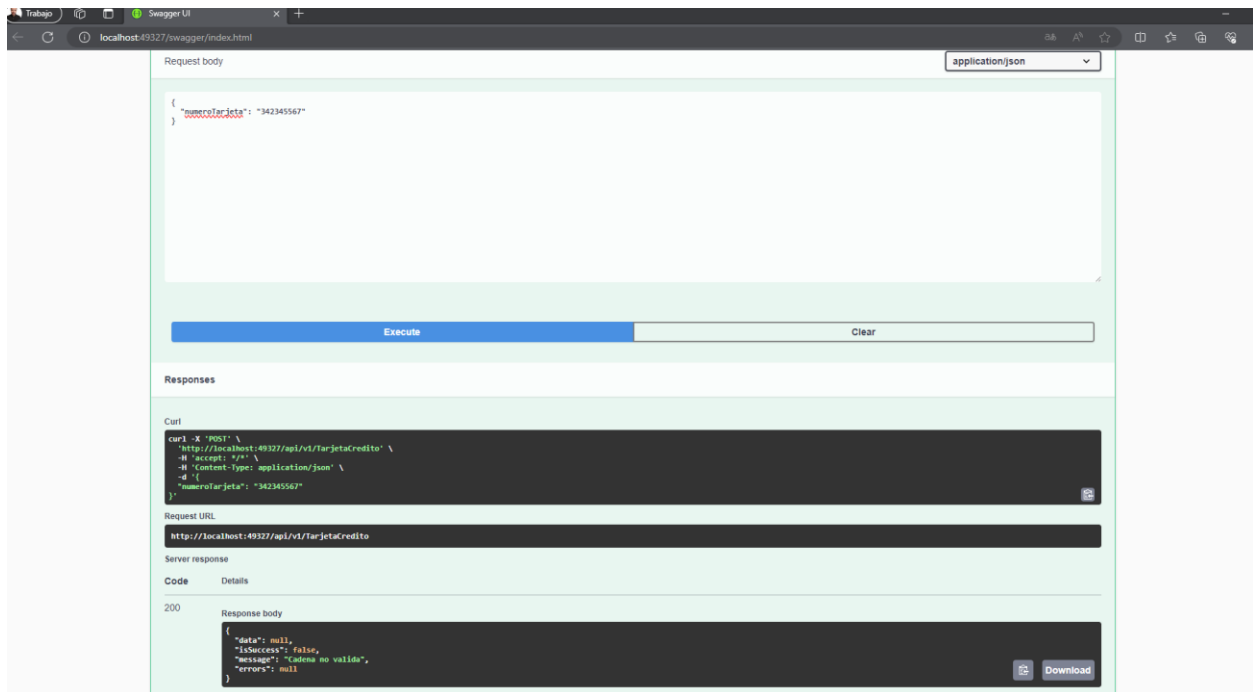


Validando una cadena que no corresponde a una tarjeta de crédito.



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": "d8666f0948334547ad4f05321ca768f644ec82a3f76084a9728fa2876336b34",
    "numeroTarjetaEncriptado": "MHFG8+knYGHviiyE1QAn/Tkk9j0hYerrxJfmMM9mao=",
    "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.

```
/// <summary>
/// Proceso para sanitizar un numero de tarjeta de credito, mediante la libreria fluentvalidation. Vease capa de aplicacion proyecto Validartor para
/// </summary>
/// <param name="tarjetaCredito"></param>
/// <returns></returns>
1 reference | oarellan, 10 days ago | 1 author, 1 change
public async Task<Response<ComprobacionDto>> SanitizaTarjetaCredito(TarjetaCreditoDto tarjetaCredito)
{
    var response = new Response<ComprobacionDto>();
    var validationRequest = await _tarjetaCreditoValidador.ValidateAsync(tarjetaCredito).ConfigureAwait(false);
    if (!validationRequest.IsValid)
    {
        response.IsSuccess = false;
        var errors = validationRequest.Errors.Select(x => x.ErrorMessage).ToList();
        string erroresConcatenados = string.Join(" ", errors);
        response.Message = erroresConcatenados;
        return response;
    }
    response.IsSuccess = true;
    return response;
}
```

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

```
public Response<string> EnmascaraNumeroTarjeta(TarjetaCreditoDto tarjetaCredito)
{
    var responseDomain = new Response<string>();
    var numeroenmascarado = EnmascararTarjeta(tarjetaCredito.NumeroTarjeta);
    responseDomain.Data = numeroenmascarado;
    responseDomain.IsSuccess = true;
    return responseDomain;
}

/// <summary>
/// Metodo que enmascara los digitos de una tarjeta de credito, excepto los ultimos 4.
/// </summary>
/// <param name="digitosTarjeta"></param>
/// <returns></returns>
1 reference | oarellan, 10 days ago | 1 author, 1 change
static string EnmascararTarjeta(string digitosTarjeta)
{
    // Enmascarar todos los digitos, excepto los últimos 4
    int longitud = digitosTarjeta.Length;
    string digitosEnmascarados = new string('*', longitud - 4) + digitosTarjeta.Substring(longitud - 4);
    return digitosEnmascarados;
}
```

