Universidad de las Fuerzas Armadas – ESPE

Departamento de Ciencias de la Computación

Análisis y Diseño

Integrantes: Isaac Escobar

Eduardo Mortensen

Diego Ponce

NRC: 22426

Taller: DISEÑO CON CASOS DE USO Y DIAGRAMA DE CLASES

2. Suponga que el MHC-PMS Ver Cap5 Somerville9 los modelos se desarrollará usando un enfoque orientado a objetos. Dibuje un diagrama de caso de uso, que muestre al menos seis posibles casos de uso para este sistema.

Análisis del sistema para realizar el diagrama de casos de uso:

Paciente

- **Acciones:**
  - **Solicitar o Reprogramar Citas:** El paciente puede agendar o modificar sus citas de manera autónoma mediante la interfaz del sistema.
  - **Consultar Historial Médico:** Puede acceder a sus registros clínicos, diagnosticar y ver el tratamiento que ha seguido.
  - Realizar Pagos: Tiene la posibilidad de efectuar pagos en línea por los servicios prestados, ayudando a mantener un registro de las transacciones realizadas.

Médico/Psiquiatra

- **Acciones:**
    - **Diagnosticar y Prescribir Tratamientos:** Realiza los diagnósticos y prescribe el tratamiento adecuado para cada paciente.
    - **Consultar y Actualizar el Historial Médico:** Accede al historial clínico de los pacientes para evaluar su evolución e ingresar nuevas anotaciones, diagnósticos o tratamientos.

Recepcionista

- **Acciones:**
    - **Registrar Pacientes:** Ingresa y gestiona la información de los nuevos pacientes en el sistema.
    - **Programar y Gestionar Citas:** Se ocupa de organizar la agenda, asignar citas y, en caso necesario, reprogramarlas.
    - **Asistir en el Proceso de Pagos:** Facilita y verifica que el proceso de pago se realice correctamente, actuando como enlace entre la administración y el paciente.

Administrador

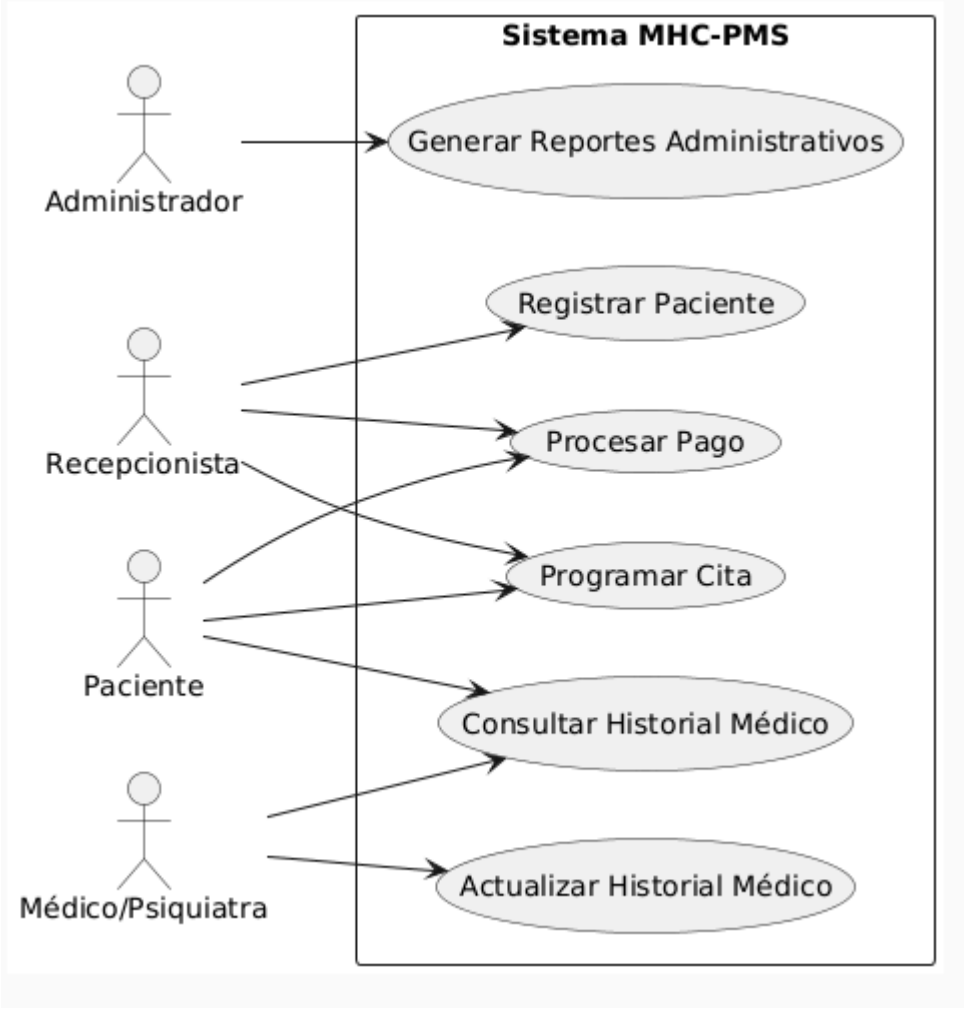
- **Acciones:**
    - **Generar Reportes Administrativos:** Recopila y analiza datos operativos, como estadísticas de citas, ingresos derivados de pagos y otros indicadores del rendimiento del sistema, para la toma de decisiones estratégicas.

