

APLICACIÓN WEB

Johanna Milena García Bautista

Programación avanzada en Lenguaje de programación I - .NET

Análisis y Desarrollo de Sistemas de la Información

SENA

INTRODUCCION

Desarrollar App Web es fundamental para la solución de problemas de un cliente en cuanto a la navegabilidad y modelo de negocio que se necesite, desarrollando varias etapas desde la maquetación en HTML donde se vera todo el diseño de la página y su navegabilidad.

Como desarrollador de Software en .Net se escribe el código interno de la App, el cual dará viabilidad a toda la navegación y función, en base al modelo Vista Controlador permite acceso a las paginas y el Crud de la App el cual enlaza una base de Datos, por la que el usuario utiliza el controlador para manipular los datos, esto lo notifica nuevamente al controlador, lo actualiza y refleja unas vistas al usuario.

Este método hace toda la lógica de la navegabilidad de la App y la funcionalidad de la misma, dando solución a los requerimientos del cliente, en este caso una empresa que brinda productos para la salud y bienestar de las personas adicional dando la oportunidad de emprender un negocio; en la App se da la posibilidad de mostrar lo que es la empresa, los productos y la oficina virtual que dará la posibilidad de manejar el negocio desde esta App, teniendo toda la Data de los clientes o prospectos registrados.

OBJETIVOS

- Desarrollar maqueta y diseño en HTML y entender el orden del lenguaje para lograr dicho diseño.
- Desarrollar el Modelo Vista Controlador para la navegabilidad de la App en .Net. con una Base de Datos enlazada, la cual se puede verificar, modificar y así mostrar un resultado al usuario por medio de un Crud básico.
- Analizar la navegabilidad de la App y llevarla a la realidad.
- Llevar todo el conocimiento enfocado a un proyecto, en este caso una pagina Web de un cliente específico y dar solución a su problema.
- Tener acceso a los datos por medio de lo Loguin y así tener la seguridad de la Data.

MOCKUPS

- **Página Inicio**



Header: Se define el dentro del <nav> y la imagen del logo, en la lista del menú por el cual se va a navegar (se repite en todas las paginas).

```
<header class="estiloheader">
  <!-- CODIGO DEL ENCABEZADO -->
  <div>
    <ul class="estilo1">
      <div class="contenedorimg">
        
      </div>
      <li class="estilo1"> <a class="estilo1a" asp-controller="Inicio" asp-action="Index"> Inicio </a></li>
      <li class="estilo1"> <a class="estilo1a" asp-controller="Nuestros" asp-action="Index"> Quienes Somos </a></li>
      <li class="estilo1"> <a class="estilo1a" asp-controller="Productos" asp-action="Index"> Nuestros Productos </a></li>
      <li class="estilo1"> <a class="estilo1a" asp-controller="Contacto" asp-action="Index"> Contacto </a></li>
    </ul>
  </div>
</header>
```

Main: Un contenedor el cual se divide dos contenedores, uno para la imagen y otro para el texto, se tiene en cuenta los estilos definidos para los contenedores.

```
<div class="estilo1">
  <!-- CODIGO DEL CONTENIDO -->
  <div class="estilo1"> Productos NIKKEN Colombia</div>
  <div class="estilo1a">
    <div class="contenedorimg">
      <div class="contenedorimgprincipal">
        
      </div>
      <div class="contenedor2">
        <div class="contenedor2principal">
          <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Suscipit ullam fugiat nihil reiciendis vel eaque. Doloremque eligendi officia quod odio laboriosam perferendis rem commodi maxime vel quos, sequi, qui expedita.
          </p>
          <p>
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Suscipit ullam fugiat nihil reiciendis vel eaque. Doloremque eligendi officia quod odio laboriosam perferendis rem commodi maxime vel quos, sequi, qui expedita.
          </p>
        </div>
      </div>
    </div>
  </div>
</div>
```

Footer: Se define el estilo y se crea contenedor para la imagen y un pequeño texto (se repite en todas las paginas).

```

<footer class="estiloFooter">
  <!--CODIGO DEL PIE-->
  <p class="estiloTextFooter"> [Follow link](file:///c:/Johanna/ADSI/III TRIMESTRE/NET
  II/Talleres/AppMockup/assets/images/logo Nikken2.png) (ctrl + click)
  <div class="contenedorInfluencia">
    
    <small class="estiloTextFooter"> COD NIKKEN Colombia 16541703</small>
  </div>
</footer>

