Resumen Sistemas 1

El ciclo de vida del sistema tiene distintas fases, fase 1 análisis, fase 2 diseño, 3 desarrollo, 4 implementación, 5 seguimiento, control y auditoria. Para llevar adelante un proyecto hay que seguir ese orden.

En el análisis se busca realizar un diagnóstico donde se identifica cual es el problema que se tiene en términos de sistemas de información.

Un proceso de software es un conjunto de actividades relacionadas que conducen a la elaboración de un producto de software.

Define quien está haciendo que, cuando y como para alcanzar un determinado objetivo.

Fases del Proceso de Ingeniería de Sistemas:

- Definición de Requerimientos:
    - Requerimientos Funcionales Abstractos (se trata de abstraer la información y conocer su uso)
    - Propiedades del Sistema (no funcionales):
        - Que debe tener el sistema
        - Que NO debe tener el sistema
- Diseño del Sistema:
    - Dividir Requerimientos
    - Identificar subsistemas
    - Asignar los requerimientos a los subsistemas
    - Especificar funcionalidad de los subsistemas
    - Definir Interfaces de los subsistemas
- Desarrollo de los subsistemas
- Integración del sistema
- Instalación del sistema
- Operación del sistema
- Evolución del sistema
- Desmantelamiento del sistema

Existen muchos diferentes procesos de software, pero todos deben incluir cuatro actividades que son fundamentales para a ingeniería de software:

- Especificación del software: Tienen que definirse tanto la funcionalidad del software como las restricciones de su operación
- Diseño e implementación del software: Debe desarrollarse el software para cumplir con las especificaciones
- Validación del software: Hay que validar el software para asegurarse de que cumple con lo que el cliente quiere

- Evolución del software: El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

Modelos de Procesos:

RUP (Rational Unified Process):

Es un proceso para el desarrollo de software orientado a objeto que utiliza UML para describir un sistema.

Creado por Rational Software Corporation en 1998

Es un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Principios RUP:

La filosofía esta basado en 6 principios clave que son los siguientes:

- Adaptar el proceso
- Equilibrar prioridades
- Demostrar valor iterativamente
- Colaboración entre equipos
- Enfocarse en la calidad
- Elevar el nivel de abstracción

Perspectivas de RUP:

Se describe desde tres perspectivas:

- Una perspectiva dinámica que demuestra las fases del modelo a través del tiempo
- Una perspectiva estática que representa las actividades del proceso que se establecen
- Una perspectiva practica que sugiere buenas practicas a usar durante el proceso

Perspectiva Dinámica:

Se identifican 4 fases discretas en el proceso de software

Las fases están mas estrechamente vinculadas con la empresa que con las preocupaciones técnicas

Fase 1: Concepción:

- La meta de la fase de concepción es establecer un caso empresarial para el sistema. Deben identificarse todas las entidades externas (personas y sistemas)
- Luego se usa esta información para valorar la aportación del sistema hacia la empresa
- Si esta aportación es menos, entonces el proyecto puede cancelarse después de esta fase

Fase 2: Elaboración

- Las metas son:
    - Desarrollar la comprensión del problema del dominio
    - Establecer un marco conceptual arquitectónico para el sistema
    - Diseñar el plan del proyecto e identificar los riesgos clave del proyecto
- Al completar esta fase, debe tenerse un modelo de requerimientos para el sistema que podría ser una serie de casos de uso del UML, una descripción arquitectónica y un plan de desarrollo para el software.

Fase 3: Construcción.

- La fase de construcción incluye diseño, programación y pruebas del sistema.
- Partes del sistema se desarrollan en paralelo y se integran durante esta fase.
- Al completar esta, debe tenerse un sistema de software funcionando y la documentación relacionada y lista para entregarse al usuario.

Fase 4: Transición.

- Se interesa por el cambio de sistema desde la comunidad de desarrollo hacia la comunidad de usuarios, y por ponerlo a funcionar en un ambiente real.
- Esto es algo ignorado en la mayoría de los modelos de proceso de software, aunque, en efecto es una actividad costosa y en ocasiones problemática.
- Al finalizar esta fase se debe tener un sistema de software documentado que funcione correctamente en su entorno principal.

Perspectiva estática:

Se enfoca en las actividades que tienen lugar durante el proceso de desarrollo.

