Instituto Tecnológico CTC Colonia

Analista Programador
Programación 2

Obligatorio Grupal

Carlos Rodriguez

Baltazar Boné
Fausto Clara

2025

Índice

Abstract del Proyecto

Descripción del Proyecto – Clínica Vida Sana

La aplicación Clínica Vida Sana es un sistema de gestión desarrollado en C#, para administrar las consultas médicas de la clínica. Permite registrar pacientes, médicos y turnos, así como gestionar pagos y emitir comprobantes.

El sistema está orientado al personal administrativo, quien puede agendar, modificar o cancelar consultas, verificar disponibilidad de médicos y generar reportes.

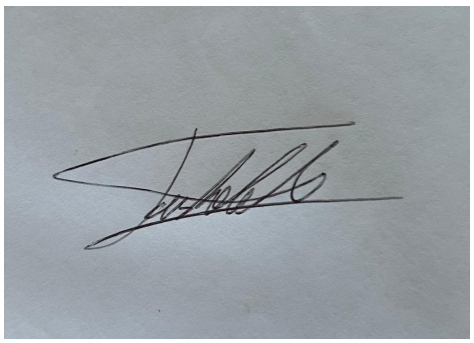
Entre sus principales funcionalidades se incluyen la gestión de usuarios, turnos y pagos, junto con la emisión de estadísticas sobre pacientes, médicos y especialidades.

El diseño del sistema se basa en una estructura modular y en principios de programación orientada a objetos, garantizando claridad, orden y facilidad para futuras ampliaciones.

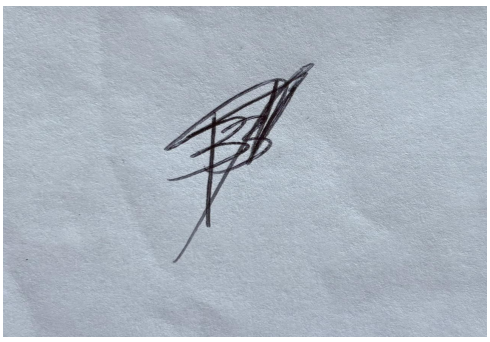
Declaración de Autoría

Nosotros, Fausto Clara y Baltazar Boné, declaramos que el trabajo que se presenta es de nuestra propia mano. Puedo asegurar que:

- La obra fue producida mientras cursaba la materia de Programación 2 con el docente Carlos Rodriguez;
- Hemos tenido en cuenta las clases dictadas por el Docente Carlos Rodriguez, quién además proporcionó el material;
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente que fue construido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.
- Ninguna parte de este trabajo ha sido realizada exclusivamente con Inteligencia Artificial.



Firma Fausto Clara



Firma Baltazar Boné

Cronograma de Trabajo

Fase	Tarea / Actividad	Responsable	Duracion estimada	Fecha de inicio	Fecha de fin
1	Preparación y planificación: definir estructura del proyecto, clases y relaciones	Baltazar y Fausto	**1 día**	**20/10/2025**	**20/10/2025**
2	Implementación de clases básicas: Paciente, Médico, Administrativo, Consulta, Pago	Baltazar y Fausto	**2 días**	**21/10/2025**	**22/10/2025**
3	Implementación de la clase Clinica con métodos CRUD y validaciones	Baltazar y Fausto	**3 días**	**23/10/2025**	**26/10/2025**
4	Carga de datos iniciales: CargaDeDatosIniciales.cs	Baltazar y Fausto	**1 día**	**26/10/2025**	**26/10/2025**
5	Implementación del menú y Program.cs: entradas, validaciones, navegación	Baltazar y Fausto	**3 días**	**27/10/2025**	**29/10/2025**

Diccionario de Clases

Clase: Persona (abstracta)

Descripción: Clase base para todas las personas del sistema (pacientes, médicos, administrativos). Contiene los datos comunes y garantiza la validación básica de información.

Atributos:

ID: int -> Identificador único de la persona (autoasignado).

Nombre: string -> Nombre de la persona.

Apellido: string -> Apellido de la persona.

NombreUsuario: string -> Email o nombre de usuario para login.

Contrasenia: string -> Contraseña de acceso.

Métodos:

ToString(): string -> Devuelve un resumen de la persona (ID, nombre y apellido).

Relaciones: Clase base para Paciente, Medico y Administrativo.

Clase: Paciente

Descripción: Representa a un paciente registrado en la clínica. Hereda de Persona.

Atributos:

NumeroDocumento : string -> Documento de identidad del paciente.

Telefono : string -> Número de teléfono (solo dígitos).

