Versión: 03
Código: GFPI-F-147

Proceso Gestión de Formación Profesional Integral
Formato Bitácora seguimiento Etapa productiva

REGIONAL CAUCA

CENTRO DE COMERCIO Y SERVICIO

BITÁCORA DE SEGUIMIENTO ETAPA PRODUCTIVA

Nombre de la empresa donde está realizando la etapa productiva	NIT	BITACORA N°	Período
SERVIPARAMO SAS	**890116102-1**	**2**	11/10/2024-240/10/2024

Nombre del jefe inmediato/Responsable	Teléfono de contacto	Correo electrónico
JAMES CALVO	312 381 1324	jcalvo@seviparamo.co

Seleccione con una "X" el tipo de modalidad de etapa productiva

CONTRATO DE APRENDIZAJE		VÍNCULO LABORAL O CONTRACTUAL	X	PROYECTO PRODUCTIVO		APOYO A UNA UNIDAD PRODUCTIVA FAMILIAR		APOYO A INSTITUCIÓN ESTATAL NACIONAL,TERRITORIAL , O A UNA ONG, O A ENTIDAD SIN ANIMO DE LUCRO		MONITORIA		PASANTIA	

Nombre del aprendiz	Documento Id.	Teléfono de contacto	Correo electrónico institucional
RODNEY ZAPATA PALACIO	72209311	**312 681 0844**	rodney.zapata@soy.sena.edu.co

Número de ficha	Programa de formación
2675810	TECNOLOGO EN ANALISIS Y DESARROLLO DE SOFTWARE

DESCRIPCIÓN DE LA ACTIVIDAD *(Ingrese cuantas filas sean necesarias)*	FECHA INICIO	FECHA FIN	EVIDENCIA DE CUMPLIMIENTO	OBSERVACIONES, INASISTENCIAS Y/O DIFICULTADES PRESENTADAS
Agregue una tabla llamada Factura_token, que guarde el token que debe ser enviado, en la validacion de credenciales para emitir la factura. La idea es tratar de emitir la mayor cantidad de facturas posibles con un token, y no estar generando un token por cada factura, para administrar mejor los recursos.	11-oct	11-oct	La evidencia con captura de pantallas, las envio en archivo de word llamado evidencia_bitacora_02	
Generamos codigo en c# para la generacion del Token, este token debe usarse en dos ambiente : 1. ambiente de prueba 2. ambiente productivo, con dos bases de datos independiente	13-oct	13-oct		
Guardamos el token generado en la base de datos, para ser utilizado en el proceso de Emisión de factura, el código se realiza en C#.	14-oct	14-oct		
Habilitar protocolo TLS 1.2, A partir del 31 de octubre de 2024, todas las conexiones a nuestros servicios deberán utilizar mínimo TLS 1.2, un protocolo que garantiza una mayor seguridad en la transmisión de datos. Es crucial que sus sistemas y aplicaciones sean compatibles con TLS 1.2 para evitar problemas de conexión o acceso.	15-oct	15-oct		
Clase ApiRespons en C# que se utiliza para mapear la respuesta JSON del servicio	16-oct	16-oct		
Consulto de credencias en Base de Datos	17-oct	17-oct		
Guardar el JSON devuelto en un archivo txt, para más tarde Deserializarlo.	18-oct	18-oct		
Método escrito en C# para Deserializar JSON	21-oct	21-oct		
Guardamos respuesta del Response, una vez serializado en campos de la base de datos:	22-oct	22-oct		
Metodo que consume el servicio Web	23-oct	23-oct		

Medado principal que llama el servicio web y ejecuta cada uno de los métodos asociado como son: consultar las credenciales, traer el token autorizado, llamar al servicio y pasarle en el body el XML de la factura, además verificar si la emisión fue exitosa,Dicha respuesta la rerializamos y finalmente la guardamos en la base de datos.	24-oct	24-oct		