Se aplica la función SHA256

```

    /// </summary>
    /// <param name="numeroTarjetaCredito"></param>
    /// <returns></returns>
    2 references | oarellan, 10 days ago | 1 author, 1 change
    public Response<string> CalculaSHA256(string numeroTarjetaCredito)
    {
        var responseDomain = new Response<string>();

        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] inputBytes = Encoding.UTF8.GetBytes(numeroTarjetaCredito);
            byte[] hashBytes = sha256.ComputeHash(inputBytes);

            StringBuilder stringBuilder = new StringBuilder();

            foreach (byte b in hashBytes)
            {
                stringBuilder.Append(b.ToString("x2"));
            }

            responseDomain.Data = stringBuilder.ToString();
        }

        responseDomain.IsSuccess = true;
        return responseDomain;
    }
    responseDomain.Data  Q View "d8666f09483334547ad4f05321ca768f644ec82a3f76084a9728fa2876336b34"
    responseDomain.IsSuccess = true;
    return responseDomain; ≤ 1ms elapsed
}

```

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

```

//Sanitizo tarjeta con fluenvalidation
var responseSanitizada = await SanitizaTarjetaCredito(tarjetaCredito);
if (responseSanitizada.IsSuccess)
{
    //Se guarda el numero de tarjeta enmascarado en una variable de respuesta.
    response.Data.NumeroTarjetaEnmascarada = EnmascaraTarjetaCredito(tarjetaCredito).Data;

    //Se guarda el numero de tarjeta de credito en sha256 en una variable de respuesta.
    response.Data.Sha256Hash = CalculaSHA256(tarjetaCredito.NumeroTarjeta).Data;

    // Generar una clave y un vector de inicialización (IV)
    byte[] key = GenerateRandomKey(32); // Clave de 256 bits (32 bytes)
    byte[] iv = GenerateRandomIV(16); // IV de 128 bits (16 bytes)

    //Se guarda el numero de tarjeta encriptado en una variable de respuesta.
    response.Data.NumeroTarjetaEncriptado = EncriptarAES256(tarjetaCredito.NumeroTarjeta, key, iv).Data; ≤ 1ms elapsed

    //Se guarda el numero de tarjeta desencriptado en una variable de respuesta.
    response.Data.NumeroTarjetaDesEncriptado = DesEncriptarAES256(response.Data.NumeroTarjetaEncriptado, key, iv).Data;

    //Comparamos Sha256 original con Sha256 Desencriptado, Si la comparativa es true entonces el proceso fue correcto.
    if (response.Data.Sha256Hash == CalculaSHA256(response.Data.NumeroTarjetaDesEncriptado).Data)
    {
        response.Data.IsTarjetaOriginalADesencriptada = true;
    }
}

```

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[]> EncriptarStringBytesAES(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.WriteLine(numeroTarjetaCredito);
                }

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

    return responseDomain;
}
```

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 descriptado el cual también pasa por un proceso de SHA256. Si las cadenas son iguales, el método para encriptar y descriptar fue exitoso y asignamos una bandera en true a la variable IsTarjetaOriginalADescriptada.

```
//Se guarda el numero de tarjeta de crédito en sha256 en una variable de respuesta.
response.Data.Sha256Hash = CalculaSHA256(tarjetaCredito.NumeroTarjeta).Data;

// Generar una clave y un vector de inicialización (IV)
byte[] key = GenerateRandomKey(32); // Clave de 256 bits (32 bytes)
byte[] iv = GenerateRandomIV(16); // IV de 128 bits (16 bytes)

//Se guarda el numero de tarjeta encriptado en una variable de respuesta.
response.Data.NumeroTarjetaEncriptado = EncriptarAES256(tarjetaCredito.NumeroTarjeta, key, iv).Data;

//Se guarda el numero de tarjeta descriptado en una variable de respuesta.
response.Data.NumeroTarjetaDesEncriptado = DesEncriptarAES256(response.Data.NumeroTarjetaEncriptado, key, iv).Data;

//Comparamos Sha256 original con Sha256 Descriptado, Si la comparativa es true entonces el proceso fue correcto.
if (response.Data.Sha256Hash == CalculaSHA256(response.Data.NumeroTarjetaDesEncriptado).Data)
{
    response.Data.IsTarjetaOriginalADescriptada = true;
}
response.IsSuccess = true;
return response;
}

return rsponseSanitizada;
}
```