DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA: INGENIERÍA DE SOFTWARE

NRC: 27837

ASIGNATURA: ANÁLISIS Y DISEÑO DE SOFTWARE

TEMA: Taller de Pensamiento Crítico con CU (entrevista) y con CU (PlantUML)

Nombre:

- Marcelo Acuña
- Abner Arboleda
- Christian Bonifaz

DOCENTE: PhD. Jenny Ruiz

FECHA: 21 de Octubre del 2025

Parte 1. Lectura y análisis

RF01 - Gestión de Medicamentos

Número de requisito	RF01
Nombre de requisito	Gestión de Medicamentos
Característica	Registrar, editar y eliminar medicamentos ilimitadamente
Descripción del requisito	El usuario podrá ingresar nombre del medicamento, presentación, dosis, frecuencia, horario, duración y notas. Se permite personalizar con íconos y recordatorios. Medicamentos pueden modificarse o eliminarse en cualquier momento.
Fuente del requisito	Análisis funcional Medisafe
Requisito relacionado	RNF01, RNF02, RNF03, RNF05, RNF06
Prioridad del requisito	Alta
Versión del requisito	V1.0

3.2.1 RF01 - Gestión de Medicamentos
El sistema permitirá registrar, editar y eliminar medicamentos sin restricciones de cantidad.

- El usuario podrá ingresar datos como nombre del medicamento, presentación (pastilla, jarabe, etc.), dosis, frecuencia, horario, duración y notas adicionales.
- Los medicamentos podrán personalizarse con íconos y configuraciones de recordatorio.
- Se podrán modificar o eliminar registros en cualquier momento.

Identifica:

1. **El actor principal en cada requisito.**

   Actor: El Usuario, pues en la descripción se indica que: "El usuario podrá ingresar datos...", siendo este la entidad que interactúa con estas funciones.

2. **Las acciones principales que el sistema debe ejecutar.**

   Las acciones que el sistema debe permitir son:

   - Registrar Medicamentos
   - Editar Medicamentos (mencionado como "editar" y "modificar")
   - Eliminar Medicamentos
   - Personalizar Medicamentos (mencionado como "personalizarse con íconos y configuraciones de recordatorio")

   Cabe mencionar que el ingreso de datos como "nombre, presentación, dosis" no es una acción separada, sino los atributos o el flujo principal que ocurre durante la acción de Registrar o Editar.

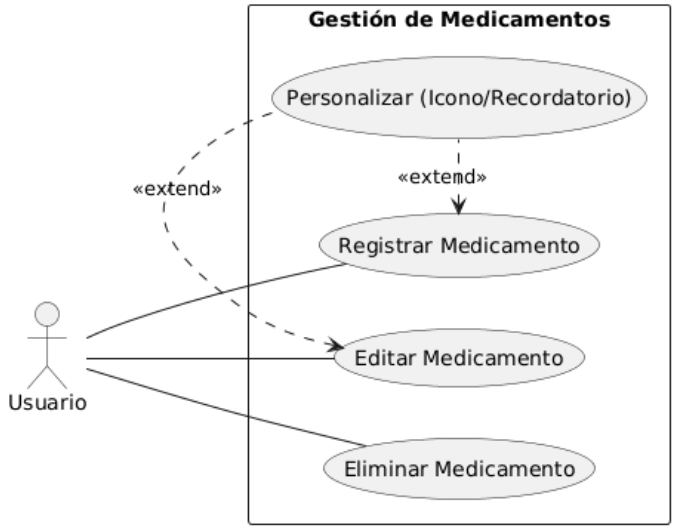
3. **Las relaciones de inclusión (<<include>>) o extensión (<<extend>>) que aparecen en los diagramas UML.**

