UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA INFORMACIÓN

DOCUMENTO TECNICO - ANTRO

ALUMNOS:

- DIAZ BERNAL EMILIO YAEL
- LLAMAS VALLE JOSÉ SEBASTIÁN
- SANTACRUZ DE LUNA JOSUE FERNANDO

CARRERA: LICENCIATURA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES

SEMESTRE: 6°

MATERIA: DESARROLLO WEB

DOCENTE: MARGARITA MONDRAGON ARELLANO

FECHA: 24 DE MAYO DEL 2024

Introducción

Este proyecto se enfoca en la gestión eficiente de un almacén dentro de un antro, garantizando que esté siempre organizado y bien surtido. Utilizando una base de datos en SQL Server, se implementan cuatro funcionalidades principales para facilitar esta gestión:

Alta: Permite registrar nuevos productos en el sistema, asegurando que el inventario se mantenga actualizado y completo.

Baja: Facilita la eliminación de productos del inventario cuando sea necesario, asegurando que solo los artículos relevantes estén disponibles en el almacén.

Actualización: Permite realizar modificaciones en la información de los productos existentes, como cambios de precio, descripción u otra información relevante.

Consultas: Proporciona la capacidad de realizar búsquedas y consultas rápidas en la base de datos, permitiendo a los usuarios obtener información detallada sobre el inventario disponible.

Cada una de estas funcionalidades está diseñada para optimizar la administración del almacén, garantizando que los procesos de gestión de inventario sean eficientes y precisos.

BD En Sql Server Y Uso De Entity Framework

La migración se llevó a cabo desde Visual Studio con el uso de dos proyectos, una biblioteca de clases y una plantilla de ASP.NET Core Web API.

En ambos proyectos fue necesario instalar los paquetes NuGet Microsoft Entity Framework Tools y Microsoft Entity Framework SqlServer

Dentro de la biblioteca de clases, agregamos una clase que contendrá los atributos que se desee que tenga la tabla, según las tablas necesarias, se crea una clase nueva.

```
public class Producto
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public  int Productoid {  get; set; }
    public string NombreProducto {  get; set; }
    public string Descripcion { get; set; }
    public double Precio { get; set; }
    public int Cantidad { get; set; }
    public string Imagen { get; set; }
}
```

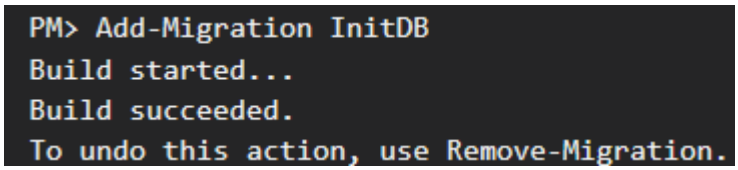
Después, abrimos el archivo “appsettings.json”, para agregar la cadena de conexión que necesita el nombre del servidor, la base de datos, así como directivas para confiar en los certificados de SQL Server.

```
"AllowedHosts": "*",
"ConnectionStrings": {
  "AntroConnection": "Server=DESKTOP-T75BOGG\\SQLEXPRESS;Database=Antro;Trusted_Connection=True;MultipleActiveResultSets=True;TrustServerCertificate=True"
}
```

Ahora, en la clase, “Program.cs”, se deben agregar las siguientes líneas, que indicaran que a la aplicación que se debe hacer la migración al ejecutarse

```
builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<AntroContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("AntroConnection"
))

);

var app = builder.Build();


using(var scope = app.Services.CreateScope())

{

    var context = scope.ServiceProvider.GetRequiredService<AntroContext>();

    context.Database.Migrate();

}
```

Por último, en la consola de administrador de paquetes, debemos de indicar que se inicie la migración con el comando “Add-Migration InitDB”



Alta

En este apartado se detalla el proceso de implementación del módulo de "Alta de Información" en un sistema de gestión de almacén. El presente documento detalla el proceso de implementación del módulo de "Alta de Información" en un sistema de gestión de almacén

