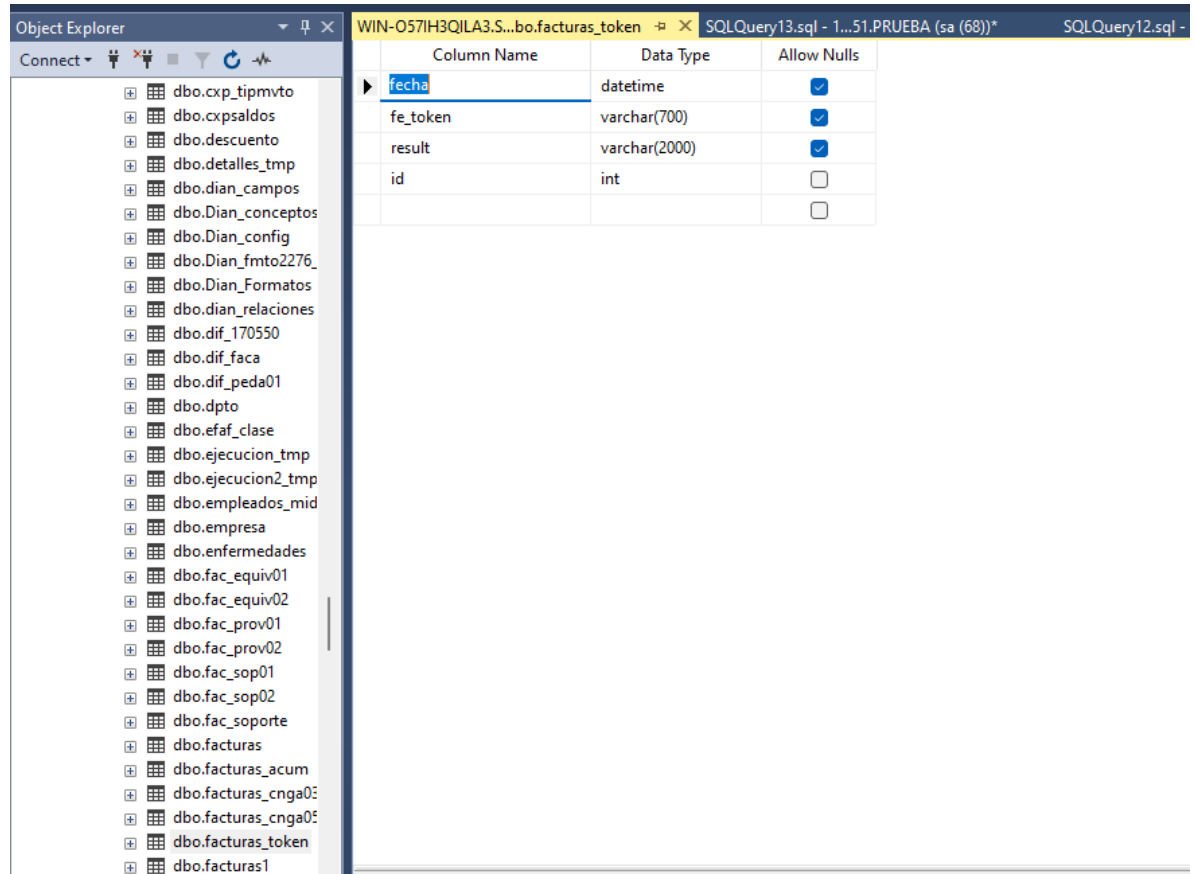


Evidencias Bitácora No 2			
Ficha	Aprendiz	cedula	periodo
2675810	RODNEY ZAPATA PALACIO	72209311	11/10/2024 – 24/10/2024

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

```

10 namespace TokenV03NetCore
11 {
12     internal class Program
13     {
14         public static string respuesta_token="";
15         public static string token="";
16     }
17     2 referencias
18     public class WebServiceClient
19     {
20         1 referencia
21         public async Task CallWebServiceAsync()
22         {
23             // Crear una instancia de HttpClient
24             using var client = new HttpClient();
25
26             // Ambiente de prueba
27             client.DefaultRequestHeaders.Add("X-Who", "dc1986b27eee4bb29a9b8196dd6df4cac"); // Reemplaza el valor con el adecuado
28
29             // URI del servicio al que quieres hacer la petición
30             var uri = "https://plcolabbeta.azure-api.net/Auth/Login"; // Cambia esta URL por la de tu API
31
32             // Crear el objeto JSON que deseas enviar
33             var jsonBody = new
34             {
35                 u = "890116102", // Cambia los valores por los reales si es necesario
36                 p = "abc123$!"
37             };
38         }
39     }

```

- 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#.

