

Casos de Uso (Use Case)

Introducción

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

- [Actor](#).
- [Casos de Uso](#).
- [Relaciones de Uso, Herencia y Comunicación](#).

Elementos

- **Actor:**



Una definición previa, es que un **Actor** es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra **rol**, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

- **Caso de Uso:**



Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

- **Relaciones:**

- **Asociación** 

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

- **Dependencia o Instanciación** 

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

- **Generalización** 

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso** (<<uses>>) o de **Herencia** (<<extends>>).

Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores).

extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde esta la duda clásica de **usar** o **heredar**.

Ejemplo:

Como ejemplo esta el caso de una Máquina Recicladora:

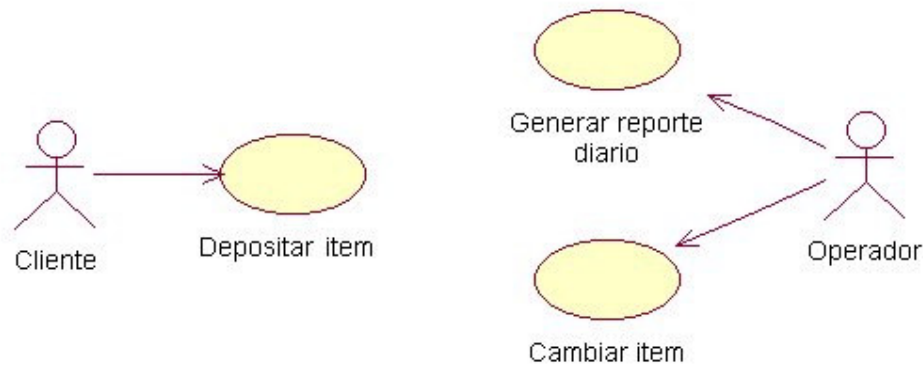
Sistema que controla una máquina de reciclamiento de botellas, tarros y jabas. El sistema debe controlar y/o aceptar:

- Registrar el número de ítemes ingresados.
- Imprimir un recibo cuando el usuario lo solicita:
 - a. Describe lo depositado
 - b. El valor de cada ítem
 - c. Total
- El usuario/cliente presiona el botón de comienzo
- Existe un operador que desea saber lo siguiente:
 - a. Cuantos ítemes han sido retornados en el día.
 - b. Al final de cada día el operador solicita un resumen de todo lo depositado en el día.
- El operador debe además poder cambiar:
 - a. Información asociada a ítemes.
 - b. Dar una alarma en el caso de que:
 - i. Ítem se atora.
 - ii. No hay más papel.

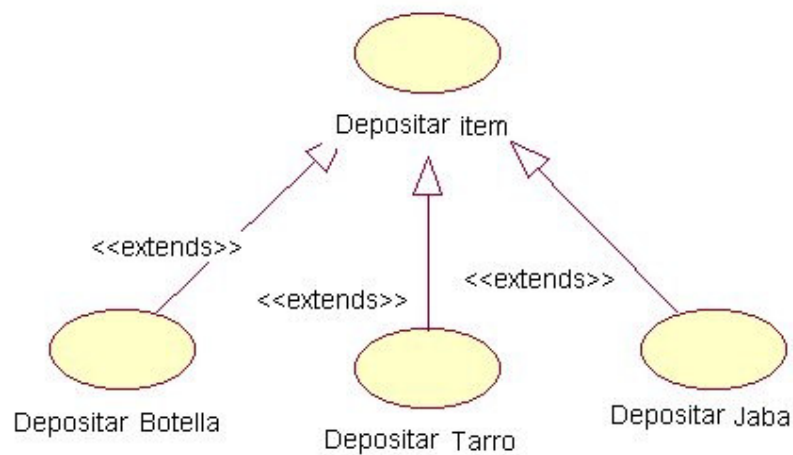
Como una primera aproximación identificamos a los actores que interactuan con el sistema:



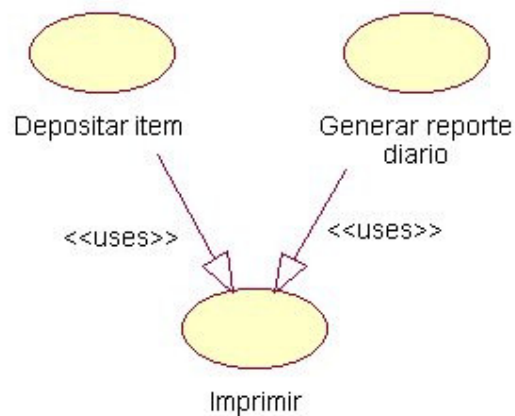
Luego, tenemos que un Cliente puede Depositar Ítemes y un Operador puede cambiar la información de un Ítem o bien puede Imprimir un informe:



Además podemos notar que un item puede ser una Botella, un Tarro o una Jaba.



Otro aspecto es la impresión de comprobantes, que puede ser realizada después de depositar algún item por un cliente o bien puede ser realizada a petición de un operador.



Entonces, el diseño completo del diagrama Use Case es:

