Universidad de las Fuerzas Armadas ESPE

Actividad

Taller 6 – Casos de Uso

Asignatura

Analisis y Diseno de Software

Docente

Ing. Jenny Ruiz

Grupo 6

Integrantes

- Jordan Guaman
- Caetano Flores
- Anthony Morales
- Leonardo Narvaez

Fecha

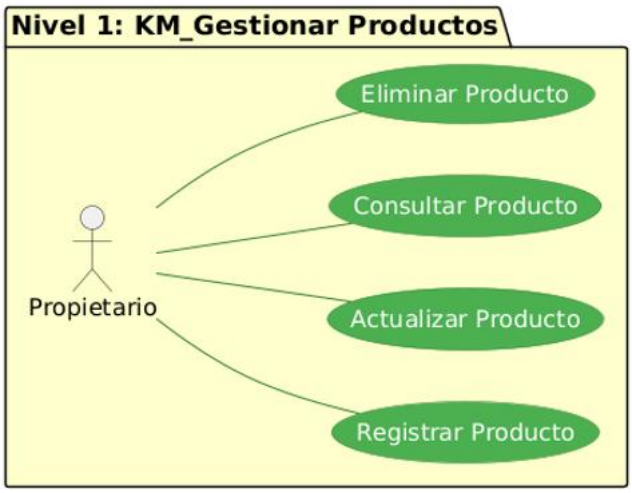
28/10/2025

CU01: Gestionar Productos

CU1.0	Gestionar Producto
Actor	Propietario
Descripción	El sistema deberá permitir al actor. Registrar, editar, consultar y eliminar los datos del producto. Toda la información se gestionará y almacenará en la base de datos.

Commented [AM1]: En descripción que datos del producto?

Commented [A2R1]: Gracias por la observación. Siguiendo buenas prácticas en la elaboración de casos de uso por niveles, en los **casos de nivel 0** solo se menciona de forma general el tipo de datos gestionados (por ejemplo: "datos del producto", "datos del cliente", etc.), sin entrar en los atributos específicos.



Codigo PlantUML

```
package "Nivel 1: KM_Gestionar Productos" #FFFFCC {
   actor Propietario_GP as "Propietario"

   (Registrar Producto) as UC_RP
   (Actualizar Producto) as UC_AP
   (Consultar Producto) as UC_CP
   (Eliminar Producto) as UC_EP

   Propietario_GP -- UC_RP
   Propietario_GP -- UC_AP
   Propietario_GP -- UC_CP
   Propietario_GP -- UC_EP
}
```

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

La diferencia más notable reside en la precisión y la granularidad de la descripción funcional. El Caso de Uso textual ("Gestionar Producto") es de alto nivel y resulta impreciso al agrupar cuatro funciones operacionales (CRUD) en una sola unidad. En contraste, el diagrama PlantUML fuerza la descomposición en Casos de Uso atómicos (Registrar, Actualizar, etc.), lo que aumenta significativamente la precisión para el desarrollo de software.

Además, el modelado gráfico ofrece una claridad visual superior al explicitar el alcance del sistema y las interacciones del Actor, facilitando la validación. Mientras que el texto es un resumen, PlantUML ayuda a validar la completitud al requerir la explicitación de las relaciones UML (<<include>> o <<extend>>), si son necesarias.

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

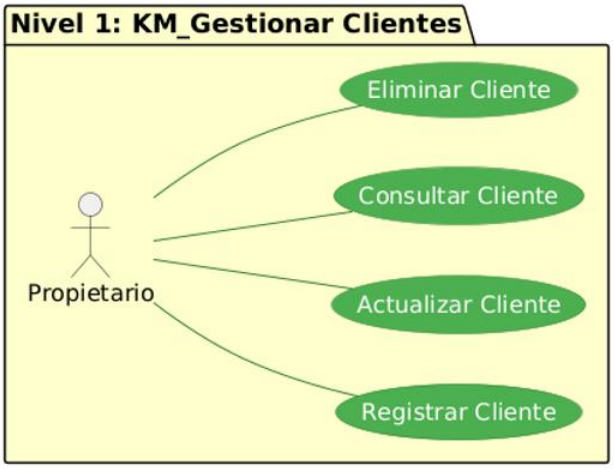
PlantUML facilita el trabajo del analista mediante la automatización y la trazabilidad. Permite la generación rápida de diagramas complejos a partir de código simple de texto, lo que acelera el modelado y permite al analista centrarse en la lógica UML.docx, 2, 9]. Adicionalmente, el *script* es versionable, lo que establece una trazabilidad directa entre el requisito y su modelo, y estandariza la comunicación con el desarrollador, asegurando un diseño preciso.

