CENTRO DE CIENCIAS BASICAS

DEPARTAMENTO DE SISTEMAS DE INFORMACION

ACADEMIA DE INGENIERIA DE SOFTWARE

DOCUMENTO TECNICO

MATERIA: DESARROLLO WEB

MAESTRA: MARGARITA MONDRAGON ARELLANO

INTEGRANTES:

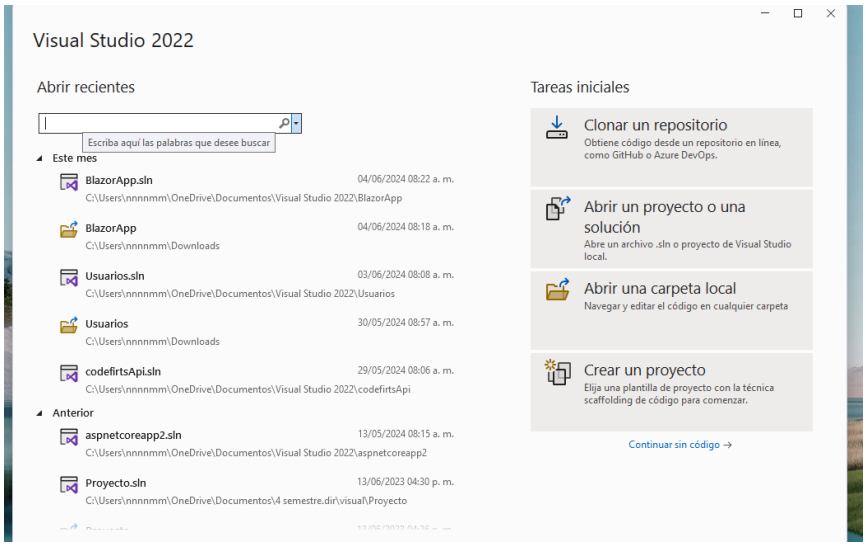
ITZEL ADRIANA PALOS FLORES

MINERVA CATALINA PEREZ GONZALEZ

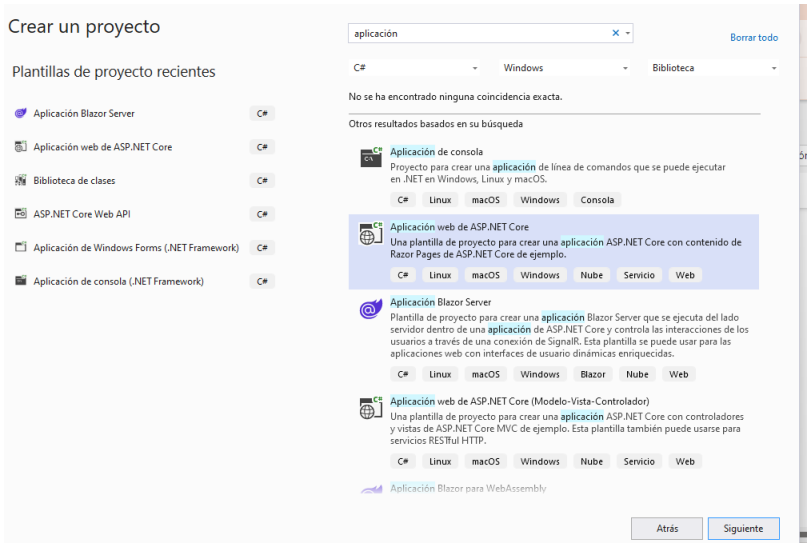
ALONDRA REYES MARTINEZ

EDWIN JESUS ROSALES AGUILAR

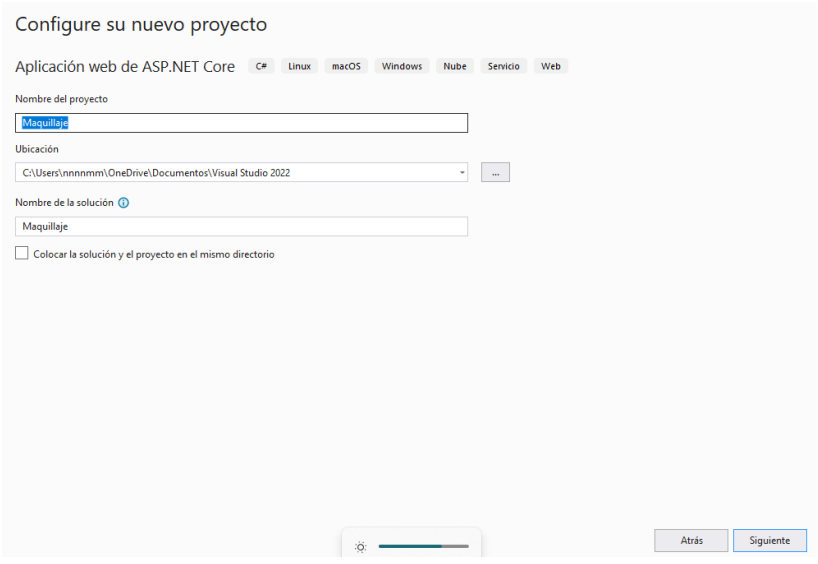
Hemos iniciado el desarrollo de un proyecto utilizando Microsoft Visual Studio