```

- **Pagina Nosotros**



Main: Se define <h1> como el titulo de la página, un contenedor el cual se divide dos contenedores, uno para la imagen y otro para el texto, se tiene en cuenta los estilos definidos para los contenedores.

```

<h1 class="estiloH1"> Nuestra Empresa </h1>

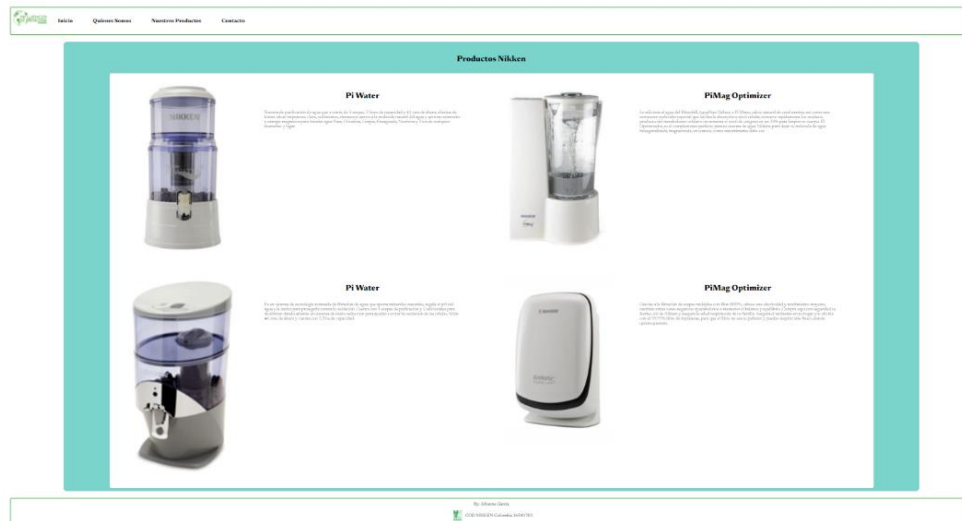
<section class="estiloSection">
  <div class="contenedor40">
    <h3 class="estiloH1"> Comunidad de Bienestar Nikken </h3>

    <p class="estiloTextos">
      Nikken se ha dedicado a pensar en tu bienestar y en cómo mejorar tu calidad de vida. Desde 1975, te ofrece el balance natural para tu cuerpo, tu familia y entorno, equilibrando los pilares de la salud: agua, aire, descanso y nutrición. Piensa en prevención y tu organismo y tu familia te lo agradecerán. Nuestros sistemas de purificación de agua, limpian y dan a cada molécula su estructura perfecta. Los sistemas de aire, dejan el 99.9% del aire puro, los sistemas de sueño te dan un descanso profundo y reparador; así, cada uno de los productos Nikken son claves para mejorar la calidad de vida de los que más amas.
    </p>
    <br>
    <p class="estiloTextos">
      Nuestros sistemas de purificación de agua, limpian y dan a cada molécula su estructura perfecta. Los sistemas de aire, dejan el 99.9% del aire puro, los sistemas de sueño te dan un descanso profundo y reparador; así, cada uno de los productos Nikken son claves para mejorar la calidad de vida de los que más amas.
    </p>
  </div>

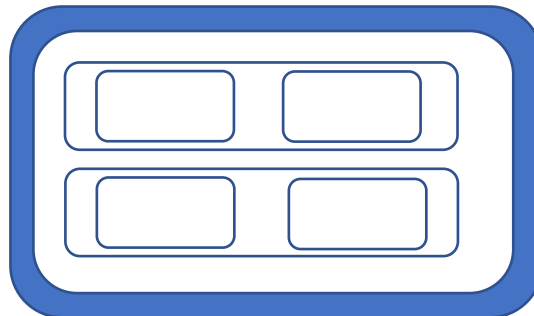
  <div class="contenedor50">
    <div class="contenedorImgPrincipal">
      
    </div>
  </div>
</section>

```

- **Pagina Productos**



Main: se definió un contenedor de 100% llamado contenedorProducto, dentro un contenedor más pequeño llamado contenedorProductos, el cual se colocó un contenedor de 40% para la imagen u otro de 70% para el texto, estos dos están dentro de un contenedor y se unieron dos de estos con flex.



- **Pagina Contacto**



Main: se definió un contenedor con <h1> el cual genera el título y dentro un contenedor con el texto.

CODIGO .NET

1. HTML

Lenguaje basado en etiquetas para desarrollo de paginas web.