Se crea "Agregar.cshtml" donde:

Se desarrolla un formulario para capturar los datos necesarios para agregar un nuevo producto al inventario.

```
<form method="post" enctype="multipart/form-data">
    <!-- Campo Nombre del Producto -->
    <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Nombre</label>
        <div class="col-sm-8">
            <input class="form-control" asp-for="ProductoDTO.NombreProducto">
            <span asp-validation-for="ProductoDTO.NombreProducto" class="text-danger"></span>
        </div>
    </div>

    <!-- Campo de la descripcion del prodcucto-->
    <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Descripcion</label>
        <div class="col-sm-8">
            <textarea class="form-control" asp-for="ProductoDTO.Descripcion"></textarea>
            <span asp-validation-for="ProductoDTO.Descripcion" class="text-danger"></span>
        </div>
    </div>
```

```
    <!-- Campo Precio del Producto -->
    <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Precio</label>
        <div class="col-sm-8">
            <input class="form-control" asp-for="ProductoDTO.Precio">
            <span asp-validation-for="ProductoDTO.Precio" class="text-danger"></span>
        </div>
    </div>

    <!-- Campo Cantidad del Producto -->
    <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Cantidad</label>
        <div class="col-sm-8">
            <input class="form-control" asp-for="ProductoDTO.Cantidad">
            <span asp-validation-for="ProductoDTO.Cantidad" class="text-danger"></span>
        </div>
    </div>

    <!-- Campo Imagen del Producto -->
    <div class="row mb-3">
        <label class="col-sm-4 col-form-label">Image</label>
        <div class="col-sm-8">
            <input class="form-control" asp-for="ProductoDTO.Imagen">
            <span asp-validation-for="ProductoDTO.Imagen" class="text-danger"></span>
        </div>
    </div>


 <!-- Botones de acción -->
```

```
    <div class="row mb-3">
        <!-- Botón Agregar -->
        <div class="offset-sm-4 col-sm-4 d-grid">
            <button type="submit" class="btn btn-primary">Agregar</button>
        </div>
        <!-- Botón Cancelar -->
        <div class="col-sm-4 d-grid">
            <a class="btn btn-outline-primary" href="/Admin/Products/Index" role="button">Cancelar</a>
        </div>
    </div>


</form>
```

Se implementan validaciones de entrada para garantizar la integridad de los datos ingresados por el usuario.

```
        <!-- VALIDACION Y MENSAJES DE ERROR  -->
        @if (Model.errorMessage.Length > 0)
        {
            <div class='alert alert-warning alert-dismissible fade show' role='alert'>
                <strong>@Model.errorMessage</strong>
                <button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'></button>
            </div>
        }
        else if (Model.successMessage.Length > 0)
        {
            <div class='alert alert-success alert-dismissible fade show' role='alert'>
                <strong>@Model.successMessage</strong>
```

```
            <button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'></button>
        </div>
    }
```

Se crea "Agregar.cshtml.cs" donde:

Se vincula los datos del formulario

```
[BindProperty]
public ProductoDTO ProductoDTO { get; set; } = new ProductoDTO();
```