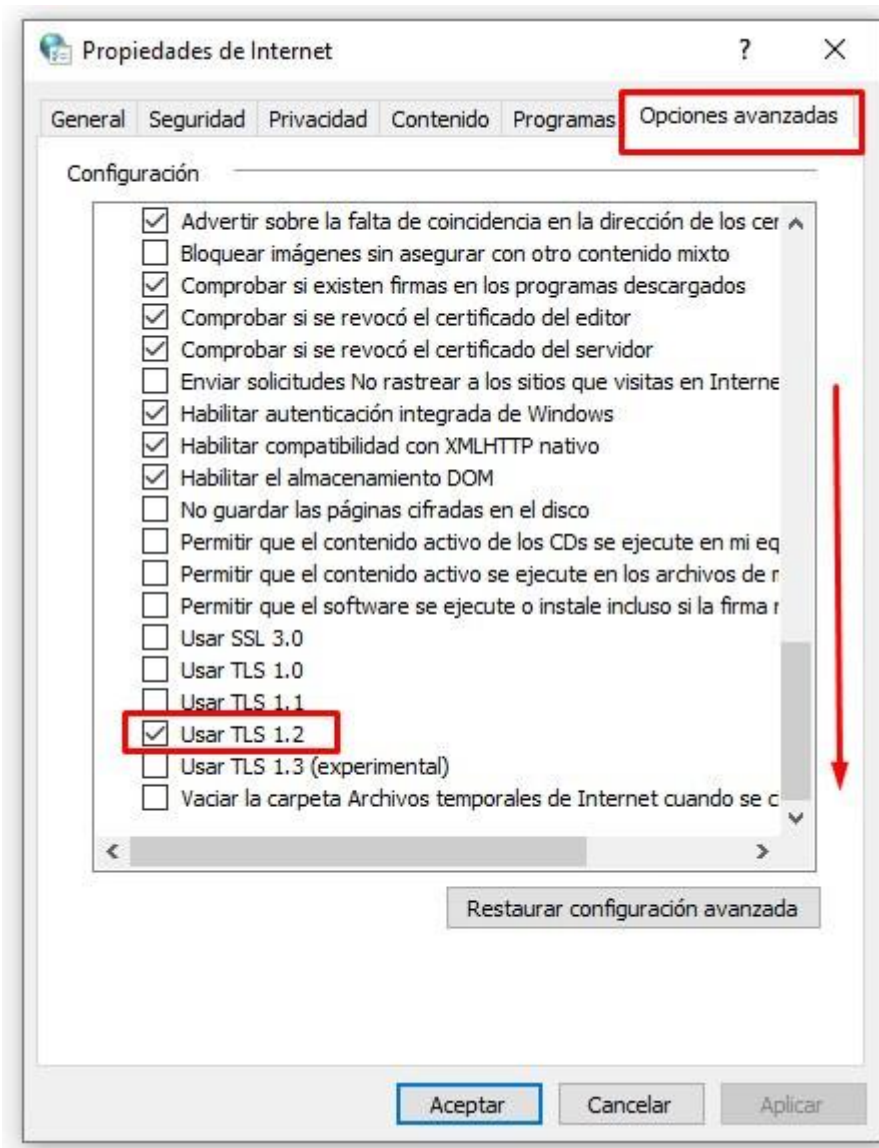
```

98
99 1 referencia
100 public static void guardarBD()
101 {
102     //guardamos el resultado de la consulta
103     SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01.;User ID=sa;");
104     SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.6;Initial Catalog=SERVIPARAMO;Persist Security Info=True;Password=Clave01.;User ID=sa;");
105     conexion1.Open();
106     string cadena_respuesta = "update com_cxca01 set fe_respuesta_cufer=@respuesta_cufer, fe_uid=@uid, fe_codigo_cufer=@cufer, fe_token=@token, fe_fecha=@fecha";
107     string cadena_respuesta = "insert facturas_token(fecha,fe_token,result) values(getdate(), @token, @respuesta_token)";
108
109     SqlCommand comando = new SqlCommand(cadena_respuesta, conexion1);
110     comando.Parameters.AddWithValue("@token", token);
111     comando.Parameters.AddWithValue("@respuesta_token", respuesta_token);
112
113     if (comando.ExecuteNonQuery() > 0)
114         Console.WriteLine("ok update sql cufer");
115     else
116         Console.WriteLine("error update sql cufer");
117
118     conexion1.Close();
119 }
120
121 0 referencias
122 public static async Task Main(string[] args)
123 {
124     WebServiceClient client = new WebServiceClient();
125     await client.CallWebServiceAsync(); // Llamar el método que consume la API
126     Console.ReadKey();
127 }

