



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	7	26/12/2024-10/01/2025

Nombre del jefe inmediato/Responsable	Teléfono de contacto	Correo electrónico
JAMES CALVO	312 681 1324	<a href="mailto:jcalvo@seviparamo.co">jcalvo@seviparamo.co</a>

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		APORTA 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	<a href="mailto:rodney.zapata@soy.sena.edu.co">rodney.zapata@soy.sena.edu.co</a>


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

DESCRIPCIÓN DE LA ACTIVIDAD <i>(Ingrese cuantas filas sean necesarias)</i>	FECHA INICIO	FECHA FIN	EVIDENCIA DE CUMPLIMIENTO	OBSERVACIONES, INASISTENCIAS Y/O DIFICULTADES PRESENTADAS
Creación de un API REST en C# que se encargue del CRUD, empezamos con el Modelo de Clientes. El cual mapea las tablas y sus relaciones dentro de la aplicación, permitiendo manipular los datos usando código en lugar de SQL directo.	26-dic	27-dic	La evidencia con captura de pantallas en formato PDF	
Creamos el DBContext de Entity Framework Core (EF Core) La cual actúa como puente entre la aplicación y la base de datos. Se utiliza para: Gestionar la conexión a la base de datos: Se encarga de abrir y cerrar la conexión automáticamente. Mapear entidades a tablas: Permite definir cómo las clases de C# (entidades) se traducen en tablas de la base de datos. Ejecutar consultas LINQ: Facilita la recuperación de datos mediante LINQ to Entities.	28-dic	30-dic	La evidencia con captura de pantallas en formato PDF	
Creamos el controlador de Clientes. Su función principal es manejar las solicitudes HTTP, procesar la lógica de negocio y devolver una respuesta, generalmente en formato JSON o HTML.	02-ene	03-ene	La evidencia con captura de pantallas en formato PDF	
Controlador de clientes, que consulta un cliente por Id.	07-ene	08-ene	La evidencia con captura de pantallas en formato PDF	
Controlador de clientes, que modifica un cliente por Id.	09-ene	10-ene	La evidencia con captura de pantallas en formato PDF	

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

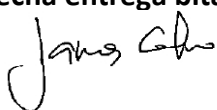
RODNEY ZAPATA PALACIO

Nombre del Aprendiz



Firma del aprendiz

Fecha entrega bitácora



Firma del jefe inmediato (Si es del caso)

Nombre del Instructor de Seguimiento

Firma de instructor de seguimiento

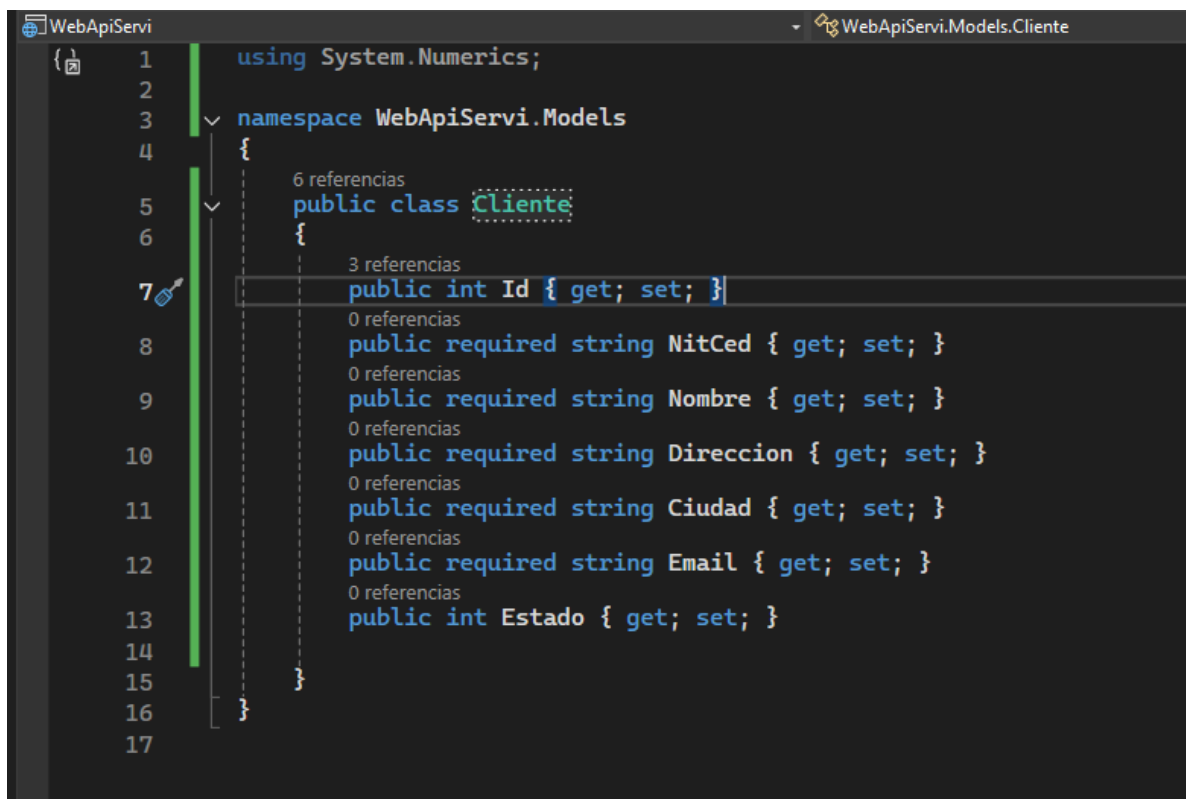
MARTHA ISABEL ORDOÑEZ ARDILA

Nombre del Instructor de Seguimiento

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 7			
Ficha	Aprendiz	cedula	periodo
2675810	RODNEY ZAPATA PALACIO	72209311	26/12/2024 – 10/01/2025