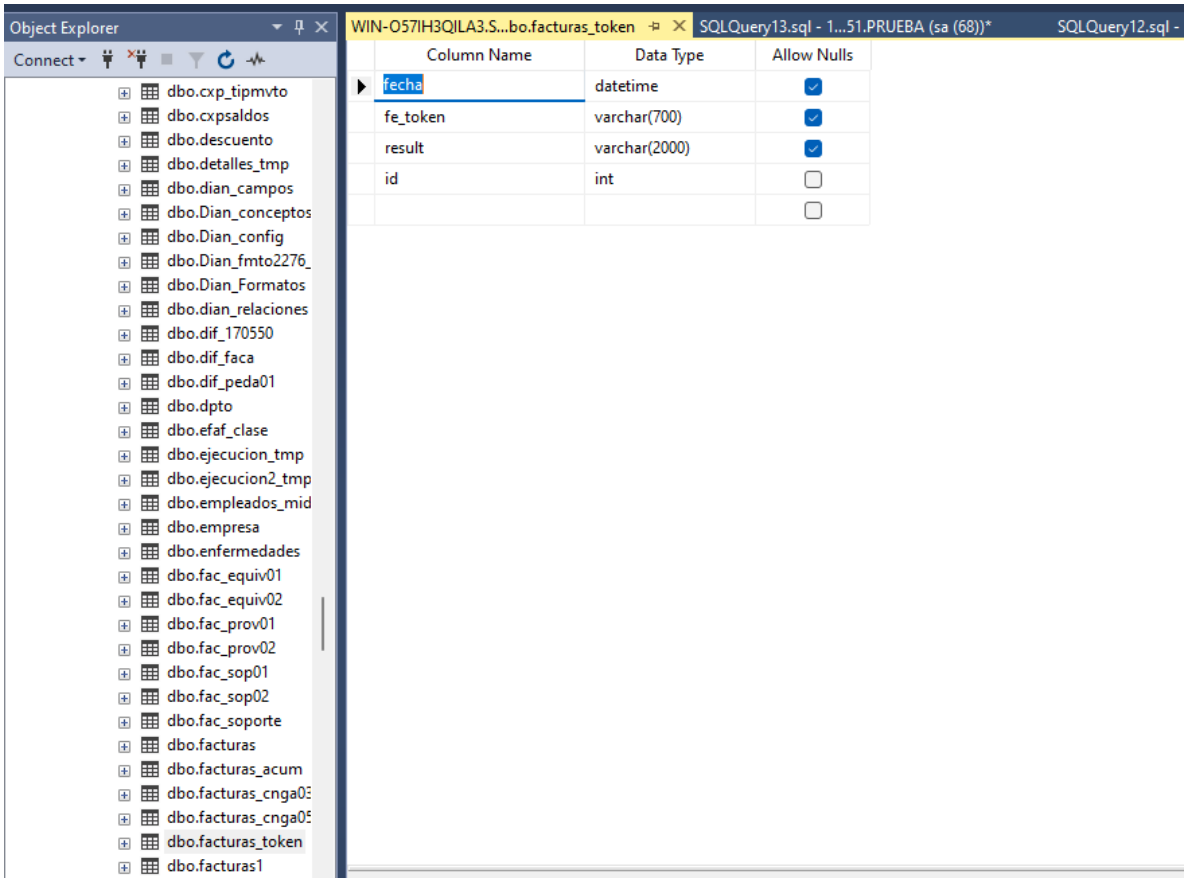
Aprendiz: *recuerde diligenciar completamente el informe y entregarlo o subirlo al espacio asignado para este.*

RODNEY ZAPATA PALACIO		
Nombre del Aprendiz	**Firma del aprendiz**	**Fecha entrega bitácora**
MARTHA ISABEL ORDOÑEZ ARDILA		
Nombre del Instructor de Seguimiento	**Firma de instructor de seguimiento**	**Firma del jefe inmediato (Si es del caso)**

Nota: LOS DATOS PROPORCIONADOS SERÁN TRATADOS DE ACUERDO CON LA POLÍTICA DE TRATAMIENTO DE DATOS PERSONALES DEL SENA Y A LA LEY 1581 DE 2012.

