



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

ANÁLISIS Y DISEÑO DE SOFTWARE

NRC 27837

GRUPO #4:

- Mideros Samir
- Miranda Alison
- Morán David
- Vivanco Gabriel

TAREA # PARCIAL 1

TEMA:

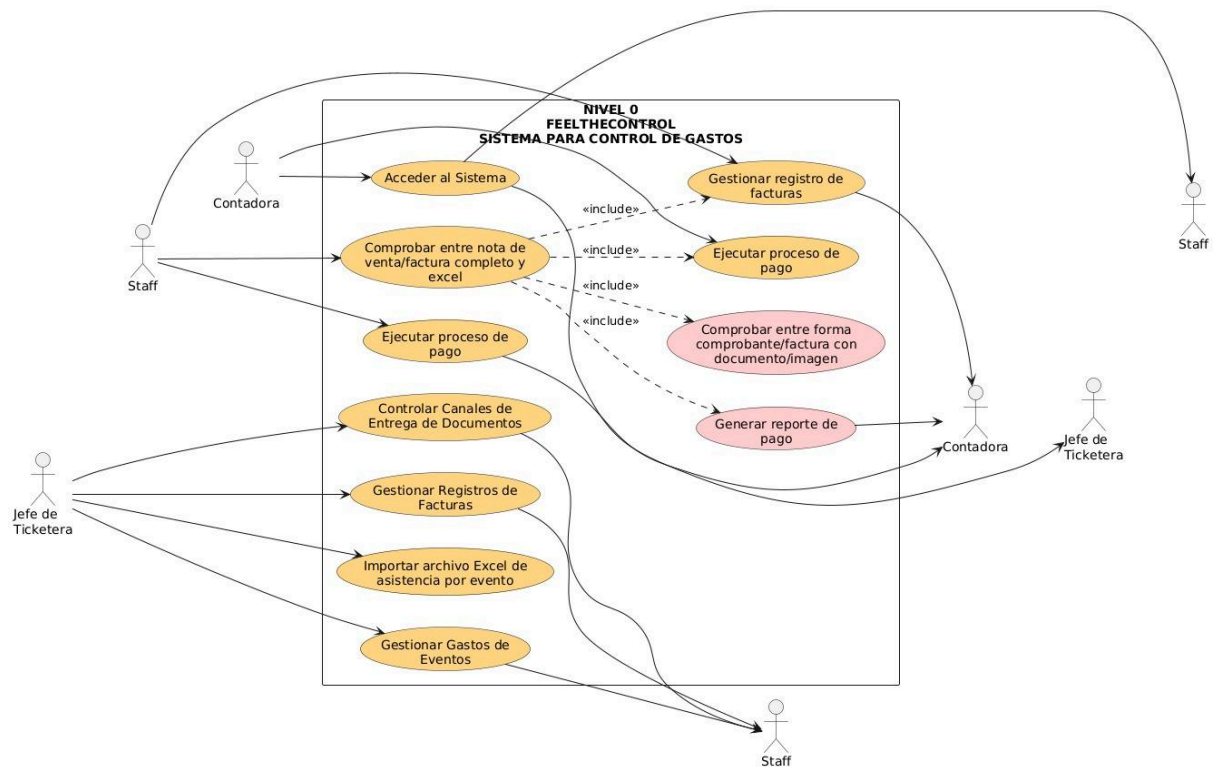
“Casos de Uso - PlantUML”

TUTOR: Jenny Ruiz

Fecha de entrega: 27/10/2025

NIVEL 0

SISTEMA "Control de Gastos"



CÓDIGO

@startuml

Left to right direction

skinparam packageStyle rectangle

' Actores lado izquierdo

actor "Jefe de\nTicketera" as JefeL

actor "Staff" as StaffL

actor "Contadora" as ContadoraL

' Forzar posición vertical de actores izquierda

JefeL -[hidden]down- StaffL

StaffL -[hidden]down- ContadoraL

package "NIVEL 0\nFEELTHECONTROL\nSISTEMA PARA CONTROL DE GASTOS" {

' Casos de uso normales (color amarillo-naranja)

usecase "Importar archivo Excel de\nasistencia por evento" as UC_IMPORT #FFD580

usecase "Gestionar Gastos de\nEventos" as UC_GASTOS #FFD580

usecase "Controlar Canales de\nEntrega de Documentos" as UC_CANALES #FFD580

usecase "Gestionar Registros de\nFacturas" as UC_REGFACT #FFD580

usecase "Ejecutar proceso de\npago" as UC_PAGAR #FFD580

usecase "Gestionar registro de\nfacturas" as UC_REGFACT_2 #FFD580

usecase "Comprobar entre nota de\nventa/factura completo y\nexcel" as UC_COMPRO_N #FFD580

usecase "Ejecutar proceso de\npago" as UC_PAGAR_2 #FFD580

usecase "Acceder al Sistema" as UC_ACCEDER #FFD580

```
' Casos de uso destacados (color rojo suave)
usecase "Generar reporte de\npago" as UC_GENERAR #FFCCCC
usecase "Comprobar entre forma\ncomprobante/factura con\ndocumento/imagen" as
UC_COMPRO_F #FFCCCC
}
```

```
' Actores lado derecho
actor "Staff" as StaffR1
actor "Contadora" as ContadoraR
actor "Jefe de\nTicketera" as JefeR
actor "Staff" as StaffR2
```

```
' Forzar posición vertical de actores derecha
StaffR1 -[hidden]down- ContadoraR
ContadoraR -[hidden]down- JefeR
JefeR -[hidden]down- StaffR2
```

```
' === RELACIONES DESDE ACTORES IZQUIERDA ===
```

```
' Jefe de Ticketera (izquierda)
JefeL --> UC_IMPORT
JefeL --> UC_GASTOS
JefeL --> UC_CANALES
JefeL --> UC_REGFACT
```

```
' Staff (izquierda)
StaffL --> UC_REGFACT_2
StaffL --> UC_COMPRO_N
StaffL --> UC_PAGAR
```

```
' Contadora (izquierda)
ContadoraL --> UC_PAGAR_2
ContadoraL --> UC_ACCEDER
```

```
' === RELACIONES HACIA ACTORES DERECHA ===
```

```
' Staff (derecha superior)
UC_GASTOS --> StaffR1
UC_CANALES --> StaffR1
UC_REGFACT --> StaffR1
```

```
' Contadora (derecha)
UC_REGFACT_2 --> ContadoraR
UC_PAGAR --> ContadoraR
UC_GENERAR --> ContadoraR
```

```
' Jefe de Ticketera (derecha)
UC_ACCEDER --> JefeR
```

```
' Staff (derecha inferior)
UC_ACCEDER --> StaffR2
```

' === RELACIONES INCLUDE ENTRE CASOS DE USO ===

' Comprobar entre nota de venta/factura completo y excel tiene includes con:

UC_COMPRO_N ..> UC_REGFACT_2 : <<include>>

UC_COMPRO_N ..> UC_GENERAR : <<include>>

UC_COMPRO_N ..> UC_COMPRO_F : <<include>>

UC_COMPRO_N ..> UC_PAGAR_2 : <<include>>

@enduml

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.)