FechaNacimiento : DateTime -> Fecha de nacimiento del paciente.

Email : string -> Correo electrónico del paciente.

ObraSocial : string -> Obra social (opcional).

Métodos:

ToString() : string -> Devuelve la información completa del paciente, incluyendo datos de contacto y obra social.

Relaciones: Un Paciente puede tener múltiples Consulta (turnos médicos).

Clase: Medico

Descripción: Representa a un médico de la clínica. Hereda de Persona.

Atributos:

Especialidad : string -> Ejemplo: Pediatría, Dermatología, Clínica Médica.

Matricula : string -> Matrícula profesional del médico.

DiasAtencion : List<string> -> Días de la semana en que atiende.

HorariosDisponibles : List<TimeSpan> -> Horarios disponibles en bloques de 30 minutos.

Métodos:

ToString() : string -> Devuelve la información completa del médico, incluyendo especialidad, matrícula, días y horarios disponibles.

Relaciones: Un Medico puede tener múltiples Consulta asignadas.

Clase: Administrativo

Descripción: Representa al personal administrativo o recepcionista de la clínica. Hereda de Persona.

Atributos:

NumeroDocumento : string -> Documento de identidad del administrativo.

Métodos:

ToString() : string -> Devuelve la información básica del administrativo, incluyendo su número de documento.

Relaciones: Administra Paciente, Medico, Consulta y Pago dentro del sistema.

Clase: Pago

Descripción: Representa un pago realizado por un paciente para una consulta médica.

Atributos:

ID : int -> Identificador único del pago (autoasignado).

IdConsulta : int -> Identificador de la consulta asociada al pago.

FechaPago : DateTime -> Fecha en que se realizó el pago.

Monto : decimal -> Monto abonado.

Metodo : MetodoPago -> Forma de pago (Efectivo, Débito o Crédito).

Métodos:

ToString() : string -> Devuelve un resumen del pago incluyendo ID, consulta asociada, fecha, monto y método.

Relaciones:

Cada Pago está asociado a una única Consulta.

Enumeraciones:

MetodoPago -> Enum que define los posibles métodos de pago: Efectivo, Debito, Credito.

Clase: Consulta

Descripción: Representa un turno médico agendado entre un paciente y un médico en la clínica.

Atributos:

ID : int → Identificador único de la consulta (autoasignado).

IdPaciente : int → Identificador del paciente asociado.

IdMedico : int → Identificador del médico asignado.

FechaHora : DateTime → Fecha y hora del turno (bloques de 30 minutos).

Estado : EstadoConsulta → Estado del turno (Agendada, Realizada, Cancelada).

Métodos:

Cancelar() : void → Cancela la consulta si no ha sido realizada.

MarcarComoRealizada() : void → Marca la consulta como realizada si no está cancelada.

Reprogramar(DateTime nuevaFechaHora) : void → Cambia la fecha y hora respetando las reglas de anticipación y duración de 30 minutos.

ToString() : string → Devuelve un resumen de la consulta, incluyendo paciente, médico, fecha/hora y estado.

Relaciones:

Cada Consulta está asociada a un Paciente y a un Medico.

Puede estar asociada a un Pago.

Enumeraciones:

EstadoConsulta → Enum con los posibles estados de la consulta:

Agendada

Realizada

Cancelada

Clase: Clinica

Descripción: Clase principal que gestiona todas las entidades del sistema (pacientes, médicos, administrativos, consultas y pagos). Actúa como “controlador” de la clínica, centralizando la lógica de negocio.

Atributos:

pacientes : List<Paciente> → Lista de todos los pacientes registrados.

medicos : List<Medico> → Lista de todos los médicos de la clínica.

administrativos : List<Administrativo> → Lista de todos los usuarios administrativos.

consultas : List<Consulta> → Lista de todas las consultas médicas agendadas.

pagos : List<Pago> → Lista de todos los pagos realizados.

Métodos:

Gestión de pacientes:

AgregarPaciente(Paciente nuevoPaciente) : void → Agrega un paciente, verificando que no exista otro con el mismo documento.

ObtenerPacientes() : List<Paciente> → Devuelve los pacientes ordenados alfabéticamente por apellido.

Gestión de médicos:

AgregarMedico(Medico nuevoMedico) : void → Agrega un médico verificando matrícula única.

ObtenerMedicosPorEspecialidad(string especialidad) : List<Medico> → Devuelve médicos de una especialidad.

ObtenerMedicosPorDia(string dia) : List<Medico> → Devuelve médicos disponibles un día específico.

ObtenerTodosLosMedicos() : List<Medico> → Devuelve todos los médicos.