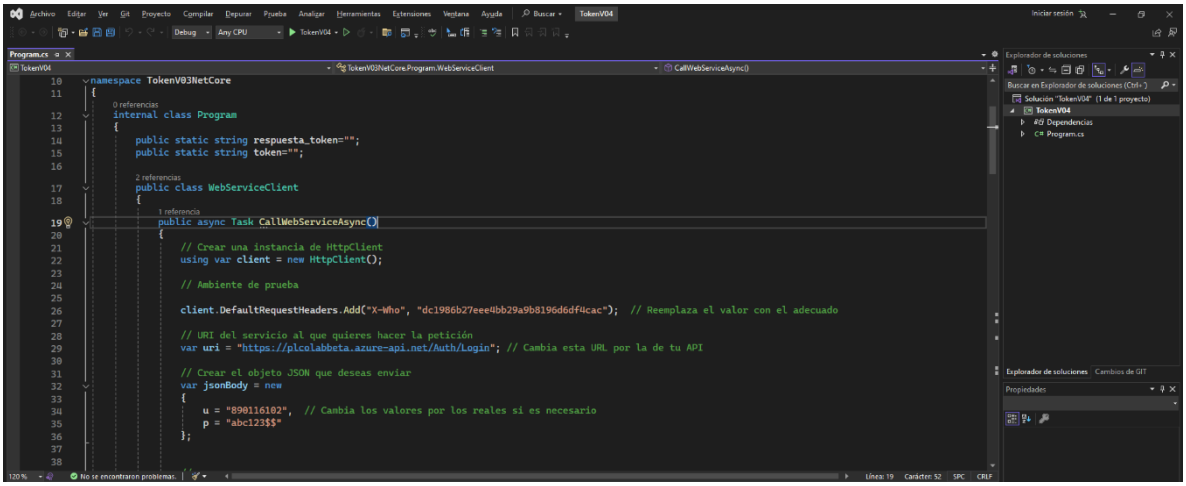
Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**

1. Fecha: 11/10/2024 Agregue una tabla llamada Factura_token, que guarde el token que debe ser enviado, en la validación de credenciales para emitir la factura.



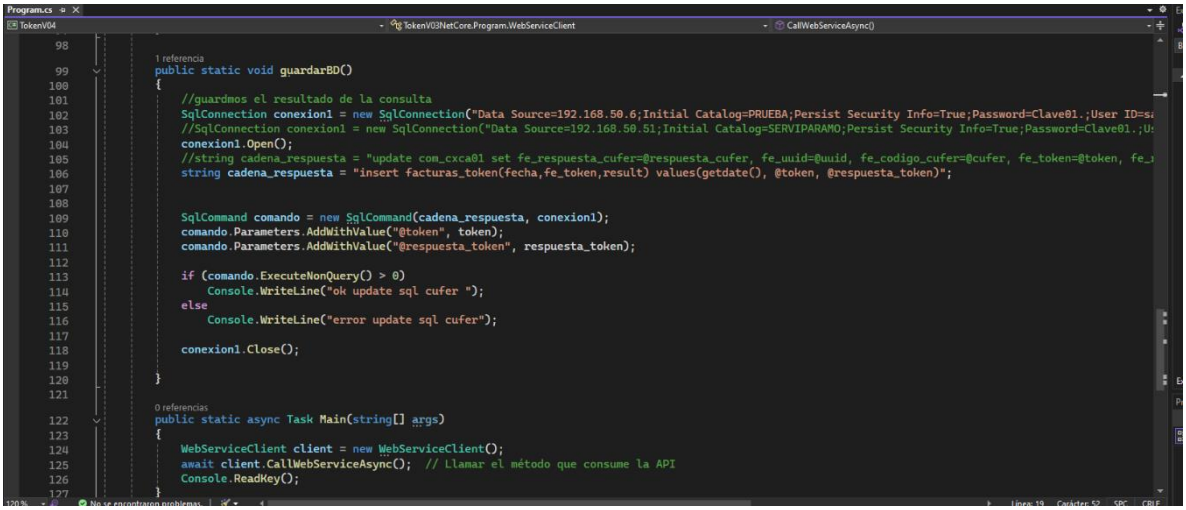
2. Fecha 13/10/2024 Generamos código en c#. Para la generación del Token, este token debe usarse en dos ambiente : 1. ambiente de prueba 2. ambiente productivo, con dos bases de datos independiente.

Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**



```
namespace TokenV03NetCore
{
    internal class Program
    {
        public static string respuesta_token="";
        public static string token="";

        public class WebServiceClient
        {
            public async Task CallWebServiceAsync()
            {
                // Crear una instancia de HttpClient
                using var client = new HttpClient();

                // Ambiente de prueba

                client.DefaultRequestHeaders.Add("X-Who", "dc1986b27eee4bb29a9b8196d6df4cac");  // Reemplaza el valor con el adecuado

                // URI del servicio al que quieres hacer la petición
                var uri = "https://plcolabbeta.azure-api.net/Auth/Login"; // Cambia esta URL por la de tu API

                // Crear el objeto JSON que deseas enviar
                var jsonBody = new
                {
                    u = "890116102",  // Cambia los valores por los reales si es necesario
                    p = "abc123$$"
                };
```