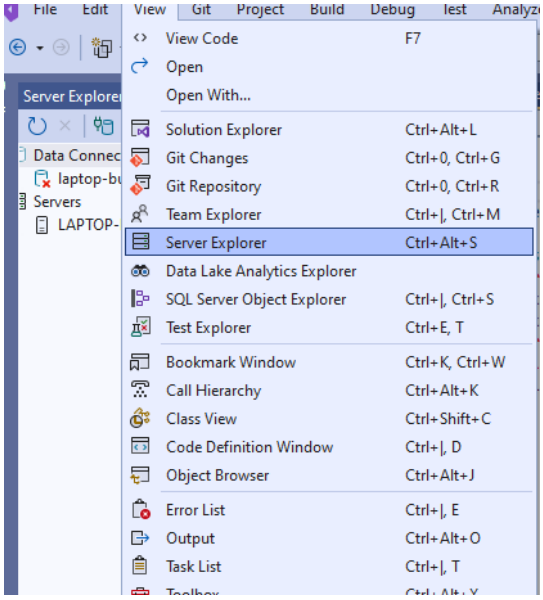
Estamos utilizando la aplicación web desarrollada en ASP.NET Core

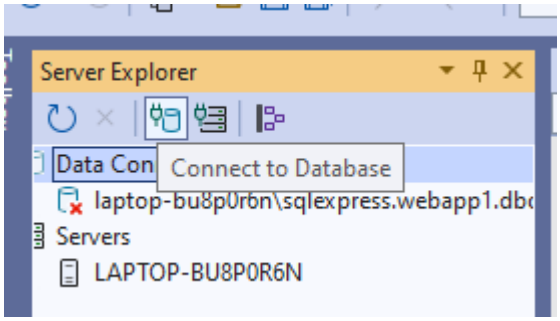


Nombramos el proyecto como maquillaje

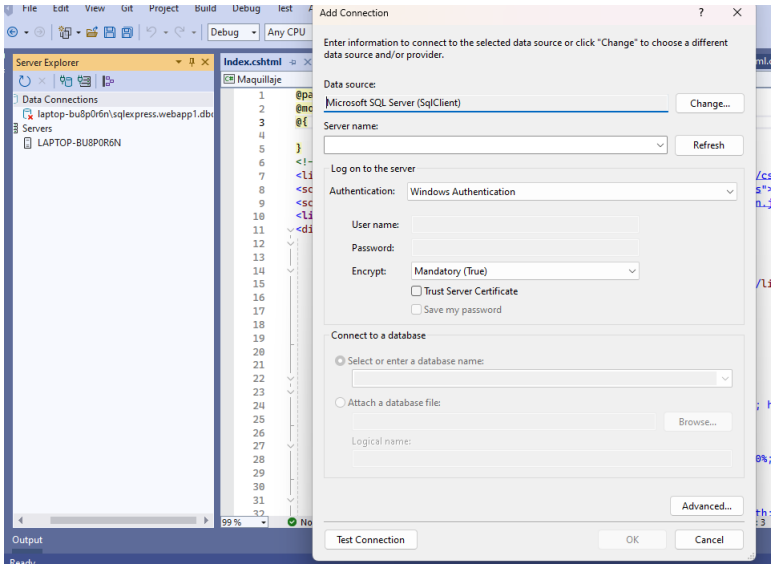


Establecemos una conexión a la base de datos



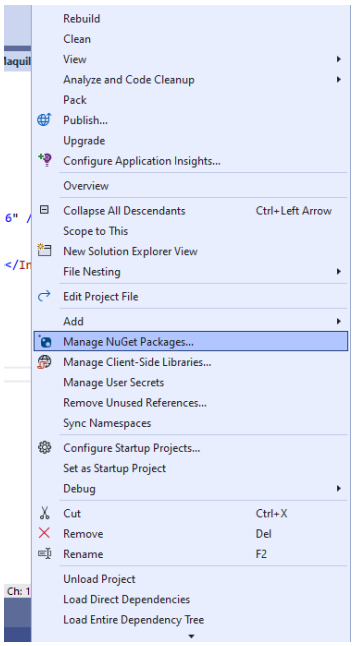


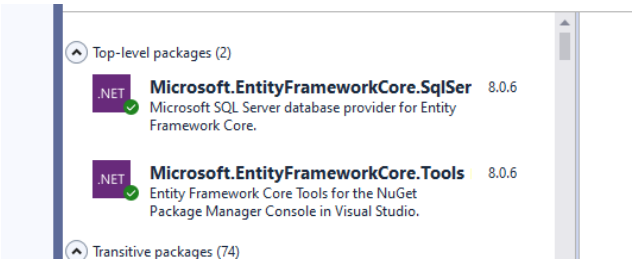
Nombramos a la base de datos



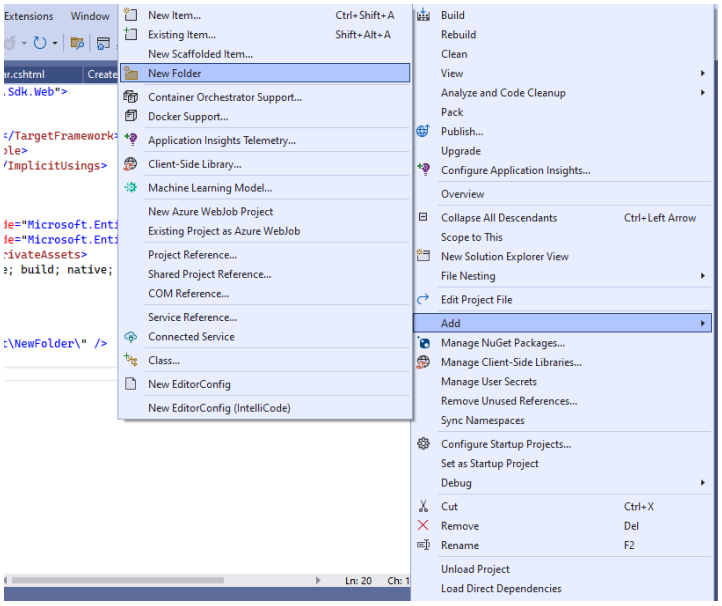
Agregamos la ruta correspondiente en el archivo appsettings.json

Realizamos la instalación de los paquetes necesarios

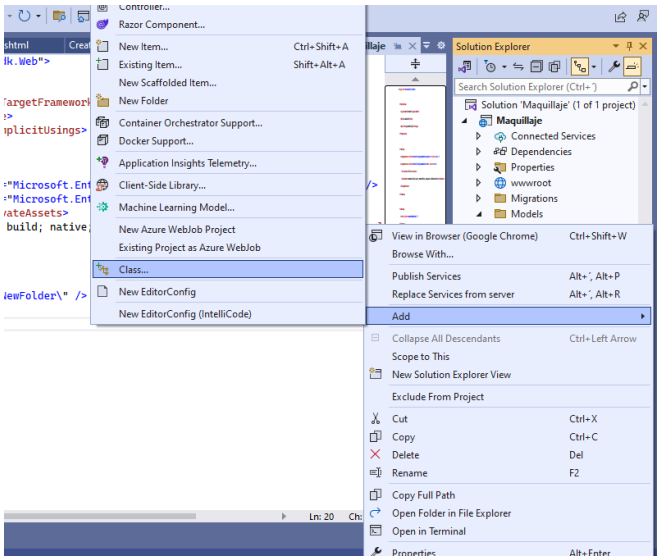




Añadimos una carpeta denominada Services



Agregamos una clase llamada Maquillaje context

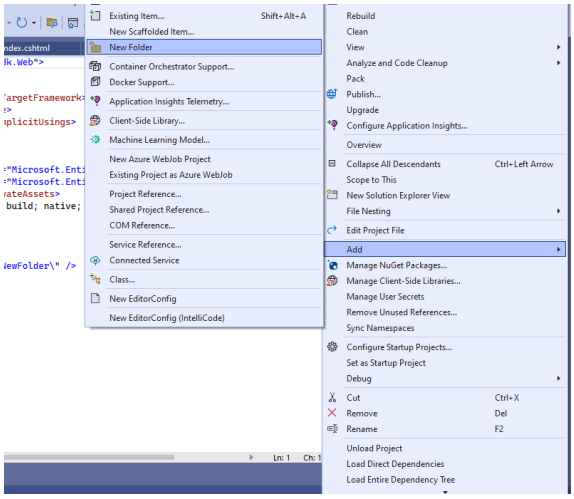



```
using Microsoft.EntityFrameworkCore;
