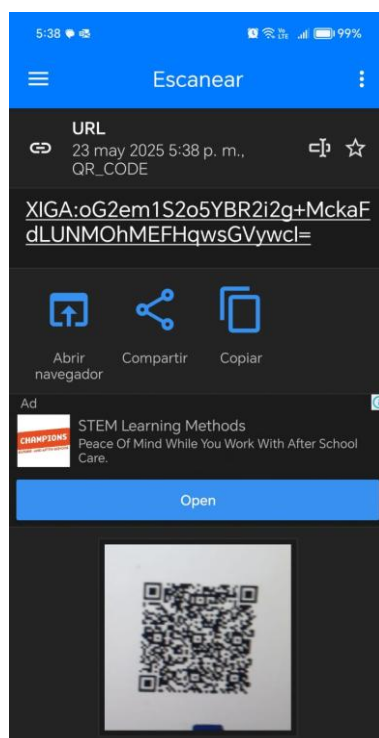


63. Evidencia de identificador de cliente-dispositivo almacenado en QR

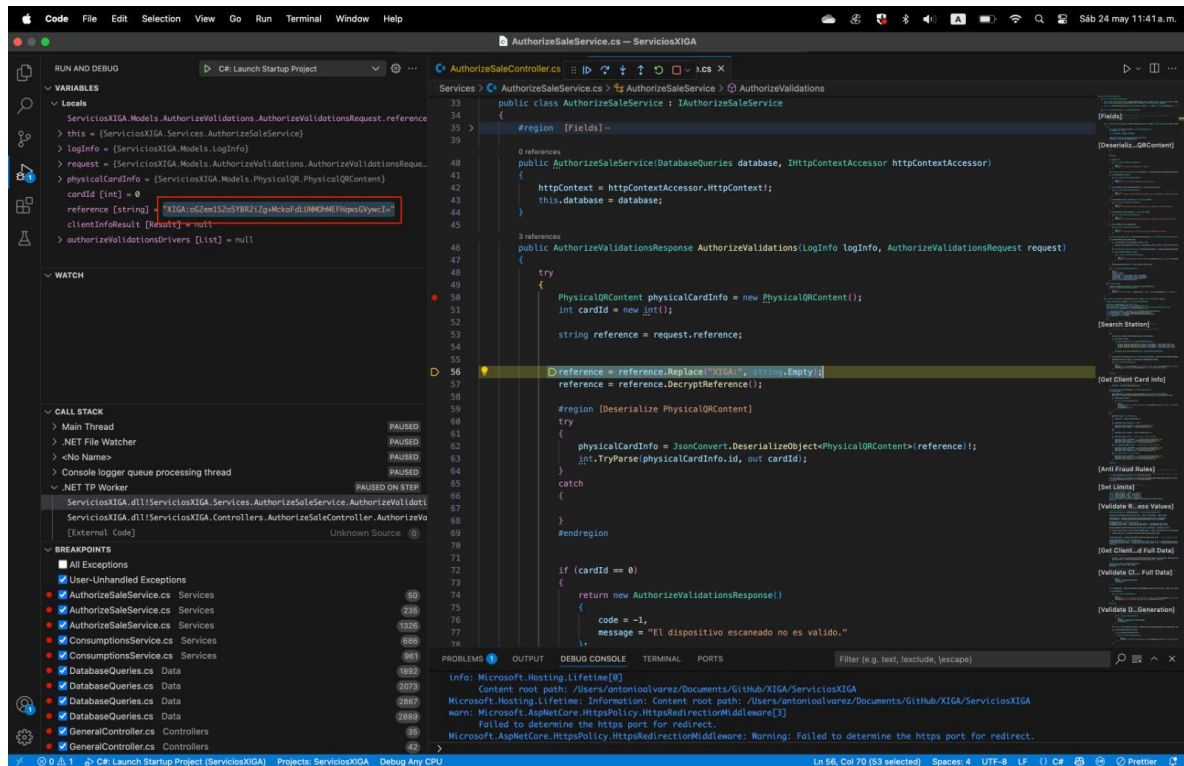
La imagen muestra ejemplar de tarjeta de QR, para el cliente **9801** Corporativo de Energía Servicios y Desarrollo de negocios S.A de C.V. vehículo **1308**, asignada a Merced Ortiz, la cual presenta código QR en la parte superior derecha donde se almacena un identificador del dispositivo (tarjeta) encriptado con el algoritmo 3DES, además de otros datos desencryptados en las siguientes imágenes.