Código PlantUML:

```
@startuml
left to right direction

actor "Administrador" as Admin
actor "Recepcionista" as Recep
actor "Médico/Psiquiatra" as Medico
actor "Paciente" as Paciente

rectangle "Sistema MHC-PMS" {
   usecase "Registrar Paciente" as UC1
```

```
    usecase "Programar Cita" as UC2
    usecase "Consultar Historial Médico" as UC3
    usecase "Actualizar Historial Médico" as UC4
    usecase "Procesar Pago" as UC5
    usecase "Generar Reportes Administrativos" as UC6
}

Admin --> UC6
Recep --> UC1
Paciente --> UC2
Recep --> UC2
Medico --> UC3
Paciente --> UC3
Medico --> UC4
Paciente --> UC5
Recep --> UC5
@enduml
```

Diagrama Generado de PlantUML



LINK PlantUML:
[VP7DQkim48NtUeh1-t39J-xoak35pSPW4le0Wpnr1sGf7OsxQEa3zJdwOhKLYU8KwjqvvplYM3f4HT5nSCRHeu864EwV53eMiih16uDMWq1rx0Rs75Mm2r81HaXEYSzawMevp0YcE3abRZu_EhRXJnlvUMHKkJ4DpMw1MhHCNYb5HHWpBq7VEuBgmb5fG6XEzQfjBXMy6PY-CPB5lFEAFaqKM9prKA-NN2kX5nmchcR5p6oMJ1ry79rEp8cZ1c5qa7yZ5xRBmj7gY8vVVsdiViom5DFQlY3x9VAVFCb4dEaQH2d2zpielulOMt_Dkp4fWjNgNpBl_MUrDkK-ih4fSLQtXyXoUqzlb_dkFj-Nqx8waE_6mNq1 (468×486)]

3. Con la notación gráfica UML para clases de objetos, diseñe las siguientes clases de objetos, e identifique los atributos y las operaciones. Use su experiencia para decidir sobre los atributos y las operaciones que deban asociarse con estos objetos:

■ Teléfono.