1. Fecha: 26/12/2024. Creación de un API REST en C# que se encargue del CRUD, empezamos con el Modelo de Clientes.  
El cual mapea las tablas y sus relaciones dentro de la aplicación, permitiendo manipular los datos usando código en lugar de SQL directo.



```

1  using System.Numerics;
2
3  namespace WebApiServi.Models
4  {
5      6 referencias
6      public class Cliente
7      {
8          3 referencias
9          public int Id { get; set; }
10         0 referencias
11         public required string NitCed { get; set; }
12         0 referencias
13         public required string Nombre { get; set; }
14         0 referencias
15         public required string Direccion { get; set; }
16         0 referencias
17         public required string Ciudad { get; set; }
18         0 referencias
19         public required string Email { get; set; }
20         0 referencias
21         public int Estado { get; set; }
22     }
23 }

```

2. 28/12/2024 Creamos el DbContext de Entity Framework Core (EF Core)  
La cual actúa como puente entre la aplicación y la base de datos. Se utiliza para:  
  
Gestionar la conexión a la base de datos: Se encarga de abrir y cerrar la conexión automáticamente.

Evidencias Bitácora No 7			
Ficha	Aprendiz	cedula	periodo
2675810	RODNEY ZAPATA PALACIO	72209311	26/12/2024 – 10/01/2025

Mapear entidades a tablas: Permite definir cómo las clases de C# (entidades) se traducen en tablas de la base de datos.

Ejecutar consultas LINQ: Facilita la recuperación de datos mediante LINQ to Entities.

```

1  using Microsoft.EntityFrameworkCore;
2  using WebApiServi.Models;
3
4  namespace WebApiServi.Context
5  {
6      public class AppDbContext: DbContext
7      {
8          public AppDbContext(DbContextOptions<AppDbContext> options):base(options)
9          {
10             }
11
12         public DbSet<Cliente> Clientes { get; set; }
13         public DbSet<FacturaElectronica> FacturasElectronicas { get; set; }
14
15         // Método para ejecutar el procedimiento almacenado
16
17
18         public async Task<List<FacturaElectronica>> ObtenerFacturaElectronica(int numfac)
19         {
20             return await this.Set<FacturaElectronica>()
21                 .FromSqlRaw("EXEC _XML_factura_electronica @p0", numfac)
22                 .ToListAsync();
23         }
24     }
25 }
26
27

```

Administrar transacciones: Garantiza la integridad de los datos mediante transacciones.

Aplicar migraciones: Permite crear y actualizar la estructura de la base de datos mediante código.

- 02/01/2025 al 03/01/2025. Creamos el controlador de Clientes. Su función principal es manejar las solicitudes HTTP, procesar la lógica de negocio y devolver una respuesta, generalmente en formato JSON o HTML.

Evidencias Bitácora No 7			
Ficha	Aprendiz	cedula	periodo
2675810	RODNEY ZAPATA PALACIO	72209311	26/12/2024 – 10/01/2025

```

WebApiServi
WebApiServi.Controllers.CientesController

7      using Microsoft.EntityFrameworkCore;
8      using WebApiServi.Context;
9      using WebApiServi.Models;
10
11     namespace WebApiServi.Controllers
12     {
13         [Route("api/[controller]")]
14         [ApiController]
15         public class CientesController : ControllerBase
16         {
17             private readonly AppDbContext _context;
18
19             public CientesController(AppDbContext context)
20             {
21                 _context = context;
22             }
23
24             // GET: api/Cientes
25             [HttpGet]
26             public async Task<ActionResult<IEnumerable<Cliente>>> GetCientes()
27             {
28                 return await _context.Cientes.ToListAsync();
29             }
30

```

4. 07/01/2025 al 08/01/2025 Controlador de clientes, que consulta un cliente por Id.

```

30
31     // GET: api/Cientes/5
32     [HttpGet("{id}")]
33     public async Task<ActionResult<Cliente>> GetCliente(int id)
34     {
35         var cliente = await _context.Cientes.FindAsync(id);
36
37         if (cliente == null)
38         {
39             return NotFound();
40         }
41
42         return cliente;
43     }
44

```

Evidencias Bitácora No 7			
Ficha	Aprendiz	cedula	periodo
2675810	RODNEY ZAPATA PALACIO	72209311	26/12/2024 – 10/01/2025

5. 09/01/2025 al 10/01/2025 Controlador de clientes, que modifica un cliente por Id.

```

45 // PUT: api/Clientes/5
46 // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
47 [HttpPut("{id}")]
48 0 referencias
49 public async Task<IActionResult> PutCliente(int id, Cliente cliente)
50 {
51     if (id != cliente.Id)
52     {
53         return BadRequest();
54     }
55     _context.Entry(cliente).State = EntityState.Modified;
56
57     try
58     {
59         await _context.SaveChangesAsync();
60     }
61     catch (DbUpdateConcurrencyException)
62     {
63         if (!ClienteExists(id))
64         {
65             return NotFound();
66         }
67         else
68         {
69             throw;
70         }
71     }
72

```