using Maquillaje.Models;

namespace Maquillaje.Services
{
    public class MaquillajeContext : DbContext
    {
        public MaquillajeContext(DbContextOptions options):base(options) {

        }

        public DbSet<Producto> Productos { get; set; }
    }
}
```

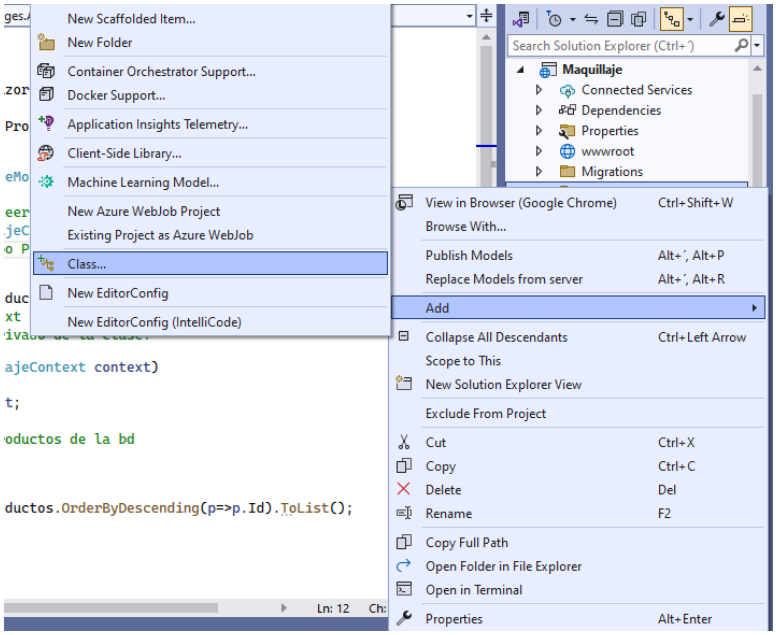
Agregamos una función que realiza llamados a la base de datos

```
builder.Services.AddRazorPages();

//Es la función para ser llamado a la base
//Se requiere un parámetro para ser llamado (options)
builder.Services.AddDbContext<MaquillajeContext>(options =>
{
    //se agregó ! al final para poder regresar valores nulos y no haya fallos
    string connectionString = builder.Configuration.GetConnectionString("DefaultConnection"
    options.UseSqlServer(connectionString);
});
```

