Sistema de Atención de Pacientes - Triage de 3 Niveles

Fase 0: PROYECTO DE PROGRAMACION

Jesus Daniel Daza Bermudez

Andrés Felipe Currea

Andrés Felipe Rodríguez

1. ¿Qué debemos hacer?

Desarrollar un sistema tecnológico para la gestión del proceso de clasificación de pacientes en urgencias mediante Triage de 3 niveles. Este sistema permitirá:

- Registrar pacientes.
- Evaluar su nivel de urgencia según el triage.
- Gestionar la información médica y el historial de cada paciente.
- Asignar camas para casos críticos.
- Administrar el personal médico y de enfermería.
- Gestionar el proceso de facturación según los servicios prestados.

2. ¿Qué se necesita?

Para el desarrollo del sistema, se requieren los siguientes componentes:

- **Entorno de desarrollo en Java** (VS Code o IntelliJ IDEA).

- **Base de datos** para almacenar información de pacientes, médicos, enfermeros y diagnósticos.
- **Interfaz de usuario** para la gestión de registros y la visualización de datos.
- **Algoritmo de clasificación de triage** basado en los síntomas del paciente.
- **Implementación de funcionalidades CRUD** (Crear, Leer, Actualizar, Eliminar) para la gestión de usuarios y datos médicos.
- **Módulo de facturación**, que genere los cobros según los servicios utilizados

3. Elementos (objetos) que se identifican

- **Paciente** (nombre, edad, síntomas, nivel de triage, historial médico).
- **Médico** (nombre, especialidad, ID, pacientes asignados).
- **Enfermero** (nombre, ID, pacientes a su cargo).
- **Triage** (nivel de urgencia, criterios de evaluación, patologías clasificadas).
- **Historial Médico** (diagnósticos, tratamientos, ID del paciente).
- **Factura** (ID del paciente, servicios prestados, costo total).
- **Cama** (número, disponibilidad, ID del paciente asignado).

4. ¿Qué objetos se incluye?

Para estructurar el sistema, se incluyen las siguientes clases en Java:

- **Clase `Persona`** (superclase de Paciente, Médico y Enfermero).

- **Clase `Paciente`** (almacena datos personales, nivel de triage y historial médico).
- **Clase `Medico`** (gestiona diagnósticos y tratamientos de los pacientes asignados).
- **Clase `Enfermero`** (registra pacientes y asigna triage).
- **Clase `Triage`** (determina el nivel de urgencia según los síntomas del paciente).
- **Clase `HistorialMedico`** (almacena diagnósticos, tratamientos y actualizaciones del paciente).
- **Clase `Factura`** (gestiona los cobros por los servicios prestados en urgencias).
- **Clase `Cama`** (controla la asignación de camas a pacientes críticos).
- **Clase `AdministracionPersonal`** (gestiona el registro y asignación de personal médico y de enfermería).

Fase I: Requerimientos

- Elaborar listado de requerimientos del contexto planteado. Donde cada requerimiento se debe documentar haciendo uso de la siguiente tabla.

Nombre	Registro de Pacientes
Descripción	Permite ingresar los datos de un paciente.
Entradas	
Datos del paciente (nombre, edad, síntomas)	
Salidas	
Confirmación de registro	

Nombre	Evaluación de Triage
Descripción	Asigna un nivel de urgencia según los síntomas.
Entradas	
Datos del paciente	
Salidas	
Nivel de triage asignado	

Nombre	Asignación de Cama
Descripción	Si el paciente es crítico, se le asigna una cama.
Entradas	
Nivel de urgencia	
Salidas	
Cama asignada o lista de espera	