Se les llama flujos de trabajo.

En el proceso se identifican seis flujos de trabajo de proceso centrales y tres flujos de trabajo de apoyo centrales.

Como RUP y UML se desarrollaron juntos la descripción del flujo de trabajo se apoyan sobre los modelos UML asociados.

Flujos de trabajo:

- Modelo del negocio: Se modelan los procesos del negocio usando los casos de uso de la empresa.
- Requerimientos: Se identifican los actores y como interactúan con el sistema.
- Análisis y Diseño: Se crea y documenta un modelo de diseño utilizando modelos arquitectónicos, de componentes, de objetos y de secuencias.

- Implementación: Se implementan y estructuran los componentes del sistema en subsistemas de implementación.
- Pruebas: Las pruebas son un proceso iterativo que se realiza en conjunto con la implementación. Las pruebas del sistema siguen al completar la implementación.
- Despliegue: Se crea la liberación del producto hacia el usuario y se instala en su lugar de trabajo.
- Administración de la configuración y del cambio: Gestiona los cambios al sistema.
- Administración del proyecto: Gestiona el desarrollo del sistema.
- Entorno: Pone a disposición del equipo de desarrollo de software las herramientas adecuadas de software.

Características del ciclo de vida de RUP:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

UML Modelos y Diagramas:

Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto de software desde cada una de las perspectivas de interés.

¿Qué es un producto de software?

Un producto de software NO ES el código y los ejecutable de un sistema.

Un producto de software es el conjunto de artefactos que se necesitan para representarlo en forma comprensible para:

- Las maquinas.
- Los trabajadores.
- Los usuarios.
- Los interesados.

El concepto artefacto es aplicable a cualquier tipo de información creada, cambiada o utilizada por los trabajadores en el desarrollo del sistema.

Ejemplos:

- Diagramas UML y su texto.

- Bocetos de interfaz.
- Planes de prueba.

Un modelo es una abstracción que considera un cierto propósito. Captura una vista de un sistema del mundo real.

Describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

Un diagrama es una representación gráfica de una colección de elementos de modelado.

Los diagramas UML tienen una jerarquía, diferenciado en mayor parte por dos ramas:

- Los diagramas de Estructura:
    - Diagramas de Paquetes.
    - Diagrama de Clases.
    - Diagrama de Despliegue.
    - Diagrama de Objetos
    - Diagrama de Componentes
    - Diagrama de Estructuras Compuestas.

- Los diagramas de Comportamiento:
    - Diagrama de Actividad.
    - Diagrama de Estados.
    - Diagrama de Casos de Uso.
    - Diagrama de Interacción.

UML define una notación que se expresa como diagramas sirven para representar modelos/subsistemas o partes de ellos.

El 80% de los problemas pueden modelarse usando alrededor del 20% de UML.

Casos de Uso:

Son una herramienta para el modelado de los requerimientos.

Documentan el comportamiento del sistema desde el punto de vista del usuario, ayuda con tres de los aspectos del desarrollo:

- La captura de requisitos.

- La planificación de las iteraciones del desarrollo.
- La validación de los sistemas.

El diagrama muestra todos los casos de uso de un sistema dado.

Un caso de uso individual representa un tipo de tarea a la cual el sistema debe dar soporte.

Es una descripción de un proceso fin-a-fin, relativamente largo, que incluye varias etapas o transacciones.

Es una manera específica de usar el sistema, es una historia que describe un uso particular del sistema.

Escenario:

Es una secuencia de acciones e interacciones entre los usuarios y el sistema.

Actor:

Un actor representa el rol jugado por una persona o cosa que actúa con el sistema, seria mejor usar la palabra rol, pero algunos piensan que actor fue usado debido a una mala traducción del sueco.

Los actores son los que toman beneficio del sistema, “Beneficiarios”.

Es importante identificar los beneficiarios de cada caso de uso. Si un caso de uso tiene valor para un determinado actor, ese actor permanecerá conectado al caso de uso a lo largo del desarrollo.

Los actores pueden no ser humanos:

- Teclado
- Internet
- Otra aplicación

Importante: Solo son actores aquellos que utilizaran el sistema. El resto de los interesados no se identifican en el diagrama.

