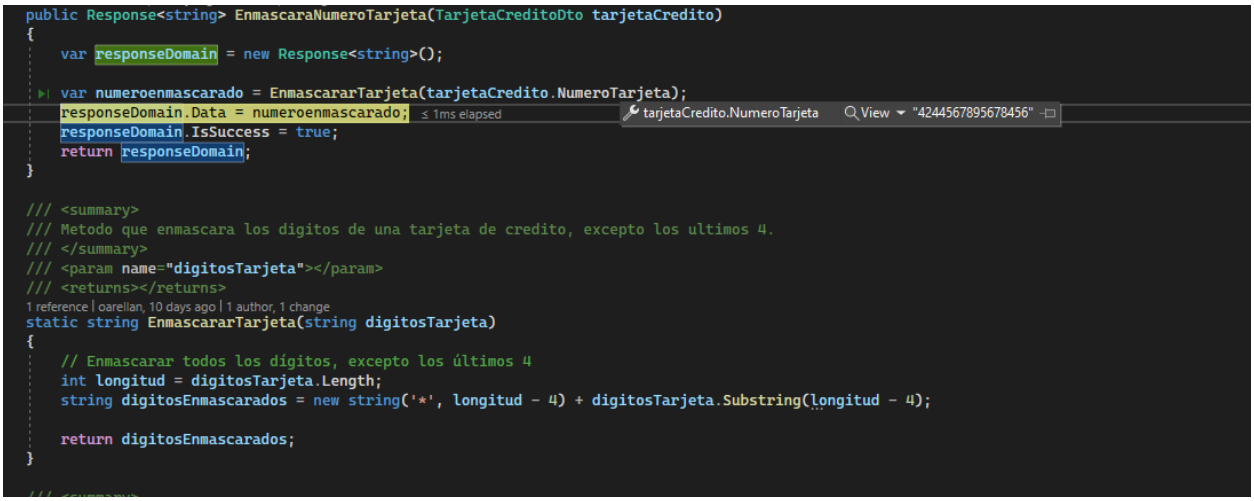

```
{
  "data": {
    "numeroTarjeta": "4244567895678456",
    "numeroTarjetaEnmascarada": "************8456",
    "sha256Hash": "d8666f09483334547ad4f05321ca768f644ec82a3f76084a9728fa2876336b34",
    "numeroTarjetaEncriptado": "MHFG8+knYGHHviiyE1QAn/Tkk9jOhYerrxJfmMM9mao=",
    "numeroTarjetaDesEncriptado": "4244567895678456",
    "isTarjetaOriginalADesencriptada": true
  },
  "isSuccess": true,
  "message": null,
  "errors": null
}
```

El seguimiento del numero de tarjeta ingresado se ve en las siguientes pantallas.

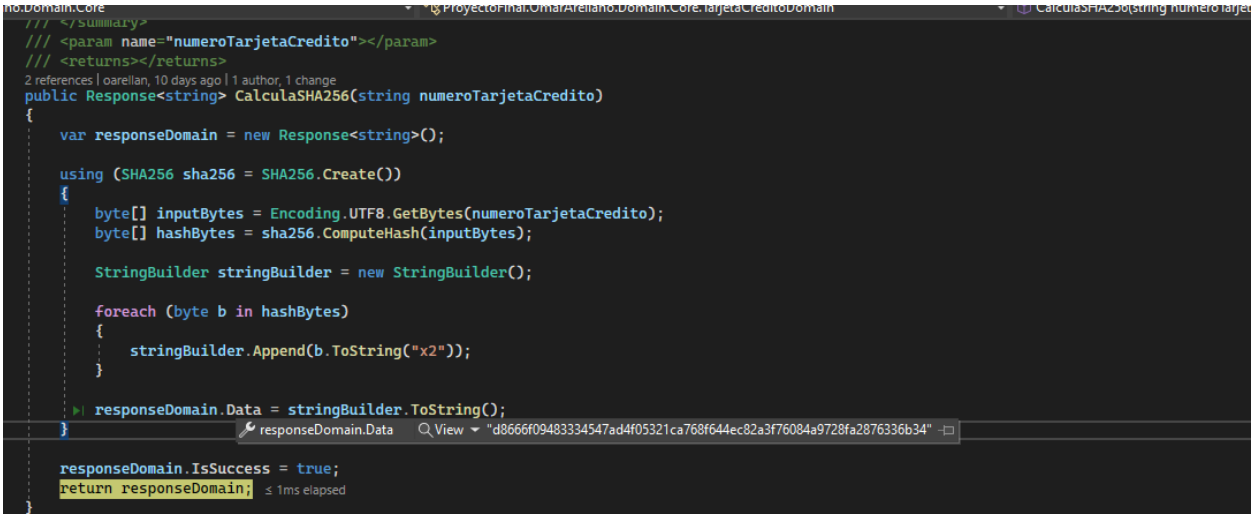
Se muestra cómo se sanitiza por medio de fluentvalidation la cadena ingresada.