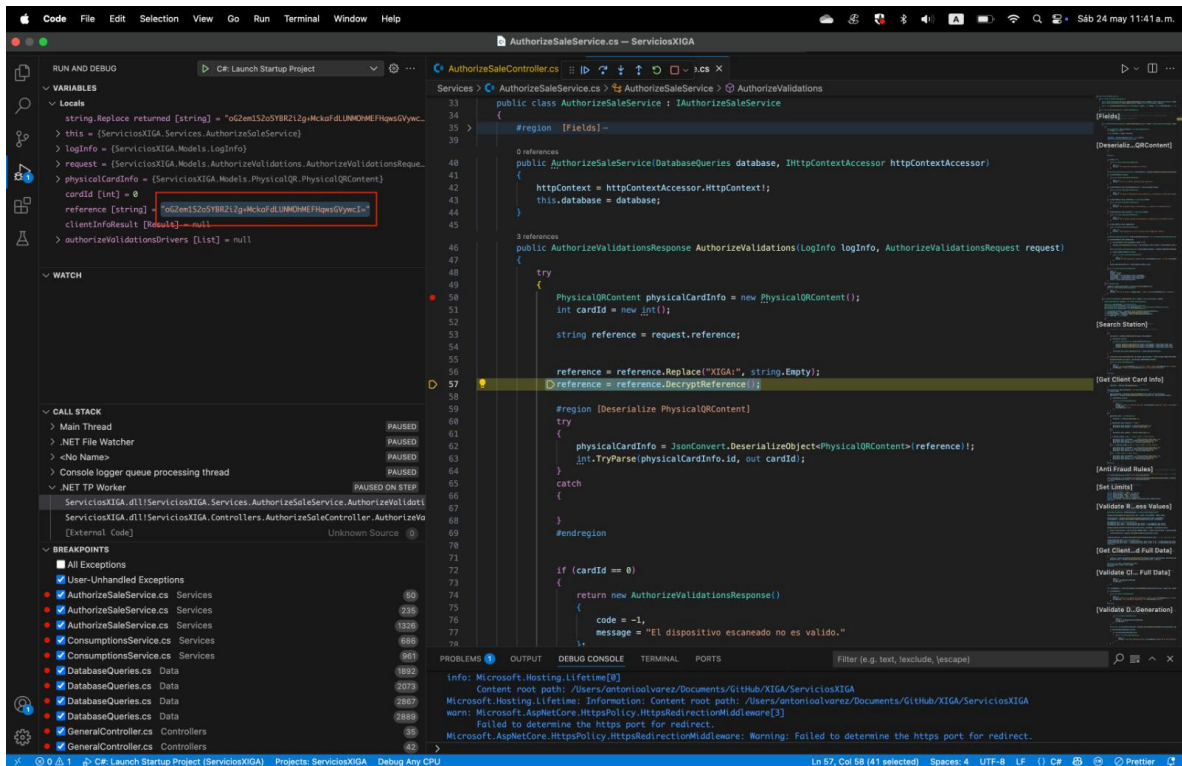
La imagen muestra captura de pantalla de un celular escaneando con la cámara el QR de la Tarjeta, a fin de obtener el identificador del dispositivo almacenado en el QR la cual muestra un hash encriptado con el prefijo “XIGA:”



La imagen muestra la consola de debug de Visual Studio donde se observa una traza paso a paso de la ejecución del código que procesa el escaneo de la tarjeta QR, pantalla donde se observa el mismo hash obtenido con el teléfono:

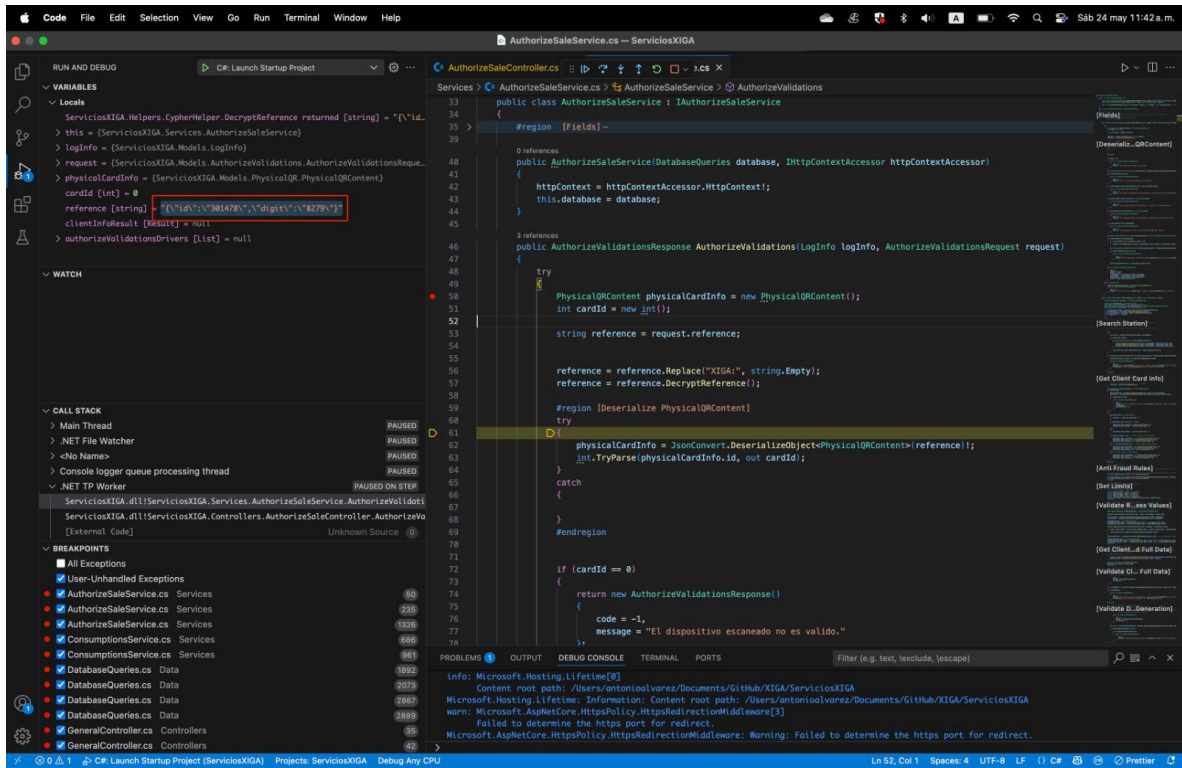


La imagen muestra la parte del código encargado de descryptar el hash, misma que se realiza con una librería nativa de C# que hace una implementación del algoritmo 3DES, que aunque es un algoritmo sencillo, es lo suficientemente seguro pues solo se utiliza para cifrar un identificador que se mapea en número de cliente y vehículo.



```
33 public class AuthorizeSaleService : IAuthorizeSaleService
34 {
35     #region [Fields]
36
37     0 references
38     public AuthorizeSaleService(DatabaseQueries database, IHttpContextAccessor httpContextAccessor)
39     {
40         httpContext = httpContextAccessor.HttpContext!;
41         this.database = database;
42     }
43
44     3 references
45     public AuthorizationsResponse AuthorizeValidations(LogInfo logInfo, AuthorizationsRequest request)
46     {
47         try
48         {
49             PhysicalQRContent physicalCardInfo = new PhysicalQRContent();
50             int cardId = new_int();
51
52             string reference = request.reference;
53
54             reference = reference.Replace("XIGA:", string.Empty);
55             reference = reference.DecryptReference();
56
57             #region [Deserialize PhysicalQRContent]
58             try
59             {
60                 physicalCardInfo = JsonConvert.DeserializeObject<PhysicalQRContent>(reference);
61                 _int.TryParse(physicalCardInfo.id, out cardId);
62             }
63             catch
64             {
65             }
66             #endregion
67
68             if (cardId == 0)
69             {
70                 return new AuthorizationsResponse()
71                 {
72                     code = -1,
73                     message = "El dispositivo escaneado no es valido."
74                 };
75             }
76         }
77         catch
78         {
79         }
80     }
81 }
```

La imagen muestra consola de debug de Visual Studio donde se observa el hash desencriptado apreciándose el identificador de CardID con el valor **301478**



La imagen muestra captura de pantalla del registro en la base de datos de SQL donde se observa que el identificador CardID **301478** mapea el número de cliente **9801** y vehículo **1308** los cuales son los mismos que aparecen en la fotografía del ejemplar de la tarjeta presentado al inicio.

The screenshot shows the DBeaver SQL client interface. The top toolbar includes buttons for 'Commit', 'Rollback', 'Auto', and 'Script-59'. The main query editor contains the following SQL statement:

```
select card_id, hologram_digit_security, clientNumber, vehicle, Name, Identification from GasmartCard.dbo.gmc_card where card_id = 301478
```

The results pane displays a single row of data with the following columns and values:

card_id	hologram_digit_security	clientNumber	vehicle	Name	Identification
301478	8279	9801	1308	MERCED ORTIZ MELCHOR	NISSAN TIIDA -APC363A

The bottom status bar indicates '1 row(s) fetched - 0.460s, on 2025-05-24 at 11:44:26'.