Precisión:

- **Manual (Entrevistas):** Permiten una mayor libertad en la disposición de los elementos. Esto puede hacer que los casos de uso relacionados estén visualmente más cerca, facilitando una comprensión más intuitiva de los flujos. Un ejemplo de esto son las especificaciones textuales detalladas y el diagrama general en el documento de especificaciones.
- **PlantUML:** Tienen una precisión semántica mayor. El hecho de escribirlos como código obliga al analista a definir explícitamente todas las relaciones, lo que resulta en un modelo más riguroso.

Completitud:

- **Manual:** Existe un mayor riesgo de omitir relaciones o actores secundarios durante el dibujo manual. La distribución puede ser más intuitiva, pero potencialmente menos rigurosa.
- **PlantUML:** Al requerir un código estructurado, la herramienta fuerza al analista a ser exhaustivo. Cada actor, caso de uso y relación debe declararse explícitamente, lo que reduce significativamente las omisiones.

Claridad Visual:

- **Manual:** Ofrecen un mayor control estético y de diseño. El analista puede usar colores conscientemente o agrupar elementos para guiar la vista de forma natural.
- **PlantUML:** La disposición es automática, lo que puede crear cruces de líneas confusos o distribuciones poco intuitivas. Aunque se pueden usar directivas para mejorar el diseño, el control es mucho más limitado que en el diseño manual.

Consistencia (Punto adicional del documento):

- **Manual:** Son vulnerables a inconsistencias en la notación UML entre diferentes diagramas o analistas.

- **PlantUML:** Garantiza la consistencia sintáctica y el uso de la notación UML estándar en todos los diagramas generados.

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.)

Facilita el trabajo:

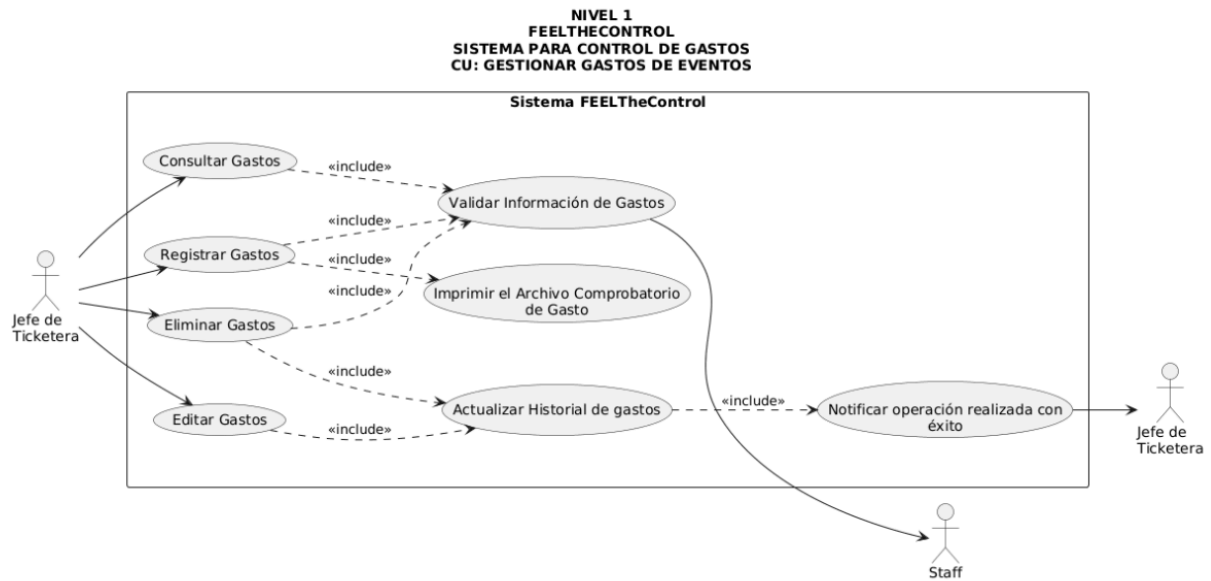
- **Automatización y Velocidad:** Permite generar diagramas automáticamente a partir de texto. Si se necesita un cambio (como agregar un actor), solo se edita una línea de código y el diagrama se actualiza solo, sin necesidad de redibujar manualmente. Esto fue crucial para reflejar ajustes en la lógica del sistema de forma rápida.
- **Trazabilidad y Versionamiento:** Al ser basado en texto, el código de PlantUML se puede gestionar con herramientas como Git, lo que permite un control de cambios y un historial completo de modificaciones (trazabilidad).
- **Consistencia:** Asegura que todos los diagramas usen una notación UML estandarizada. En el proyecto, esto fue fundamental para mantener la coherencia en los diferentes niveles de abstracción (Nivel 0 y Nivel 1).
- **Colaboración y Comprensión:** Se menciona que PlantUML "facilitó la comprensión del sistema y promovió el trabajo colaborativo eficiente".

Limita el trabajo:

- **Control Visual Limitado:** El analista tiene poco control sobre el layout (diseño) automático, que puede resultar poco flexible o confuso.
- **Curva de Aprendizaje:** Requiere que el analista aprenda la sintaxis específica de PlantUML para poder crear los diagramas.
- **Comunicación con No Técnicos:** El código puede ser menos intuitivo para usuarios finales o stakeholders que no están familiarizados con él. Esto dificulta su uso en sesiones colaborativas en vivo con ellos.
- **Mantenimiento de Diagramas Complejos:** A medida que un diagrama crece en complejidad, el código de PlantUML que lo genera puede volverse difícil de mantener.