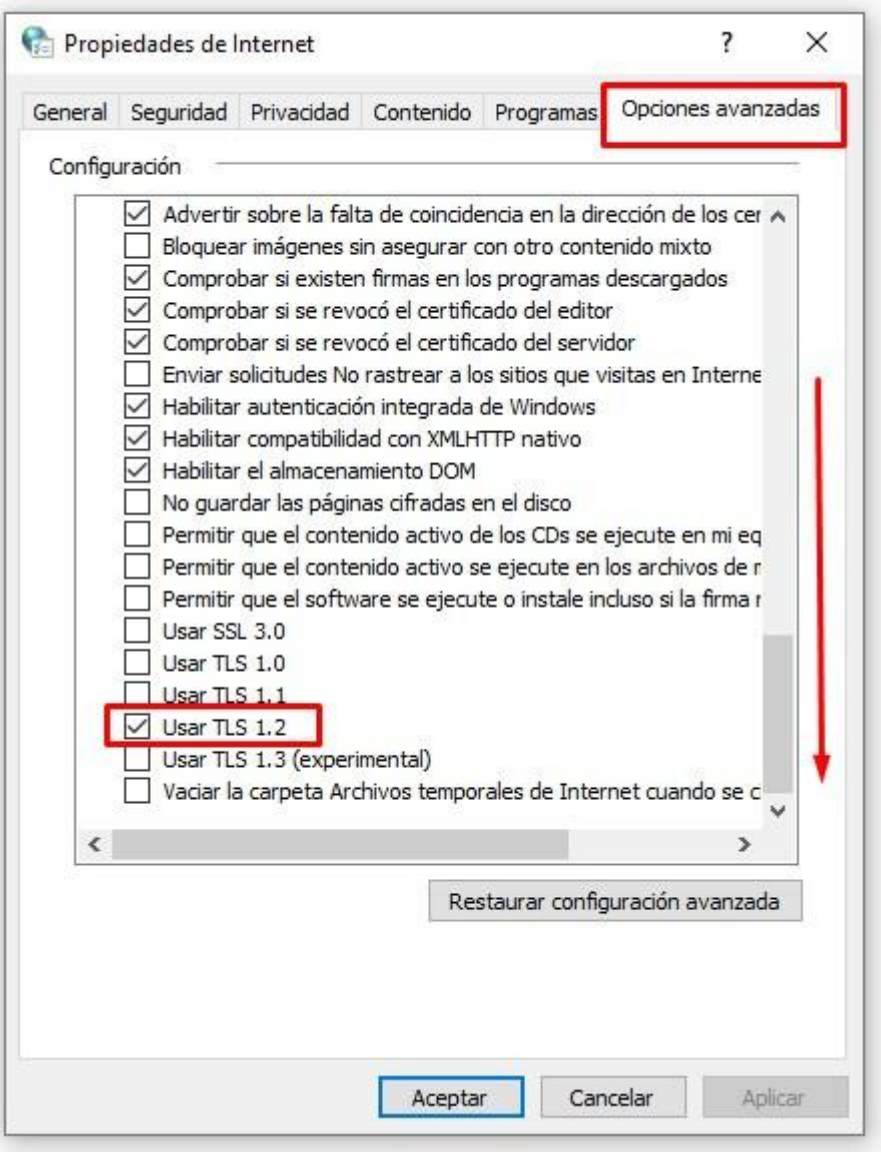
3. 14/10/2024 Guardamos el token generado en la base de datos, para ser utilizado en el proceso de Emisión de factura, el código se realiza en C#.



```
        public static void guardarBD()
        {
            //guardmos el resultado de la consulta
            SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01.;User ID=s
            //SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.51;Initial Catalog=SERVIPARAMO;Persist Security Info=True;Password=Clave01.;U
            conexion1.Open();
            //string cadena_respuesta = "update com_cxca01 set fe_respuesta_cufer=@respuesta_cufer, fe_uuid=@uuid, fe_codigo_cufer=@cufer, fe_token=@token, fe_
            string cadena_respuesta = "insert facturas_token(fecha,fe_token,result) values(getdate(), @token, @respuesta_token)";


            SqlCommand comando = new SqlCommand(cadena_respuesta, conexion1);
            comando.Parameters.AddWithValue("@token", token);
            comando.Parameters.AddWithValue("@respuesta_token", respuesta_token);

            if (comando.ExecuteNonQuery() > 0)
                Console.WriteLine("ok update sql cufer ");
            else
                Console.WriteLine("error update sql cufer");

            conexion1.Close();

        }

        public static async Task Main(string[] args)
        {
            WebServiceClient client = new WebServiceClient();
            await client.CallWebServiceAsync();  // Llamar el método que consume la API
            Console.ReadKey();
        }
```