- Relación <<extend>> (Opcional):
    - Como se menciona en la descripción dice: "Los medicamentos podrán personalizarse...".
    - La palabra clave aquí es "podrán", lo que implica que la personalización es una funcionalidad opcional.
    - Un usuario puede registrar o editar un medicamento y elegir si desea o no personalizarlo, por lo tanto, Personalizar Medicamentos es una extensión de los casos de uso Registrar Medicamento y Editar Medicamento.
- Relación <<include>> (Obligatoria):
    - Una relación <<include>> implica que una acción debe ocurrir obligatoriamente como parte de otra, y usualmente se usa para no repetir lógica.
    - En este requisito, no existe en la descripción una relación <<include>>, las acciones de Registrar, Editar y Eliminar son independientes y la personalización es opcional (extend), no obligatoria (include).

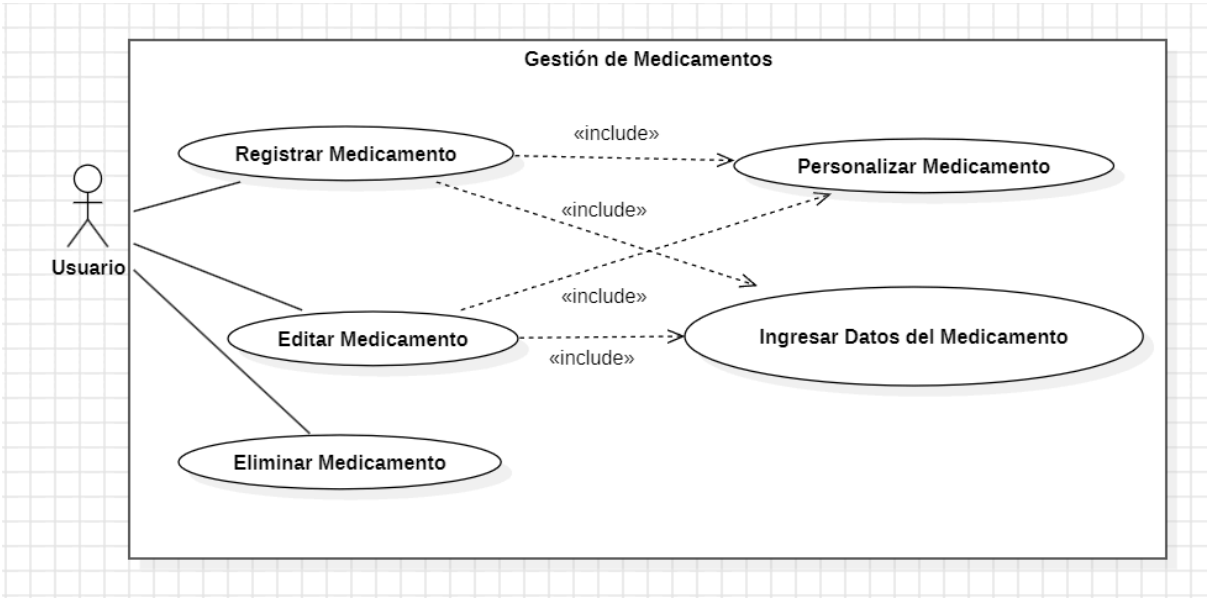
Caso de Uso en PlantUML (Analizado)

Código:

```
@startuml
left to right direction
actor Usuario

rectangle "Gestión de Medicamentos" {
   usecase "Registrar Medicamento" as UC1
   usecase "Editar Medicamento" as UC2
   usecase "Eliminar Medicamento" as UC3
   usecase "Personalizar (Icono/Recordatorio)" as UC_Pers

   Usuario -- UC1
   Usuario -- UC2
   Usuario -- UC3

   (UC1) <. (UC_Pers) : <<extend>>
   (UC2) <. (UC_Pers) : <<extend>>
}
@enduml
```



Caso de Uso en StartUML (Generado con IA)



Comparación

1. ¿Qué diferencias observas entre los Casos de Uso derivados de entrevistas o descripciones textuales y los Casos de Uso generados automáticamente en PlantUML? (Piensa en la precisión, completitud y claridad visual del modelo.)