NIVEL 1

SUBSISTEMA “Gestionar gastos de eventos”



CÓDIGO

@startuml

Left to right direction

skinparam packageStyle rectangle

title NIVEL 1\nFEELTHECONTROL\nSISTEMA PARA CONTROL DE GASTOS\nCU: GESTIONAR GASTOS DE EVENTOS

actor "Jefe de\nTicketera" as JefeTicketeraIzq

actor "Staff" as Staff

actor "Jefe de\nTicketera" as JefeTicketeraDer

```

rectangle "Sistema FEELTheControl" {
    usecase "Consultar Gastos" as UC1
    usecase "Registrar Gastos" as UC2
    usecase "Eliminar Gastos" as UC3
    usecase "Editar Gastos" as UC4
    usecase "Validar Información de Gastos" as UC5
    usecase "Imprimir el Archivo Comprobatorio\nde Gasto" as UC6
    usecase "Actualizar Historial de gastos" as UC7
    usecase "Notificar operación realizada con\néxito" as UC8
}

```

JefeTicketeraIzq --> UC1

JefeTicketeraIzq --> UC2

JefeTicketeraIzq --> UC3

JefeTicketeraIzq --> UC4

UC1 ..> UC5 : <<include>>

UC2 ..> UC5 : <<include>>

UC2 ..> UC6 : <<include>>

UC3 ..> UC5 : <<include>>

UC3 ..> UC7 : <<include>>

```
UC4 ..> UC7 : <<include>>
```

```
UC5 --> Staff
```

```
UC8 --> JefeTicketeraDer
```

```
UC7 ..> UC8 : <<include>>
```

```
@enduml
```

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.)

En este caso, los casos de uso en texto y los hechos en PlantUML muestran prácticamente la misma información del sistema FEELTheControl, pero la diferencia está en cómo se trabajan y representan. Con PlantUML el proceso es más rápido y colaborativo, ya que varios pueden editar el mismo código y generar el diagrama automáticamente sin tener que hacerlo a mano. Además, si usamos herramientas de IA, podemos agilizar aún más el proceso, aunque antes debemos alimentar a la IA con un buen análisis previo, para que entienda correctamente el sistema y genere un modelo coherente. En cambio, de manera manual tenemos la libertad de ir construyendo el diagrama según entendemos el proyecto, lo que puede ser más intuitivo, pero también más lento y menos uniforme.

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 mucho la creación de diagramas porque permite representar los casos de uso de forma clara, rápida y fácil de actualizar. Es ideal cuando se trabaja en equipo o cuando el proyecto cambia con frecuencia. Sin embargo, requiere conocer un poco de su lenguaje o apoyarse en IA, y aunque el diagrama sea visualmente claro, sigue siendo necesario complementar con la descripción textual para explicar los detalles del flujo.

Precisión:

- **Manual** : Permite mayor libertad en la disposición espacial y agrupación lógica de elementos. Los casos de uso relacionados están visualmente más cerca, facilitando la comprensión de flujos.
- **PlantUML** : La precisión semántica es mayor porque el código obliga a definir explícitamente todas las relaciones. Sin embargo, la disposición automática puede no ser óptima para la comprensión humana.

Completitud:

- **Manual**: Existe riesgo de omitir relaciones o actores secundarios. Se ve una distribución más intuitiva pero potencialmente menos rigurosa.
- **PlantUML**: Al requerir código estructurado, fuerza al analista a ser exhaustivo. Cada relación, actor y caso de uso debe declararse explícitamente, reduciendo omisiones.

Claridad Visual:

- **Manual:** Mayor control estético. Muestra un uso consciente del color (azul vs. púrpura) para diferenciar tipos de casos de uso, y una disposición que guía el ojo naturalmente.
- **PlantUML:** La disposición automática. Puede crear cruces de líneas confusos y distribuciones poco intuitivas. Aunque existen directivas para mejorar el layout, el control es limitado comparado con el diseño manual.

Consistencia:

- **Manual:** Vulnerable a inconsistencias de notación entre diferentes diagramas o analistas.
- **PlantUML:** Garantiza consistencia sintáctica y notación UML estándar en todos los diagramas.

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.)

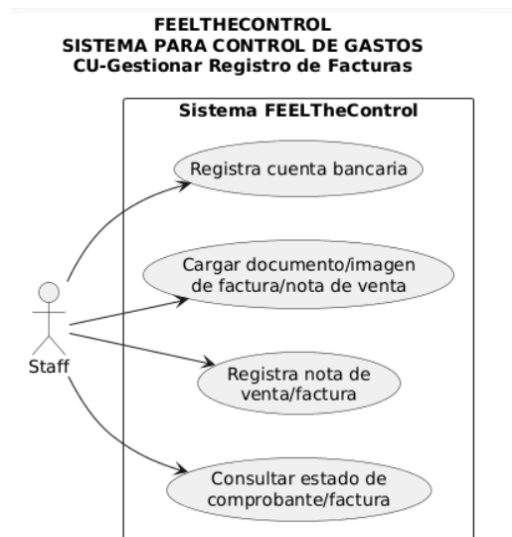
Facilita:

- Versionamiento: Control de cambios con Git
- Velocidad: Actualizaciones rápidas editando texto
- Consistencia: Notación UML estandarizada siempre
- Trazabilidad: Historial completo de modificaciones

Limita:

- Curva de aprendizaje: Requiere conocer sintaxis
- Diagramas complejos: Código difícil de mantener
- Personalización: Limitado para colores, espaciado y agrupación visual

SUBSISTEMA “Gestionar registro de facturas”



CÓDIGO

@startuml

Left to right direction

skinparam packageStyle rectangle

title NIVEL 1\nFEELTHECONTROL\nSISTEMA PARA CONTROL DE GASTOS\nCU-Gestionar Registro de Facturas

actor "Staff" as Staff

```

rectangle "Sistema FEELTheControl" {
    usecase "Consultar estado de\ncomprobante/factura" as UC1
    usecase "Registra nota de\nventa/factura" as UC2
    usecase "Cargar documento/imagen\nde factura/nota de venta" as UC3
    usecase "Registra cuenta bancaria" as UC4
}
  
