UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

Avance Semestral

Integrantes:
Ian Ramos
Ivan Justavino
Cristhofer Villarreal
Jean Janten

Profesor:
Victor Sarmiento

Grupo:
1S3212

Asignatura:
Progrmacion

Tema:
Sistema de gestion de hospital universitario

2025

Sistema de Gestión de Hospital Universitario

Descripción: Control de pacientes, médicos, citas, tratamientos, historial clínico y farmacia.
Lista de clases:

- Hospital
- Paciente
- Médico
- Cita
- Tratamiento
- Receta
- Farmacia
- Medicamento
- InventarioMedicamentos
- HistorialClinico
- Especialidad
- Usuario
- Rol
- Auditoria
- Administrador

Tabla 1 - Escenarios, clases involucradas, explicación

Escenario clave	Clases involucradas	Explicación breve
Agendar citas y evitar duplicidad	Cita, Paciente, Médico, Especialidad, Usuario	El sistema valida disponibilidad del médico y solapa de horarios; evita citas duplicadas por paciente y hora.
Registrar tratamientos y recetas	Tratamiento, Receta, Médico, Paciente, HistorialClinico	Un médico crea un tratamiento y receta asociada; se guarda en el historial del paciente.
Control de inventario de medicamentos	InventarioMedicamentos, Medicamento, Farmacia, Auditoria	Control de stock, alertas y registro de movimientos.
Generar historial clínico por paciente	HistorialClinico, Paciente, Cita, Tratamiento, Receta	Genera un historial cronológico de consultas y tratamientos.
Organización de médicos por especialidad	Médico, Especialidad, Hospital	Asocia médicos a especialidades y permite filtrar por área médica.
Validación de roles y auditoría de registros	Usuario, Rol, Auditoria, Administrador	Controla el acceso según rol y registra acciones para auditoría.

Tabla 2 - Clases, atributos y métodos
Cada clase incluye sus modificadores de acceso.

Clase	Atributos	Métodos
Hospital	- id: String - nombre: String - direccion: String	+ agregarMedico(m: Médico): void + buscarMedicoPorEspecialidad(e: Especialidad): List<Médico>
Paciente	- idPaciente: String - nombre: String - fechaNacimiento: Date	+ registrar(): void + getEdad(): int
Médico	- idMedico: String - nombre: String - especialidades: List<Especialidad>	+ObtenerCitasenFecha()
Cita	- idCita: String - paciente: Paciente - medico: Médico - fechaHora: DateTime - estado: String	+ confirmar(): void + cancelar(): void
Tratamiento	- idTrat: String - descripcion: String - medico: Médico	+ agregarObservacion(obs: String): void + finalizar(): void
Receta	- idReceta: String - tratamiento: Tratamiento - items: List<Medicamento>	+ agregarItem(m: Medicamento, qty: int): void + validar(): boolean
Farmacia	- idFarmacia: String - inventario: InventarioMedicamentos	+ solicitarMedicamento(m: Medicamento, qty: int): boolean + dispensarReceta(r: Receta): boolean
Medicamento	- idMed: String - nombre: String - precio: double	+ consultarStock(): int
InventarioMedicamentos	- registros: Map<Medicamento, int>	+ registrarEntrada(m: Medicamento, qty: int): void + registrarSalida(m: Medicamento, qty: int): boolean
HistorialClinico	- idHist: String - paciente: Paciente - entradas: List<EntradaHistorial>	+ agregarEntrada(e: EntradaHistorial): void + obtenerCronologico(): List<EntradaHistorial>
Especialidad	- idEsp: String - nombre: String	+ asignarMedico(m: Médico): void
Usuario	- idUsuario: String - username: String - hashPassword: String - rol: Rol	+ autenticar(password: String): boolean

Rol	- idRol: String - nombre: String	+ tienePermiso(perm: String): boolean
Auditoria	- idAudit: String - accion: String - usuario: Usuario - timestamp: LocalDatetime	+ registrarAccion(a: String, u: Usuario): void
Administrador	- usuario: Usuario - nivelAcceso: int - generadorReportes: GeneradorReportes	+ crearUsuario(u: Usuario): void + generarReporteEventos(): Report
EntradaHistofial	- fecha: LocalDateTime - tipo: String - referenciaID: String	+getFecha(): LocalDateTime +getTipo(): String +getReferencia(): String
GestorCitas	- citasRegistradas: List<Cita>	+agendarCita(p: Paciente, m: Medico, fechaHora: LocalDateTime): boolean
GestorPacientes	- pacientesRegistrados: List<Paciente> - medicosRegistrados: List<Medico>	+ registrarPaciente(p: Paciente): Paciente + registrarMedico(m: Medico): Medico + crearHistorial(p: Paciente): HistorialClinico
GeneradorReportes	- auditorias: List<Auditorias - >	+ generarReporteEventos(auditorias: List<Auditoria>): Report

Tabla 3 -Explicación de las relaciones y multiplicidad

Relacion	***Tipo***	***Multiplicidad***
Hospital -Médico	Agregación	Un hospital agrega varios médicos, pero un médico puede pertenecer a varios hospitales.
Médico - Especialidad	Asociación	Un médico puede tener varias especialidades, y una especialidad puede aplicarse a varios médicos.
Médico - Cita	Asociación	Un médico puede tener múltiples citas asignadas.
Paciente - Cita	Asociación	Un paciente puede tener múltiples citas.
Cita – Tratamiento	Dependencia	Un tratamiento puede generarse a partir de una cita concluida.
Tratamiento – Receta	Composición	Cada tratamiento crea una receta que no puede existir fuera de él.
Receta - Medicamento	Asociación	Una receta contiene varios medicamentos, y un medicamento puede estar en varias recetas.
Farmacia - InventarioMedicamentos	Composición	El inventario depende directamente de la farmacia.
InventarioMedicamentos – Medicamento	Agregación	La farmacia mantiene una lista de medicamentos, pero estos pueden existir fuera del inventario.
Paciente - HistorialClinico	Composición	El historial clínico pertenece exclusivamente a un paciente. Si se elimina el paciente, desaparece su historial.
HistorialClinico - EntradaHistorial	Composición	Cada entrada solo existe dentro del historial correspondiente.

Usuario –-Rol	Asociación	Cada usuario tiene un rol único, aunque un rol puede asignarse a varios usuarios.
Administrador -Usuario	Herencia	El administrador hereda atributos y métodos de usuario.
Auditoria - Usuario	Dependencia	La auditoría registra las acciones que realiza un usuario.
EntradaHospital - Historial Clinico	Composición	Un historial clínico puede tener muchas entradas, pero una entrada no existe sin su historiale.
GestorCitas-Citas	Asociación	Un GestorCitas puede gestionar múltiples citas, pero cada cita es gestionada por un único gestor.
GestorCitas - Paciente	Dependencia	Cada vez que GestorCitas agenda una cita, interactúa con un único médico para validar su disponibilidad.
GestorCitas – Medico	Dependencia	Para crear una cita, GestorCitas utiliza exactamente un paciente, ya que una cita no puede existir sin un paciente asociado.