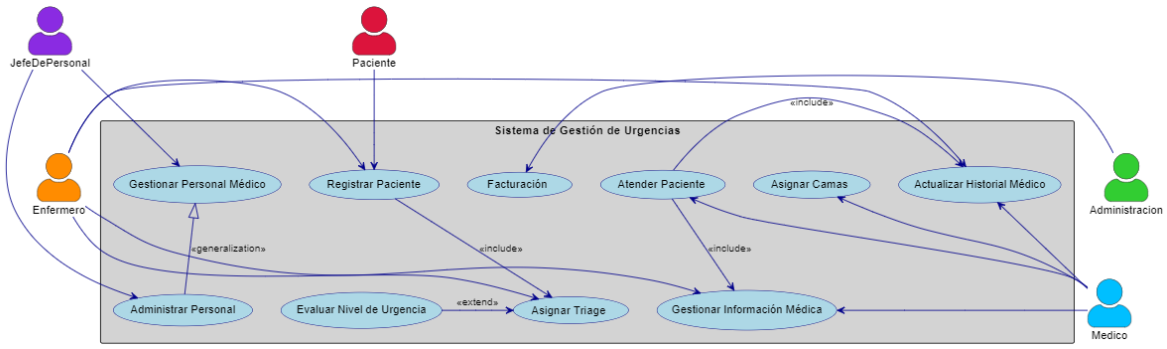
Nombre	Registro de Diagnóstico
Descripción	Permite a un médico registrar su diagnóstico.
Entradas	
ID del paciente, diagnóstico	
Salidas	
Confirmación de registro	

Nombre	Facturación
Descripción	Genera una factura según la atención recibida.
Entradas	
Servicios utilizados	
Salidas	
Monto a pagar	

Fase II Diseño

Etapa de diseño que permite determinar cuáles son las clases que conformaran el sistema