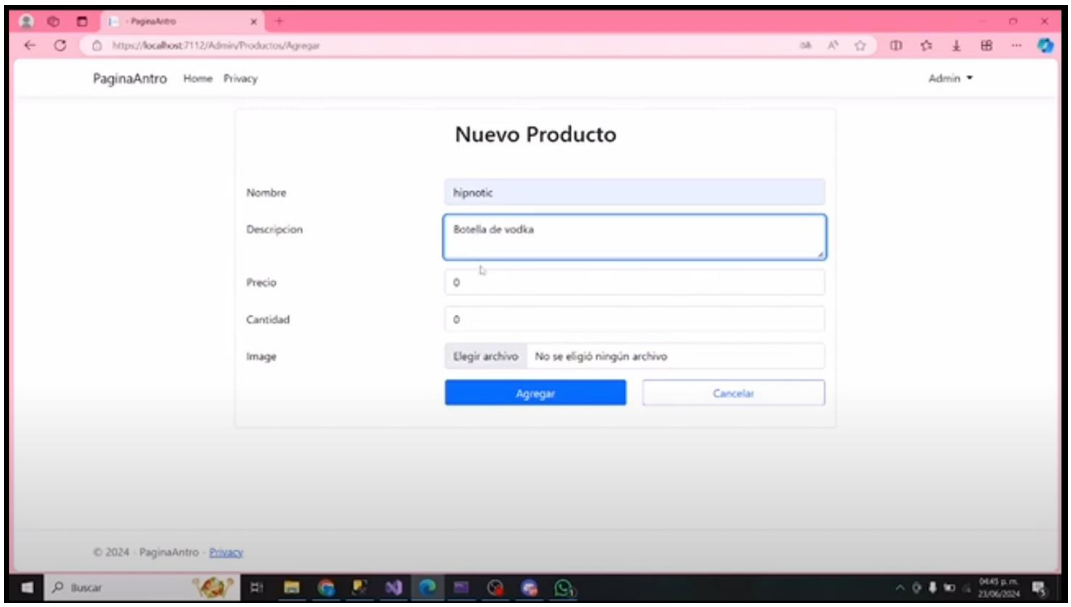
Se crea un constructor que inyecte el entorno web yel contexto de la base de datos

```
public AgregarModel(IWebHostEnvironment environment, ApplicationDbContext context)
    {
        this.environment = environment;
        this.context = context;
    }
```

Se crea el metodo que se ejecutara al enviar el formulario (HTTP POST)

```
public void OnPost()
{
    // Validación para asegurar que se ha seleccionado una imagen
    if (ProductoDTO.Imagen == null)
    {
        ModelState.AddModelError("ProductoDTO.Imagen", "Archivo de imagen requerido");
    }
```

```
if (!ModelState.IsValid)
{
    errorMessage = "Ingresa todos los campos";
    return;
}
```



Baja

En este aparatado se detalla el proceso de implementación del módulo de "Baja de Información" en un sistema de gestión de almacén desarrollado con Visual Studio, utilizando Entity Framework para la interacción con una base de datos SQL Server. El objetivo es permitir la eliminación de productos del inventario de manera eficiente y segura.

Se crea "Eliminar.cshtml.cs

Se desarrolla un constructor que inyecta el entorno web y el contexto de la base de datos

```
public EliminarModel(IWebHostEnvironment environment, ApplicationDbContext context)
{
    this.environment = environment;
    this.context = context;
}
```

Se desarrolla un método que ejecuta al cargar la pagina

```
public void OnGet(int? id)
{
    if (id == null)
    {
        Response.Redirect("/Admin/Productos/Index");
        return;
    }

    // Busca el producto por su ID en la base de datos
    var product = context.Producto.Find(id);
    if (product == null)
    {
```

```
            Response.Redirect("/Admin/Productos/Index");
            return;
        }

        // Obtiene la ruta completa de la imagen y la elimina del servidor
        string imageFullPath = environment.WebRootPath + "/productos/" + product.Imagen;
        System.IO.File.Delete(imageFullPath);

        // Elimina el producto de la base de datos y guarda los cambios
        context.Producto.Remove(product);
        context.SaveChanges();

        // Redirige al índice de productos después de eliminar el producto
        Response.Redirect("/Admin/Productos/Index");
    }
```

Actualización

En este apartado se detalla el proceso de implementación del módulo de "Actualización de Información" en un sistema de gestión de almacén desarrollado con Visual Studio. Utilizaremos Entity Framework para interactuar con una base de datos SQL Server, permitiendo modificar datos existentes de manera eficiente y sin errores.