Características de los Casos de Uso:

- Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario.
- Se puede considerar que, hasta cierto punto, cada caso de uso es independiente de los demás.
- Permiten definir los límites del sistema y las relaciones entre el sistema y su entorno.
- Un caso de uso NO es un diagrama, NO es un símbolo dentro de un diagrama, es una forma de describir un escenario de interacción usuario-sistema.
- Un diagrama de caso de uso es una forma de tener una visión general de todos los casos de uso, sus relaciones con los actores y con otros casos de uso.

Características de los Escenarios:

- Hay muchas variaciones sobre cómo se puede describir un caso de uso ya que UML no define ningún estándar al respecto.
- Cuando lo utilice en forma profesional, seleccione o diseñe una o mas plantilla que considere adecuadas para sus necesidades.
- Permiten definir los limites del sistema y las relaciones entre el sistema y su entorno.
- Conozca bien la plantilla que va a utilizar, sepa para que sirve cada campo.

Utilización de los casos de uso:

Para captura de requerimientos:

- Proporcionando una forma estructurada de abordarlos:
    - Identificar los actores.
    - Para cada actor, averiguar:
        - Lo que necesita del sistema: es decir, que casos de uso hay que tienen valor para ellos.
        - Cualquier otra interacción que esperan tener con el sistema, esto es, en que casos de uso podrían formar parte para el beneficio de otro.
- Para propósitos de priorización de trabajo.
- Para planificación de iteraciones del desarrollo.

En el desarrollo: Planificación:

- Antes de realizar una se necesita tener una lista con todos los casos de uso el sistema junto con:
    - Una buena idea de lo que significa cada uno.
    - Un entendimiento de quien quiere cada uno y cuánto.
    - El conocimiento de que casos de uso tienen más riesgo.
    - Un plan de cuánto tiempo llevaría implementar cada caso de uso

En el desarrollo: Aspectos Políticos:

- Si se han capturado los requisitos en función de los actores y los casos de uso, es probable que se tenga una buena idea de que casos de uso son más importantes para que personas.
- Tan pronto como sea posible, se quiere asegurar que todo aquel que tiene poder para hundir el sistema no tiene una buena razón para hacerlo.

En el desarrollo: Aspectos Técnicos:

- Hay que entregar primero los casos de uso con mayor riesgo, para abordar los mayores riesgos cuando todavía se tiene la contingencia de abordaros, y asi uno no se queda atado en un diseño que no permita tratar los casos de uso más difíciles.
- En otras circunstancias, se pueden abordar primero las partes más fáciles de un problema.

- La manera en que los casos de uso son descriptos, varia a lo largo del proceso de desarrollo.
    - Casos de uso del negocio.
    - Casos de uso del sistema.
- Para empezar, es importante identificar que se debería alcanzar en cada caso de uso, no como debería alcanzarse.
- Mas tarde se elegirá una implementación.
    - Pueden cambiar los actores.

En el desarrollo: Validación:

- Un diseño correcto permite que se ejecute cada caso de uso; es decir, realiza cada caso de uso.
- Técnicas:
    - Tomar de uno en uno los casos de uso y comprobar que el sistema permite ejecutar dicho caso de uso.
    - La misma técnica puede utilizarse para derivar pruebas del sistema.

Posibles problemas con los casos de uso:

- Hay peligro de construir un sistema que no sea orientado a objetos.
- Hay peligro de tener un diseño de requisitos erróneo.
- Hay peligro de perder requisitos si se pone demasiada confianza en el proceso sugerido para encontrar los actores y después encontrar los casos de uso que necesita cada actor.

Modelo de Casos de Uso:

¿Cómo se desarrolla un diagrama de Casos de Uso?

Antes de hacer un caso de uso es necesario tratar de entender los requerimientos del sistema. Trate de expresar lo que el sistema debe hacer.

En base a esto, responda:

- ¿Cuáles son las tareas del/los actores involucrados?
- ¿Qué datos debe el actor crear, guardar, modificar, destruir, leer?
- ¿Debe el actor informar al sistema de cambios externos ocurridos?
- ¿Debe el sistema informar al actor de cambios internos?

Asociaciones entre Casos de Uso:

Asociaciones:

- Colaboraciones.
- Extensiones.
- Inclusiones.

- Herencia.