Agregamos una carpeta llamada Modelos



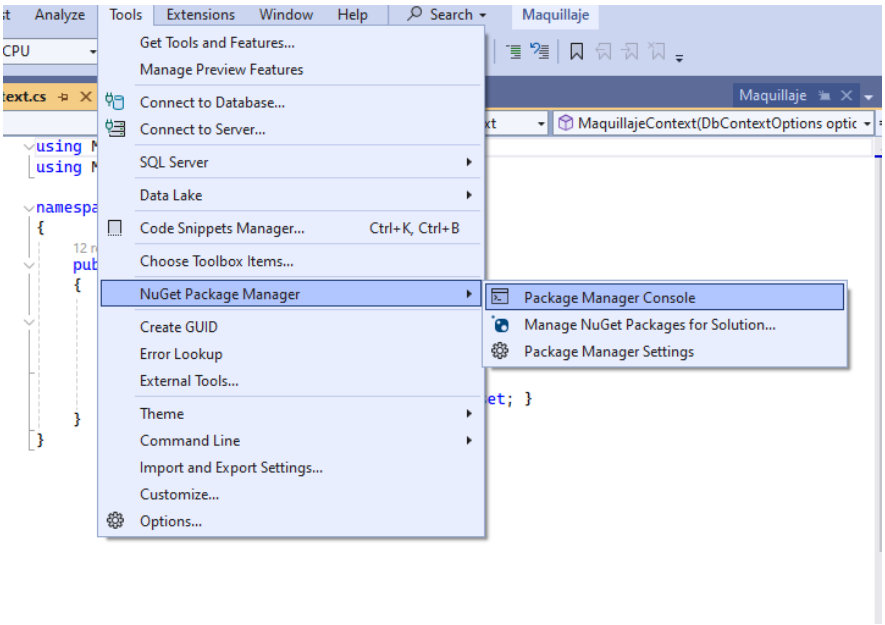
En esa misma carpeta, creamos una clase llamada Producto

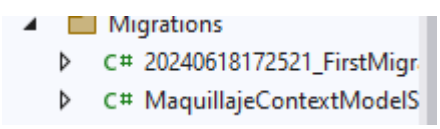


Aquí realizamos la estructura de la base de datos que incluye una tabla con cinco atributos. En MaquillajeContext, se implementará la lógica para guardar todos los registros relacionados con esta estructura

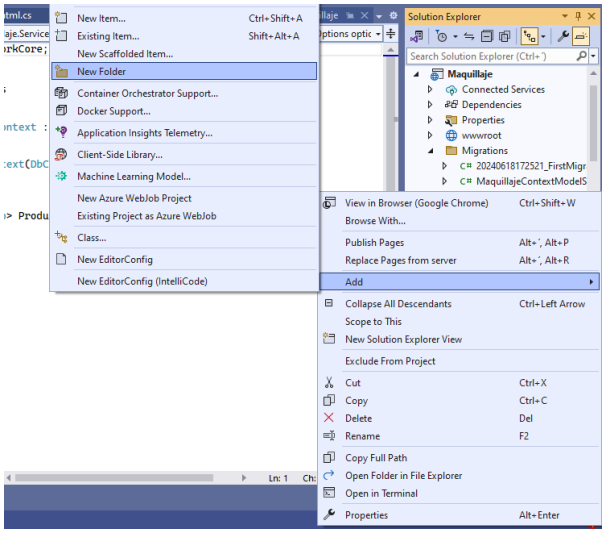
```
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;

namespace Maquillaje.Models
{
    7 references
    public class Producto
    {
        5 references
        public int Id { get; set; }
        [MaxLength(100)]
        4 references
        public string Nombre { get; set; } = "";
        [MaxLength(100)]
        4 references
        public string Marca { get; set; } = "";
        [MaxLength(100)]
        4 references
        public string Tono { get; set; } = "";
        [Precision(12,2)]
        4 references
        public decimal Precio { get; set; }
        [MaxLength(100)]
        7 references
        public string Imagen { get; set; } = "";
    }
}
```