- Elaborar el Diagrama de Casos de Uso

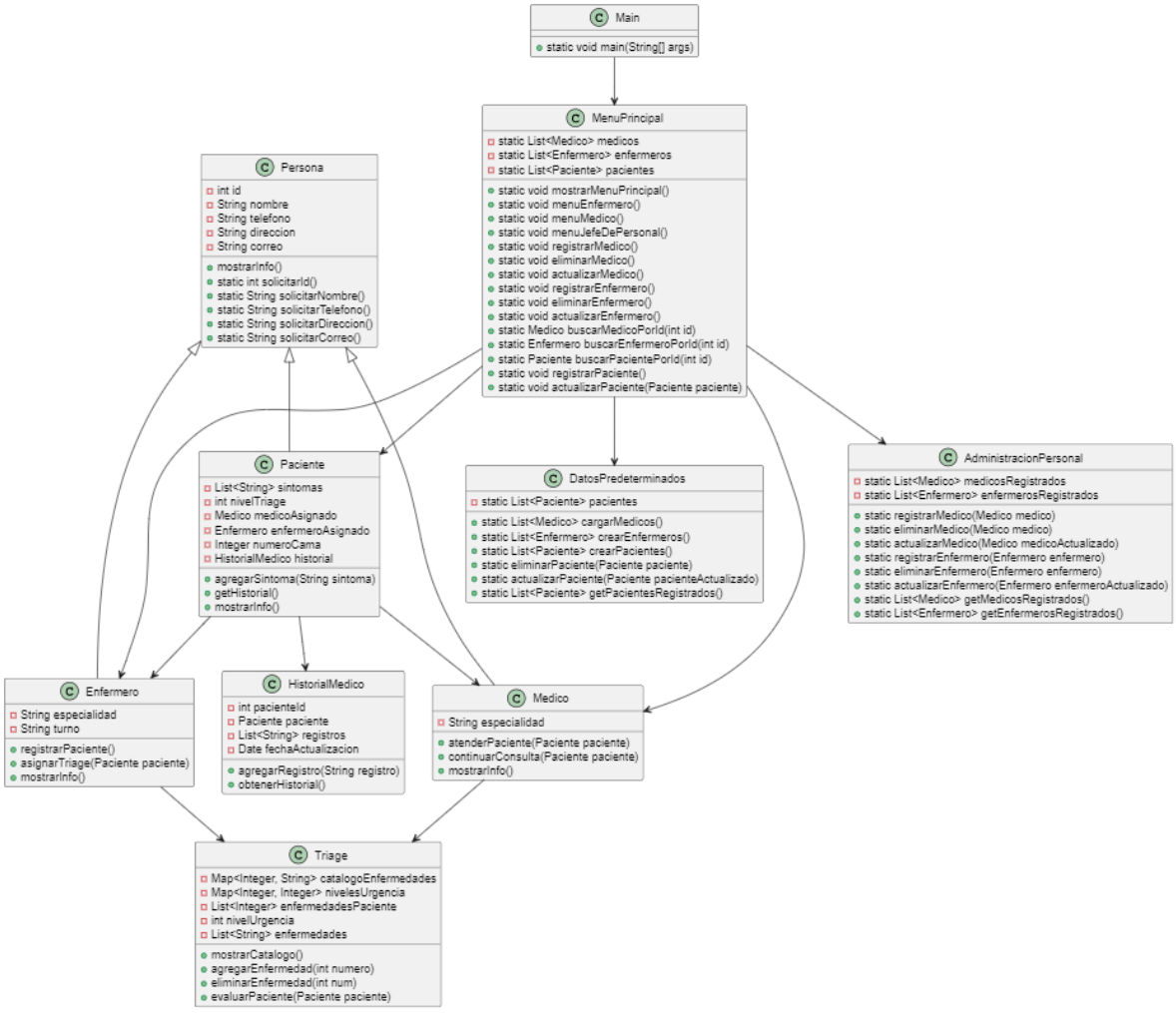


Elaborar lista de sustantivos del enunciado del proyecto para determinar cuáles son las clases del sistema.

- A partir del análisis del problema, se identifican las siguientes clases:
- **Persona** (superclase) → (nombre, ID, contacto).
- **Paciente** (hereda de Persona) → (síntomas, nivel de triage, historial médico).
- **Médico** (hereda de Persona) → (especialidad, pacientes asignados).
- **Enfermero** (hereda de Persona) → (pacientes a su cargo, triage asignado).
- **Triage** → (niveles de urgencia, criterios de evaluación).
- **HistorialMedico** → (diagnósticos, tratamientos, ID de paciente).
- **Factura** → (ID del paciente, servicios prestados, costo total).
- **Cama** → (número, disponibilidad, ID del paciente asignado).
- **AdministracionPersonal** → (gestión de personal médico y enfermeros).

- Elaborar el Diagrama de Clases del Proyecto completo

  Construir representación gráfica de las clases identificadas y asociar, las características a cada clase (atributos) y las operaciones que se van realizar (métodos).



Fase III

En esta fase se da inicio a la codificación de las clases diseñadas en la fase II

- Codificación de los atributos teniendo en cuenta el tipo y la visibilidad

- Construcción de los métodos que realizan las operaciones de las clases.
- Asignar el nombre del método, tipo de retorno de dato, lista de parámetros

Documentación Métodos de la clase

Nombre del Método	mostrarInfo
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Método abstracto que debe ser implementado por las subclases para mostrar la información de la persona.
Firma del Método (encabezado)	public abstract void mostrarInfo();

Nombre del Método	solicitarId
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	int
Solución planteada para el método	Solicita un ID al usuario y valida que sea un número entero mayor que 0 antes de devolverlo.
Firma del Método (encabezado)	`public static int solicitarId()`

Nombre del Método	solicitarNombre
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	String
Solución planteada para el método	Solicita un nombre al usuario y valida que solo contenga letras y espacios antes de devolverlo.
Firma del Método (encabezado)	`public static String solicitarNombre()`

Nombre del Método	solicitarTelefono
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	String
Solución planteada para el método	Solicita un número de teléfono al usuario y valida que tenga exactamente 10 dígitos numéricos.
Firma del Método (encabezado)	`public static String solicitarTelefono()`

Nombre del Método	solicitarDireccion
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	String
Solución planteada para el método	Solicita una dirección al usuario y verifica que no esté vacía antes de devolverla.
Firma del Método (encabezado)	`public static String solicitarDireccion()`

Nombre del Método	solicitarCorreo
Entrada (lista de Parámetros)	Ninguno (Entrada por consola)
Resultado (tipo de dato de retorno)	String
Solución planteada para el método	Solicita un correo electrónico al usuario y valida su formato usando una expresión regular antes de devolverlo.
Firma del Método (encabezado)	public static String solicitarCorreo()

Nombre del Método	agregarSintoma
Entrada (lista de Parámetros)	String sintoma
Resultado (tipo de dato de retorno)	Void
Solución planteada para el método	Agrega un síntoma a la lista de síntomas del paciente.
Firma del Método (encabezado)	`public void agregarSintoma(String sintoma)`

Nombre del Método	asignarMedico
Entrada (lista de Parámetros)	Medico medico
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Asigna un médico al paciente.
Firma del Método (encabezado)	`public void asignarMedico(Medico medico)`

Nombre del Método	asignarEnfermero
Entrada (lista de Parámetros)	Enfermero enfermero
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Asigna un enfermero al paciente.
Firma del Método (encabezado)	`public void asignarEnfermero(Enfermero enfermero)`

Nombre del Método	asignarCama
Entrada (lista de Parámetros)	int numeroCama
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Asigna un número de cama al paciente.
Firma del Método (encabezado)	`public void asignarCama(int numeroCama)`

Nombre del Método	eliminarMedico
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Elimina la asignación del médico al paciente, estableciendo `null` en el atributo correspondiente.
Firma del Método (encabezado)	`public void eliminarMedico()`

Nombre del Método	eliminarEnfermero
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Elimina la asignación del enfermero al paciente, estableciendo `null` en el atributo correspondiente.