La diferencia clave no es la cantidad de casos de uso, sino el significado de las relaciones.

- **Precisión:**
    - **Modelo Analizado:** Usa <<extend>>, lo que interpreta correctamente que la personalización es opcional.
    - **Modelo "IA":** Usa <<include>> lo que es un error de análisis crítico, pues <<include>> significa que la personalización es obligatoria y el diagrama está forzando al usuario a personalizar un medicamento *cada vez* que lo registra o edita, lo cual contradice el requisito.
- **Completitud y Claridad:**

- **Modelo "IA":** Intenta ser más completo creando un caso de uso separado para "Ingresar Datos del Medicamento".
- **Problema:** Esto es un error de modelado pues "Ingresar datos" no es un objetivo del usuario; es el flujo de pasos dentro del objetivo "Registrar Medicamento".
- **Resultado:** El diagrama automático es menos claro y mezcla niveles de abstracción confundiendo objetivos con pasos, lo que lo hace más confuso.

En resumen el modelo analizado es semánticamente correcto y claro porque modela los objetivos opcionales del usuario, por otro lado el modelo automático es semánticamente incorrecto forzando la personalización y confuso modelando pasos como si fueran objetivos.

2. ¿De qué manera el uso de PlantUML facilita (o limita) el trabajo del analista al modelar los requisitos funcionales? (Considera aspectos como automatización, trazabilidad y comunicación entre analista y desarrollador.)

PlantUML es una herramienta de doble filo, ya que es fantástica para la agilidad, pero peligrosa si no se domina la semántica de UML.

Cómo Facilita (Automatización y Trazabilidad)

1. **Agilidad y Automatización:** Es increíblemente rápido y un analista puede modificar el texto y regenerar el diagrama en segundos siendo esto ideal para iterar en vivo durante una entrevista de requisitos.
2. **Trazabilidad:** Como el diagrama es código de texto, se puede guardar en un sistema de control de versiones como Git.
    - **Analista-Desarrollador:** El desarrollador y el analista pueden ver el historial de cambios del diagrama para entender cuándo y por qué cambió un requisito.
3. **Consistencia:** El motor de PlantUML se encarga del diseño, asegurando que todos los diagramas de la empresa tengan el mismo estilo visual.

Cómo Limita (Rigidez y Falsa Confianza)

1. **Rigidez del Auto-Layout:** El analista pierde el control visual fino pues no se puede arrastrar los objetos de manera personalizable y en diagramas muy complejos, el diseño automático puede volverse caótico y difícil de leer.
2. **Enfoque en la Sintaxis, No en la Semántica:**
    - Una IA puede centrarse en hacer que el diagrama compile teniendo una sintaxis correcta en lugar de asegurar que el diagrama signifique lo correcto.
    - En el diagrama automático, la IA usó la sintaxis de `include` sin pensar si el requisito era opcional u obligatorio.
3. **Curva de Aprendizaje:** Aunque es simple, requiere aprender la sintaxis, lo que puede ser una barrera inicial para analistas acostumbrados a herramientas visuales de arrastrar y soltar.

RF02 - Recordatorios de Medicación

Número de requisito	RF02
Nombre de requisito	Recordatorios de Medicación
Característica	Generar alertas automáticas de medicación
Descripción del requisito	Las alertas incluirán sonido, vibración y mensajes en pantalla, incluso bloqueada. El usuario podrá marcar como tomada, posponer o registrar omisión con motivo. Las notificaciones persistirán hasta que se realice una acción.
Fuente del requisito	Análisis funcional Medisafe + mejora propuesta
Requisito relacionado	RNF01, RNF02, RNF03, RNF04, RNF05
Prioridad del requisito	Alta
Versión del requisito	V1.0

3.2.2 RF02 - Recordatorios de Medicación

El sistema generará alertas automáticas para cada toma de medicamento programada.
• Las alertas incluirán sonido, vibración y mensaje en pantalla, incluso en la pantalla de bloqueo.
• El usuario podrá marcar la dosis como tomar, posponer o indicar que fue omitida y registrar el motivo.
• Las notificaciones persistirán hasta que se realice una acción.