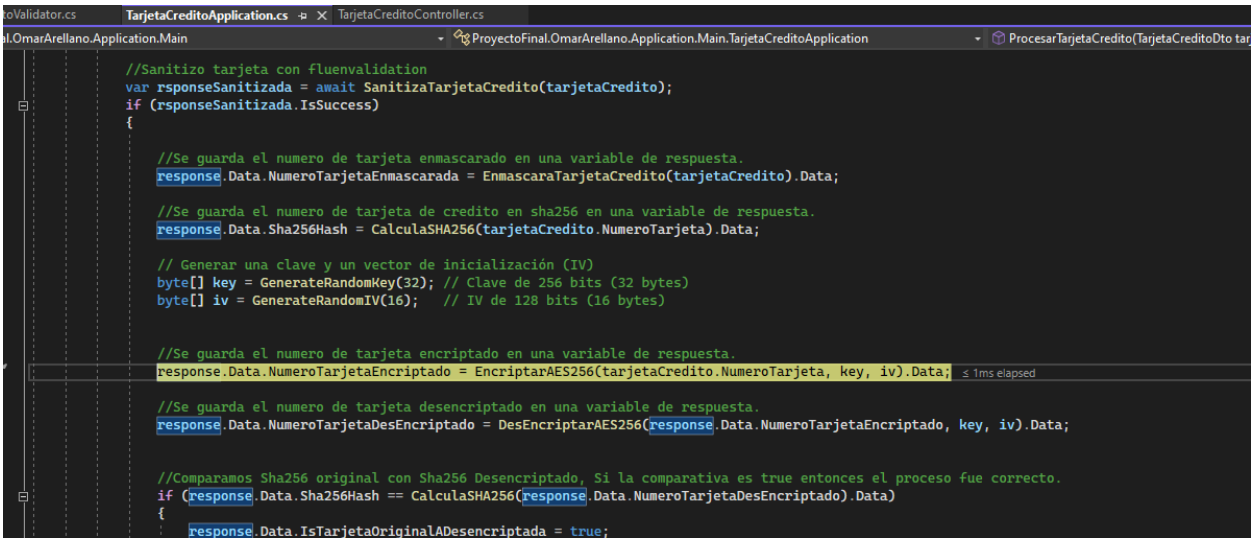
Se implementa un método para que el número de tarjeta ingresado sea enmascarado para su presentación en pantalla.



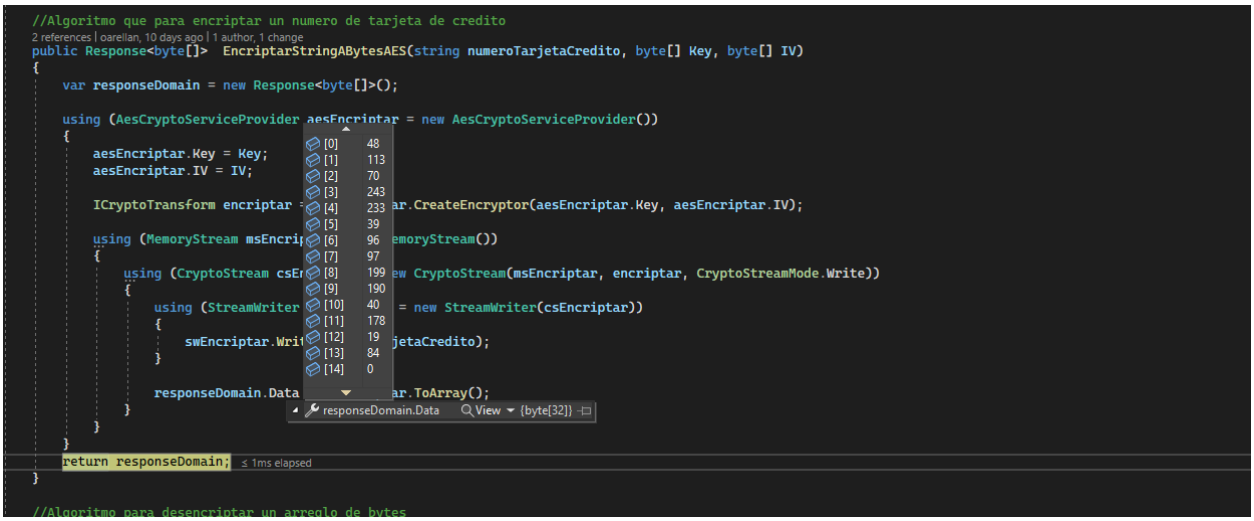
Se aplica la función SHA256



Se genera un vector de inicialización y una llave para la encriptación y desencriptación de la cadena ingresada.