Se realizó la migración hacia la base de datos

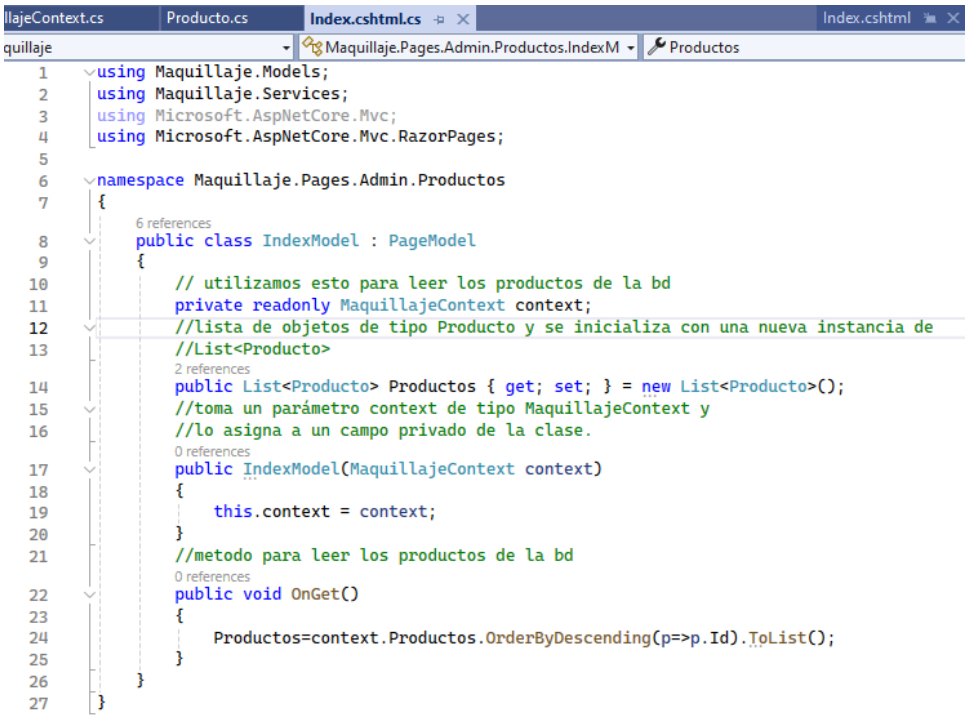




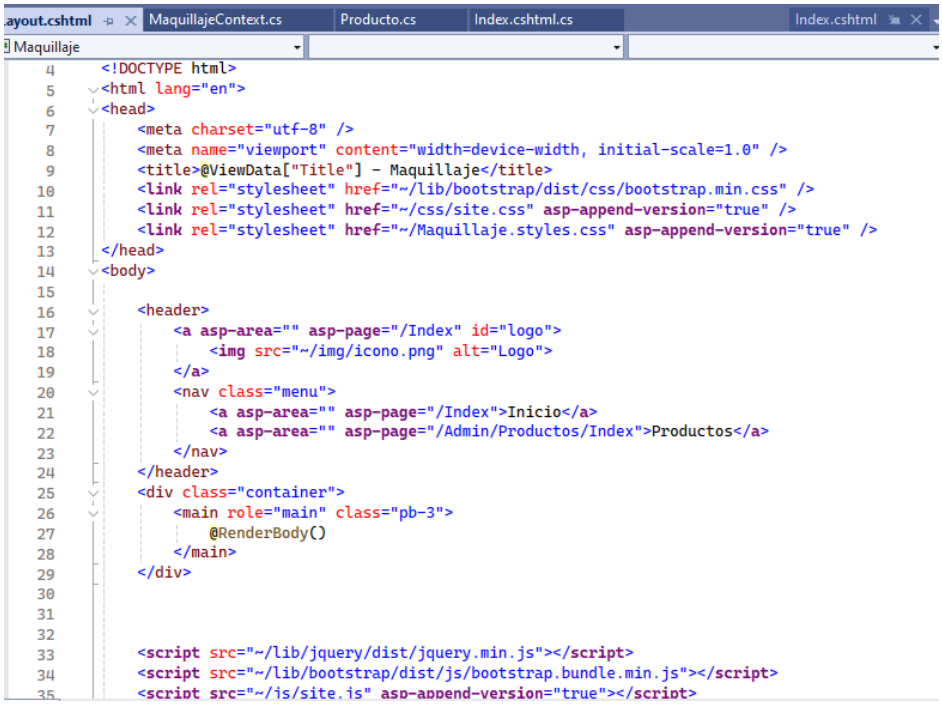
Se añade una carpeta adicional llamada Admin



Aquí se leen los productos desde la base de datos



En esta sección se encuentra el diseño del menú de la aplicación, el cual será compartido por todos los demás archivos



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Maquillaje</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Maquillaje.styles.css" asp-append-version="true" />
</head>
<body>

    <header>
        <a asp-area="" asp-page="/Index" id="logo">
            <img src="~/img/icono.png" alt="Logo">
        </a>
        <nav class="menu">
            <a asp-area="" asp-page="/Index">Inicio</a>
            <a asp-area="" asp-page="/Admin/Productos/Index">Productos</a>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>



