UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

Proyecto *"Sistema Web de Gestión Veterinaria para Mascotas"*

Curso: *Programación Web II*

Docente: *Ing. Patrick Cuadros Quiroga*

Integrantes:

Ccalli Chata, Joel Robert ***(2017057528)***
Apaza Mamani, Edward Hernan ***(2018060915)***

Tacna – Perú
2025

Estándar de Programación

1. Introducción

Este documento establece las normas y estándares de programación para garantizar la calidad, mantenibilidad y escalabilidad del código en los proyectos de software.

---

2. Convenciones de Nomenclatura

2.1. Clases y Objetos

- Usar PascalCase para nombres de clases y objetos.
- Ejemplo:
- public class ClienteService {
- // Código
- }

2.2. Variables y Métodos

- Usar camelCase para variables y nombres de métodos.
- Ejemplo:
- public string obtenerNombreCompleto() {
- string nombreCompleto = "Juan Perez";
- return nombreCompleto;
- }

2.3. Constantes

- Usar UPPER_CASE para constantes.
- Ejemplo:
- public const int MAX_INTENTOS = 5;

2.4. Base de Datos

- Usar PascalCase para nombres de tablas y columnas.
- Ejemplo:
- CREATE TABLE Usuarios (
- UsuarioID INT PRIMARY KEY,

- Nombre VARCHAR(50),
- Apellido VARCHAR(50)
-);

---

3. Organización del Código

3.1. Estructura del Proyecto

El proyecto debe seguir la arquitectura MVC con una estructura ordenada:

Proyecto

│— Proyecto.sln

│— Proyecto.Web (Capa de Presentación)

│— Proyecto.Core (Lógica de Negocio)

│— Proyecto.Infrastructure (Acceso a Datos)

│— Proyecto.Tests (Pruebas Unitarias)

---

4. Comentarios y Documentación

- Usar comentarios XML en C#.
- Ejemplo:
- /// <summary>
- /// Obtiene una lista de usuarios activos.
- /// </summary>
- /// <returns>Lista de usuarios</returns>
- public List<Usuario> ObtenerUsuariosActivos() {
- return usuarios.Where(u => u.Activo).ToList();
- }

---

5. Manejo de Errores

- Usar try-catch para capturar excepciones.
- Ejemplo:

- try {
- int resultado = CalcularDivision(10, 0);
- } catch (Exception ex) {
- Console.WriteLine("Error: " + ex.Message);
- }