1. El actor principal en cada requisito.
    a. **Paciente:** Es el usuario que interactúa con las notificaciones.
2. Las acciones principales que el sistema debe ejecutar.
    a. **(Marcar Dosis como Tomada):** Una acción opcional que el Paciente puede realizar.
    b. **(Posponer Recordatorio):** Otra acción opcional del Paciente.
    c. **(Registrar Dosis Omitida):** La tercera acción opcional del Paciente.
3. Las relaciones de inclusión (<<include>>) o extensión (<<extend>>) que aparecen en los diagramas UML.
    a. Usamos una relación de ***extensión*** para mostrar que los casos de uso UC_Tomada, UC_Posponer y UC_Omitida son opcionales y solo pueden ocurrir bajo la condición del caso de uso base (UC_Alert). Es decir, no puedes marcar una dosis como tomada si la alerta no se ha generado primero.

Código:

```
@startuml
left to right direction

actor Usuario

rectangle "Recordatorios de Medicación" {
   usecase "Responder a Alerta" as UC_Alert
   usecase "Marcar Dosis Tomada" as UC_Taken
   usecase "Posponer Dosis" as UC_Postpone
```

```
    usecase "Indicar Dosis Omitida" as UC_Omit
    usecase "Registrar Motivo de Omisión" as UC_Reason

    Usuario -- UC_Alert

    UC_Taken --|> UC_Alert
    UC_Postpone --|> UC_Alert
    UC_Omit --|> UC_Alert

    (UC_Omit) .> (UC_Reason) : <<include>>
}
@enduml
```

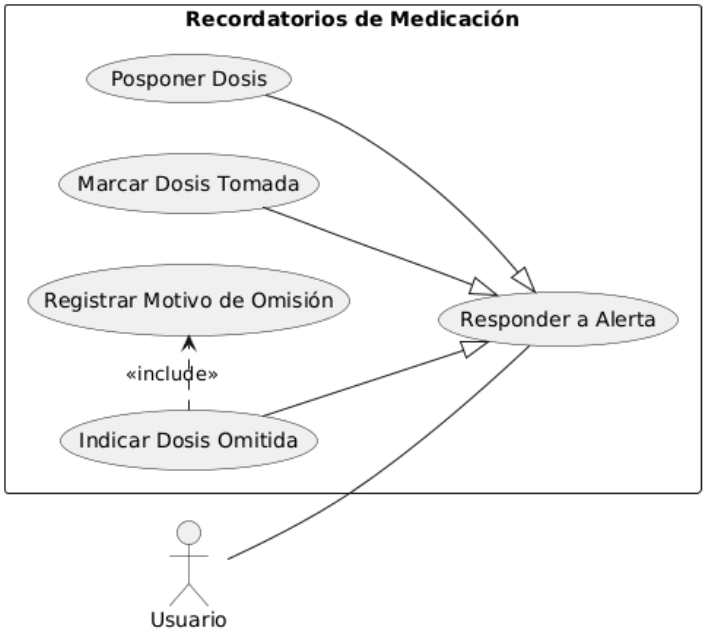
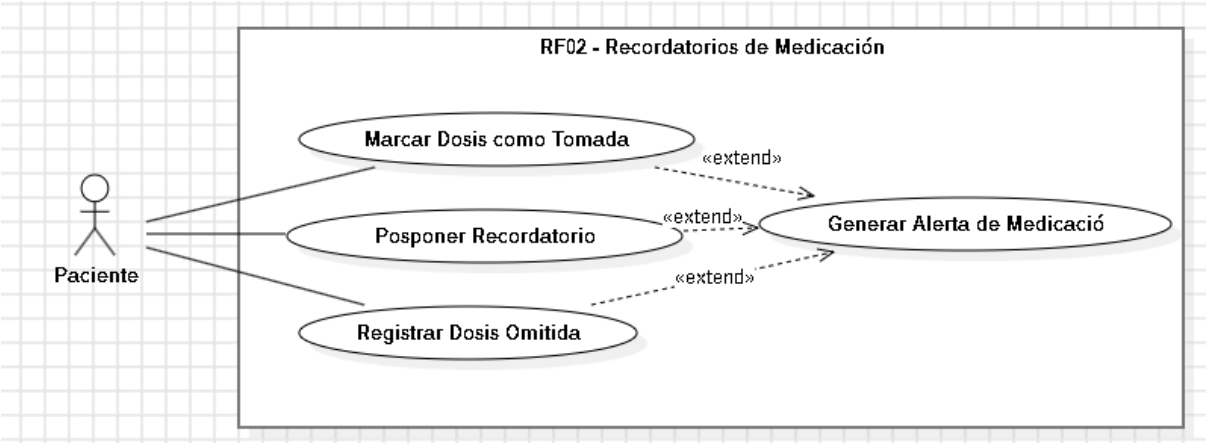