Se crea "Editar.cshtml.cs"

Se desarrolla un formulario para capturar los datos necesarios para agregar un nuevo producto al inventario.

```
    <form method="post" enctype="multipart/form-data">
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">ID</label>
            <div class="col-sm-8">
                <input  readonly class="form-control-plaintext"
value="@Model.Producto.Productoid">


            </div>
        </div>
        <!-- Campo del nombre del prodcucto-->
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Nombre</label>
            <div class="col-sm-8">
                <input class="form-control" asp-
for="ProductoDTO.NombreProducto">
                <span asp-validation-for="ProductoDTO.NombreProducto"
class="text-danger"></span>
            </div>
        </div>


        <!-- Campo de la decripcion del prodcucto-->
```

```
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Descripcion</label>
            <div class="col-sm-8">
                <textarea class="form-control" asp-
for="ProductoDTO.Descripcion"></textarea>
                <span asp-validation-for="ProductoDTO.Descripcion" class="text-
danger"></span>
            </div>
        </div>

        <!-- Campo del precio del prodcucto-->
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Precio</label>
            <div class="col-sm-8">
                <input class="form-control" asp-for="ProductoDTO.Precio">
                <span asp-validation-for="ProductoDTO.Precio" class="text-
danger"></span>
            </div>
        </div>

        <!-- Campo de la cantidad del producto-->
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Cantidad</label>
            <div class="col-sm-8">
                <input class="form-control" asp-for="ProductoDTO.Cantidad">
                <span asp-validation-for="ProductoDTO.Cantidad" class="text-
danger"></span>
            </div>
        </div>
```

```
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label"></label>
            <div class="col-sm-8">
                <img src="~/productos/@Model.Producto.Imagen" width="150"/>
            </div>
        </div>
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Imagen</label>
            <div class="col-sm-8">
                <input class="form-control" asp-for="ProductoDTO.Imagen">
                <span asp-validation-for="ProductoDTO.Imagen" class="text-
danger"></span>
            </div>
        </div>
```

Se crean botones de accion Agregar y Cancelar

```
        <div class="row mb-3">
            <!-- Botón Agregar -->
            <div class="offset-sm-4 col-sm-4 d-grid">
                <button type="submit" class="btn btn-primary">Agregar</button>
            </div>
            <!-- Botón Cancelar -->
            <div class="col-sm-4 d-grid">
                    <a class="btn btn-outline-primary" href="/Admin/Products/Index"
role="button">Cancelar</a>
            </div>
        </div>
```

Se crea "Editar.cshtml"

Se implementan validaciones de entrada para garantizar la integridad de los datos ingresados por el usuario.

```
@if (Model.errorMessage.Length > 0)
{
    <div class='alert alert-warning alert-dismissible fade show' role='alert'>
        <strong>@Model.errorMessage</strong>
        <button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'></button>
    </div>
}
else if (Model.successMessage.Length > 0)
{
    <div class='alert alert-success alert-dismissible fade show' role='alert'>
        <strong>@Model.successMessage</strong>
        <button type='button' class='btn-close' data-bs-dismiss='alert' aria-label='Close'></button>
    </div>
}
```

Se crea "Editar.cshtml.cs"

Se crean propiedad para vincular los datos del formulario de edición, para mantener el producto a editar y para manejar mensajes de error y éxito

```
[BindProperty]

public ProductoDTO ProductoDTO { get; set; } = new ProductoDTO();

public Producto Producto { get; set; } = new Producto();
```