```

- 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



- 16/10/2024 Clase ApiResponse 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

```

98
99 2 referencias
100 public class ApiResponse
101 {
102     [JsonPropertyName("isSuccess")]
103     public bool IsSuccess { get; set; }
104
105     [JsonPropertyName("requestId")]
106     public string RequestId { get; set; }
107
108     [JsonPropertyName("shortRequestId")]
109     public string ShortRequestId { get; set; }
110
111     [JsonPropertyName("fileNameInteroperability")]
112     public string FileNameInteroperability { get; set; }
113
114     [JsonPropertyName("UUID")]
115     public string UUID { get; set; }
116
117     [JsonPropertyName("QR")]
118     public string QR { get; set; }
119
120     [JsonPropertyName("documentNumber")]
121     public string DocumentNumber { get; set; }
122
123     [JsonPropertyName("LDF")]

```

6. 17/10/2024 Consulta de credenciales en Base de Datos

```

158
159 0 referencias
160 public static void ConsultBD()
161 {
162     //Consultamo el procedimiento almacenado
163     string cadenaConexion2 = "Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01;User ID=sa;Connect Timeout=30";
164     //string cadenaConexion2 = "Data Source=192.168.50.6;Initial Catalog=SERVIPARAMO;Persist Security Info=True;Password=Clave01;User ID=sa;Connect Timeout=30";
165     SqlConnection con2 = new SqlConnection(cadenaConexion2);
166     con2.Open();
167     string cad_select2 = "exec _XML_factura_electronica " + "" + no_factura + "";
168
169     SqlCommand com2 = new SqlCommand(cad_select2, con2);
170     using (SqlDataReader reader2 = com2.ExecuteReader())
171     {
172         while (reader2.Read())
173         {
174             //Console.WriteLine(reader["xml_factura"].ToString());
175             token = reader2["token"].ToString();
176             tipoServicio = reader2["tipo"].ToString();
177             // Console.WriteLine(reader2["token"].ToString());
178             result_xml = reader2["fe_xml_factura"].ToString();
179             keycontrol = reader2["keycontrol"].ToString();
180             x_who = reader2["x_who"].ToString();
181             x_uri = reader2["x_uri"].ToString();
182
183             //Console.ReadLine();
184         }
185     }
186     con2.Close();
187     Console.WriteLine("Variables consultadas");
188     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**.

```

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

```

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

```

246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
1 referencia
public static void deserializarJson()
{
    string nombJson = @"C:\serviparamo\datos\error_" + no_factura + ".txt";

    // Leemos archivo plano guardado anteriormente para separar el cufer, token,
    //string path = @"D:\ServiparamoDSAv03\product.json";
    using (StreamReader jsonStream = File.OpenText(nombJson))
    {
        var json2 = jsonStream.ReadToEnd();
        string jsonResponse=json2.Trim();
        Console.WriteLine(json2);
        try
        {
            // Deserializar el JSON
            ApiResponse response = JsonSerializer.Deserialize<ApiResponse>(jsonResponse);

            // Acceder a los datos deserializados
            Console.WriteLine($"Éxito: {response.IsSuccess}");
            Console.WriteLine($"*ID de solicitud: {response.RequestId}");
            Console.WriteLine($"*QR: {response.QR}");

            Console.WriteLine($"*URL : {response.URL}");
            //url = response.URL;
            url = response.URL;
            uuid = response.UUID;
            cufer = response.RequestId;