La principal limitación es que la generación automática puede limitar la libertad creativa del analista para realizar personalizaciones visuales no estándar que sí son posibles en herramientas de dibujo. También requiere que el analista domine la sintaxis para el manejo de relaciones complejas.

CU02: Gestionar Productos

CU2.0	Gestionar Cliente
Actor	Propietario
Descripción	El sistema deberá permitir al actor. Registrar, editar, consultar y eliminar los datos del cliente. Toda la información se gestionará y almacenará en la base de **datos**.

Commented [AM3]: Que datos del cliente¿?



Codigo PlantUML:

```
package "Nivel 1: KM_Gestionar Clientes" #FFFFCC {
   actor Propietario_GC as "Propietario"

   (Registrar Cliente)
   (Actualizar Cliente)
   (Consultar Cliente)
   (Eliminar Cliente)

   Propietario_GC -- (Registrar Cliente)
   Propietario_GC -- (Actualizar Cliente)
   Propietario_GC -- (Consultar Cliente)
   Propietario_GC -- (Eliminar Cliente)
}
```

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

La diferencia más notable se centra en la precisión y la granularidad de la descripción funcional. El Caso de Uso textual (CU2.0) es de alto nivel y resulta impreciso al agrupar cuatro funciones operacionales (Registrar, editar, consultar y eliminar) bajo el único concepto "Gestionar Cliente". En contraste, el diagrama generado con PlantUML fuerza la descomposición en cuatro Casos de Uso atómicos (Registrar Cliente, Actualizar Cliente, etc.), lo que aumenta la precisión necesaria para el desarrollo del software.

Además, el modelado gráfico ofrece una claridad visual superior al explicitar el límite del sistema, el Actor ("Propietario") y las interacciones funcionales. Mientras que el texto es un resumen, PlantUML ayuda a validar la completitud al requerir la explicitación formal de las relaciones UML (<<include>> o <<extend>>) si estas fueran aplicables al flujo de eventos del Caso de Uso.

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

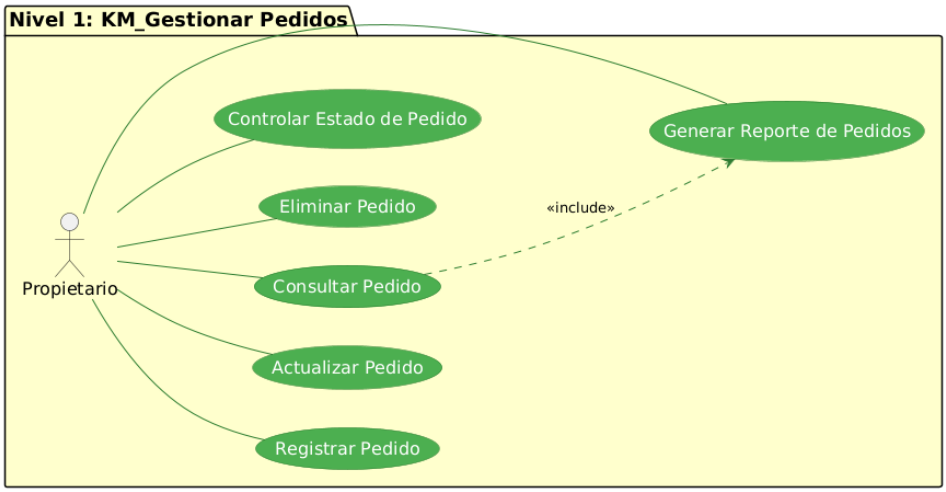
PlantUML facilita el trabajo del analista mediante la automatización y la trazabilidad. Permite la generación rápida de diagramas complejos a partir de código simple de texto, lo que acelera el modelado y permite al analista centrarse en la lógica UML en lugar de en el dibujo.docx, 9]. El código es versionable, lo que establece una trazabilidad directa entre el requisito textual, su modelo gráfico y el código de desarrollo. Esto también estandariza la comunicación con el desarrollador, asegurando un diseño preciso.