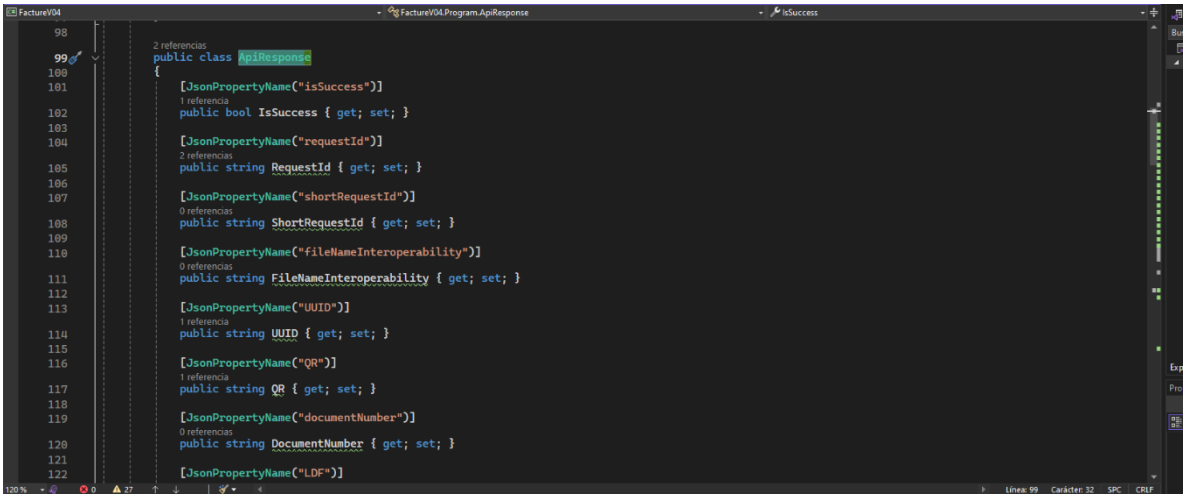
4. 15/10/2024 Habilitar protocolo TLS 1.2, A partir del **31 de octubre de 2024**, todas las conexiones a nuestros servicios deberán **utilizar mínimo TLS 1.2**, un protocolo que garantiza una mayor seguridad en la transmisión de datos. Es crucial que sus sistemas y aplicaciones sean compatibles con TLS 1.2 para **evitar problemas** de conexión o acceso.

Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**



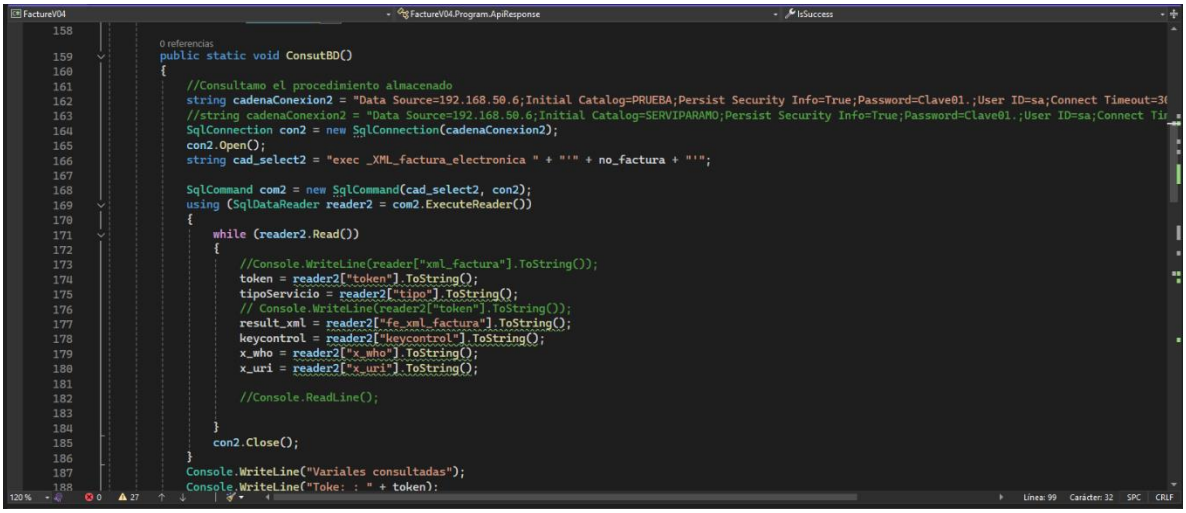
5. 16/10/2024 Clase ApiRespons en C# que se utiliza para mapear la respuesta de la API en formato JSON.

Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**



```
FactureV04 > FactureV04.Program.ApiResponse > IsSuccess

2 referencias
public class ApiResponse
{
    [JsonPropertyName("isSuccess")]
    1 referencia
    public bool IsSuccess { get; set; }

    [JsonPropertyName("requestId")]
    2 referencias
    public string RequestId { get; set; }

    [JsonPropertyName("shortRequestId")]
    0 referencias
    public string ShortRequestId { get; set; }

    [JsonPropertyName("fileNameInteroperability")]
    0 referencias
    public string FileNameInteroperability { get; set; }

    [JsonPropertyName("UUID")]
    1 referencia
    public string UUID { get; set; }

    [JsonPropertyName("QR")]
    1 referencia
    public string QR { get; set; }

    [JsonPropertyName("documentNumber")]
    0 referencias
    public string DocumentNumber { get; set; }

    [JsonPropertyName("LDF")]
```

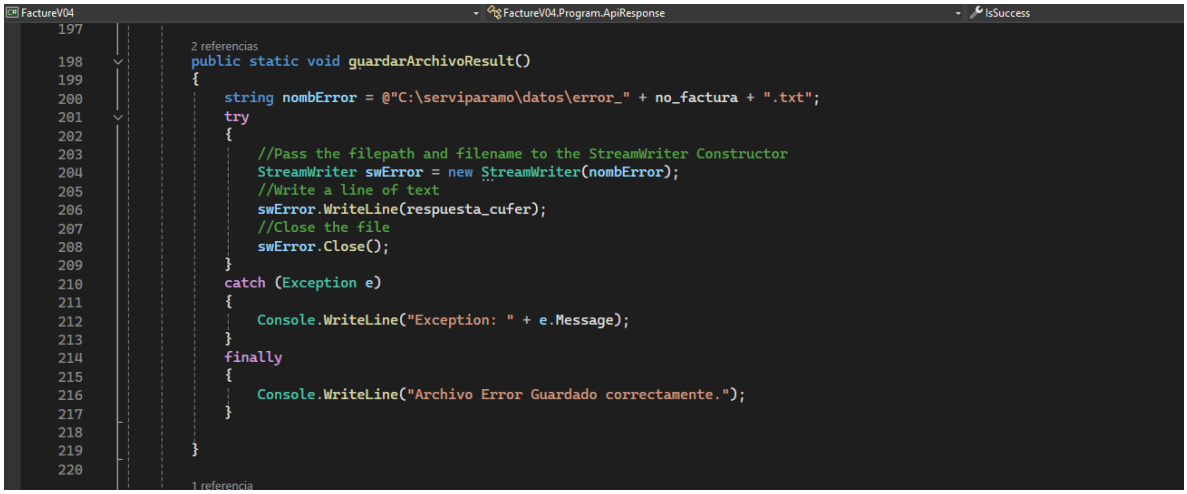
6. 17/10/2024 Consulto de credencias en Base de Datos



```
FactureV04 > FactureV04.Program.ApiResponse > IsSuccess

0 referencias
public static void ConsutBD()
{
    //Consultamo el procedimiento almacenado
    string cadenaConexion2 = "Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01.;User ID=sa;Connect Timeout=3(
    //string cadenaConexion2 = "Data Source=192.168.50.6;Initial Catalog=SERVIPARAMO;Persist Security Info=True;Password=Clave01.;User ID=sa;Connect Tir
    SqlConnection con2 = new SqlConnection(cadenaConexion2);
    con2.Open();
    string cad_select2 = "exec _XML_factura_electronica " + "'" + no_factura + "'";

    SqlCommand com2 = new SqlCommand(cad_select2, con2);
    using (SqlDataReader reader2 = com2.ExecuteReader())
    {
        while (reader2.Read())
        {
            //Console.WriteLine(reader["xml_factura"].ToString());
            token = reader2["token"].ToString();
            tipoServicio = reader2["tipo"].ToString();
            // Console.WriteLine(reader2["token"].ToString());
            result_xml = reader2["fe_xml_factura"].ToString();
            keycontrol = reader2["keycontrol"].ToString();
            x_who = reader2["x_who"].ToString();
            x_uri = reader2["x_uri"].ToString();

            //Console.ReadLine();

        }
        con2.Close();
    }
    Console.WriteLine("Variales consultadas");
    Console.WriteLine("Toke: : " + token);
```