            // Guardamo el resultado en la base de datos
            guardarBD();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"*Error al deserializar: {ex.Message}");
        }
    }
}

```

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:

```

221 public static void guardarBD()
222 {
223     //guardamos el resultado de la consulta
224     SqlConnection conexion1 = new SqlConnection("Data Source=192.168.50.6;Initial Catalog=PRUEBA;Persist Security Info=True;Password=Clave01.;User ID=sa");
225     conexion1.Open();
226     string cadena_respuesta = "update com_cxca01 set fe_respuesta_cufer=@respuesta_cufer, fe_uuid=@uuid, fe_codigo_cufer=@cufer, fe_token=@token, fe_rej";
227
228
229     SqlCommand comando = new SqlCommand(cadena_respuesta, conexion1);
230     comando.Parameters.AddWithValue("@respuesta_cufer", respuesta_cufer);
231     comando.Parameters.AddWithValue("@uuid", uuid);
232     comando.Parameters.AddWithValue("@cufer", cufer);
233     comando.Parameters.AddWithValue("@token", token);
234     comando.Parameters.AddWithValue("@url", url);
235     comando.Parameters.AddWithValue("@numfac", no_factura);
236
237     if (comando.ExecuteNonQuery() > 0)
238     {
239         Console.WriteLine("ok update sql cufer ");
240     }
241     else
242     {
243         Console.WriteLine("error update sql cufer");
244     }
245     conexion1.Close();
246 }

```

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

```

33 public async Task CallWebServiceAsync()
34 {
35     // Crear una instancia de HttpClient
36     using var client = new HttpClient();
37
38     // Ambiente de prueba
39
40     client.DefaultRequestHeaders.Add("X-Who", x_who);
41     client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);
42     client.DefaultRequestHeaders.Add("X-REF-DOCUMENTTYPE", "FACTURA-UBL");
43     client.DefaultRequestHeaders.Add("X-KEYCONTROL", keycontrol);
44     client.DefaultRequestHeaders.Add("X-REF-TEMPLATE", tipoServicio);
45
46     // URI del servicio al que quieres hacer la petición
47     //var uri = "https://plcolabbeta.azure-api.net/Auth/Login"; // Cambia esta URL por la de tu API
48     var uri = x_uri;
49
50     //Cuerpo del XML lo traigo desde un campo de la base de datos
51     string xmlBody = result_xml;
52
53
54     // Convertir el cuerpo a bytes
55     var content = new StringContent(xmlBody, Encoding.UTF8, "application/xml");
56
57     try
58     {
59         // Hacer la solicitud POST enviando el contenido
60         HttpResponseMessage response = await client.PostAsync(uri, content);
61
62         // Leer y mostrar el contenido de la respuesta
63         string result = await response.Content.ReadAsStringAsync();
64         Console.WriteLine("Respuesta: " + result);
65
66         // Verificar el estado de la respuesta
67         if (response.IsSuccessStatusCode)
68         {
69             Console.WriteLine("Solicitud exitosa.");
70             respuesta_cufer = result;
71             token =
72             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 reerializamos y finalmente la guardamos en la base de datos.

```

24
25 //variables que retorna desde el servicio
26 public static string uuid = "";
27 public static string cufer = "";
28 public static string url = "";
29
30
31 0 referencias
32 public class WebServiceClient
33 {
34     0 referencias
35     public async Task CallWebServiceAsync()
36 }
37
38
39
40
41 2 referencias
42 public class ApiResponse
43 {
44 }
45
46
47 0 referencias
48 public static void ConsutBD()
49 {
50 }
51
52 2 referencias
53 public static void guardarArchivoResult()
54 {
55 }
56
57 1 referencia
58 public static void guardarBD()
59 {
60 }
61
62 1 referencia
63 public static void desealizarJson()
64 {
65 }
66
67 0 referencias
68 public static async Task Main(string[] args)
69 {
70     Console.WriteLine("Cantidad de argumentos: {0}", args.Length);
71     foreach (string numfac in args)
72     {
73         Console.WriteLine("fact: {0}", numfac);
74         no_factura = numfac;
75     }
76     no_factura = "SETT122699";
77     desealizarJson();
78 }
79
80

```