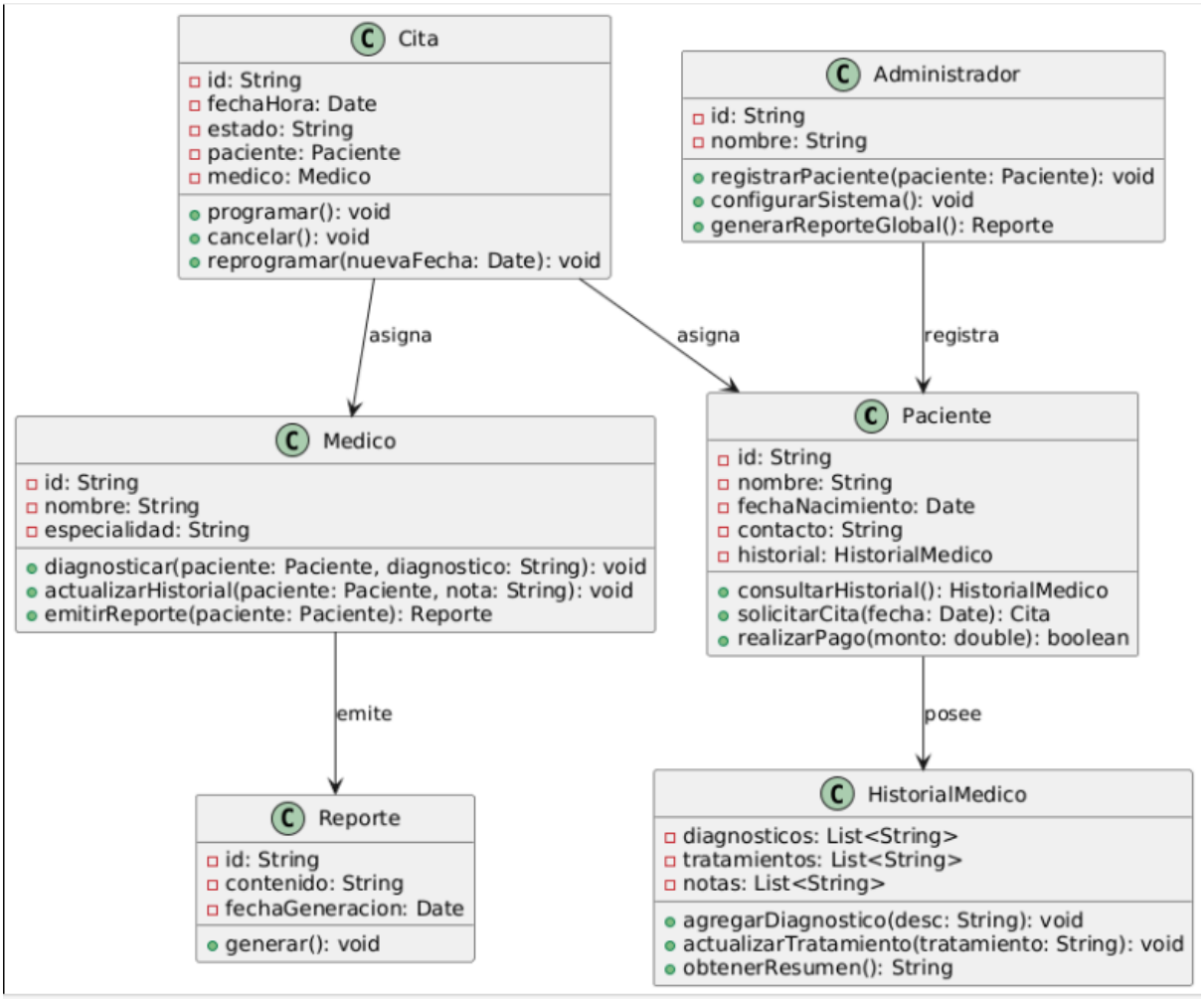
■ Impresora para computadora personal.
■ Sistema de estéreo personal.
■ Cuenta bancaria.
■ Catálogo de biblioteca.

Código de PlantUML:

```
@startuml
' Definición de la clase Paciente
class Paciente {
  -id: String
  -nombre: String
  -fechaNacimiento: Date
  -contacto: String
  -historial: HistorialMedico
  +consultarHistorial(): HistorialMedico
  +solicitarCita(fecha: Date): Cita
  +realizarPago(monto: double): boolean
}

' Definición de la clase HistorialMedico
class HistorialMedico {
  -diagnosticos: List<String>
  -tratamientos: List<String>
  -notas: List<String>
  +agregarDiagnostico(desc: String): void
  +actualizarTratamiento(tratamiento: String): void
  +obtenerResumen(): String
}

' Definición de la clase Medico/Psiquiatra
class Medico {
  -id: String
  -nombre: String
  -especialidad: String
  +diagnosticar(paciente: Paciente, diagnostico: String): void
  +actualizarHistorial(paciente: Paciente, nota: String): void
  +emitirReporte(paciente: Paciente): Reporte
}

' Definición de la clase Cita
class Cita {
```

```
  -id: String
  -fechaHora: Date
  -estado: String
  -paciente: Paciente
  -medico: Medico
  +programar(): void
  +cancelar(): void
  +reprogramar(nuevaFecha: Date): void
}

' Definición de la clase Administrador
class Administrador {
  -id: String
  -nombre: String
  +registrarPaciente(paciente: Paciente): void
  +configurarSistema(): void
  +generarReporteGlobal(): Reporte
}

' Definición de la clase Reporte
class Reporte {
  -id: String
  -contenido: String
  -fechaGeneracion: Date
  +generar(): void
}

' Relaciones entre clases
Paciente --> HistorialMedico : posee
Medico --> Reporte : emite
Cita --> Paciente : asigna
Cita --> Medico : asigna
Administrador --> Paciente : registra
Administrador --> "Sistema MHC-PMS" : configura
@enduml
```

Diagrama Generado de PlantUML:



LINK PlantUML:

ZLH1Zjiw3Dtx5DfzVqo3xerYC4M3pYpQ8fZf1HY9yH2GnPIYPz6Yf-eHUh5ALXmx
YH5a5_6zP_4zKdc82g9zvwl_p1ft5CZItp_1E3GUZFKGqMp04WR5AX_ZyMX-LS
Qio3NcHOL2cq-1kwtWlB93-mhVagJB8cxC6XGpO3aeMEKv-PMYiX3unZoDFx-Y
8yi9lKkAs7i5EMBr_ul4o3xPICHFf50F7PHx4psNCaSGFFq4sK3BTST3QuxxhS-a
BRD72DNlwaeiPnUNTCwh9lH7q0QEcWgnCLyIvKCnVPzH5L0eyIoWWHKkotVG
2hOWw-dBjSDenpIJYJScDp2jziNhz-cYUdRfWeQtYW7b6MFVOSWn5yhrH8hbzvj
8Fte25JXaKkgtxGl6FLe2JmxcpBifGf1wVrZ1vhYCxymiuwiXJCkpz9MSze8SEr9A
SUnP51U4YNe0hoPKjg-4adykH58Up1CBJ4y5euBZEUcoWrpjq0tkfvUm5su5Ef1
wPiP2iEXFQu8JDVJu1f-dLpFIhZhxwBfKZohWM8h5qzejuyzzj8D0Hb-BOKzMEEo
exHF_9ScmWxcdDgymZ5DxzBm5NzyugP5Kd8od9G-MWs8WnnT3V1pkjyIXv3Z
hgJx9z1dzGCDeCAXWQI5Mes6pMjrV_B4qPiyHiIh7J1cxJ53ULwpoYcLWJ2uZ4A
aDC455FW6dCpiJZzEf7Z2ullF_0000 (808×669)

Justificación (Análisis):

Sección 1: Clases, Acciones y Atributos (Con Justificación)

1. **Clase: Paciente**
    - **Atributos:**
        - `id: String` – Identifica unívocamente a cada paciente.
        - `nombre: String` – Almacena el nombre completo del paciente para efectos de identificación y tratamiento.
        - `fechaNacimiento: Date` – Permite conocer la edad, lo cual puede influir en las decisiones de tratamiento.
        - `contacto: String` – Facilita la comunicación con el paciente.
        - `historial: HistorialMedico` – Relaciona a cada paciente con su historial clínico.
    - **Operaciones (Acciones):**
        - `consultarHistorial(): HistorialMedico` – Permite que el paciente vea su historial de diagnósticos y tratamientos.
        - `solicitarCita(fecha: Date): Cita` – Habilita la gestión del agendamiento de citas.
        - `realizarPago(monto: double): Boolean` – Gestiona el proceso de pago en línea por los servicios médicos.
    - **Justificación:** El paciente es el actor central del sistema. Sus atributos básicos aseguran la identificación y comunicación, mientras que las operaciones responden a las necesidades de auto-gestión (consulta, agendamiento y pagos) que demandan sistemas modernos de atención en salud.
2. **Clase: HistorialMedico**
    - **Atributos:**
        - `diagnosticos: List<String>` – Registra los diagnósticos emitidos en cada consulta.

        - `tratamientos: List<String>` – Lista los tratamientos aplicados o en curso.
        - `notas: List<String>` – Almacena observaciones u otros comentarios clínicos relevantes.
    - **Operaciones (Acciones):**
        - `agregarDiagnostico(desc: String): void` – Permite incluir nuevos diagnósticos en el historial.
        - `actualizarTratamiento(tratamiento: String): void` – Facilita la incorporación o modificación de tratamientos.
        - `obtenerResumen(): String` – Ofrece un resumen consolidado del historial completo.
    - **Justificación:** El historial médico constituye el núcleo de la información clínica de cada paciente. Separarlo en una clase independiente ayuda a mantener la integridad de la información y facilita su actualización y consulta, respetando el principio de encapsulación.
3. **Clase: Medico (o Psiquiatra)**
    - **Atributos:**
        - `id: String`, `nombre: String` – Identifican al profesional.
        - `especialidad: String` – Especifica la rama de la medicina en la que se especializa, relevante para derivar tratamientos adecuados.
    - **Operaciones (Acciones):**
        - `diagnosticar(paciente: Paciente, diagnostico: String): void` – Registra el diagnóstico basado en la evaluación clínica del paciente.
        - `actualizarHistorial(paciente: Paciente, nota: String): void` – Agrega o modifica observaciones en el historial del paciente.
        - `emitirReporte(paciente: Paciente): Reporte` – Genera un reporte sobre el estado y evolución del paciente.
    - **Justificación:** Como proveedor principal del servicio de atención, el médico necesita operaciones para documentar diagnósticos y

tratamientos, así como para generar reportes que aseguren la continuidad del cuidado y faciliten la evaluación de resultados.