---

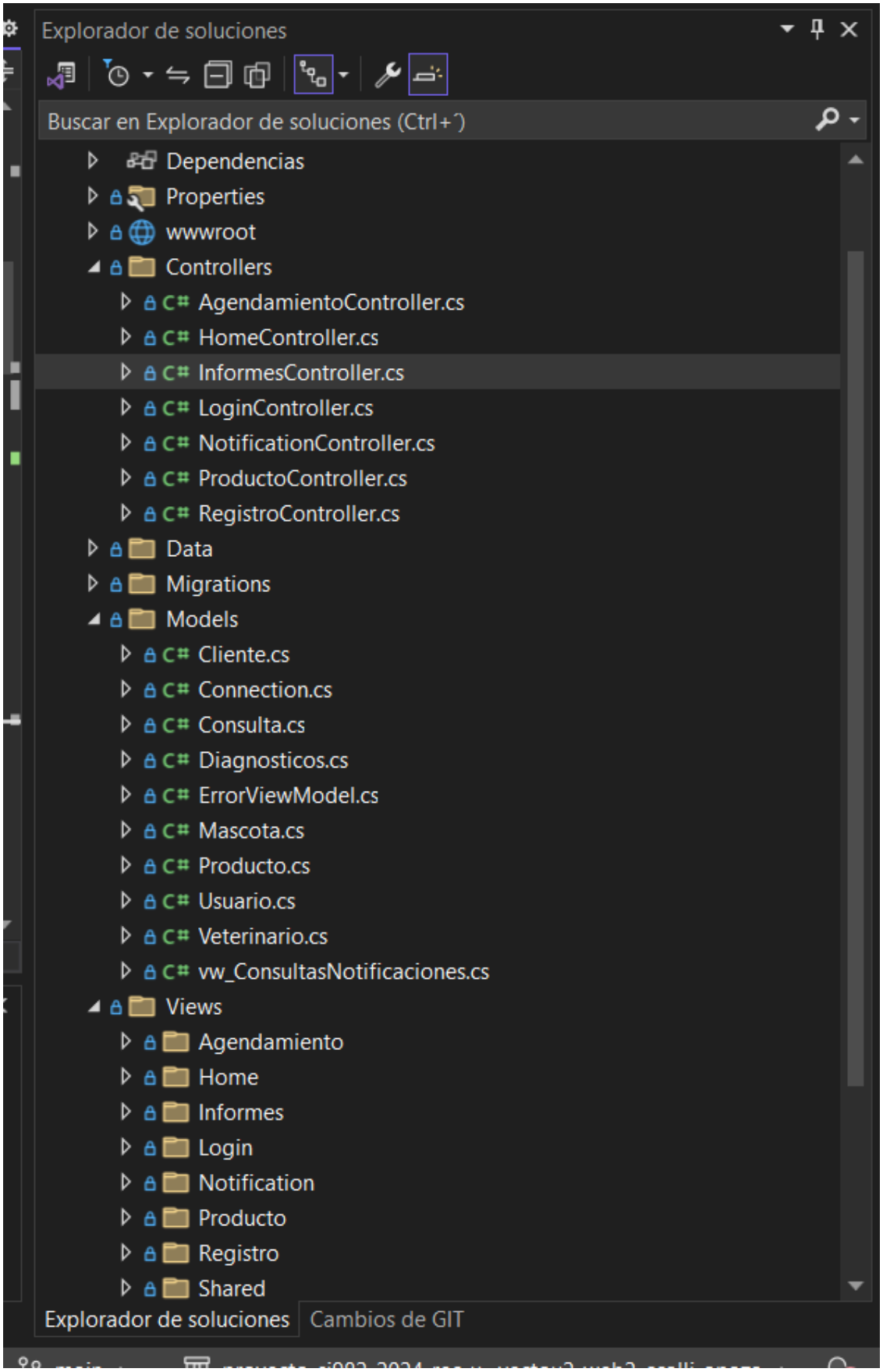
6. Seguridad

- No almacenar contraseñas en texto plano.
- Usar appsettings.json para configuraciones sensibles:
- {
- "ConnectionStrings": {
- "DefaultConnection": "Server=myserver;Database=mydb;User Id=myuser;Password=mypassword;"
- }
- }

Controlador: Informes

```
48      public async Task<IActionResult> CreateDiagnostico(Diagnostico diagnostico)
49      {
50          _context.Diagnosticos.Add(diagnostico);
51          await _context.SaveChangesAsync();
52          return RedirectToAction("Index", "Informes");
53      }
54
        0 referencias
55      public async Task<IActionResult> GenerarPdfDiagnostico(int idConsulta)
56      {
57          var consulta = await _context.Consultas
58              .Include(c => c.Mascota)
59                  .ThenInclude(m => m.Cliente)
60              .Include(c => c.Diagnostico)
61              .FirstOrDefaultAsync(c => c.IdConsulta == idConsulta);
62
63          if (consulta == null || consulta.Diagnostico == null)
64          {
65              return NotFound();
66          }
```

Estructura de Proyecto:



Controlador: Agendamiento

```
0 referencias
public IActionResult ReprogramarConsulta(int IdConsulta, string NuevaFecha, string NuevaHora)
{
    var consulta = _context.Consultas.Find(IdConsulta);
    if (consulta == null)
    {
        TempData["Error"] = "Consulta no encontrada.";
        return RedirectToAction("Index");
    }

    consulta.Fecha = DateTime.Parse(NuevaFecha);
    consulta.Hora = NuevaHora;
    _context.Consultas.Update(consulta);
    _context.SaveChanges();

    return RedirectToAction("Index");
}

[HttpPost]
0 referencias
public IActionResult CancelarConsulta(int IdConsulta)
{
    var consulta = _context.Consultas.Find(IdConsulta);
    if (consulta != null)
    {
        _context.Consultas.Remove(consulta);
        _context.SaveChanges();
        return RedirectToAction("Index");
    }

    TempData["Error"] = "Consulta no encontrada.";
    return RedirectToAction("Index");
}
```