```

Staff --> UC1

Staff --> UC2

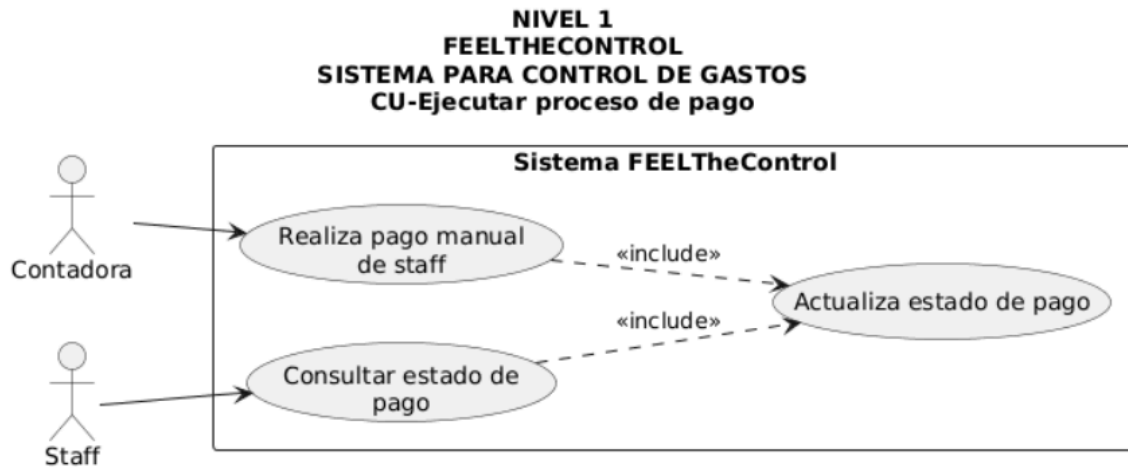
Staff --> UC3

Staff --> UC4

@enduml

1) Diferencias entre Casos de Uso derivados de entrevistas/descripciones textuales y los Casos de Uso generados automáticamente en PlantUML

SUBSISTEMA “Ejecutar proceso de pago”



CÓDIGO

@startuml

left to right direction

skinparam packageStyle rectangle

title NIVEL 1\nFEELTHECONTROL\nSISTEMA PARA CONTROL DE GASTOS\nCU-Ejecutar proceso de pago

actor "Contadora" as Contadora

actor "Staff" as Staff

```

rectangle "Sistema FEELTheControl" {
    usecase "Actualiza estado de pago" as UC1
    usecase "Realiza pago manual\nde staff" as UC2
    usecase "Consultar estado de\npago" as UC3
}
  
```

Contadora --> UC2

Staff --> UC3

UC2 ..> UC1 : <<include>>

UC3 ..> UC1 : <<include>>

@enduml

SUBSISTEMA “Controlar canales de entrega de docuemntos”

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.)

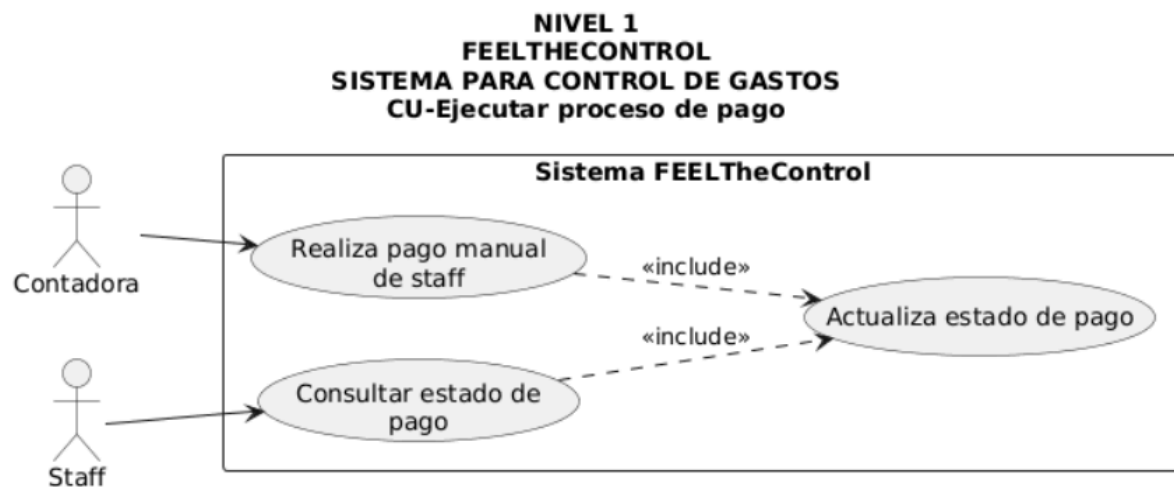
En el caso de uso “Ejecutar proceso de pago”, el modelo realizado a partir de entrevistas representa únicamente las acciones principales que el usuario mencionó, como realizar el

pago manual y consultar su estado. Este enfoque es intuitivo y refleja el lenguaje del negocio, pero puede dejar actividades implícitas sin detallar. En cambio, el modelo generado en PlantUML descompone el proceso en subcasos adicionales y aplica relaciones como «include» y «extend», logrando mayor precisión y estructura formal. Esto hace más claro el flujo interno del sistema, aunque requiere validar que los elementos agregados coincidan con la realidad del proceso.

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 convertir descripciones en diagramas estructurados, claros y fáciles de actualizar, mejorando la comunicación con desarrolladores y asegurando trazabilidad. Sin embargo, su uso exige comprensión del negocio y de la notación UML, ya que el modelo puede volverse técnicamente correcto pero funcionalmente incorrecto si no se analiza bien la información. Por tanto, PlantUML es una herramienta útil que potencia la documentación, pero el razonamiento y la validación siguen siendo responsabilidad del analista.



