

## Anexo

### Tratamiento de imágenes de productos con ASP.NET Core MVC

Lo más habitual es almacenar en la tabla de productos el nombre de la imagen, mientras que los archivos de imagen correspondientes a cada producto se suelen ubicar en una carpeta del proyecto, por ejemplo, en la carpeta `~/wwwroot/imagenes`. El nombre de archivo que almacena la imagen de cada producto, además, puede coincidir con el valor de la propiedad `Id` del producto.

Una forma habitual de gestionar las imágenes es asignar una imagen predeterminada al crear un nuevo producto, de manera que, posteriormente, el usuario puede cambiar la imagen del producto mediante el procesamiento correspondiente. En este caso, la solicitud de ejecución de la acción para cambiar la imagen de cada producto puede realizarse a través de un enlace que especifica la dirección URL correspondiente. Esta dirección URL estará formateada según el enrutamiento MVC establecido para la aplicación Web, de modo que se especifique a la acción que permite cambiar la imagen y al producto en concreto al cual se desea cambiar la imagen mediante su valor `Id`. Este enlace puede situarse en la lista de productos que se obtiene mediante la acción `Index()` del controlador `ProductosController.cs`.

A continuación, se presenta un ejemplo que permite comprender el procesamiento que permite cambiar la imagen de un producto ya existente. Este procesamiento se realiza añadiendo la acción `CambiarImagen()` en el controlador `ProductosController.cs` y creando la vista correspondiente a esa nueva acción, tal como se puede apreciar en el siguiente código.

**Controlador: `ProductosController.cs`**

**Método de Acción: `CambiarImagen()`**

```
// GET: Productos/CambiarImagen/5
public async Task<IActionResult> CambiarImagen(int? id)
{
    if (id == null || _context.Productos == null)
    {
        return NotFound();
    }

    var producto = await _context.Productos
        .Include(p => p.Categoria)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (producto == null)
    {
        return NotFound();
    }

    return View(producto);
}

// POST: Productos/CambiarImagen/5
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> CambiarImagen(int? id, IFormFile imagen)
{
    if (id == null)
    {
        return NotFound();
    }
}
```

```
}

var producto = await _context.Productos.FindAsync(id);
if (producto == null)
{
    return NotFound();
}

if (imagen == null)
{
    return NotFound();
}

if (ModelState.IsValid)
{
    // Copiar archivo de imagen
    string strRutaImagenes = Path.Combine(_webHostEnvironment.WebRootPath, "imagenes");
    string strExtension = Path.GetExtension(imagen.FileName);
    string strNombreFichero = producto.Id.ToString() + strExtension;
    string strRutaFichero = Path.Combine(strRutaImagenes, strNombreFichero);
    using (var fileStream = new FileStream(strRutaFichero, FileMode.Create))
    {
        imagen.CopyTo(fileStream);
    }

    // Actualizar producto con nueva imagen
    producto.Imagen = strNombreFichero;
    try
    {
        _context.Update(producto);
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!ProductoExists(producto.Id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
}

return View(producto);
}
```

**Controlador: *ProductosController.cs***

**Inyección de dependencia en el constructor**

Los controladores de ASP.NET Core MVC solicitan las dependencias de forma explícita a través de los constructores. Los servicios se agregan como un parámetro del constructor. Es necesario especificar la inyección de dependencia del servicio *IWebHostEnvironment* en el controlador *ProductosController.cs*, tal como se resalta a continuación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MvcTienda.Data;
using MvcTienda.Models;

namespace MvcTienda.Controllers
{
    public class ProductosController : Controller
    {
        private readonly MvcTiendaContexto _context;
        private readonly IWebHostEnvironment _webHostEnvironment;

        public ProductosController(MvcTiendaContexto context, IWebHostEnvironment
HostEnvironment)
        {
            _context = context;
            _webHostEnvironment = HostEnvironment;