4. **Clase: Cita**
    - **Atributos:**
        - `id: String` – Identificador único de la cita.
        - `fechaHora: Date` – Indica cuándo se llevará a cabo la cita.
        - `estado: String` – Permite conocer el estado actual de la cita (programada, cancelada, reprogramada).
        - `paciente: Paciente` – Referencia al paciente asociado.
        - `medico: Medico` – Referencia al médico asignado.
    - **Operaciones (Acciones):**
        - `programar(): void` – Define y fija la cita inicial.
        - `cancelar(): void` – Permite cancelar la cita si fuese necesario.
        - `reprogramar(nuevaFecha: Date): void` – Actualiza la fecha y hora de la cita.
    - **Justificación:** La clase Cita es esencial para articular la interacción temporal entre pacientes y profesionales. Su diseño facilita la gestión de la agenda, contribuyendo a la organización y al manejo de imprevistos.
5. **Clase: Administrador**
    - **Atributos:**
        - `id: String`
        - `nombre: String`
    - **Operaciones (Acciones):**
        - `registrarPaciente(paciente: Paciente): void` – Permite el registro formal de nuevos pacientes en el sistema.
        - `configurarSistema(): void` – Habilita la modificación de parámetros y configuración general del sistema.
        - `generarReporteGlobal(): Reporte` – Consolida datos del sistema para generar reportes a nivel global.
    - **Justificación:** El administrador es clave para garantizar el correcto funcionamiento y mantenimiento del sistema. Sus operaciones

aseguran el control sobre el ingreso de datos y la configuración de todo el entorno, lo cual es indispensable para la seguridad y eficiencia.

6. **Clase: Reporte**
    - **Atributos:**
        - `id: String`
        - `contenido: String` – Representa la información detallada del reporte.
        - `fechaGeneracion: Date` – Fecha en que se generó el reporte.
    - **Operaciones (Acciones):**
        - `generar(): void` – Procesa y consolida la información necesaria para la elaboración del reporte.
    - **Justificación:** Los reportes son fundamentales para evaluar el desempeño del sistema, el seguimiento del tratamiento de los pacientes y para tomar decisiones estratégicas. Esta clase permite transformar datos operativos en información clara y procesable.

Sección 2: Relaciones Entre Clases y su Justificación

1. **Paciente → HistorialMedico ("posee")**
    - **Descripción:** Cada objeto `Paciente` contiene una instancia de `HistorialMedico`.
    - **Justificación:** Esta relación, generalmente una **composición**, indica que el historial es parte integral del perfil del paciente. Sin un paciente, no tendría sentido un historial clínico, lo que garantiza que la información clínica esté siempre asociada a un usuario específico.
2. **Cita → Paciente**
    - **Descripción:** Cada `Cita` está asociada a un `Paciente` que solicita o tiene asignada la cita.
    - **Justificación:** La existencia de una cita depende de la participación activa del paciente, asegurando que la programación y gestión de citas sean dirigidas a un usuario específico.
3. **Cita → Medico**
    - **Descripción:** Cada `Cita` también se vincula a un `Medico`

responsable de la atención del paciente durante esa cita.
    - **Justificación:** Esta relación es esencial para coordinar el servicio de atención médica, garantizando que cada cita tenga un profesional asignado para brindar el tratamiento correspondiente.
4. **Medico → Reporte**
    - **Descripción:** El `Medico` emite o genera un `Reporte` relacionado con la evolución y estado del paciente.
    - **Justificación:** Permite evaluar el seguimiento clínico a través de reportes detallados, facilitando la toma de decisiones y la mejora continua del servicio de salud mental.
5. **Administrador → Paciente**
    - **Descripción:** El `Administrador` utiliza la operación `registrarPaciente(paciente)` para incorporar nuevos pacientes al sistema.
    - **Justificación:** Garantiza un control centralizado y seguro del ingreso de datos a la plataforma, fundamental para la integridad y la actualización de la base de datos de usuarios.
6. **Administrador → Sistema MHC-PMS (Configuración del Sistema)**
    - **Descripción:** A través de la operación `configurarSistema()`, el `Administrador` establece parámetros operativos y de seguridad para el funcionamiento del sistema.
    - **Justificación:** Esta relación permite que el administrador supervise y ajuste el entorno del sistema para asegurar su estabilidad, rendimiento y la conformidad con normativas de seguridad y manejo de datos.