CÓDIGO

```
@startuml
```

```
left to right direction
```

```
skinparam packageStyle rectangle
```

```
title CU: CONTROLAR CANALES DE ENTREGA DE DOCUMENTOS
```

```
actor "Jefe de\nTicketera" as JefeTicketeraIzq
```

```
actor "Jefe de\nTicketera" as JefeTicketeraDer
```

```
rectangle "Sistema FEELTheControl" {
```

```
    usecase "Registrar canal de entrega preferido" as UC1
```

```

    usecase "Verificar recepción de documentos\npor canal" as UC2
    usecase "Archivar documentos según canal" as UC3
    usecase "Reasignar canal de entrega para un\nusuario" as UC4
    usecase "Notificar confirmación de recepción de\ndocumentos al jefe del
Ticketera" as UC5
    usecase "Reportar canal incorrecto al jefe de la Ticketera" as UC6
    usecase "Validar integridad del archivo recibido" as UC7
}

JefeTicketeraIzq --> UC1
JefeTicketeraIzq --> UC4

UC1 ..> UC2 : <<include>>
UC2 ..> UC3 : <<include>>

UC2 ..> UC5 : <<include>>
UC2 ..> UC6 : <<extend>>
UC2 ..> UC7 : <<include>>

UC5 --> JefeTicketeraDer
UC6 --> JefeTicketeraDer

@enduml

```

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.)

El diagrama PlantUML muestra una disposición automática compacta y simétrica, con casos de uso alineados horizontalmente y espaciado uniforme. Las líneas <<include>> son perfectamente punteadas y consistentes. Una novedad importante fue que al generar desde requisitos funcionales textuales, surgieron ambigüedades en la redacción que llevaron a modelar los diagramas de manera diferente entre el manual y PlantUML, aunque ambos mantuvieron coherencia lógica con los requisitos originales.

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 generando automáticamente formas perfectas, alineación simétrica y notación UML estándar desde código simple, ahorrando tiempo de dibujo manual y garantizando consistencia visual. Además, obligó al analista a clarificar requisitos funcionales ambiguos o mal redactados durante la traducción a código, actuando como mecanismo de validación temprana. Limita porque el analista no puede destacar visualmente casos de uso principales, agrupar elementos relacionados por proximidad, ni ajustar ángulos de flechas

para mejorar legibilidad, resultando en diagramas técnicamente correctos pero potencialmente menos intuitivos. Reintentar Claude puede cometer errores. Verifique las respuestas.

PLANT UML COMO HERRAMIENTA

El proyecto "Control de Gastos" empleó PlantUML como la herramienta central para modelar los casos de uso del sistema. Esta elección permitió generar diagramas claros y estructurados directamente desde código, facilitó la comprensión del sistema y promovió el trabajo colaborativo eficiente.

PlantUML facilitó la representación visual de las interacciones clave entre los actores (Jefe de Ticketera, Staff y Contadora) y los procesos esenciales como Gestionar Gastos de Eventos, Controlar Canales de Entrega de Documentos y Ejecutar Proceso de Pago.

La naturaleza basada en texto de PlantUML resultó ser fundamental para mantener la coherencia y la trazabilidad en los diferentes niveles de abstracción del sistema (nivel 0 y nivel 1).

Al escribir el código UML, cualquier ajuste en la lógica o el diseño del sistema se reflejaba de manera rápida y sencilla en los diagramas, evitando la necesidad de redibujar completamente. Esta agilidad en la actualización fue crucial durante la fase de refinamiento del diseño, asegurando que los modelos estuvieran siempre alineados con la evolución del proyecto.

VENTAJAS DE PLANT UML EN EL PROYECTO

1. Ventajas de PlantUML

PlantUML fue una mejor opción en el proyecto porque ahorra mucho tiempo y esfuerzo. En lugar de dibujar cada actor, cada línea o cada flecha a mano como en Lucidchart o Draw.io, con PlantUML solo se escribe un pequeño código que describe el diagrama, y el programa lo genera automáticamente. Esto significa que si luego quieres cambiar algo, por ejemplo agregar un nuevo actor o corregir una relación, no tienes que borrar y mover figuras una por una simplemente cambias una línea de texto y el diagrama se actualiza. Esa rapidez hace que el trabajo sea más ágil, ordenado y profesional.

Además, PlantUML te ayuda a mantener coherencia y organización en todo el proyecto. Como todo se escribe con un formato similar, todos los diagramas tienen el mismo estilo, estructura y orden, sin errores de conexión o tamaños desiguales. Esto facilita mucho cuando un proyecto tiene varios niveles o subsistemas, como en este caso con el "Nivel 0" y los subsistemas del "Nivel 1".

Otra gran ventaja es que PlantUML favorece el trabajo en equipo. Al ser texto, se puede guardar junto al código del proyecto en un repositorio (por ejemplo, en GitHub), y varios integrantes pueden editar o revisar los diagramas al mismo tiempo. Esto es algo que no se puede hacer tan fácilmente con diagramas hechos a mano o en programas visuales.

A nivel laboral, PlantUML sí es muy útil. Muchas empresas de software lo usan porque se integra con entornos de desarrollo, documentación técnica o control de versiones. En el trabajo real, cuando se actualiza el código, los diagramas pueden actualizarse automáticamente, manteniendo la documentación al día. Eso hace que los proyectos sean más eficientes y profesionales, y que la información siempre esté sincronizada.

2. Desventajas de PlantUML

La principal desventaja es que no es tan visual ni tan intuitivo al principio. Hay que aprender su sintaxis, o sea, los comandos con los que se describe el diagrama. Entonces, si una persona no tiene experiencia programando, puede resultar difícil o poco amigable al inicio.

También, a diferencia de herramientas gráficas, PlantUML no permite diseños muy decorativos. Los diagramas se ven bien, pero no puedes personalizar tanto los colores, las formas o el estilo visual como en otras plataformas.

Por último, aunque es una herramienta potente, no todas las empresas la usan, especialmente aquellas que prefieren trabajar con diagramas más “vistosos” o hechos de forma manual. Pero en proyectos técnicos o de ingeniería de software, sigue siendo una de las mejores opciones.

3. IA COMO ASISTENTE DE APOYO

Durante el desarrollo del proyecto, la inteligencia artificial fue una herramienta de apoyo que facilitó el trabajo del grupo, especialmente en las fases de corrección y organización del código PlantUML. Su uso fue muy útil para aclarar la sintaxis de los comandos, sugerir mejoras visuales en los diagramas y ayudar a mantener la coherencia entre los diferentes casos de uso del sistema.