Controlador: Producto

```
[HttpPost]
0 referencias
public async Task<IActionResult> Create(Producto producto)
{
    _context.Add(producto);
    await _context.SaveChangesAsync();
    // ViewBag.Productos = await _context.Productos.ToListAsync(); // No longer using ViewBag
    return View("Index", await _context.Productos.ToListAsync()); // Return to Index view with updated list
}

[HttpPost]
0 referencias
public async Task<IActionResult> Delete(int id)
{
    var producto = await _context.Productos.FindAsync(id);
    if (producto == null)
    {
        return NotFound(); // Handle case where product is not found
    }
    _context.Productos.Remove(producto);
    await _context.SaveChangesAsync();

    return View("Index", await _context.Productos.ToListAsync());
}
```

No se encontraron problemas.

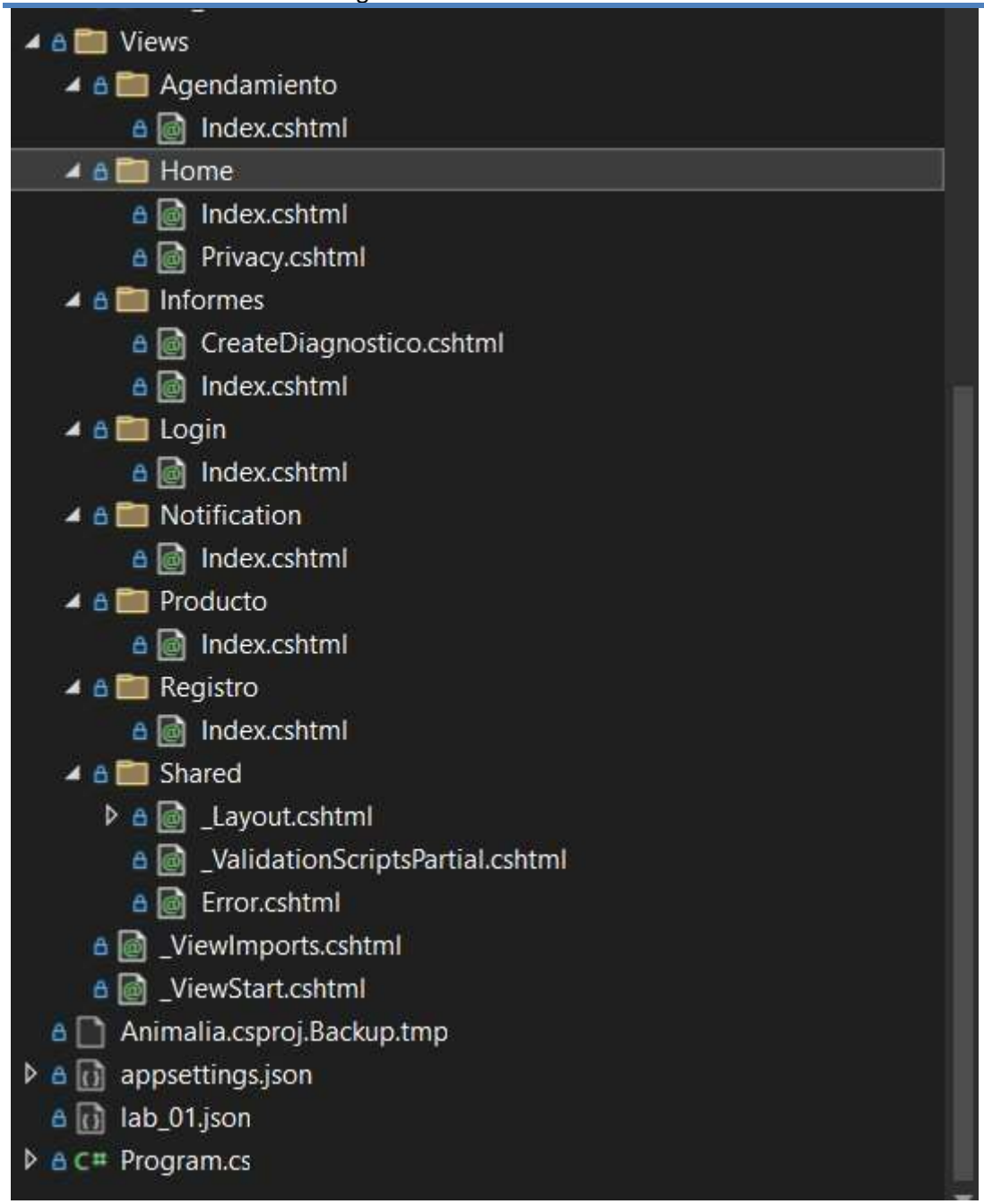
Modelo: Mascota

```
6 referencias
public class Mascota
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    2 referencias
    public int IdMascota { get; set; }

    [Required]
    [StringLength(100)]
    8 referencias
    public string Nombre { get; set; }

    [Required]
    [StringLength(50)]
    3 referencias
    public string Especie { get; set; }

    [StringLength(50)]
    2 referencias
    public string Raza { get; set; }

    2 referencias
    public int? Edad { get; set; }

    [StringLength(50)]
    3 referencias
    public string Color { get; set; }

    [Required]
    2 referencias
    public int IdCliente { get; set; }

    [ForeignKey("IdCliente")]
    9 referencias
    public Cliente Cliente { get; set; }
}
```