    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
```

Aquí se presenta la estructura de la página principal

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
<!-- Estructura de la página de inicio-->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
<div class="container">
    <div id="myCarousel" class="carousel slide" data-ride="carousel">
        <!-- Indicators -->
        <ol class="carousel-indicators">
            <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
            <li data-target="#myCarousel" data-slide-to="1"></li>
            <li data-target="#myCarousel" data-slide-to="2"></li>
            <li data-target="#myCarousel" data-slide-to="3"></li>
        </ol>

        <!-- Wrapper for slides -->
        <div class="carousel-inner">
            <div class="item active">
                <img src="img/carrusel1.jpg" alt="Estudio" style="width:100%; height: 400px; object-fit: cover;">
            </div>

            <div class="item">
                <img src="img/carrusel2.jpg" alt="Productos" style="width:100%; height: 400px; object-fit: cover;">
            </div>

            <div class="item">
```

En esta sección se exhibe la estructura técnica de la página principal de productos, donde se pueden visualizar los productos junto con la inclusión de botones adicionales

```
@page
@model Maquillaje.Pages.Admin.Productos.IndexModel
@{
}
<!-- Estrucutra de la página donde se muestran los productos registrados-->
<!-- Contiene un botón para agregar producto que nos mandará a otra página-->
<!-- Contiene una tabla y cada registro dos botones (eliminar y editar)
    el botón de editar nos mandará a otra página
-->

<h2 class="text-center mb-5">Lista de productos de maquillaje</h2>
<div class="row mb-5">
    <div class="col">
        <a class="btn btn-primary" href="/Admin/Productos/Create">Nuevo producto</a>
    </div>
    <div class="col">

    </div>
</div>

<table class="table">
    <tr>
        <th>Id</th>
        <th>Nombre</th>
        <th>Marca</th>
        <th>Tono</th>
        <th>Precio</th>
        <th>Imagen</th>
    </tr>
    @foreach(var producto in Model.Productos)
    {
        <tr>
```

Funcionalidades de la página principal de producto

```
using Maquillaje.Models;
using Maquillaje.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace Maquillaje.Pages.Admin.Productos
{
    6 references
    public class IndexModel : PageModel
    {
        // utilizamos esto para leer los productos de la bd
        private readonly MaquillajeContext context;
        //lista de objetos de tipo Producto y se inicializa con una nueva instancia de
        //List<Producto>
        2 references
        public List<Producto> Productos { get; set; } = new List<Producto>();
        //toma un parámetro context de tipo MaquillajeContext y
        //lo asigna a un campo privado de la clase.
        0 references
        public IndexModel(MaquillajeContext context)
        {
            this.context = context;
        }
        //metodo para leer los productos de la bd
        0 references
        public void OnGet()
        {
            Productos=context.Productos.OrderByDescending(p=>p.Id).ToList();
        }
    }
}
```

Aquí se crean todos los productos de maquillaje con sus datos e imagen, y se envían a la base de datos

```
using Maquillaje.Models;
using Maquillaje.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace Maquillaje.Pages.Admin.Productos
{
    6 references
    public class CreateModel : PageModel
    {
        // campo privados y solo de lectura para proporcionar información sobre el entorno web
        private readonly IWebHostEnvironment environment;
        // campo privados y solo de lectura para interactura con la bd
        private readonly MaquillajeContext context;

        //se utiliza para especificar que una propiedad de un modelo de página debe ser
        //enlazada automáticamente a los datos enviados en una solicitud HTTP.
        [BindProperty]

        //Define una propuedad en una clase ProductoDto ,
        //inicializada con nuevas instancias de sus respectivos tipos.
        21 references
        public ProductoDto ProductoDto { get; set; } = new ProductoDto();

        //Constructor con dos parámetros y asigna los parámetros a los campos
        0 references
        public CreateModel(IWebHostEnvironment environment, MaquillajeContext context)
        {
            this.environment = environment;
            this.context = context;
        }
        0 references
        public void OnGet()
        {
        }

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

        //manejar las solicitudes HTTP POST enviadas a la página.
        0 references
        public void OnPost()
        {

            if (ProductoDto.Imagen == null)
            {
                ModelState.AddModelError("ProductoDto.Imagen", "La imagen no es requerido");
            }

            //si no se han completado todos los campos nos muestra un error
            if (!ModelState.IsValid)
            {
                errorMessage = "Por favor llena todos los campos";
                return;
            }
```

```
            //guardar imagen
            string newFileName = DateTime.Now.ToString("yyyyMMddHHmmssfff");
            newFileName += Path.GetExtension(ProductoDto.Imagen!.FileName);

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

            // guardar producto en la bd
            Producto producto = new Producto()
            {
                Nombre=ProductoDto.Nombre,
                Marca=ProductoDto.Marca,
                Tono=ProductoDto.Tono,
                Precio=ProductoDto.Precio,
                Imagen=newFileName,
            };
            //añadimos el producto
            context.Productos.Add(producto);
            context.SaveChanges();

            //limpiar el formulario
            ProductoDto.Nombre = "";
            ProductoDto.Marca = "";
            ProductoDto.Tono = "";
            ProductoDto.Precio = 0;
            ProductoDto.Imagen = null;

            ModelState.Clear();
            successMessage = "Producto creado correctamente";
            Response.Redirect("/Admin/Productos/Index");
        }
    }
}
```