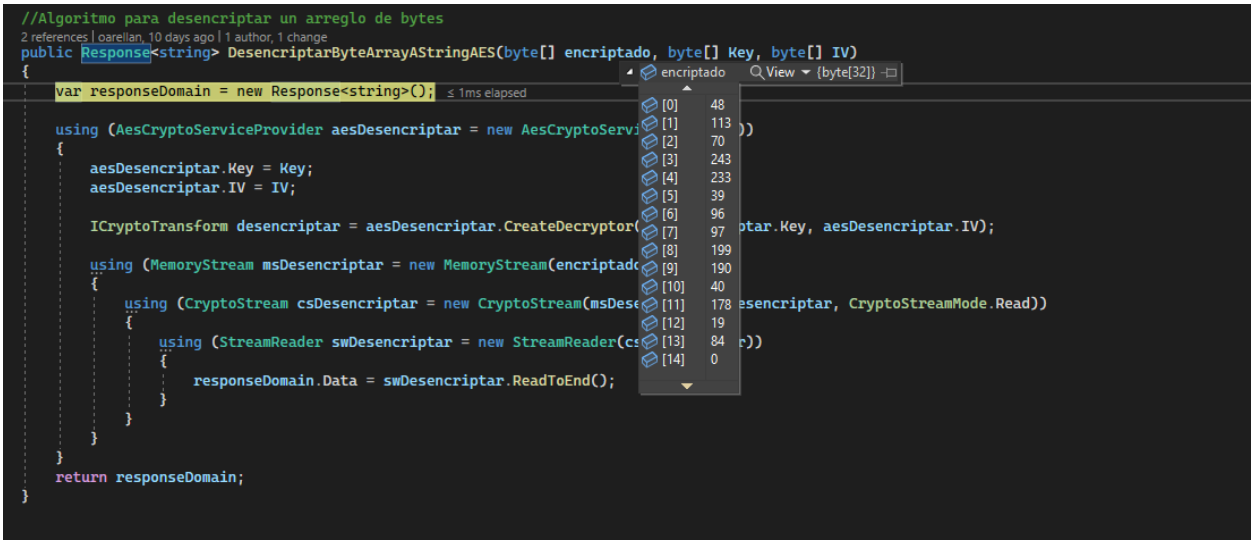


Se encripta la cadena ingresada a esta se le aplica la llave y el vector de inicialización.



```
//Algoritmo que para encriptar un numero de tarjeta de credito
2 references | oarellan, 10 days ago | 1 author, 1 change
public Response<byte[]>  EncriptarStringABytesAES(string numeroTarjetaCredito, byte[] Key, byte[] IV)
{
    var responseDomain = new Response<byte[]>();

    using (AesCryptoServiceProvider aesEncriptar = new AesCryptoServiceProvider())
    {
        aesEncriptar.Key = Key;
        aesEncriptar.IV = IV;

        ICryptoTransform encriptar = aesEncriptar.CreateEncryptor(aesEncriptar.Key, aesEncriptar.IV);

        using (MemoryStream msEncriptar = new MemoryStream())
        {
            using (CryptoStream csEncriptar = new CryptoStream(msEncriptar, encriptar, CryptoStreamMode.Write))
            {
                using (StreamWriter swEncriptar = new StreamWriter(csEncriptar))
                {
                    swEncriptar.Write(numeroTarjetaCredito);
                }

                responseDomain.Data = msEncriptar.ToArray();
            }
        }
    }
    return responseDomain;
}

//Algoritmo para desencriptar un arreglo de bytes
```

Se toma la cadena de bytes encriptado y se procede a desencriptar aplicando la misma llave y el vector de inicialización.



```
//Algoritmo para desencriptar un arreglo de bytes
2 references | oarellan, 10 days ago | 1 author, 1 change
public Response<string> DesencriptarByteArrayAStringAES(byte[] encriptado, byte[] Key, byte[] IV)
{
    var responseDomain = new Response<string>();

    using (AesCryptoServiceProvider aesDesencriptar = new AesCryptoServiceProvider())
    {
        aesDesencriptar.Key = Key;
        aesDesencriptar.IV = IV;

        ICryptoTransform desencriptar = aesDesencriptar.CreateDecryptor(aesDesencriptar.Key, aesDesencriptar.IV);

        using (MemoryStream msDesencriptar = new MemoryStream(encriptado))
        {
            using (CryptoStream csDesencriptar = new CryptoStream(msDesencriptar, desencriptar, CryptoStreamMode.Read))
            {
                using (StreamReader swDesencriptar = new StreamReader(csDesencriptar))
                {
                    responseDomain.Data = swDesencriptar.ReadToEnd();
                }
            }
        }
    }
    return responseDomain;
}
```

La cadena original que se le aplico el SHA256 se compara con el numero de tarjeta desencriptado el cual también pasa por un proceso de SHA256. Si las cadenas son iguales, el método para encriptar y desencriptar fue exitoso y asignamos una bandera en true a la variable IsTarjetaOriginalADesencriptada.