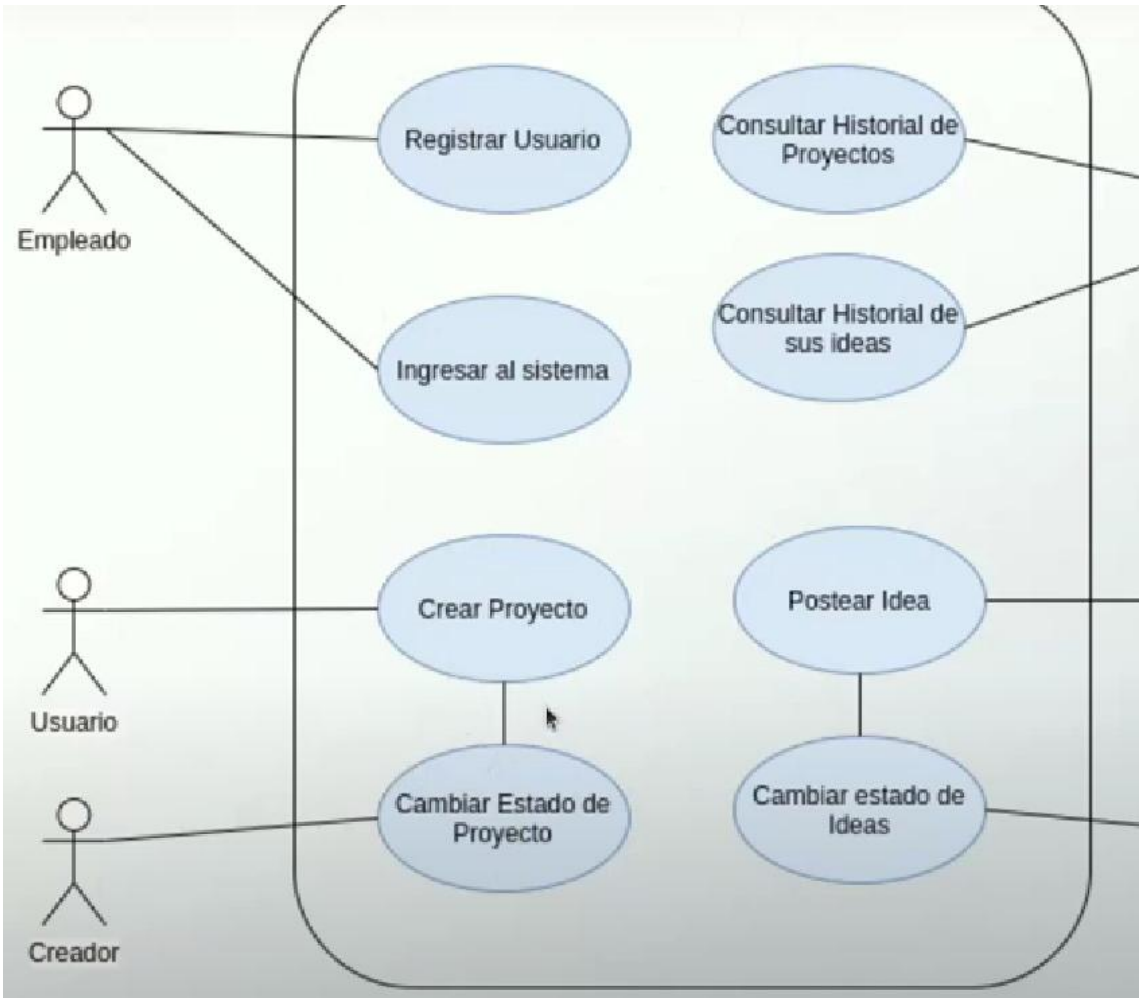
Estereotipo:

Un elemento de texto que al ser aplicado a otro elemento define su categoría. Es uno de los mecanismos de extensión del lenguaje toda vez que permite cambiar o complementar la semántica de cualquier elemento.

Es una cadena de texto encerrada entre los símbolos de comillas francesas (<< >>).