La principal limitación es que la generación automática de PlantUML puede limitar la libertad creativa del analista para realizar personalizaciones visuales no estándar que sí son posibles en herramientas de dibujo manual. También requiere una curva de aprendizaje inicial para dominar la sintaxis para el manejo de estructuras complejas.

CU03: Gestionar Pedido

CU3.0	Gestionar Pedido
Actor	Propietario
Descripción	El sistema deberá permitir al actor. Registrar, actualizar, consultar, eliminar los datos del pedido (especificados más adelante). Toda la información en la base de datos para su posterior seguimiento.

Commented [AM4]: Que datos del pedido?¿



Codigo PlantUML:

```
package "Nivel 1: KM_Gestionar Pedidos" #FFFFCC {
  actor Propietario_GPE as "Propietario"

  (Registrar Pedido)
  (Actualizar Pedido)
  (Consultar Pedido) as UC_CONP
  (Eliminar Pedido)
  (Controlar Estado de Pedido)
  (Generar Reporte de Pedidos) as UC_GR
  UC_CONP ..> UC_GR : <<include>>

  Propietario_GPE -- (Registrar Pedido)
  Propietario_GPE -- (Actualizar Pedido)
  Propietario_GPE -- UC_CONP
  Propietario_GPE -- (Eliminar Pedido)
  Propietario_GPE -- (Controlar Estado de Pedido)
  Propietario_GPE -- UC_GR
}
```

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

La principal diferencia se centra en la precisión funcional y la explicitación de relaciones lógicas del sistema. El Caso de Uso textual (CU3.0) es un concepto impreciso que agrupa todas las

operaciones de pedido (CRUD) bajo el título "Gestionar Pedido". El diagrama PlantUML fuerza la descomposición en funciones atómicas (Registrar, Actualizar, Consultar, Eliminar, etc.), lo que aumenta la precisión necesaria para el desarrollo.

En este nuevo ejemplo, la completitud del modelo gráfico con PlantUML es superior, ya que logra modelar la relación <<include>> entre "Generar Reporte de Pedidos" y "Consultar Pedido". Esta relación, crucial para la lógica del sistema, está completamente ausente en la descripción textual, demostrando cómo el modelado gráfico refina la claridad visual y la documentación lógica.

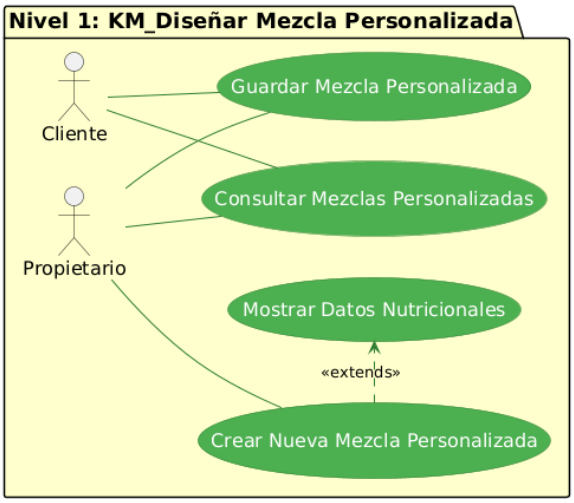
2. ¿De qué manera el uso de PlantUML facilita (o limita) el trabajo del analista al modelar los requisitos funcionales?

PlantUML facilita el trabajo del analista mediante la automatización y la documentación precisa de la lógica UML. Permite la generación rápida de diagramas complejos a partir de código simple de texto, lo cual acelera el modelado y permite al analista centrarse en la lógica UML.docx, 9]. La capacidad de representar relaciones avanzadas (<<include>> o <<extend>>) mediante texto asegura una documentación lógica rigurosa y establece una trazabilidad directa entre el requisito y su modelo.

