**Descripción general del sistema**

El sistema web fue diseñado con el propósito de automatizar los procesos administrativos, académicos y médicos del internado, garantizando la trazabilidad, la integridad de los datos y la eficiencia operativa.  
Permite la gestión de residentes, usuarios, asistencia, calificaciones, consultas médicas y reportes consolidados desde una interfaz web accesible y segura.

El sistema se implementó bajo el patrón **Modelo–Vista–Controlador (MVC)**, integrando una interfaz moderna en HTML y Bootstrap, un backend robusto desarrollado en **ASP.NET (C#)** y una base de datos relacional en **Microsoft SQL Server**.

**Arquitectura del sistema**

La arquitectura se basa en el paradigma **multicapa**, dividiendo la solución en tres niveles principales:

1. **Capa de presentación (Frontend)**
   * Desarrollada en **HTML5, CSS3 y Bootstrap 5**.
   * Interactúa con el backend mediante controladores y peticiones HTTP seguras (HTTPS).
   * Cumple las normas de **UX/UI**: diseño responsivo, accesible, con retroalimentación visual, navegación intuitiva y consistencia de componentes.
2. **Capa de negocio (Lógica de aplicación)**
   * Implementada en **C# bajo ASP.NET Core MVC**.
   * Contiene toda la lógica empresarial, validaciones, cifrado y control de flujo.
   * Separa las operaciones del acceso a datos para mantener alta cohesión y bajo acoplamiento.
3. **Capa de datos (Persistencia)**
   * Gestionada en **SQL Server 2019**, con procedimientos almacenados parametrizados.
   * Incluye índices y claves foráneas para optimizar rendimiento y mantener integridad referencial.
   * Se implementaron roles y usuarios restringidos para mayor seguridad.

**Figura 1**  
Arquitectura general del sistema MVC.

**Diseño de la base de datos**

La base de datos se diseñó conforme al principio de **normalización en tercera forma normal (3FN)**, evitando redundancias y garantizando la integridad referencial entre las entidades.

**Entidades principales:**

|  |  |  |
| --- | --- | --- |
| **Tabla** | **Descripción** | **Campos clave** |
| **Usuarios** | Almacena credenciales y roles de acceso | UsuarioID (PK), RolID (FK), Nombre, Correo, HashContraseña, Estado |
| **Roles** | Define los perfiles de acceso del sistema | RolID (PK), NombreRol, Descripción |
| **Residentes** | Información personal y estado académico de las residentes | ResidenteID (PK), DPI, NombreCompleto, Tutor, Estado |
| **HistorialMedico** | Registro de consultas médicas | MedicoID (PK), ResidenteID (FK), Diagnóstico, Tratamiento, Fecha |
| **HistorialAcademico** | Registro de promedios y notas anuales | AcadID (PK), ResidenteID (FK), Promedio, Año |
| **Asistencia** | Control diario de asistencia | AsistenciaID (PK), ResidenteID (FK), Fecha, Estado, CursoID |
| **Calificaciones** | Registro de notas por curso | CalificacionID (PK), ResidenteID (FK), CursoID, Nota, Periodo |
| **Consultas** | Registro de atención médica | ConsultaID (PK), ResidenteID (FK), MédicoID, Diagnóstico |
| **Medicamentos** | Control de inventario farmacéutico | MedicamentoID (PK), Nombre, StockActual, FechaVencimiento |
| **Reportes** | Consolidación de información por módulo | ReporteID (PK), Tipo, FechaGeneracion |

**Relaciones:**

* Usuarios → Roles (1:N)
* Residentes → Asistencia, Calificaciones, HistorialMedico (1:N)
* Consultas → Medicamentos (N:M, controlado por tabla intermedia MovimientosMedicamentos)

**Figura 2**  
*Diagrama entidad–relación simplificado.*

**Tecnologías utilizadas**

|  |  |  |
| --- | --- | --- |
| **Componente** | **Tecnología** | **Descripción** |
| Lenguaje backend | **C# (.NET 8 MVC)** | Implementación del servidor, lógica de negocio y controladores |
| Frontend | **HTML5 / CSS3 / Bootstrap 5** | Interfaz web moderna, responsiva y accesible |
| Base de datos | **SQL Server 2019** | Motor relacional con soporte para procedimientos y vistas |
| ORM / Conexión | **Entity Framework Core** | Mapeo objeto-relacional, migraciones y consultas LINQ |
| Autenticación | **ASP.NET Identity / JWT** | Control de sesión y roles seguros |
| Gráficos | **Chart.js** | Visualización de indicadores |
| Exportación | **iTextSharp / EPPlus** | Generación de reportes en PDF y Excel |
| Control de versiones | **GitHub / Git** | Control de cambios y colaboración |
| IDE | **Visual Studio 2022**, **SQL Server Management Studio** | Desarrollo y administración del entorno |

**Estructura del proyecto**

**El sistema sigue la convención estándar de ASP.NET MVC:**

**/Internado**

**│**

**├── /Controllers → Controladores de cada módulo (UsuariosController, ResidentesController…)**

**├── /Models → Modelos de datos (DTO y entidades EF)**

**├── /Views → Vistas Razor/HTML (Bootstrap)**

**│ ├── /Usuarios**

**│ ├── /Residentes**

**│ ├── /Asistencia**

**│ ├── /Calificaciones**

**│ ├── /Medico**

**│ └── /Reportes**

**├── /wwwroot → Recursos estáticos (CSS, JS, imágenes)**

**├── /Services → Lógica de negocio e interfaces de servicios**

**├── /Data → Contexto de Entity Framework y scripts SQL**

**└── appsettings.json → Cadenas de conexión, variables de entorno y configuración JWT**

**Configuración del entorno**

**Requisitos mínimos del servidor:**

* Windows 10/11 o Windows Server 2019
* .NET 8 Runtime / SDK
* Microsoft SQL Server 2019
* IIS 10 habilitado con ASP.NET Core Hosting Bundle
* Visual Studio 2022 (opcional para desarrollo)

**Pasos de configuración:**

1. Clonar el repositorio con git clone.
2. Crear la base de datos InternadoDB en SQL Server.
3. Ejecutar los scripts de creación y procedimientos almacenados incluidos en /Data/Scripts.
4. Editar appsettings.json con la cadena de conexión:
5. Ejecutar las migraciones:
6. Ejecutar el proyecto con dotnet run o desde Visual Studio (F5).
7. Acceder desde el navegador a: <https://localhost:5001/>

**Despliegue (deployment)**

**En entorno local (IIS):**

1. Publicar el proyecto desde Visual Studio (Build → Publish).
2. Copiar la carpeta /publish al servidor IIS.
3. Crear un nuevo *Application Pool* con .NET CLR v8.0.
4. Configurar el sitio web y enlazar el puerto 443 (HTTPS).
5. Verificar la conexión con la base de datos y los permisos de lectura/escritura.

**En entorno en la nube (Azure App Service):**

1. Crear App Service con runtime .NET 8.
2. Configurar Azure SQL Database y cadena de conexión.
3. Habilitar autenticación SSL.
4. Publicar desde Visual Studio → *Publish to Azure*.
5. Activar monitorización de logs desde Application Insights.

**Mantenimiento preventivo y correctivo**

**Mantenimiento preventivo**

* **Actualizaciones mensuales** de dependencias .NET y paquetes NuGet.
* **Revisión de logs** para detectar errores o accesos no autorizados.
* **Optimización de índices** en SQL Server y ejecución de DBCC CHECKDB.
* **Respaldo automático** diario de la base de datos.
* **Pruebas funcionales** trimestrales en cada módulo.
* **Verificación UX/UI**: validación de compatibilidad en navegadores y dispositivos móviles.

**Mantenimiento correctivo**

* Aplicar *hotfixes* ante fallas identificadas en controladores o procedimientos.
* Corregir vulnerabilidades detectadas (inyección, XSS, CSRF).
* Ajustar reportes y vistas cuando cambien los requerimientos institucionales.
* Documentar cada cambio en el repositorio con número de versión.

**Conclusiones técnicas**

El sistema fue desarrollado bajo estándares profesionales, cumpliendo los criterios de **seguridad, escalabilidad y mantenibilidad**.  
La arquitectura MVC permitió separar responsabilidades, garantizando un código limpio y fácil de mantener.  
La base de datos normalizada en SQL Server asegura la consistencia de la información, y la interfaz HTML/Bootstrap ofrece una experiencia de usuario intuitiva y accesible.