            // GET: Productos
            public async Task<IActionResult> Index()
            {
                var mvcTiendaContexto = _context.Productos.Include(p => p.Categoria);
                return View(await mvcTiendaContexto.ToListAsync());
            }

            . . .
        }
    }
}
```

**Vista: CambiarImagen.cshtml**

**Carpeta de Vistas: ~/Views/Productos**

```
@model MvcTienda.Models.Producto

@{
    ViewData["Title"] = "CambiarImagen";
}

<h1>Cambiar imagen</h1>

<div>
    <h4>Producto</h4>
    <hr />
    <dl class="row">
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Descripcion)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Descripcion)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Texto)
        </dt>
    </dl>
</div>
```

```
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Texto)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Precio)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Precio)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Stock)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Stock)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Escaparate)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Escaparate)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Categoria)
</dt>
<dd class="col-sm-10">
    @Html.DisplayFor(model => model.Categoria.Descripcion)
</dd>
<dt class="col-sm-2">
    @Html.DisplayNameFor(model => model.Imagen)
</dt>
<dd class="col-sm-10">
    <div>
        <div>
            @{
                var nombreImagen = "imagen-no-disponible.jpg";
                if (Model.Imagen != null)
                {
                    nombreImagen = Model.Imagen;
                }
            }
            
        </div>
        <br />
        <div>
            <form enctype="multipart/form-data"
                asp-action="CambiarImagen">
                <div class="form-group" hidden>
                    <label asp-for="Id" class="control-label"></label>
                    <input asp-for="Id" class="form-control" />
                </div>
                <div class="form-group">
                    <input type="file" asp-for="Imagen"
                        class="form-control-file">
                </div>
                <br />
                <div class="form-group">
                    <input type="submit" value="Actualizar imagen"
                        class="btn btn-sm btn-primary rounded-0" />
                </div>
            </form>
        </div>
    </div>
</dd>
```







```

    </div>
  </form>
</div>
</div>
</dd>
</dl>
</div>

<div>
  <a asp-action="Index">Volver</a>
</div>

```

Finalmente, será necesario añadir la solicitud de la ejecución de la acción *CambiarImagen()* mediante el enlace correspondiente que se incluirá en la lista de productos que se presenta visualmente a través de la vista *Index.cshtml* de la carpeta de vistas *Productos*.

MvcTienda Inicio Privacidad Pedidos Clientes Productos Categorías Estados Detalles							Register Login
<h2>Productos</h2>							
<a href="#">Nuevo</a>							
Descripción	Texto	Precio	Stock	Escaparate	Imagen	Categoría	
Taladradora A34/F. Dos velocidades con percutor	Taladradora de dos velocidades con percutor. Especialmente diseñada para trabajos de bricolaje	45,91		<input type="radio"/>		Herramientas eléctricas	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>
Taladradora A1003/P. Profesional. Red 230 V.	Taladradora A1003/P de gama profesional. Resistente y duradera para trabajo profesional de altas prestaciones. Conexión a red de 230 V.	128,69		<input type="radio"/>		Herramientas eléctricas	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>
Rack pequeño 50 cm. con alimentación	Rack pequeño de 50 cm. de altura para instalaciones de redes informáticas	83,54		<input checked="" type="radio"/>		Material de redes informáticas	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>
Rack mediano 100 cm	Rack mediano de 100 cm para instalaciones de redes informáticas	268,35		<input checked="" type="radio"/>		Material de redes informáticas	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>
Cable eléctrico monopolar de 2,5 mm. Rollo 5 m.	Cable eléctrico monopolar de 2,5 mm. de diámetro. Rollo de 5 m. Varios colores: azul, gris, marrón, negro y verde-amarillo	12,55		<input type="radio"/>		Material eléctrico	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>
Pack de 4 bombillas led de 5 W. E27	Pack de 4 bombillas led de 5W/230V. Casquillo normalizado E27	15,83		<input checked="" type="radio"/>		Iluminación	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>   <a href="#">Cambiar imagen</a>