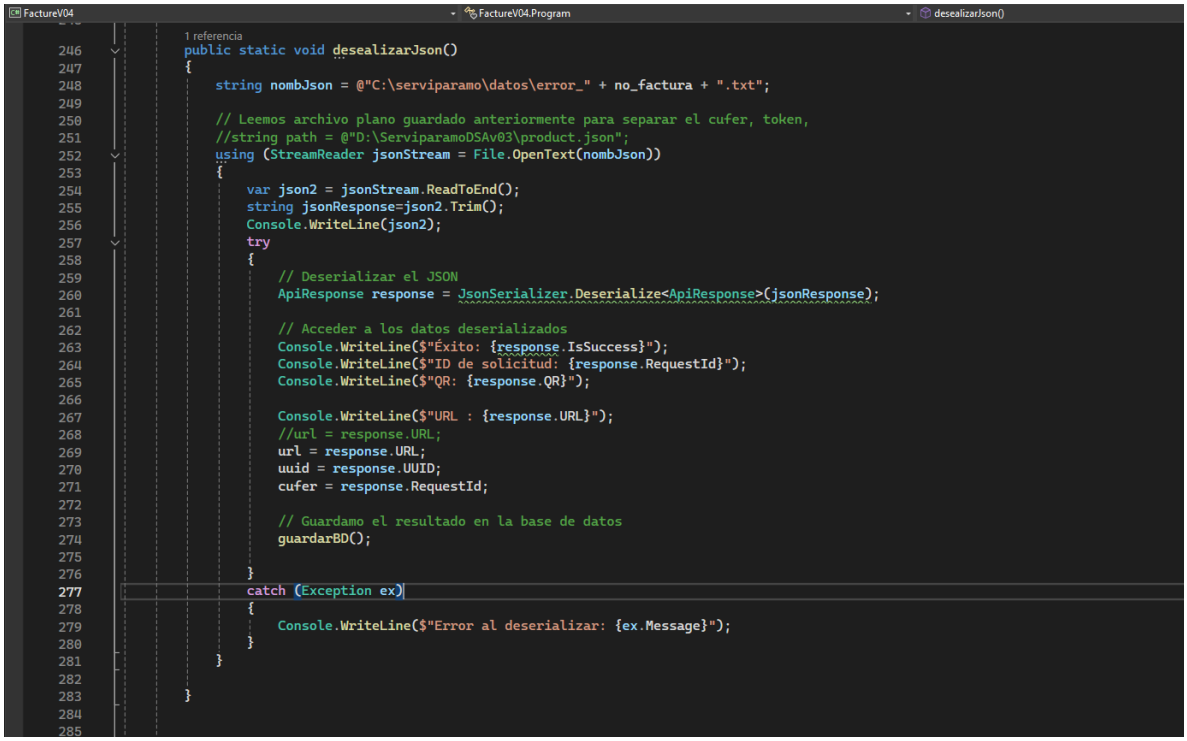
Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**

7. 18/10/2024 Guardar el JSON devuelto en un archivo txt, para más tarde **Deserializarlo.**



```
FactureV04 | FactureV04.Program.ApiResponse | IsSuccess
197
        2 referencias
198     public static void guardarArchivoResult()
199     {
200         string nombError = @"C:\serviparamo\datos\error_" + no_factura + ".txt";
201         try
202         {
203             //Pass the filepath and filename to the StreamWriter Constructor
204             StreamWriter swError = new StreamWriter(nombError);
205             //Write a line of text
206             swError.WriteLine(respuesta_cufer);
207             //Close the file
208             swError.Close();
209         }
210         catch (Exception e)
211         {
212             Console.WriteLine("Exception: " + e.Message);
213         }
214         finally
215         {
216             Console.WriteLine("Archivo Error Guardado correctamente.");
217         }
218
219     }
220
        1 referencia
```