Una de las principales ventajas de contar con este tipo de herramientas es la retroalimentación inmediata que ofrece. Cuando uno escribe el código UML o busca mejorar la presentación del diagrama, la IA permite detectar errores de forma rápida o sugerir alternativas más claras. Esto nos ayudó a concentrarnos en la parte conceptual del modelado, sin perder tanto tiempo en detalles técnicos o de formato.

Además, trabajar con IA fomenta el aprendizaje autónomo: cada recomendación se convierte en una oportunidad para entender mejor cómo funciona la herramienta. En nuestro caso, permitió que cada integrante del grupo fortaleciera su comprensión sobre las relaciones de inclusión, extensión y los roles de los actores en el sistema.

También determinamos que depender completamente de la IA puede limitar el desarrollo de nuestras propias habilidades y conocimiento. Nos sirvió para mejorar la calidad del trabajo y aumentar la productividad, pero el análisis, la interpretación y las decisiones finales fueron

realizadas por nosotros. Por ende, podemos afirmar que la IA cumplió su verdadero propósito: apoyar al estudiante sin reemplazar su razonamiento.

ANÁLISIS CRÍTICO

Este proyecto evidenció un equilibrio entre la automatización y la comprensión conceptual en el análisis de casos de uso.

Aspectos positivos

El uso de PlantUML permitió representar el sistema de forma ordenada, clara y modular.

La IA apoyó la interpretación y validación de los requisitos, facilitando la identificación de actores, relaciones y dependencias.

La colaboración se volvió más eficiente gracias a herramientas digitales que permitieron iterar, corregir y validar el modelo rápidamente.

Aspectos a mejorar

Existe riesgo de depender demasiado de la IA y reducir la práctica manual en el modelado UML.

La sintaxis de PlantUML puede generar una curva de aprendizaje inicial, especialmente para quienes no están familiarizados con modelado estructurado.

Para obtener precisión, es necesario revisar el código y validar visualmente los diagramas, evitando duplicidades o relaciones incorrectas.

Conclusión

El uso conjunto de PlantUML y la IA demuestra que la tecnología puede potenciar el análisis de requisitos y acelerar la documentación. Sin embargo, el razonamiento humano sigue siendo el elemento central. La IA no reemplaza la capacidad del analista para interpretar el negocio, priorizar requisitos y entender las necesidades del usuario. Tanto PlantUML como la IA son herramientas de apoyo que mejoran el proceso, pero la comprensión del modelo y la coherencia funcional dependen del analista.

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.)

Los casos de uso obtenidos a partir de entrevistas y descripciones textuales suelen mostrar la perspectiva del usuario final, incluyendo matices, excepciones del negocio y detalles contextuales. Estos modelos capturan la intención y el flujo narrativo de los actores sin preocuparse inicialmente por la estructura formal. Sin embargo, pueden contener ambigüedades o depender de interpretaciones personales del analista.

En contraste, los casos de uso generados mediante PlantUML presentan una estructura mucho más formal y estandarizada. Su representación es más clara y precisa en cuanto a relaciones, dependencias, actores e inclusiones, lo que facilita la verificación y la trazabilidad. No

obstante, esta “automaticidad” puede omitir matices o comportamientos implícitos si no fueron correctamente interpretados o incorporados desde las entrevistas.

Textuales → más ricos en contexto humano pero susceptibles a ambigüedad.

PlantUML → más precisos y visualmente claros, pero dependen de la calidad del análisis previo.

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.)

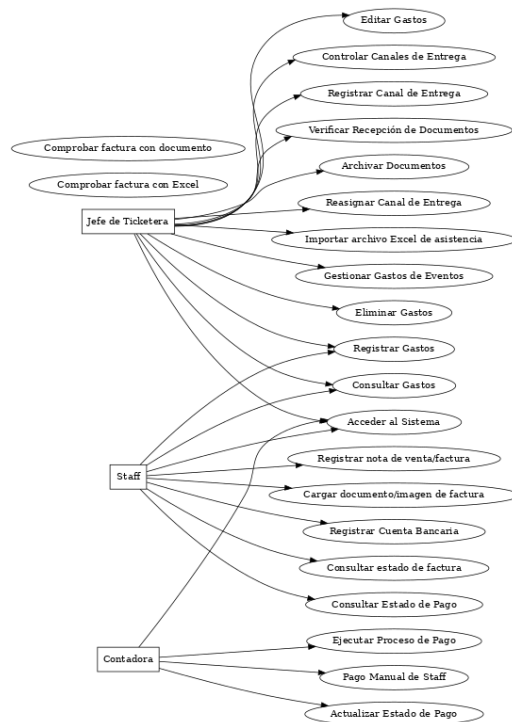
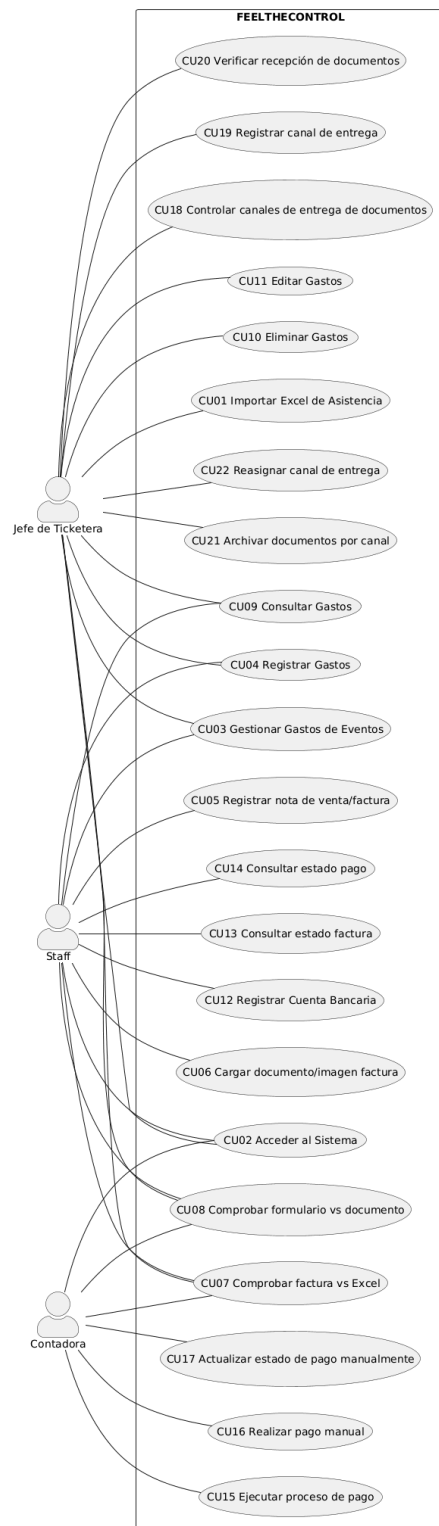
PlantUML facilita el trabajo del analista al ofrecer un lenguaje declarativo que permite automatizar la generación de diagramas de forma rápida, reproducible y fácil de mantener. Esto incrementa la trazabilidad, ya que cualquier cambio en requisitos puede reflejarse inmediatamente en el modelo. También mejora la comunicación entre analista y desarrolladores, al evitar interpretaciones subjetivas y proporcionar un formato estándar.