Firma del Método (encabezado)	`public void eliminarEnfermero()`

Nombre del Método	mostrarInfo
Entrada (lista de Parámetros)	Ninguno (Por consola)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra en consola la información del paciente, incluyendo datos personales, síntomas, nivel de triage, asignaciones médicas y su historial médico.
Firma del Método (encabezado)	`public void mostrarInfo()`

Nombre del Método	mostrarInfo
Entrada (lista de Parámetros)	Ninguno (Por consola)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra en consola la información del médico, incluyendo su ID, nombre, teléfono, dirección, correo y especialidad.
Firma del Método (encabezado)	`public void mostrarInfo()`

Nombre del Método	atenderPaciente
Entrada (lista de Parámetros)	Paciente paciente
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Imprime un mensaje indicando que el médico está atendiendo al paciente y registra la atención en el historial médico del paciente.
Firma del Método (encabezado)	`public void atenderPaciente(Paciente paciente)`

Nombre del Método	mostrarInfo
Entrada (lista de Parámetros)	Ninguno (Por consola)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra en consola la información del enfermero, incluyendo su ID, nombre, teléfono, dirección, correo, especialidad y turno
Firma del Método (encabezado)	`public void mostrarInfo()`

Nombre del Método	registrarPaciente
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void

Solución planteada para el método	Solicita información para registrar un nuevo paciente, lo crea, lo muestra en consola y agrega un registro en su historial médico. Luego, llama a `asignarTriage` para definir el nivel de urgencia del paciente.
Firma del Método (encabezado)	`public void registrarPaciente()`
Nombre del Método	asignarTriage
Entrada (lista de Parámetros)	Paciente paciente
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra un catálogo de enfermedades, permite al enfermero ingresar enfermedades al historial del paciente y luego evalúa el diagnóstico.
Firma del Método (encabezado)	public void asignarTriage(Paciente paciente)

Nombre del Método	mostrarCatalogo
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra el catálogo de enfermedades con su número correspondiente en la consola.
Firma del Método (encabezado)	public void mostrarCatalogo()

Nombre del Método	agregarEnfermedad
Entrada (lista de Parámetros)	`numero` (int)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega la enfermedad seleccionada por el paciente a su historial y ajusta el nivel de urgencia si es necesario.
Firma del Método (encabezado)	public void agregarEnfermedad(int numero)

Nombre del Método	evaluarPaciente
Entrada (lista de Parámetros)	`paciente` (Paciente)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Muestra el diagnóstico del paciente, su nivel de urgencia y las enfermedades detectadas, ordenadas por nivel de urgencia.
Firma del Método (encabezado)	public void evaluarPaciente(Paciente paciente)

Nombre del Método	**agregarEnfermedad** *(privado)*
Entrada (lista de Parámetros)	`numero` (int), `enfermedad` (String), `nivel` (int)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega una enfermedad al catálogo junto con su nivel de urgencia.
Firma del Método (encabezado)	private void agregarEnfermedad(int numero, String enfermedad, int nivel)

Nombre del Método	**agregarEnfermedades** *(privado)*
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega una lista predefinida de enfermedades con su nivel de urgencia al catálogo.
Firma del Método (encabezado)	private void agregarEnfermedades()

Nombre del Método	registrarMedico
Entrada (lista de Parámetros)	`medico` (Medico)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega un médico a la lista de médicos registrados. `asignarTriage` para definir el nivel de urgencia del paciente.
Firma del Método (encabezado)	`public static void registrarMedico(Medico medico)`

Nombre del Método	eliminarMedico
Entrada (lista de Parámetros)	`medico` (Medico)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Elimina un médico de la lista de médicos registrados.
Firma del Método (encabezado)	`public static void eliminarMedico(Medico medico)`

Nombre del Método	registrarEnfermero
Entrada (lista de Parámetros)	Enfermero (Enfermero)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega un enfermero a la lista de enfermeros registrados.