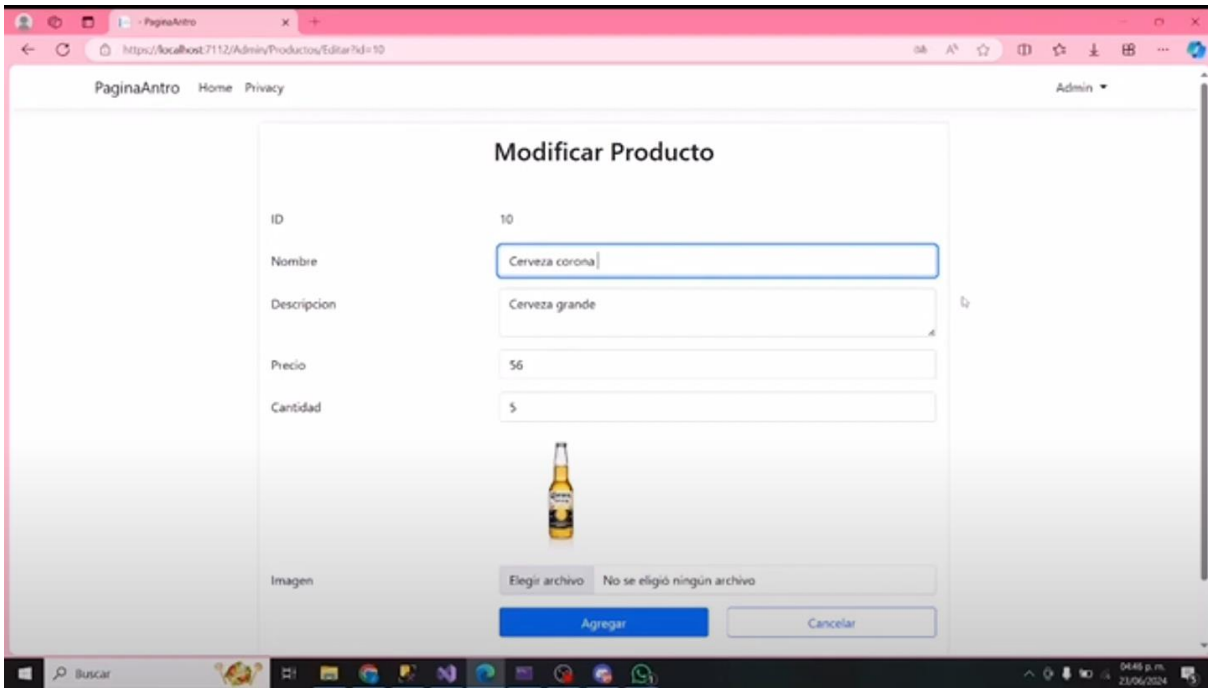
```
public string errorMessage = "";
public string successMessage = "";
```

Se crea un método que se ejecuta al enviar el formulario de edición.

```
public void OnPost(int? id)
{
    if (id == null)
    {
        Response.Redirect("/Admin/Productos/Index");
        return;
    }

    if (!ModelState.IsValid)
    {
        errorMessage = "Completa todos los campos";
        return;
    }

    var product = context.Producto.Find(id);
    if (product == null)
    {
        Response.Redirect("/Admin/Productos/Index");
        return;
    }


    // Actualiza el archivo de imagen si se ha proporcionado uno nuevo
```

```
    string newFileName = product.Imagen;
    if (ProductoDTO.Imagen != null)
    {
        newFileName = DateTime.Now.ToString("yyyyMMddHHmmssfff");
        newFileName += Path.GetExtension(ProductoDTO.Imagen!.FileName);

        string imageFullPath = environment.WebRootPath + "/Productos/" +
newFileName;
        using (var stream = System.IO.File.Create(imageFullPath))
        {
            ProductoDTO.Imagen.CopyTo(stream);
        }

        // Elimina la imagen antigua del servidor
        string oldImageFullPath = environment.WebRootPath + "/Productos/" +
product.Imagen;
        System.IO.File.Delete(oldImageFullPath);
    }


    // Actualiza los datos del producto en la base de datos
    product.NombreProducto = ProductoDTO.NombreProducto;
    product.Descripcion = ProductoDTO.Descripcion;
    product.Precio = ProductoDTO.Precio;
    product.Cantidad = ProductoDTO.Cantidad;
    product.Imagen = newFileName;

    context.SaveChanges();
```

```
        Producto = product;

        // Establece un mensaje de éxito y redirige al índice de productos
        successMessage = "Producto modificado exitosamente";

        Response.Redirect("/Admin/Productos/Index");
    }
```



Consultas

En este apartado se detalla el proceso de implementación del módulo de "Consulta de Información" en un sistema de gestión de almacén desarrollado con Visual Studio. Utilizaremos Entity Framework para interactuar con una base de datos SQL Server, permitiendo a los usuarios buscar y visualizar información del inventario de manera eficiente y sin errores.