Gestión de administrativos:

AgregarAdministrativo(Administrativo nuevoAdministrativo) : void → Agrega un administrativo verificando documento único.

Login(string nombreUsuario, string contrasenia) : bool → Verifica credenciales de un administrativo.

VerificarContrasenia(string usuario, string contrasenia) : bool → Verifica la contraseña de un administrativo.

CambiarContrasenia(string usuario, string nuevaContrasenia) : void → Permite cambiar la contraseña de un administrativo.

Gestión de consultas:

AgendarConsulta(int idPaciente, int idMedico, DateTime fechaHora) : void → Agenda una consulta respetando reglas de negocio (máximo 6 turnos, 30 min, 24h de anticipación, no duplicar consulta paciente-médico el mismo día).

CancelarConsulta(int idConsulta) : void → Cancela una consulta existente.

ReprogramarConsulta(int idConsulta, DateTime nuevaFechaHora) : void → Reprograma una consulta verificando reglas de negocio.

ObtenerConsultasPorPaciente(int idPaciente) : List<Consulta> → Devuelve el historial de consultas de un paciente.

ObtenerConsultasPorMedico(int idMedico) : List<Consulta> → Devuelve todas las consultas de un médico.

Gestión de pagos:

RegistrarPago(int idConsulta, decimal monto, MetodoPago metodo, bool emitirComprobante = true) : void → Registra un pago para una consulta.

EmitirComprobante(int idPago) : void → Muestra en consola el comprobante de pago.

ObtenerPagosPorPaciente(int idPaciente) : List<Pago> → Devuelve todos los pagos realizados por un paciente.

Estadísticas y reportes:

ObtenerConsultasMasFrecuentesPorEspecialidad() : List<(string Especialidad, int Cantidad)> → Devuelve ranking de especialidades más consultadas.

ObtenerMedicosMasConsultados() : List<(Medico Medico, int Cantidad)> → Ranking de médicos con más consultas realizadas.

Relaciones:

Contiene y administra listas de Paciente, Medico, Administrativo, Consulta y Pago.

Interactúa con Consulta y Pago aplicando las reglas de negocio.

Clase: CargaDeDatosIniciales

Descripción: Clase estática encargada de inicializar la aplicación con datos de prueba. Permite cargar pacientes, médicos, administrativos, consultas y pagos al iniciar el sistema, facilitando las pruebas y demostraciones.

Atributos:

No posee atributos propios; utiliza la instancia de Clinica que se pasa como parámetro para agregar los datos.

Métodos:

Cargar(Clinica clinica) : void → Carga los datos iniciales:

5 médicos con distintas especialidades y horarios.

10 pacientes con datos personales y obra social.

2 administrativos.

5 consultas agendadas respetando reglas de negocio (30 min, 24h anticipación, máximo 6 turnos por día).

2 pagos asociados a consultas.

Relaciones:

Interactúa con Clinica para agregar objetos.

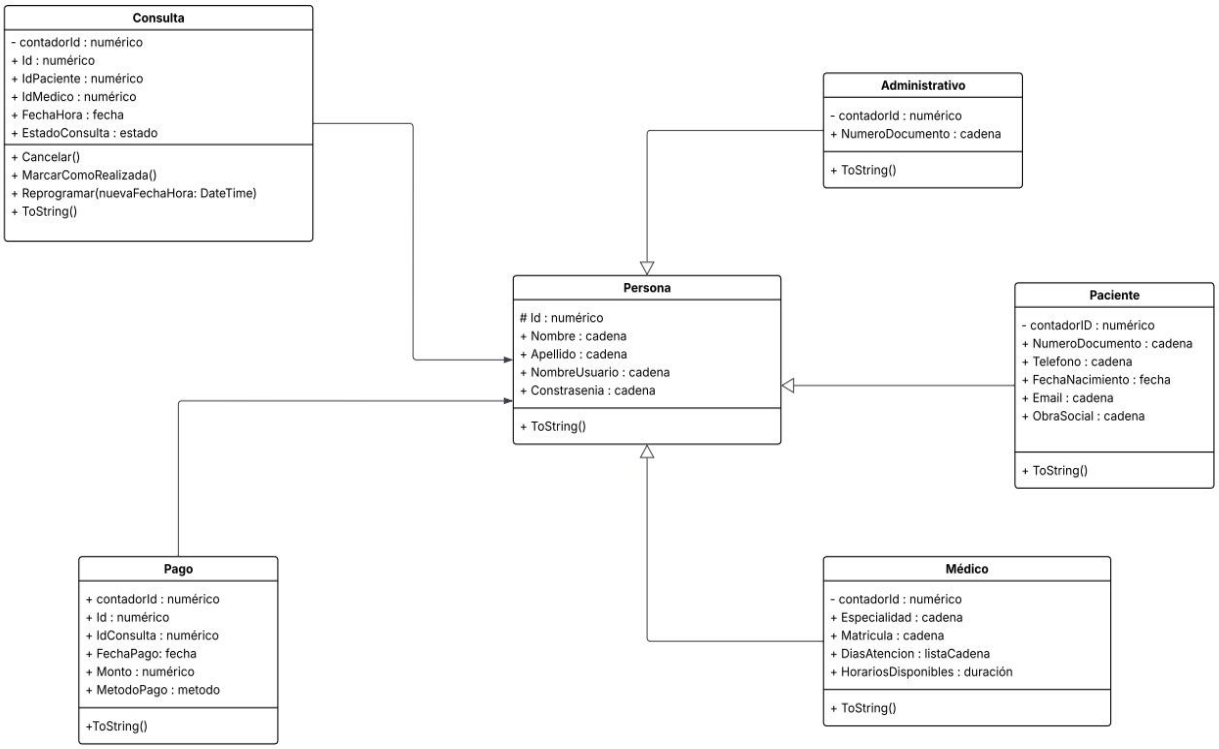
Utiliza las clases Paciente, Medico, Administrativo, Consulta y Pago.