Firma del Método (encabezado)	public static void registrarEnfermero(Enfermero enfermero)

Nombre del Método	eliminarEnfermero
Entrada (lista de Parámetros)	`enfermero` (Enfermero)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Elimina un enfermero de la lista de enfermeros registrados. historial médico. Luego, llama a `asignarTriage` para definir el nivel de urgencia del paciente.
Firma del Método (encabezado)	`public static void eliminarEnfermero(Enfermero enfermero)`

Nombre del Método	agregarRegistro
Entrada (lista de Parámetros)	`registro` (String)
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Agrega un nuevo registro al historial médico y actualiza la fecha de actualización.
Firma del Método (encabezado)	`public void agregarRegistro(String registro)`

Nombre del Método	actualizarFecha
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	void
Solución planteada para el método	Actualiza la fecha de última modificación del historial médico.
Firma del Método (encabezado)	`private void actualizarFecha()`

Nombre del Método	obtenerHistorial
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	String
Solución planteada para el método	Devuelve una cadena con el historial médico del paciente. Si no hay registros, devuelve un mensaje indicando que el historial está vacío.
Firma del Método (encabezado)	`public String obtenerHistorial()`

Nombre del Método	cargarMedicos
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Medico>
Solución planteada para el método	Crea y devuelve una lista de médicos predefinidos con sus datos.
Firma del Método (encabezado)	`public static List<Medico> cargarMedicos()`

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno

Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	`public static List<Enfermero> cargarEnfermeros()`

Nombre del Método	cargarPacientes
Entrada (lista de Parámetros)	`medicos` (List<Medico>), `enfermeros` (List<Enfermero>)
Resultado (tipo de dato de retorno)	List<Paciente>
Solución planteada para el método	Crea y devuelve una lista de pacientes predefinidos, asignándoles un médico y un enfermero.
Firma del Método (encabezado)	`public static List<Paciente> cargarPacientes(List<Medico> medicos, List<Enfermero> enfermeros)`

Nombre del Método	crearEnfermeros
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Retorna una lista de enfermeros utilizando el método `cargarEnfermeros()`.
Firma del Método (encabezado)	`public static List<Enfermero> crearEnfermeros()`

Nombre del Método	crearPacientes
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Paciente>
Solución planteada para el método	Carga listas de médicos y enfermeros y las usa para crear una lista de pacientes.
Firma del Método (encabezado)	`public static List<Paciente> crearPacientes()`

Nombre del Método	crearMedicos
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Medico>
Solución planteada para el método	Retorna una lista de médicos utilizando el método `cargarMedicos()`.
Firma del Método (encabezado)	`public static List<Medico> crearMedicos()`

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	public static List<Enfermero> cargarEnfermeros()

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	public static List<Enfermero> cargarEnfermeros()

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	public static List<Enfermero> cargarEnfermeros()

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno

Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	`public static List<Enfermero> cargarEnfermeros()`

Nombre del Método	cargarEnfermeros
Entrada (lista de Parámetros)	Ninguno
Resultado (tipo de dato de retorno)	List<Enfermero>
Solución planteada para el método	Crea y devuelve una lista de enfermeros predefinidos con sus datos
Firma del Método (encabezado)	`public static List<Enfermero> cargarEnfermeros()`

Fase IV (Construcción del Código en Lenguaje Java en el IDE Netbeans)

- Implementación de los métodos (Construcción de la parte lógica)
- Codificación de formulas
- Herencia, Polimorfismo, Clases Abstractas, Interfaces, Clases Genéricas, Patrones de diseño, Principios SOLID,

- Aplicación de todo lo aprendido en el curso de POO según enunciados y cumplimento de todos los requerimientos de su proyecto. Cumplir con lo descrito en la sección de INSTRUCCIONES del documento **Banco de Proyectos POO Enunciados curso C1A**

  https://github.com/afelipe096/Triage-de-Urgencias-.git

Fase V

- Verificar que el programa cumpla con los requerimientos establecidos según requerimientos o necesidades (Ejecución sin errores).
- Construcción del documento final de entrega
- Adjuntar evidencias de resultados de la ejecución (pantallazos).
- Sustentar el proyecto en la semana de parciales de Segundo y tercer corte, cada integrante del grupo debe participar.