Colaboración:

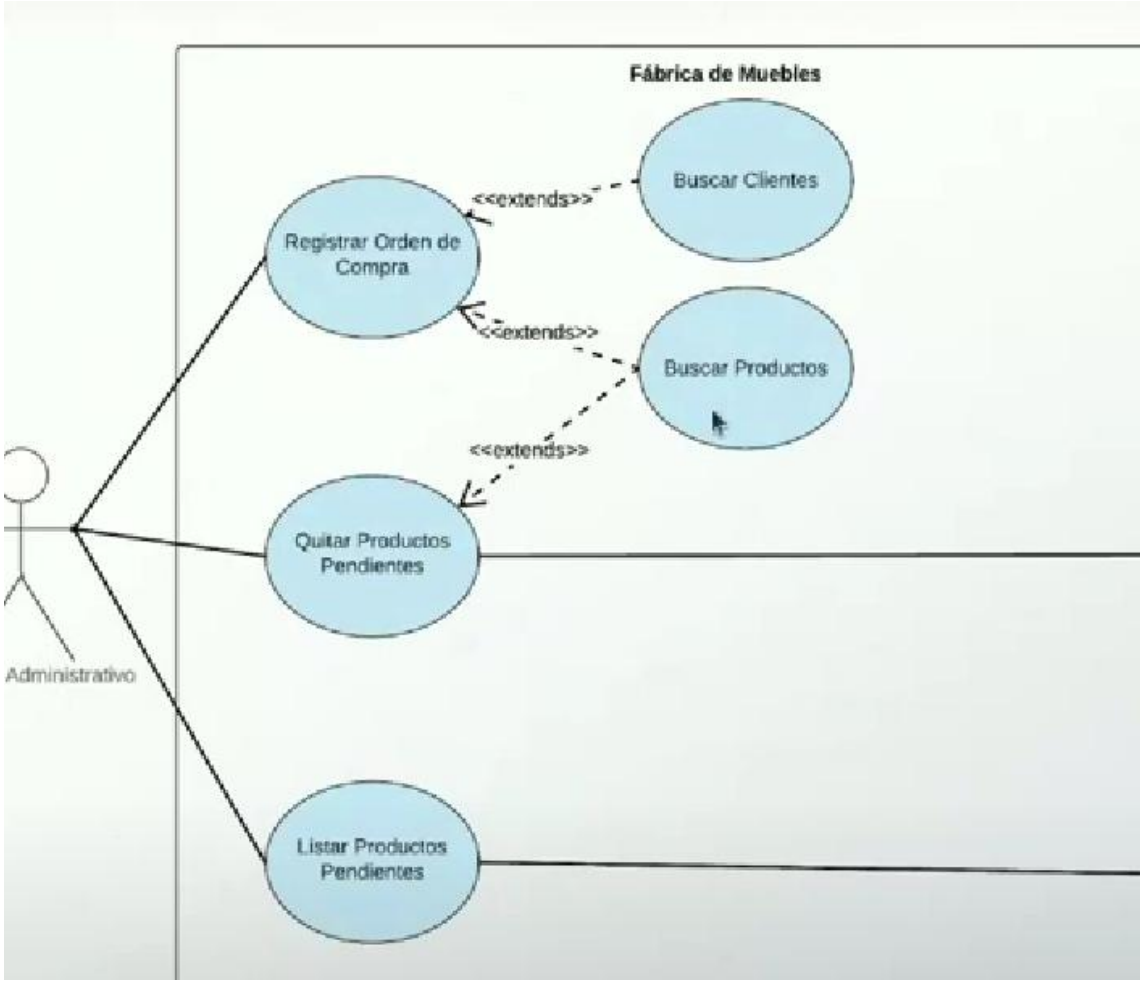
Denota un tipo de asociación donde los casos de uso asociados “colaboran estrechamente”. Se denota con una línea llena sin orientación.



Extensión:

Se denota con una flecha con línea punteada que va desde el caso que lo extiende hasta el caso base y se anota con el estereotipo <<extends>>.