Se crea "Index.cshtml.cs"

Se está declarando una propiedad pública llamada Products en una clase para la lista de productos:

```
public List<Producto> Products { get; set; } = new List<Producto>();
```

Se está definiendo un constructor (IndexModel) para la clase IndexModel.

```
public IndexModel(ApplicationDbContext context)
    {
        this.context = context;
    }
```

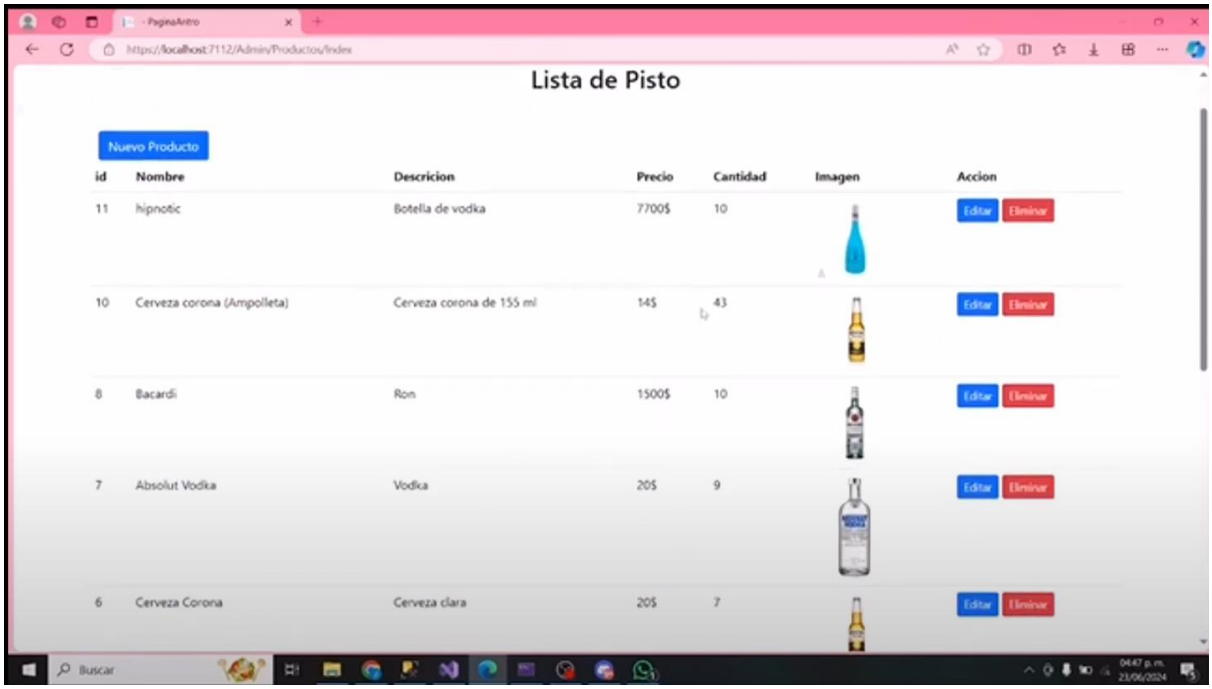
Se ordena por ID.

```
public void OnGet()
    {
        Products = context.Producto.OrderByDescending(p => p.Productoid).ToList();
    }
```

Se crea "Index.cshtml"

Se crea un enlace para agregar un nuevo producto

```
<div class="row mb-5">
   <div class="col">
      <a class='btn btn-primary' href='/Admin/Productos/Agregar'>Nuevo Producto</a>
   </div>
```



INSTALACION DE MODULOS NECESARIOS

- Instalación de Paquetes NuGet
  - Microsoft Entity Framework SqlServer
  - Microsoft Entity Framework Tools
- Dotnet sdk 8.0 para uso de plantilla de Aplicación Web ASP.NET Core