Modelo: Diagnostico

```
namespace Animalia.Models
{
    7 referencias
    public class Diagnostico
    {
        [Key]
        0 referencias
        public int IdDiagnostico { get; set; }

        [Required]
        1 referencia
        public int IdConsulta { get; set; }

        [ForeignKey("IdConsulta")]
        1 referencia
        public Consulta Consulta { get; set; }

        [Required]
        4 referencias
        public decimal Peso { get; set; }

        [MaxLength(500)]
        4 referencias
        public string Observaciones { get; set; }

        [MaxLength(500)]
        4 referencias
        public string ExamenesRealizados { get; set; }

        [Required]
        [MaxLength(1000)]
        4 referencias
        public string DiagnosticoGeneral { get; set; }

        1 referencia
        public DateTime FechaCreacion { get; set; } = DateTime.UtcNow;
    }
}
```

Vistas:



Vista General: HOME

```
<div class="container text-center mt-5">
    <h1 class="mb-4" style="color:  black; font-size: 2.5rem; font-weight: bold;">¿Qué quieres hacer?</h1>
    <div class="row justify-content-center">
        <div class="col-md-4 mb-4">
            <a href="@Url.Action("Index", "Agendamiento")" class="btn btn-lg btn-block action-card">Agendar Consulta</a>
        </div>
        <div class="col-md-4 mb-4">
            <a href="#" class="btn btn-lg btn-block action-card">Consultas del día</a>
        </div>
        <div class="col-md-4 mb-4">
            <a href="@Url.Action("Index", "Producto")" class="btn btn-lg btn-block action-card">Productos</a>
        </div>
    </div>
    <div class="row justify-content-center">
        <div class="col-md-4 mb-4">
            <a href="@Url.Action("Index", "Registro")" class="btn btn-lg btn-block action-card">Registros Médicos</a>
        </div>
        <div class="col-md-4 mb-4">
            <a href="@Url.Action("Index", "Informes")" class="btn btn-lg btn-block action-card">Informes</a>
        </div>
        <div class="col-md-4 mb-4">
            <a href="@Url.Action("Index", "Notification")" class="btn btn-lg btn-block action-card">Notificaciones</a>
        </div>
    </div>
```