[Authorize(Roles = "Administracion, Cliente")]

[HttpGet]

using Biodigestor.Data;

using Biodigestor.DTOs;

using Biodigestor.Model;

using Biodigestor.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Authorization;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

[Route("UsuarioAdministrador/[controller]")]

[ApiController]

public class ClienteController : ControllerBase

{

private readonly BiodigestorContext \_context;

public ClienteController(BiodigestorContext context)

{

\_context = context;

}

// POST: api/Clientes

[HttpPost]

public async Task<ActionResult<Cliente>> PostCliente(Cliente cliente)

{

// Validar el modelo

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

// Verificar si ya existe un cliente con el mismo DNI

var clienteExistente = await \_context.Clientes.FirstOrDefaultAsync(c => c.DNI == cliente.DNI);

if (clienteExistente != null)

{

// Retornar un conflicto si el DNI ya está registrado

return Conflict(new { message = "Ya existe un cliente con ese DNI." });

}

// Agregar el nuevo cliente si no existe un cliente con el mismo DNI

\_context.Clientes.Add(cliente);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetClienteByDni), new { dni = cliente.DNI }, cliente);

}

// GET: api/Clientes

[HttpGet("mis-datos")]

public async Task<ActionResult<ClienteDto>> GetClienteDatosTotales()

{

// Obtener el Username del usuario autenticado (suponiendo que tienes autenticación por sesión o básica)

var username = User.Identity.Name;

if (username == null)

{

return Unauthorized("No se pudo obtener el usuario autenticado.");

}

// Buscar el DNI del usuario autenticado en la tabla UsuariosRegistrados

var usuarioRegistrado = await \_context.UsuariosRegistrados.FirstOrDefaultAsync(u => u.Username == username);

if (usuarioRegistrado == null)

{

return NotFound("El usuario no está registrado.");

}

var dni = usuarioRegistrado.DNI;

// Consultar los datos del cliente autenticado basado en el DNI

var cliente = await \_context.Clientes

.Include(c => c.Domicilios)

.Include(c => c.Facturas)

.FirstOrDefaultAsync(c => c.DNI == dni);

if (cliente == null)

{

return NotFound();

}

var clienteDto = new ClienteDto

{

NumeroCliente = cliente.NumeroCliente,

DNI = cliente.DNI,

Nombre = cliente.Nombre,

Apellido = cliente.Apellido,

Email = cliente.Email,

Domicilios = cliente.Domicilios?.Select(d => new DomicilioDto

{

NumeroMedidor = d.NumeroMedidor,

Calle = d.Calle,

Numero = d.Numero,

Piso = d.Piso,

Departamento = d.Departamento

}).ToList(),

Facturas = cliente.Facturas?.Select(f => new FacturaDto

{

NumeroFactura = f.NumeroFactura,

FechaEmision = f.FechaEmision,

FechaVencimiento = f.FechaVencimiento,

ConsumoMensual = f.ConsumoMensual,

ConsumoTotal = f.ConsumoTotal

}).ToList()

};

return clienteDto;

}

// GET: api/Clientes (Administración puede ver todos los clientes)

[Authorize(Roles = "Administracion")]

[HttpGet]

public async Task<ActionResult<IEnumerable<Cliente>>> GetClientes()

{

var clientes = await \_context.Clientes

.Include(c => c.Domicilios)

.Include(c => c.Facturas)

.ToListAsync();

return clientes;

}

// GET: api/Clientes/{dni} (Solo administradores pueden usar este)

[Authorize(Roles = "Administracion")]

[HttpGet("{dni}")]

public async Task<ActionResult<Cliente>> GetClienteByDni(int dni)

{

var cliente = await \_context.Clientes.FirstOrDefaultAsync(c => c.DNI == dni);

if (cliente == null)

{

return NotFound();

}

return cliente;

}

// PUT: api/Clientes/{dni}

[HttpPut("{dni}")]

public async Task<IActionResult> PutCliente(int dni, Cliente cliente)

{

if (dni != cliente.DNI)

{

return BadRequest("El Numero de Cliente y D.N.I. no se pueden cambiar, debe ingresar los valores actuales.");

}

\_context.Entry(cliente).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ClienteExists(dni))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// DELETE: api/Clientes/{dni}

[HttpDelete("{dni}")]

public async Task<IActionResult> DeleteCliente(int dni)

{

var cliente = await \_context.Clientes.FirstOrDefaultAsync(c => c.DNI == dni);

if (cliente == null)

{

return NotFound();

}

\_context.Clientes.Remove(cliente);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool ClienteExists(int dni)

{

return \_context.Clientes.Any(e => e.DNI == dni);

}

}