8. 21/10/2024 Método escrito en C# para Deserializar JSON:



```
FactureV04 | FactureV04.Program | desealizarJson()
        1 referencia
246     public static void desealizarJson()
247     {
248         string nombJson = @"C:\serviparamo\datos\error_" + no_factura + ".txt";
249
250         // Leemos archivo plano guardado anteriormente para separar el cufer, token,
251         //string path = @"D:\ServiparamoDSAv03\product.json";
252         using (StreamReader jsonStream = File.OpenText(nombJson))
253         {
254             var json2 = jsonStream.ReadToEnd();
255             string jsonResponse=json2.Trim();
256             Console.WriteLine(json2);
257             try
258             {
259                 // Deserializar el JSON
260                 ApiResponse response = JsonSerializer.Deserialize<ApiResponse>(jsonResponse);
261
262                 // Acceder a los datos deserializados
263                 Console.WriteLine($"Éxito: {response.IsSuccess}");
264                 Console.WriteLine($"ID de solicitud: {response.RequestId}");
265                 Console.WriteLine($"QR: {response.QR}");
266
267                 Console.WriteLine($"URL : {response.URL}");
268                 //url = response.URL;
269                 url = response.URL;
270                 uuid = response.UUID;
271                 cufer = response.RequestId;
272
273                 // Guardamo el resultado en la base de datos
274                 guardarBD();
275
276             }
277             catch (Exception ex)
278             {
279                 Console.WriteLine($"Error al deserializar: {ex.Message}");
280             }
281         }
282
283     }
284
285
```

Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**

9. 22/10/2024 Guardamos respuesta del Response, una vez serializado en campos de la base de datos:

```
public static void guardarBD()
{
    //guardmos el resultado de la consulta
    SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01.;User ID=s
    conexion1.Open();
    string cadena_respuesta = "update com_cxca01 set fe_respuesta_cufer=@respuesta_cufer, fe_uuid=@uuid, fe_codigo_cufer=@cufer, fe_token=@token, fe_re


    SqlCommand comando = new SqlCommand(cadena_respuesta, conexion1);
    comando.Parameters.AddWithValue("@respuesta_cufer", respuesta_cufer);
    comando.Parameters.AddWithValue("@uuid", uuid);
    comando.Parameters.AddWithValue("@cufer", cufer);
    comando.Parameters.AddWithValue("@token", token);
    comando.Parameters.AddWithValue("@url", url);
    comando.Parameters.AddWithValue("@numfac", no_factura);

    if (comando.ExecuteNonQuery() > 0)
        Console.WriteLine("ok update sql cufer ");
    else
        Console.WriteLine("error update sql cufer");

    conexion1.Close();

}
```

10. 23/10/2024 Método que consume el servicio Web

```
public async Task CallWebServiceAsync()
{
    // Crear una instancia de HttpClient
    using var client = new HttpClient();

    // Ambiente de prueba

    client.DefaultRequestHeaders.Add("X-Who", x_who);
    client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
    client.DefaultRequestHeaders.Add("X-REF-DOCUMENTTYPE", "FACTURA-UBL");
    client.DefaultRequestHeaders.Add("X-KEYCONTROL", keycontrol);
    client.DefaultRequestHeaders.Add("X-REF-TEMPLATE", tipoServicio);

    // URI del servicio al que quieres hacer la petición
    //var uri = "https://plcolabbeta.azure-api.net/Auth/Login"; // Cambia esta URL por la de tu API
    var uri = x_uri;

    //Cuerpo del XML lo traigo desde un campo de la base de datos
    string xmlBody = result_xml;


    // Convertir el cuerpo a bytes
    var content = new StringContent(xmlBody, Encoding.UTF8, "application/xml");

    try
    {
        // Hacer la solicitud POST enviando el contenido
        HttpResponseMessage response = await client.PostAsync(uri, content);

        // Leer y mostrar el contenido de la respuesta
        string result = await response.Content.ReadAsStringAsync();
        Console.WriteLine("Respuesta: " + result);

        // Verificar el estado de la respuesta
        if (response.IsSuccessStatusCode)
        {
            Console.WriteLine("Solicitud exitosa.");
            respuesta_cufer = result;
            token =
                respuesta_cufer.Substring(
```

Evidencias Bitácora No 2			
Ficha	**Aprendiz**	**cedula**	**periodo**
2675810	**RODNEY ZAPATA PALACIO**	**72209311**	**11/10/2024 – 24/10/2024**

11. 24/10/2024 Medado principal que llama el servicio web y ejecuta cada uno de los métodos asociado como son: consultar las credenciales, traer el token autorizado, llamar al servicio y pasarle en el body el XML de la factura, además verificar si la emisión fue exitosa. Dicha respuesta la rerializamos y finalmente la guardamos en la base de datos.