<HTML>

<BODY>

<HEATHER>

<nav>

 => Lista

 => Items

<a...> => Permite hacer enlace href

<MAIN>

<h1> => Titulo

<Section> => Seccion

<div> => Contenedores => conectada => Imagen

<p> => Texto

<FOOTER>

<div> => y <Small>

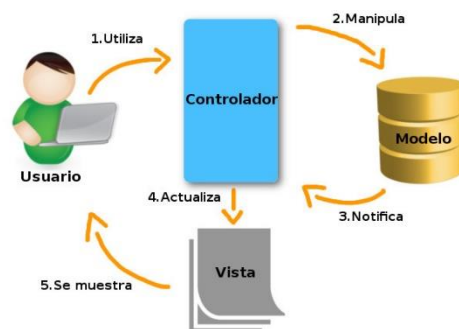
Nombre <h2> o <h3>

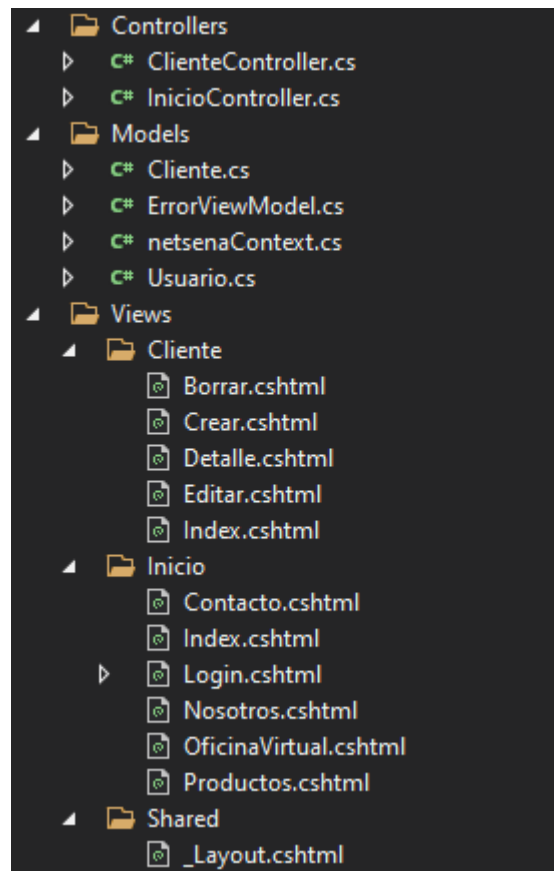
</BODY>

</HTML>

2. Modelo vista controlador MVC .NET

Modelos de tres capas que hace referencia a:





3. Visual Studio

- 3.1. Se crea una App Web basada en el modelo que se realizó anteriormente, creando un nuevo proyecto ASP.NET core web App (MVC).
Se encuentran las carpetas Controller, Models y Views.

En Views se encuentra Home y Shared

Archivo principal program.cs: Se invoca el método *IHostBuilder* y este invoca la clase *Startup* que es el más importante ya que corre los servicios y la configuración.

- 3.2. En la carpeta Views, está la carpeta Inicio; aquí se crean las vistas Index, Contacto, Nosotros, Productos y Oficina Virtual; dentro de cada una se copia el código HTML anteriormente establecido.

Se da un nombre a cada vista <h1>...</h1> y se asigna un título de página.

```
@{  
    ViewData["Title"] = "Nosotros";  
    Layout = "~/Views/Shared/_Layout.cshtml";  
}  
  
<h1 class="estiloH1"> Nuestra Empresa </h1>
```

3.3. En la carpeta Controllers se crea InicioControllers, aquí se colocan los métodos de cada una de las vistas.

```
public class InicioController : Controller  
{  
    public IActionResult Index()  
    {  
        return View();  
    }  
  
    public IActionResult Contacto()  
    {  
        return View();  
    }  
  
    public IActionResult Nosotros()  
    {  
        return View();  
    }  
  
    public IActionResult Productos()  
    {  
        return View();  
    }  
  
    [Authorize]  
    public IActionResult OficinaVirtual()  
    {  
        return View();  
    }  
}
```

3.4. En Layout se copia el <header> y el < footer> modifica el código con referente al de la maqueta en:

```
asp-controller="Inicio" asp-action="Index"> Inicio
```

3.5. En CSS se importa un ítem existente que son los estilos creando una carpeta que se llame imágenes, en estas se agrega la ~/ para su visualización y se reemplaza todo el Header y Footer de .Net por el de Visual Code en la página de Layout.