Observaciones:

Es una clase auxiliar, no forma parte de la lógica de negocio central.

Facilita la carga automática de datos para pruebas o demostraciones.

Diagrama UML de Clases



Resultado de Pruebas

PRUEBA	ENTRADA	RESULTADO ESPERADO	RESULTADO OBTENIDO	ESTADO	OBSERVACIONES
Agendar consulta	Paciente ID 1, Médico ID 2, Fecha 30/10/2025 10:00	Consulta agendada correctamente	Consulta agendada correctamente	Aprobada	----------
Cancelar consulta	Consulta ID 1	Estado de consulta cambia a Cancelada	Estado de consulta cambia a Cancelada	Aprobada	----------
Registrar pago	Consulta ID 3, Monto 1500, Efectivo	Pago registrado y comprobante emitido	Pago registrado y comprobante emitido	Aprobada	----------
Reprogramar consulta	Consulta ID 2, Nueva fecha 31/10/2025 09:30	Fecha actualizada, estado Agendada	Error: Médico ya tiene turno en ese horario	Fallida	Conflicto de horario detectado

Juego de Datos Iniciales

```
-- INSERT Pacientes (20)
INSERT INTO dbo.Pacientes (Nombre, Apellido, NombreUsuario,
Contrasenia, NumeroDocumento, FechaNacimiento, Telefono, Email,
ObraSocial) VALUES
('Ana','Gonzalez','ana.gonzalez@example.com','pass123','1234567-1','1990-
05-12','099111111','ana.gonzalez@example.com','OSE'),
('Bruno','Perez','bruno.perez@example.com','pass123','2345678-2','1985-03-
20','099222222','bruno.perez@example.com',''),
('Carla','Rodriguez','carla.rodriguez@example.com','pass123','3456789-
3','1978-08-02','099333333','carla.rodriguez@example.com','ASSE'),
('Diego','Martinez','diego.martinez@example.com','pass123','4567890-
4','1995-12-10','099444444','diego.martinez@example.com',''),
('Elena','Ferreira','elena.ferreira@example.com','pass123','5678901-5','2000-
07-07','099555555','elena.ferreira@example.com',''),
('Federico','Lopez','federico.lopez@example.com','pass123','6789012-
6','1970-01-22','099666666','federico.lopez@example.com',''),
('Gabriela','Silva','gabriela.silva@example.com','pass123','7890123-7','1982-
11-11','099777777','gabriela.silva@example.com',''),
('Hector','Diaz','hector.diaz@example.com','pass123','8901234-8','1999-02-
14','099888888','hector.diaz@example.com',''),
('Irene','Mendez','irene.mendez@example.com','pass123','9012345-9','1988-
09-09','099999999','irene.mendez@example.com',''),
('Javier','Alonso','javier.alonso@example.com','pass123','0123456-0','1993-
06-03','098000000','javier.alonso@example.com',''),
('Karla','Ruiz','karla.ruiz@example.com','pass123','1123456-1','1991-04-
04','098111111','karla.ruiz@example.com',''),
('Luis','Vargas','luis.vargas@example.com','pass123','1223456-2','1987-10-
10','098222222','luis.vargas@example.com',''),
('Marina','Gimenez','marina.gimenez@example.com','pass123','1323456-
3','1996-01-01','098333333','marina.gimenez@example.com',''),
('Nico','Santos','nico.santos@example.com','pass123','1423456-4','1983-08-
08','098444444','nico.santos@example.com',''),
('Olga','Blanco','olga.blanco@example.com','pass123','1523456-5','1975-12-
12','098555555','olga.blanco@example.com',''),