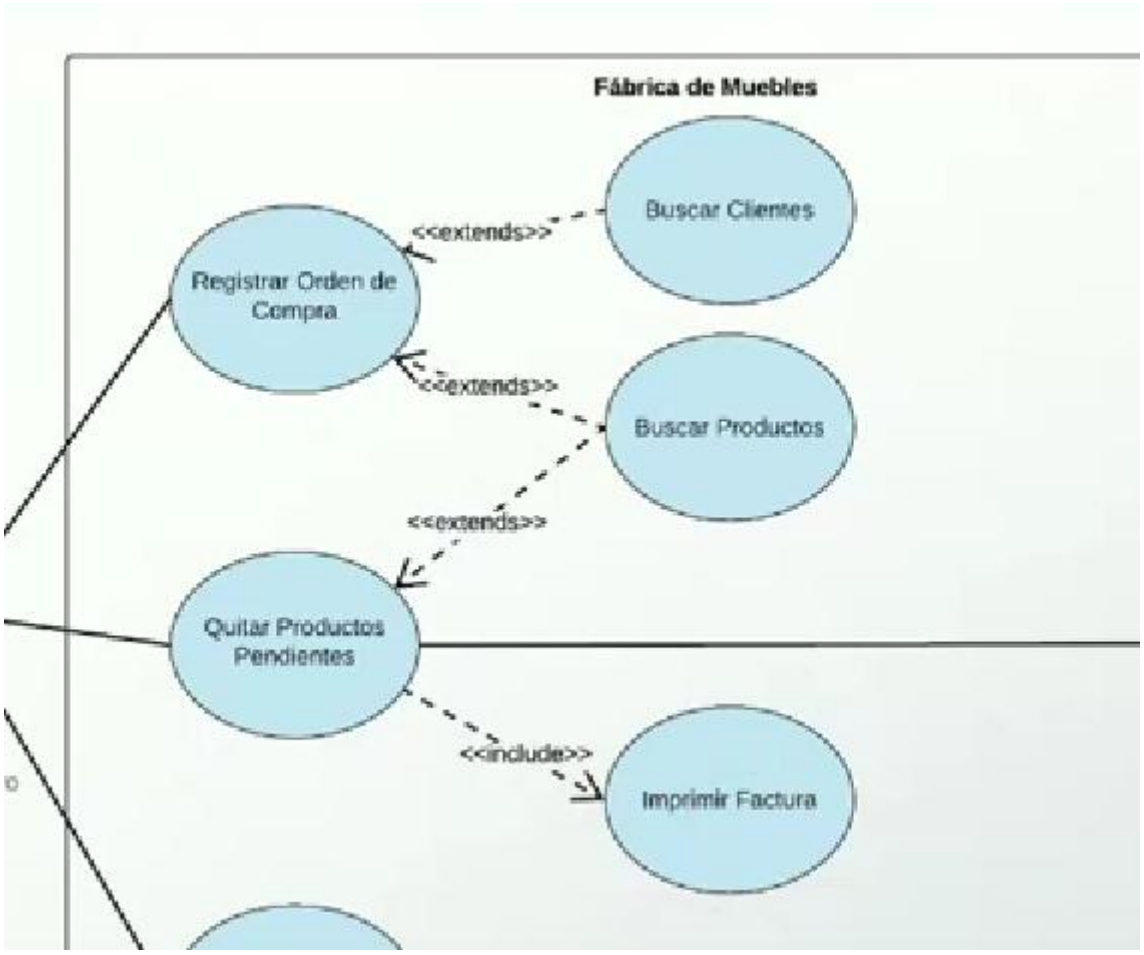
- No es un comportamiento normal, es una extensión, por lo tanto, debe ser invocado en algún flujo alternativo.
- Se usa para extender comportamientos.



Inclusión:

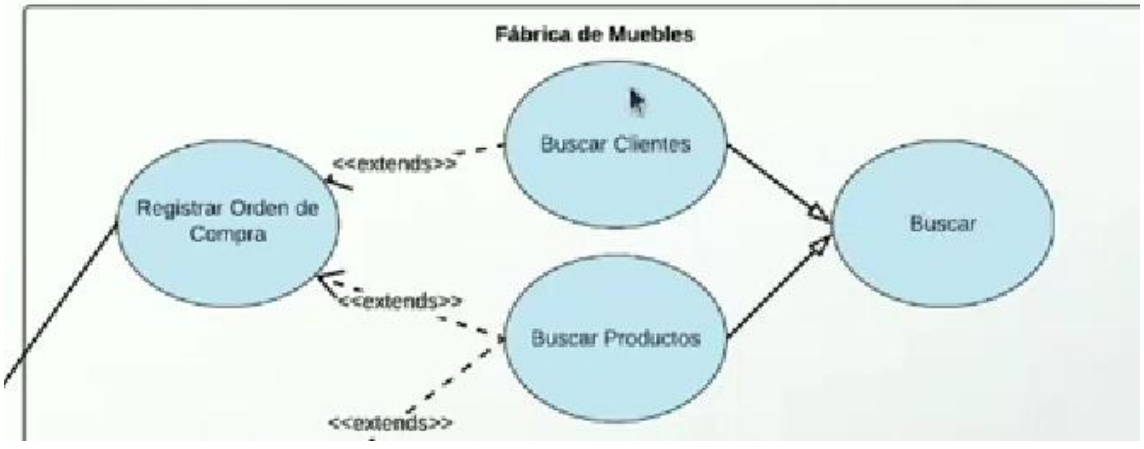
Se denota con una línea punteada que va desde el caso de uso hasta el caso incluido y se anota con el estereotipo <<include>>.