Sin embargo, PlantUML también tiene limitaciones. Requiere conocer su sintaxis, lo cual supone una curva de aprendizaje. Además, si el analista no tiene claridad conceptual en los requisitos, la herramienta puede generar modelos correctos visualmente pero incorrectos en lógica de negocio. En otras palabras, PlantUML potencia la documentación técnica, pero no reemplaza la capacidad de análisis humano para comprender el dominio del sistema.

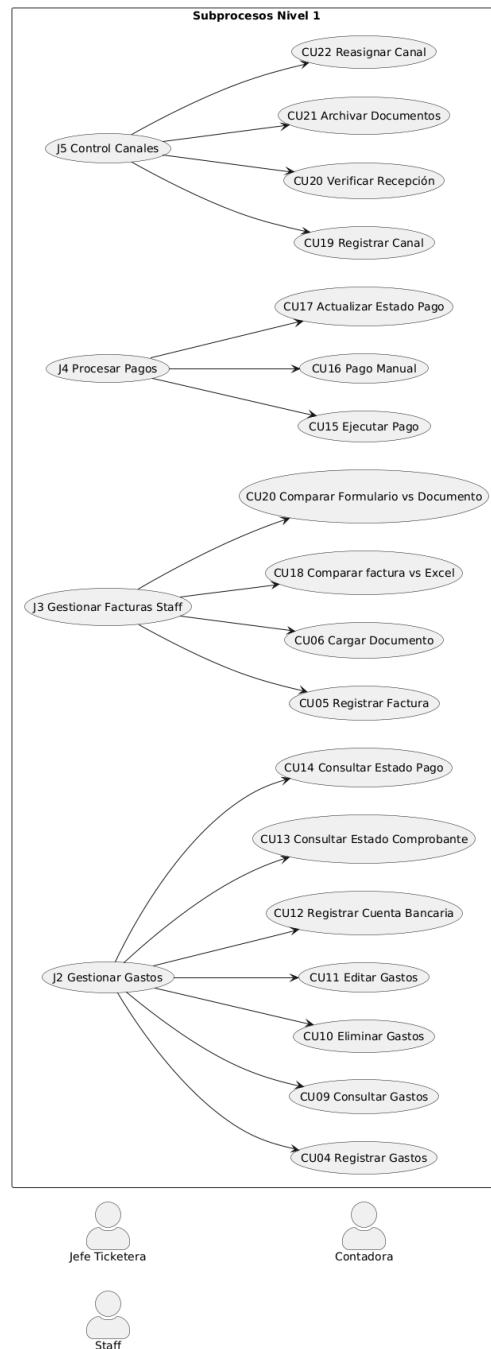
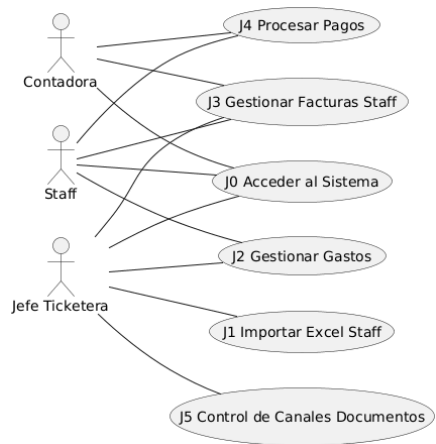
Conclusión:

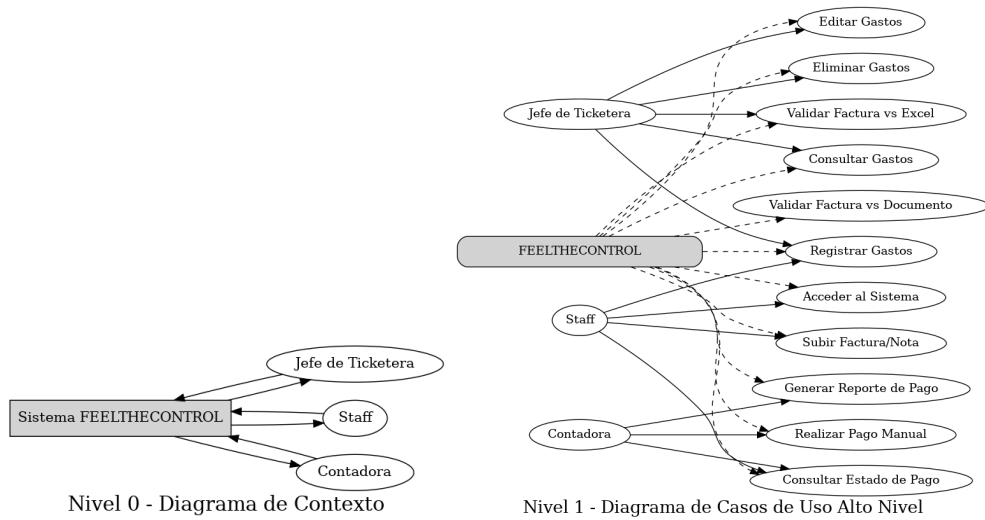
La comparación entre los casos de uso obtenidos mediante entrevistas y los diagramados en PlantUML evidencia que ambos enfoques son complementarios dentro del proceso de análisis de requisitos. Mientras que la información derivada de entrevistas aporta un entendimiento profundo del contexto del usuario, sus necesidades reales y las particularidades del negocio, la representación en PlantUML ofrece una estructura formal, precisa y estandarizada que facilita la interpretación técnica y la trazabilidad del sistema. No obstante, su efectividad depende directamente de la calidad del análisis previo, ya que la herramienta no sustituye la capacidad del analista para identificar y comprender los procesos clave. En definitiva, el valor del modelado radica en combinar la visión humana y contextual obtenida del cliente con la precisión y claridad visual que proporciona PlantUML, logrando así requisitos más completos, entendibles y alineados con la funcionalidad esperada del sistema.