4. Base de Datos

4.1 Se crea la Base de Datos en MySQL.

+ Opciones				codigo	nombre	correo
<input type="checkbox"/>				1010	julian pedroza	julipedroza@misena.edu.co
<input type="checkbox"/>				1011	nairobby rojas	nrojas@misena.edu.co
<input type="checkbox"/>				1012	teresa carrillo	terecar1010@misena.edu.co
<input type="checkbox"/>				1013	johanna garcia	joha@misena.edu.co
<input type="checkbox"/>				1015	maria memanda	marifer@sena.edu.co
<input type="checkbox"/>				1016	maritza rodriguez	maritza@sena.edu.co
<input type="checkbox"/>				1017	naydu garzon	naydug@sena.edu.co
<input type="checkbox"/>				1018	cristian garcia	crisgarcia@sena.edu.co
<input type="checkbox"/>				1019	julian catiblanco	julienc@sena.edu.co
<input type="checkbox"/>				1020	andres gomez	agomez@sena.edu.co
<input type="checkbox"/>				1021	berta mendez	bertamendez@sena.edu.co

4.2 En .Net dar click derecho sobre el nombre de la App o en el menú opción Tools se busca la opción Nuget Package Manager, se busca:

Microsoft. EntityFrameworkCore.Desing

Versión 3.1.14

Esto equivale a instalar Entity Framework.

Por consola instalar Pomelo en CMD:

```
dotnet add package Microsoft. EntityFrameworkCore.Tools --version 3.1.13
```

En AppWeb 588 aparece el código que muestra todo lo instalado y después de los Package Reference...

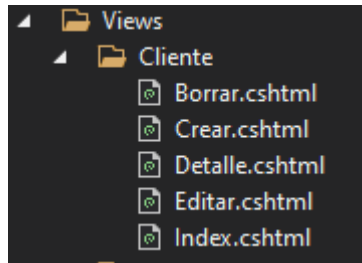
```
<DotNetCliToolReference Include="Microsoft.EntityFrameworkCore.Tools.DotNet"
Version="2.0.1" />
</ItemGroup>
```

4.3. En consola CMD colocar **dotnet restore** => Unicornio y luego **dotnet ef**.

Se verifica en .Net la carpeta ModeloCliente.CS y la variable de contexto que es NetSenaContext.cs.

5. CRUD

En las vistas, carpeta Cliente se crea Index, Crear, Detalle, Borrar.



5.1. En Index se crea la tabla, la cual la recorre un foreach.

```
<div class="contenedor80">
  <table class="table table-striped">
    <thead>
      <tr>
        <th> codigo </th>
        <th> Nombre </th>
        <th> correo </th>
        <th> </th>
      </tr>
    </thead>
    <tbody>
      <foreach (var myCliente in Model)>
        <tr>
          <td> @myCliente.Codigo </td>
          <td> @myCliente.Nombre </td>
          <td> @myCliente.Correo </td>
          <td>
            @Html.ActionLink("Editar", "Editar", new { id = myCliente.Codigo })
            @Html.ActionLink("Detalle", "Detalle", new { id = myCliente.Codigo })
            @Html.ActionLink("Borrar", "Borrar", new { id = myCliente.Codigo })
          </td>
        </tr>
      </foreach>
    </tbody>
  </table>
</div>
```

5.2. Se crea un modelo controlador para cliente ClienteController.cs; dentro de este controlador se enlaza la base de datos:

```
using AppWeb588.Models;
```

y se agregan los métodos Listar, Editar, Crear, Borrar.

5.3. Crear formulario con Label e Input.

```
<div class="contenedor50">
  <form asp-action="crear">
    <div>
      <label asp-for="Codigo" class="control-label"></label>
      <input asp-for="Codigo" class="form-control" required />
      <span class="text-danger"> @ViewData["msj"] </span>
    </div>
    <div>
      <label asp-for="Nombre" class="control-label"></label>
      <input asp-for="Nombre" class="form-control" required />
    </div>
    <div>
      <label asp-for="Correo" class="control-label"></label>
      <input asp-for="Correo" type="email" class="form-control" required />
    </div>
    <div>
      <input type="submit" class="estiloBoton" value="Guardar" />
    </div>
  </form>
</div>
```

6. Pagar

En .Net se instala: X.PagedList.Mvc.Core versión 8.0.7

6.1. En el método Listar de ClienteController.cs se modifica el código por:

```
public IActionResult Index(int ? page)
{
    var db = new netsenaContext();
    var pageNumber = page ?? 1;
    int pageSize = 6;
    var Clientes = db.Cliente.ToPagedList(pageNumber, pageSize);
    return View(Clientes);
}
```

6.2. En Index se definen los siguiente Using:

```
@using X.PagedList.Mvc.Core;
@using X.PagedList;
@model IEnumerable<AppWeb588.Models.Cliente>
```

6.3. Se define la navegación después de la tabla con JavaScript


```
<!-- definimos los objetos de navegacion-->
@Html.PagedListPager((IPagedList)Model, page => Url.Action("index", new { page = page }),
    new X.PagedList.Mvc.Core.Common.PagedListRenderOptions
    {
        DisplayItemSliceAndTotal = false,
        ContainerDivClasses = new[] { "pagination" },
        LiElementClasses = new[] { "estiloLi" },
        PageClasses = new[] { "estiloLink" },
    })

@section Scripts {
    <script>
        $(document).ready(function () {
            $('ul.pagination > li.disabled > a').addClass('page-link');
        })
    </script>
```

7. Login

Middleware: Software que asiste a una App para interactuar o comunicarse con otras App o paquetes de redes. Se encarga de tareas de gestión de datos, servicios de la App, mensajería, autenticación y gestión de la API.