- Es un comportamiento obligatorio, por lo tanto, debe ser invocado en el flujo normal.
- Se usa para incluirlo en varios casos de uso (y compartir comportamientos).



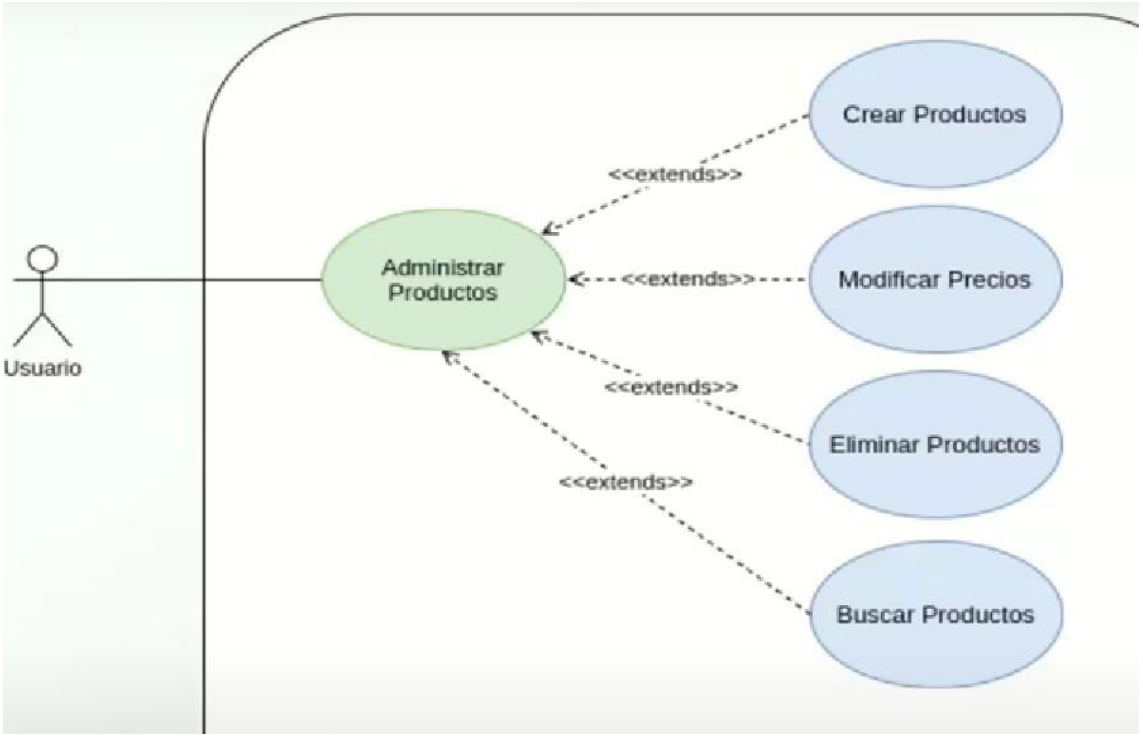
Herencia:

Se utiliza como una forma de simplificar la descripción de un comportamiento compartido por varios casos de uso.



Casos de Uso: Definiciones de CRUDs

Ejemplo de CRUD:



Se deben identificar los requerimientos funcionales:

- Es la descripción de un servicio que el software debe ofrecer para satisfacer una necesidad de los usuarios.