Aquí se presentan las funciones para ingresar los productos de maquillaje, incluyendo los campos que deben ser completados, así como botones para cancelar o aceptar la operación

```
<!-- Estrucutra de la página donde se ingresaran los productos-->
<!-- Contiene los campos para que puedan ser llenados-->
<!-- y dos botones uno para ingresar y otro para cancelar-->
<div class="row">
    <div class="col-md-8 mx-auto rounded border p-3">
        <h2 class="text-center mb-5">Nuevo producto</h2>

        @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" arial-label="Close"></button>
            </div>
        }

        <form method="post" enctype="multipart/form-data">
            <div class="row mb-3">
                <label class="col-sm-4 col-form-label">Nombre</label>
                <div class="col-sm-8">
                    <select class="form-select" asp-for="ProductoDto.Nombre">
                        <option value="Otro">Otro</option>
                        <option value="Base">Base</option>
                        <option value="Corrector">Corrector</option>
                        <option value="Iluminador">Iluminador</option>
                        <option value="Labial">Labial</option>
                        <option value="Fijador">Fijador</option>
                        <option value="Polvo">Polvo</option>
                        <option value="Gloss">Gloss</option>
                        <option value="Rimel">Rimel</option>
                        <option value="Primer">Primer</option>
                    </select>
                </div>
            </div>
```

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

            <div class="row mb-3">
                <label class="col-sm-4 col-form-label">Tono</label>
                <div class="col-sm-8">
                    <input class="form-control" asp-for="ProductoDto.Tono">
                    <span asp-validation-for="ProductoDto.Tono" class="text-danger"></span>
                </div>
            </div>

            <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>

            <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>

            <div class="row mb-3">
                <div class="offser-sm-4 col-sm-4 d-grid">
                    <button type="submit" class="btn btn-primary">Enviar</button>
                </div>
                <div class="col-sm-4 d-grid">
                    <a class="btn btn-outline-primary" href="/Admin/Productos/Index" role="button">Cancelar</a>
                </div>
            </div>

        </form>
```

Aquí se presentan las funciones para editar los productos de maquillaje, incluyendo los campos que deben ser completados, así como botones para cancelar o aceptar la operación

```
<!-- Estrucutra de la página donde se editarán los productos registrados-->
<!-- Contiene los campos con los datos del producto para que puedan ser modificados-->
<!-- y dos botones uno para enviar la modificación y otro para cancelar-->
<div class="row">
    <div class="col-md-8 mx-auto rounded border p-3">
        <h2 class="text-center mb-5">Editar producto</h2>

        @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" arial-label="Close"></button>
            </div>
        }

        <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.Id">
                </div>

            </div>

            <div class="row mb-3">
                <label class="col-sm-4 col-form-label">Nombre</label>
                <div class="col-sm-8">
                    <select class="form-select" asp-for="ProductoDto.Nombre">
                        <option value="Otro">Otro</option>
                        <option value="Base">Base</option>
                        <option value="Corrector">Corrector</option>
                        <option value="Iluminador">Iluminador</option>
                        <option value="Labial">Labial</option>
                        <option value="Fijador">Fijador</option>
```

```
            <label class="col-sm-4 col-form-label">Marca</label>
            <div class="col-sm-8">
                <input class="form-control" asp-for="ProductoDto.Marca">
                <span asp-validation-for="ProductoDto.Marca" class="text-danger"></span>
            </div>
        </div>

        <div class="row mb-3">
            <label class="col-sm-4 col-form-label">Tono</label>
            <div class="col-sm-8">
                <input class="form-control" asp-for="ProductoDto.Tono">
                <span asp-validation-for="ProductoDto.Tono" class="text-danger"></span>
            </div>
        </div>

        <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>


        <div class="row mb-3">
            <label class="col-sm-4 col-form-label"></label>
            <div class="col-sm-8">
              <img src="/Products/@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>


        <div class="row mb-3">
```

```
        <div class="row mb-3">
            <label class="col-sm-4 col-form-label"></label>
            <div class="col-sm-8">
              <img src="/Products/@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>


        <div class="row mb-3">
            <div class="offser-sm-4 col-sm-4 d-grid">
                <button type="submit" class="btn btn-primary">Enviar</button>
            </div>
            <div class="col-sm-4 d-grid">
                <a class="btn btn-outline-primary" href="/Admin/Productos/Index" role="button">Cancelar</a>
            </div>
        </div>

    </form>
  </div>
/div>
```

En esta sección se realiza la edición de todos los productos de maquillaje, buscándolos por su id. Si no se encuentra ningún producto con ese id, se regresa a la página principal. En caso contrario, se recuperan todos los datos del producto, se asignan los nuevos valores y se envían de vuelta a la base de datos

```
using Maquillaje.Models;
using Maquillaje.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using static System.Net.Mime.MediaTypeNames;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Text.RegularExpressions;

namespace Maquillaje.Pages.Admin.Productos
{
    6 references
    public class EditarModel : PageModel

    {
        // campo privados y solo de lectura para proporcionar información sobre el entorno web
        private readonly IWebHostEnvironment environment;
        // campo privados y solo de lectura para interactura con la bd
        private readonly MaquillajeContext context;