7.1. Crear la tabla en la Base de Datos

+ Opciones					id	nombre	apodo	contrasena
<input type="checkbox"/>				Borrar	1010	bruce wayne	batman	robin
<input type="checkbox"/>				Borrar	1020	clark kent	super	luisa

7.2. Agregar clase en Models llamada Usuario

```
namespace AppWeb588.Models
{
    public partial class Usuario
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string Apodo { get; set; }
        public string Contraseña { get; set; }
    }
}
```

7.3. Se modifica la variable contexto en NetsenaContext.

```
public virtual DbSet<Usuario> Usuario { get; set; }
```

7.4. Se genera un modelo de datos en Entity colocando todos los atributos de la tabla.

```
modelBuilder.Entity<Usuario>(entity =>
{
    entity.HasKey(e => e.Id)
        .HasName("PRIMARY");

    entity.ToTable("Usuario");

    entity.Property(e => e.Id)
        .HasColumnName("id")
        .HasColumnType("int(11)");

    entity.Property(e => e.Nombre)
        .HasColumnName("nombre")
        .HasColumnType("varchar(50)")
        .HasCharSet("utf8")
        .HasCollation("utf8_general_ci");

    entity.Property(e => e.Apodo)
        .HasColumnName("apodo")
        .HasColumnType("varchar(25)")
        .HasCharSet("utf8")
        .HasCollation("utf8_general_ci");

    entity.Property(e => e.Contrasena)
        .HasColumnName("contrasena")
        .HasColumnType("varchar(10)")
        .HasCharSet("utf8")
        .HasCollation("utf8_general_ci");
});

OnModelCreatingPartial(modelBuilder);
}
```

7.5. Si se quiere proteger de la vista en ClienteController.cs, para proteger antes del método se coloca:

[Authorize]

Y se agrega librería en la parte superior.

7.6. StartUp, se agregan los siguientes using:

```
using Microsoft.AspNetCore.Authentication;
```

using Microsoft.AspNetCore.Authentication.Cookies;

En la configuración de servicios

```
0 referencias
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    // configurar el metodo de autenticacion
    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
        .AddCookie(option =>
        {
            option.LoginPath = "/Login";
        });
}
```

7.7. En la vista Login.cshtml

```
<form action = "Login?ReturnUrl=@WebUtility.UrlEncode(returnUrl)" method="post">

    <div class="mb-3">
        <label class="form-label">Usuario</label>
        <input type="text" name="username" class="form-control" required>
    </div>
    <div class="mb-3">
        <label class="form-label">Contraseña</label>
        <input type="password" name="password" class="form-control" required>
    </div>

    <button type="submit" class="btn btn-primary">Iniciar Sesión</button>

</form>
```

7.8. se creo en la nueva vista tienda virtual para que el usuario pueda entrar.

```
[HttpPost("Login")]
0 referencias
public async Task<IActionResult> Validar(string username, string password, string returnUrl)
{
    //lectura de la Base de Datos y validacion

    if (username == "batman" && password == "robin")
    {
        var claims = new List<Claim>();
        claims.Add(new Claim("username", username));
        claims.Add(new Claim(ClaimTypes.NameIdentifier, username));
        claims.Add(new Claim(ClaimTypes.Name, "Bruce wine"));
        var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);
        var claimsPrincipal = new ClaimsPrincipal(claimsIdentity);
        await HttpContext.SignInAsync(claimsPrincipal);
        return Redirect(returnUrl);
    }
    else
    {
        ViewData["ReturnUrl"] = returnUrl;
        TempData["Error"] = "El Usuario o la Contraseña no son validos";
        return View("Login");
    }
}

[Authorize]
0 referencias
public async Task<IActionResult> Logout()
{
    await HttpContext.SignOutAsync();
    return Redirect("/");
}
```

CONCLUSION

Se logro poner en practica los conocimientos adquiridos por el docente y llevarlos a la práctica desarrollando una App Web, con un diseño, Crud y navegabilidad.