Diagrama en StarUML



Comparación

1. **¿Qué diferencias observas entre los Casos de Uso derivados de entrevistas o descripciones textuales y los Casos de Uso generados automáticamente en PlantUML?**
   El modelo humano es superior en los tres criterios: demuestra una alta precisión al emplear correctamente la relación de <<extend>> de UML para denotar las acciones del paciente como extensiones opcionales de la alerta del sistema (Generar Alerta), mientras que la IA falla semánticamente al usar Generalización (--|>).

2. **¿De qué manera el uso de PlantUML facilita (o limita) el trabajo del analista al modelar los requisitos funcionales?**
   (Considera aspectos como automatización, trazabilidad y comunicación entre analista y desarrollador.)
   PlantUML facilita el trabajo del analista al integrarse perfectamente en flujos de desarrollo ágil: permite una trazabilidad total de los requisitos al ser versionable en Git y mejora la comunicación con los desarrolladores al tratar los diagramas como código.

RF05 - Gestión de Perfiles Múltiples

Nombre de requisito	Gestión de Perfiles Múltiples
Característica	Manejar múltiples perfiles dentro de la misma aplicación
Descripción del requisito	Permite crear, modificar y eliminar múltiples perfiles con configuraciones e historiales independientes, útil para cuidadores de varios pacientes.
Fuente del requisito	Análisis funcional
Requisito relacionado	RNF01, RNF02, RNF03, RNF05, RNF06
Prioridad del requisito	Alta
Versión del requisito	V1.0

Descripción: Permite crear, modificar y eliminar múltiples perfiles con configuraciones e historiales independientes, útil para cuidadores de varios pacientes.

Identifica:

1. **El actor principal en cada requisito.**

   Actor: El Cuidador, pues en la descripción se indica que este podrá “crear, modificar y eliminar múltiples perfiles con configuraciones e historiales independientes”, siendo el actor que gestiona dichos perfiles dentro de la aplicación.

   Actor Secundario: El Usuario (persona de tercera edad) solo interactúa con el caso de uso Ver historial del perfil, el cual es también accesible a través de la acción Cambiar entre perfiles del Cuidador. El actor Usuario no realiza las acciones de *gestión* del perfil (crear, editar, eliminar).

2. **Las acciones principales que el sistema debe ejecutar.**

   Las acciones o casos de uso principales que el sistema **Helthy+** debe permitir, de acuerdo con el diagrama y la descripción, son:

   - **Crear perfil de usuario:** Permite establecer un nuevo perfil con configuraciones e historial independientes.
   - **Editar perfil:** Permite modificar la información o las configuraciones de un perfil existente.
   - **Eliminar perfil:** Permite la remoción total de un perfil y su historial asociado.
   - **Cambiar entre perfiles:** Permite al cuidador navegar o seleccionar un perfil distinto para operar dentro de él.
   - **Ver historial del perfil:** Muestra el registro o historial asociado a un perfil específico (accesible tanto desde Cuidador como Usuario).

