Tratamiento de las imágenes de los 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 la imagen, 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 ejecución de la acción que permite cambiar la imagen de cada producto puede realizarse a través de un enlace que especifica la dirección URL correspondiente, que estará formateada según el enrutamiento MVC establecido para la aplicación Web. Este enlace puede situarse en la lista de productos que se obtiene mediante la acción *Index()*.

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)
    {
        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.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
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*

```
@model MvcTienda.Models.Producto

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

<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>
        <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;
                        }
                    }
                    <img src="~/imagenes/@nombreImagen" height="250" width="250"
                         asp-append-version="true" />
                </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>
                        <div class="form-group">
                            <input type="submit" value="Actualizar imagen"
                class="btn btn-sm btn-primary rounded-0 text-uppercase" />
                        </div>
                    </form>
                </div>
            </div>
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
```