Sin embargo, la principal limitación es que la generación automática puede limitar la libertad creativa del analista para realizar personalizaciones visuales no estándar que sí son posibles en herramientas de dibujo. Adicionalmente, el analista debe manejar la sintaxis para el correcto uso de relaciones complejas, lo que requiere un conocimiento sólido del lenguaje UML.

CU04: Diseñar Mezcla Personalizada

CU4.0	Diseñar Mezcla Personalizada
Actores	Propietario/Cliente
Descripción	El sistema deberá permitir a los actores crear, gestionar y almacenar mezclas de productos. Los actores proporcionarán las selecciones y cantidades de los productos. La información de las mezclas se procesará y almacenará en la base de datos para su consulta y uso posterior.



Codigo PlantUML:

```
package "Nivel 1: KM_Diseñar Mezcla Personalizada" #FFFFCC {
  actor Propietario_DMP as "Propietario"
  actor Cliente_DMP as "Cliente"

  (Crear Nueva Mezcla Personalizada) as UC_CNMP
  (Consultar Mezclas Personalizadas) as UC_CMP
  (Guardar Mezcla Personalizada) as UC_GMP
  (Mostrar Datos Nutricionales) as UC_MDN

  UC_CNMP .> UC_MDN : <<extends>>

  Propietario_DMP -- UC_CNMP
  Propietario_DMP -- UC_CMP
  Propietario_DMP -- UC_GMP

  Cliente_DMP -- UC_CMP
  Cliente_DMP -- UC_GMP
}
```

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

La principal diferencia radica en la precisión y la representación estructural de la lógica del sistema. El Caso de Uso textual (CU4.0) es impreciso al agrupar funciones de creación, gestión y

almacenamiento bajo un único concepto ("Diseñar Mezcla Personalizada"). El diagrama PlantUML, sin embargo, fuerza la descomposición en funciones atómicas como "Crear Nueva Mezcla Personalizada" y "Guardar Mezcla Personalizada".

En este ejemplo, la completitud lógica del diagrama es notablemente superior. El texto no menciona flujos alternativos, mientras que el diagrama PlantUML explicita una relación <<extend>> entre "Crear Nueva Mezcla Personalizada" y "Mostrar Datos Nutricionales". Esta representación gráfica mejora la claridad visual al definir con exactitud las responsabilidades de los múltiples actores ("Propietario/Cliente") y la arquitectura funcional del sistema.

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

PlantUML facilita el trabajo del analista mediante la automatización y la documentación rigurosa de la lógica UML. Permite la generación rápida de diagramas a partir de código simple de texto, acelerando el modelado y permitiendo al analista centrarse en la lógica del diagrama, especialmente en cómo representar correctamente las relaciones complejas (<<extend>> o <<include>>).docx, 9]. Esto establece una trazabilidad directa entre el requisito textual y su modelo formal, lo que estandariza la comunicación con el desarrollador.

La principal limitación es que la generación automática puede limitar la libertad creativa del analista para realizar personalizaciones visuales no estándar que sí son posibles en herramientas de dibujo. Además, exige al analista un conocimiento sólido de la sintaxis PlantUML para aplicar correctamente las relaciones y las múltiples asociaciones de actores.

Conclusiones generales:

- El modelado gráfico con PlantUML es crucial para la precisión y completitud lógica del diseño de requisitos. La herramienta obliga al analista a descomponer requisitos textuales imprecisos en funciones atómicas y a explicitar relaciones lógicas complejas (<<include>>/<<extend>>) que son esenciales para la arquitectura del sistema, pero a menudo omitidas en las descripciones iniciales de alto nivel.
- PlantUML mejora la eficiencia operativa y la trazabilidad, actuando como un puente estandarizado entre el análisis y el desarrollo. Al automatizar la generación de diagramas a partir de código simple de texto, acelera el modelado y permite una trazabilidad directa (código versus requisito).docx, 9]. Esto estandariza el lenguaje de diseño, facilitando una comunicación precisa y menos ambigua con el equipo de desarrollo.