3. **Las relaciones de inclusión (<<include>>) o extensión (<<extend>>) que aparecen en los diagramas UML.**

   - **Relación <<include>> (Obligatoria):**
     El caso de uso “Cambiar entre perfiles” incluye obligatoriamente el caso de uso “Ver historial del perfil”, ya que para cambiar o revisar un perfil, el sistema debe mostrar su historial correspondiente.

   - **Relación <<extend>> (Opcional):**
     En este diagrama no existe ninguna relación <<extend>>, ya que todas las acciones principales son parte del flujo estándar del cuidador al gestionar perfiles.

Código:

```
@startuml
left to right direction
actor "Cuidador" as Caregiver
actor "Usuario" as User

rectangle "Gestión de perfiles múltiples" {
  usecase "Crear perfil de usuario" as UC_Create
  usecase "Editar perfil" as UC_Edit
  usecase "Eliminar perfil" as UC_Delete
  usecase "Cambiar entre perfiles" as UC_Switch
  usecase "Ver historial del perfil" as UC_View

  UC_Switch --> UC_View : <<include>>
}

Caregiver --> UC_Create
Caregiver --> UC_Edit
```

Caregiver --> UC_Delete
Caregiver --> UC_Switch
Caregiver --> UC_View
User --> UC_View
@enduml

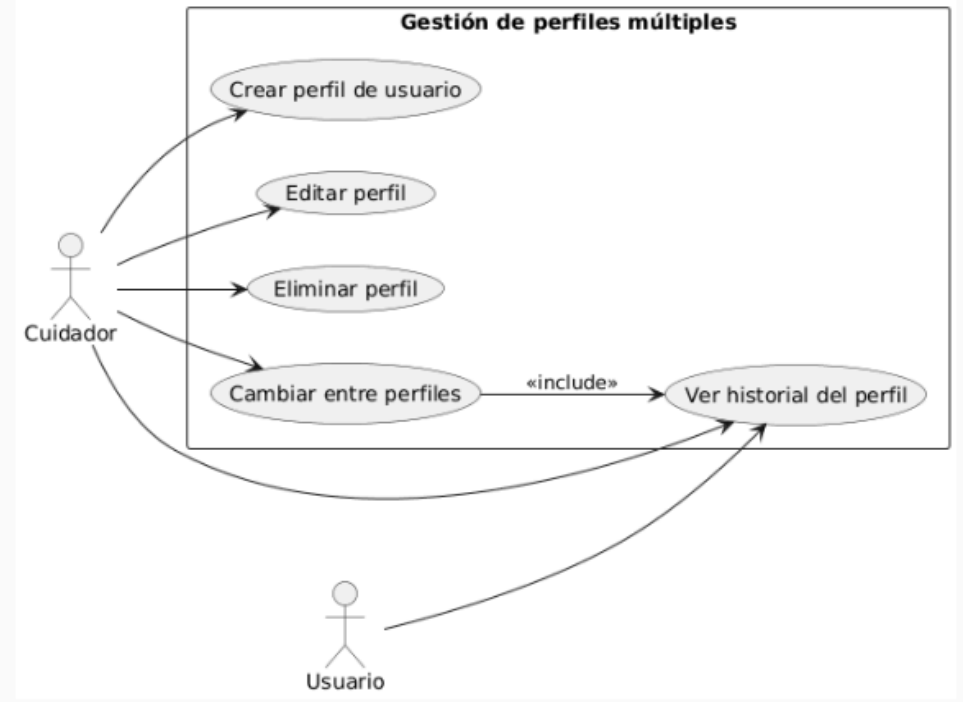
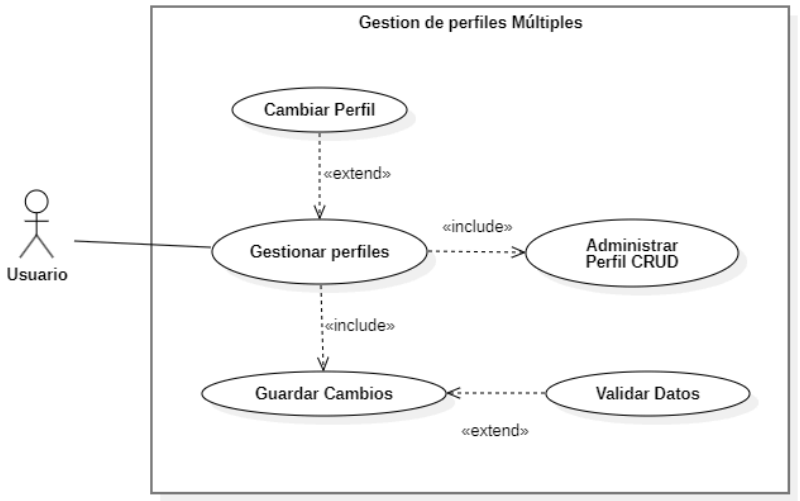