        //se utiliza para especificar que una propiedad de un modelo de página debe ser
        //enlazada automáticamente a los datos enviados en una solicitud HTTP.
        [BindProperty]

        //Define dos propiedades en una clase, ProductoDto y Producto,
        //ambas inicializadas con nuevas instancias de sus respectivos tipos.
        20 references
        public ProductoDto ProductoDto { get; set; } = new ProductoDto();
        4 references
        public Producto Producto { get; set; } = new Producto();

        // declaramos mensajes
        public string errorMessage = "";
        public string successMessage = "";

        //Constructor con dos parámetros y asigna los parámetros a los campos
        0 references
        public EditarModel(IWebHostEnvironment environment, MaquillajeContext context)
        {
            this.environment = environment;
            this.context = context;
        }

    public void OnGet(int? id)
    {
        //si no se encuentra el id vuelve a la página de productos
        if (id == null)
        {
            Response.Redirect("/Admin/Productos/Index");
            return;
        }

        //Si se encuentra un producto con el ID especificado, se devuelven los valores
        //de lo contrario, se devuelve null
        var product = context.Productos.Find(id);
        if (product == null) {
            Response.Redirect("/Admin/Productos/Index");
            return;
        }

        //Asignamos valores al objeto
        ProductoDto.Nombre = product.Nombre;
        ProductoDto.Marca = product.Marca;
        ProductoDto.Tono = product.Tono;
        ProductoDto.Precio = product.Precio;

        //Asignamos al objeto los nuevos datos ingresados
        Producto = product;

    }
```

```
public void OnPost(int? id)
{
    //si no se encuentra el id vuelve a la página de productos
    if (id == null)
    {
        Response.Redirect("/Admin/Productos/Index");
        return;
    }
    //si no se han completado todos los campos nos muestra un error
    if (!ModelState.IsValid)
    {
        errorMessage = "Por favor llene todos los campos";
        return;
    }

    //Si se encuentra un producto con el ID especificado, se devuelven los valores
    //de lo contrario, se devuelve null
    var product = context.Productos.Find(id);
    if (product == null)
    {
        Response.Redirect("/Admin/Productos/Index");
        return;
    }

    //actualizar la imagen si tenemos una nueva imagen
    string newFileName = product.Imagen;
    if (ProductoDto.Imagen != null)
    {
        newFileName = DateTime.Now.ToString("yyyyMMddHHmmssfff");
        newFileName += Path.GetExtension(ProductoDto.Imagen.FileName);
        string imageFullPath = environment.WebRootPath + "/Products/" + newFileName;
        using (var stream = System.IO.File.Create(imageFullPath))
        {
            ProductoDto.Imagen.CopyTo(stream);
        }

        //eliminar imagen vieja
        string oldImageFullPath = environment.WebRootPath + "/Products/" + product.Imagen;
        System.IO.File.Delete(oldImageFullPath);
    }
//actualizar los datos en la bd
product.Nombre = ProductoDto.Nombre;
product.Marca = ProductoDto.Marca;
product.Tono = ProductoDto.Tono;
product.Precio = ProductoDto.Precio;
product.Imagen = newFileName;

//guarda los cambios
context.SaveChanges();

//Asignamos al objeto los nuevos datos ingresados
Producto = product;
successMessage = "Producto actualizado correctamente";
//regresamos a la página de productos
Response.Redirect("/Admin/Productos/Index");
```

En esta sección se realiza la eliminación de todos los productos de maquillaje, buscándolos por su id. Si no se encuentra ningún producto con ese id, se regresa a la página principal. En caso contrario, se recuperan todos los datos del producto, se elimina y se actualiza la base de datos

```
0 referencias
public class EliminarModel : PageModel
{
    // campo privados y solo de lectura para proporcionar información sobre el entorno web
    private readonly IWebHostEnvironment environment;
    // campo privados y solo de lectura para interactura con la bd
    private readonly MaquillajeContext context;

    //Constructor con dos parámetros y asigna los parámetros a los campos
    0 references
    public EliminarModel(IWebHostEnvironment environment, MaquillajeContext context)
    {
        this.environment = environment;
        this.context = context;
    }

    //responde a solicitudes HTTP GET cargando un producto específico por id
    0 references
    public void OnGet(int? id)
    {
        //si no se encuentra el id vuelve a la página de productos
        if (id == null)
        {
            Response.Redirect("/Admin/Productos/Index");
            return;
        }

        //Si se encuentra un producto con el ID especificado, se devuelven los valores
        //de lo contrario, se devuelve null
        var producto = context.Productos.Find(id);
        if (producto == null)
        {
            Response.Redirect("/Admin/Productos/Index");
            return;
        }

        //ver la dirección de la imagen y eliminarla
        string imageFullPath = environment.WebRootPath + "/Products/" + producto.Imagen;
        System.IO.File.Delete(imageFullPath);

        //eliminar los datos del producto
        context.Productos.Remove(producto);
        //guardar los cambios
        context.SaveChanges();
        //volver a la página productos
        Response.Redirect("/Admin/Productos/Index");
    }
}
```