('Pablo','Cortes','pablo.cortes@example.com','pass123','1623456-6','1980-03-
03','098666666','pablo.cortes@example.com',''),
('Queta','Ortega','queta.ortega@example.com','pass123','1723456-7','1994-
05-05','098777777','queta.ortega@example.com',''),
('Ramon','Paz','ramon.paz@example.com','pass123','1823456-8','1969-09-
09','098888888','ramon.paz@example.com',''),
('Sofia','Nunez','sofia.nunez@example.com','pass123','1923456-9','2002-02-
02','098999999','sofia.nunez@example.com',''),
('Tomas','Iglesias','tomas.iglesias@example.com','pass123','2023456-
0','1997-07-07','097000000','tomas.iglesias@example.com','');

-- INSERT Medicos (10)
```

```
INSERT INTO dbo.Medicos (Nombre, Apellido, NombreUsuario, Contrasenia, Especialidad, Matricula, DiasAtencion, HorariosDisponibles) VALUES
('María','Lopez','maria.lopez@clinica.com','med123','Clínica Médica','M001','Lunes;Miercoles;Viernes','09:00;09:30;10:00;10:30;11:00;11:30'),
('Carlos','Fernandez','carlos.fernandez@clinica.com','med123','Pediatría','M002','Martes;Jueves','09:00;09:30;10:00;10:30;11:00;11:30'),
('Lucia','Santos','lucia.santos@clinica.com','med123','Dermatología','M003','Lunes;Martes','08:30;09:00;09:30;10:00;10:30;11:00'),
('Diego','Ramos','diego.ramos@clinica.com','med123','Cardiología','M004','Miercoles;Viernes','13:00;13:30;14:00;14:30;15:00;15:30'),
('Elisa','Vega','elisa.vega@clinica.com','med123','Ginecología','M005','Lunes;Jueves','10:00;10:30;11:00;11:30;12:00;12:30'),
('Felipe','Costa','felipe.costa@clinica.com','med123','Psiquiatría','M006','Martes;Viernes','15:00;15:30;16:00;16:30;17:00;17:30'),
('Gabriela','Ortiz','gabriela.ortiz@clinica.com','med123','Clínica Médica','M007','Miercoles;Jueves','09:00;09:30;10:00;10:30;11:00;11:30'),
('Hugo','Brito','hugo.brito@clinica.com','med123','Traumatología','M008','Lunes;Sabado','08:00;08:30;09:00;09:30;10:00;10:30'),
('Irene','Castro','irene.castro@clinica.com','med123','Endocrinología','M009','Martes;Jueves','11:00;11:30;12:00;12:30;13:00;13:30'),
('Joaquin','Perez','joaquin.perez@clinica.com','med123','Pediatría','M010','Miercoles;Viernes','08:30;09:00;09:30;10:00;10:30;11:00');

-- INSERT Administrativos (2)
INSERT INTO dbo.Administrativos (Nombre, Apellido, NombreUsuario, Contrasenia, NumeroDocumento) VALUES
('Laura','Garcia','laura.garcia@clinica.com','admin123','20123456-1'),
('Martin','Silva','martin.silva@clinica.com','admin123','20123457-2');

-- INSERT Consultas (10) -> fechas futuras en diciembre 2025
INSERT INTO dbo.Consultas (IdPaciente, IdMedico, FechaHora, Estado) VALUES
(1, 1, '2025-12-15 09:00', 0),
(2, 2, '2025-12-15 09:30', 0),
(3, 3, '2025-12-15 08:30', 1),
(4, 4, '2025-12-16 13:00', 0),
(5, 5, '2025-12-16 10:00', 0),
(6, 6, '2025-12-17 15:00', 0),
(7, 7, '2025-12-17 09:00', 1),
(8, 8, '2025-12-18 08:00', 0),
(9, 9, '2025-12-18 11:00', 0),
(10, 10, '2025-12-19 09:30', 0);

-- INSERT Pagos (2)
INSERT INTO dbo.Pagos (IdConsulta, FechaPago, Monto, Metodo) VALUES
(3, '2025-12-15 10:00', 1500.00, 0),
(7, '2025-12-17 09:30', 2000.00, 2);
```