Diagrama en StarUML



Parte 2. Pensamiento crítico.

1. **¿Qué diferencias observas entre los Casos de Uso derivados de entrevistas o descripciones textuales y los Casos de Uso generados automáticamente en PlantUML?**

Precisión:

1. **Modelo IA Analizado:**
   Usa correctamente las relaciones <<include>> para representar comportamientos obligatorios, como cuando "Cambiar entre perfiles" incluye "Ver historial del perfil", esta relación tiene sentido, ya que al cambiar de perfil, el sistema debería mostrar automáticamente su historial además, diferencia claramente los actores "Cuidador" y "Usuario", asignando correctamente qué acciones puede realizar cada uno, lo cual aporta precisión en la definición de roles y permisos dentro del sistema.

2. **Modelo IA sin análisis:**
   Presenta relaciones <<include>> y <<extend>>, pero sin un criterio claro. Por ejemplo, "Gestionar perfiles" incluye "Administrar Perfil CRUD" y "Guardar Cambios", sin justificar si estas acciones son obligatorias u opcionales, con esto se entiende que el modelo fue generado automáticamente sin considerar la semántica real de las interacciones, el resultado es un diagrama correcto en forma, pero impreciso en significado.

Completitud y Claridad Visual:

1. **Modelo IA Analizado :**
   Es más completo y claro, ya que agrupa los casos de uso principales relacionados con la gestión de perfiles de usuario y refleja una estructura coherente de objetivos del "Cuidador", también no mezcla pasos internos con metas principales del usuario y mantiene un nivel de abstracción uniforme, y la inclusión del actor "Usuario" como dependiente mejora la comprensión de las relaciones entre perfiles.

2. **Modelo IA sin análisis:**
   Aunque tiene más elementos, mezcla distintos niveles de abstracción. Por ejemplo, "Validar Datos" aparece como caso de uso independiente, cuando en realidad debería ser un paso dentro de otro flujo mayor, además, usa un solo actor ("Usuario"), lo cual simplifica en exceso la interacción real del sistema, restando claridad sobre quién ejecuta cada acción.

2. ¿De qué manera el uso de PlantUML facilita (o limita) el trabajo del analista al modelar los requisitos funcionales?

Cómo facilita (Automatización y Trazabilidad):

- **Agilidad y Automatización:**
  Permite generar diagramas en segundos a partir de texto, lo cual es ideal para ajustar o revisar modelos durante reuniones de análisis.
  Su rapidez facilita la iteración y el refinamiento continuo del modelo sin necesidad de redibujar todo.

- **Trazabilidad:**
  Al estar basado en texto, los diagramas pueden almacenarse en sistemas de control de versiones (como Git), esto permite mantener un historial de cambios y mejorar la comunicación entre analistas y desarrolladores.