1er intento

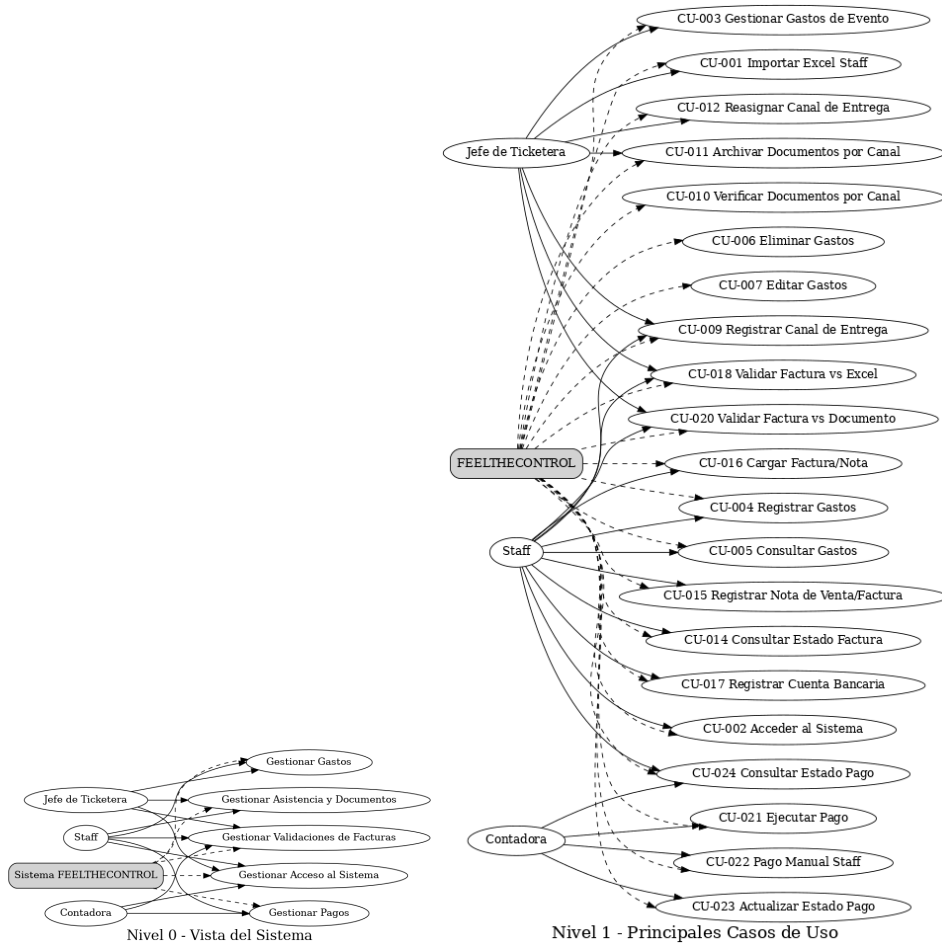


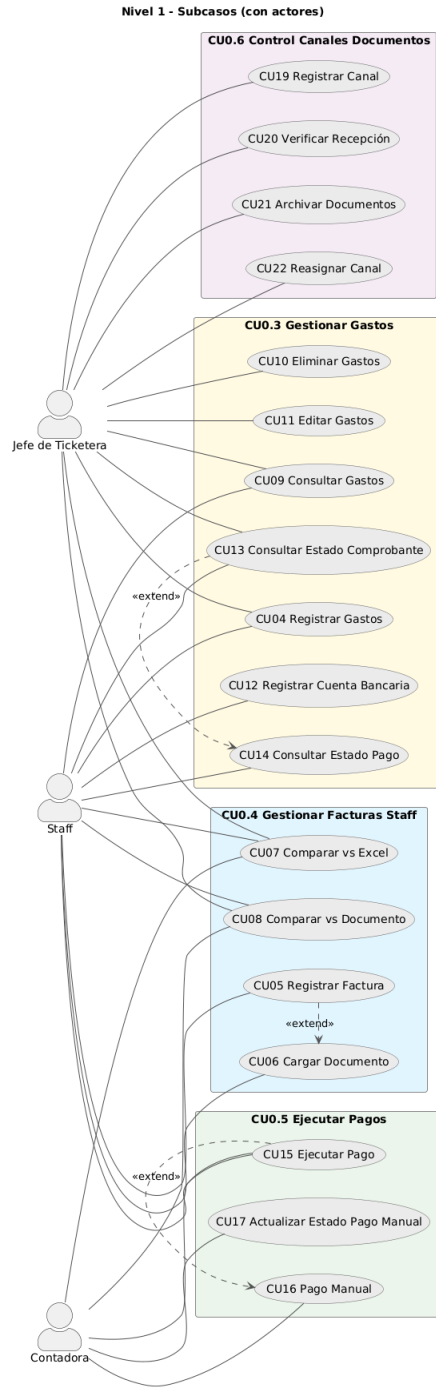
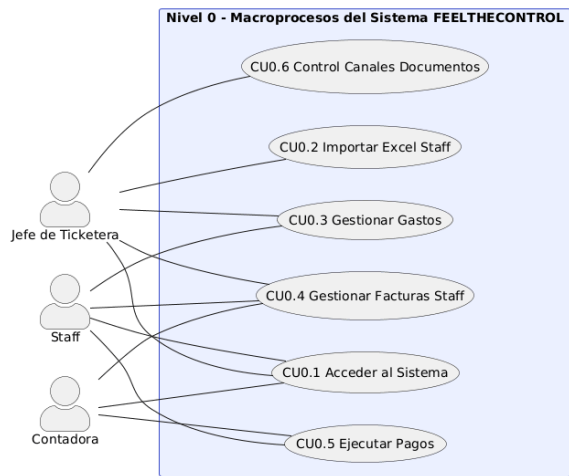
2do intento





3er intento





RÚBRICA

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.