- **Consistencia Visual:**
  PlantUML mantiene un estilo gráfico uniforme, garantizando que todos los diagramas tengan la misma presentación profesional sin necesidad de edición manual.

Cómo limita (Rigidez y Falsa Confianza):

- **Rigidez del Auto-Layout:**
  No se tiene control total sobre la disposición de los elementos, y en diagramas complejos, esto puede generar confusión visual y dificultar la lectura.

- **Falsa Confianza en la Sintaxis:**
  Un modelo puede ser sintácticamente correcto, pero semánticamente equivocado, y el segundo diagrama lo ejemplifica: aunque está bien "escrito", no representa correctamente los objetivos funcionales del sistema.

- **Curva de Aprendizaje:**
  Aunque su sintaxis es sencilla, requiere que el analista comprenda el significado de relaciones como <<include>> y <<extend>>, de lo contrario, la herramienta puede generar resultados confusos o incoherentes.

Conclusiones

- Los modelos elaborados con IA y analizados posteriormente son más precisos y coherentes que los que no están analizados y puestos con un contexto incorrecto, los cuales suelen tener errores de interpretación.
- PlantUML facilita la automatización y trazabilidad, pero requiere criterio técnico para evitar confundir corrección sintáctica con sentido funcional.
- El pensamiento crítico resulta esencial para validar los resultados y asegurar que el modelo refleje fielmente los requisitos del sistema.

Recomendaciones

- Se recomienda combinar el uso de PlantUML con un análisis humano cuidadoso, verificando siempre el significado de cada relación UML.
- Es importante capacitarse en el uso responsable de IA, utilizándose como apoyo al razonamiento y no como sustituto del juicio técnico.

Rúbrica de Evaluación

Criterio	Excelente (5)	Bueno (4)	Regular (3)	Básico (2)	Insuficiente (1)
1. Comprensión del proceso de análisis (Parte 1)	Identifica correctamente actores, acciones y relaciones en todos los requisitos; explica con precisión cómo se derivan los casos de uso a partir de los RF y su representación en PlantUML.	Reconoce la mayoría de actores, acciones y relaciones, estableciendo conexiones parciales con los RF.	Muestra comprensión general del análisis pero con errores en las relaciones o sin conexión clara entre RF y casos de uso.	Identifica algunos elementos pero sin relación entre análisis textual y modelo UML.	No identifica los elementos principales o presenta confusión en el proceso de análisis.
2. Pensamiento crítico y argumentación (Parte 2)	Compara de forma reflexiva los modelos obtenidos del análisis y los diagramas PlantUML, justificando ventajas, limitaciones y errores con fundamentos técnicos.	Presenta comparaciones válidas pero con ejemplos parciales o argumentación poco profunda.	Identifica diferencias superficiales sin justificar su impacto en la calidad del modelo.	Repite información del taller sin análisis propio.	No presenta reflexión crítica ni justificación técnica.
3. Claridad y presentación	Entrega ordenada, con redacción técnica, argumentos coherentes y uso correcto de terminología UML.	Presenta leve desorganización o errores menores de redacción.	Redacción poco clara o con terminología inexacta.	Desorden en la presentación o lenguaje no técnico.	No cumple con el formato ni con los criterios de presentación.

4. Utilidad del pensamiento crítico en el análisis de RF con IAGen	Analiza cómo la IAGen potencia la comprensión y validación de requisitos funcionales, demostrando pensamiento crítico al integrar IA en el proceso de análisis.	Menciona aportes de la IAGen con ejemplos válidos, aunque sin profundidad analítica.	Reconoce la relación entre IAGen y análisis de RF pero sin argumentar su utilidad práctica.	Menciona la IA de forma general sin vincularla al proceso de análisis de requisitos.	No reconoce la relación entre pensamiento crítico e IAGen en el análisis de requisitos.

Total: /20 puntos