APUNTE RESUMEN DISEÑO

**-REQUISITOS**

**-CASOS DE USO**

**-DIAGRAMA DE CLASES**

**REQUISITOS**

1. Requisitos Funcionales

2. Requisitos No Funcionales

3. Requisitos de Dominio

**Requisitos funcionales**

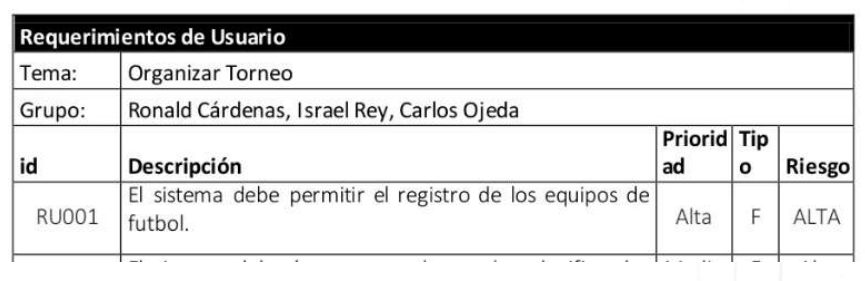
Los requirimientos funcionales son las características que tendrá el sistema o app, son las que actúa directo con el usuario (pantallas,opciones,funciones del sistema) . algo puntual que el sistema deba hacer

Ejemplo: El sistema enviará un correo electrónico al cliente cuando se registre el pago,

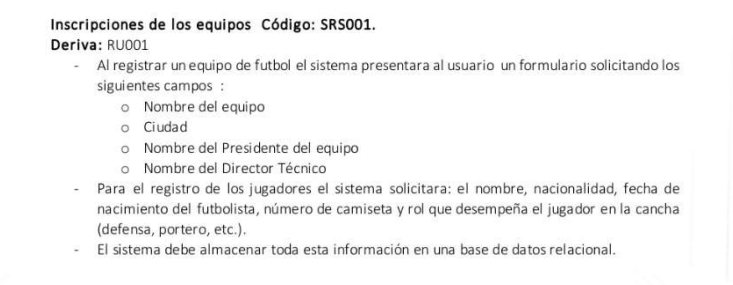
El sistema pedirá un Uber cuando se registre la ubicación y las dos partes acepten

**Dentro de los Requisitos Funcionales tenemos:**

**Requisitos de usuario :** Están especificados desde el punto de vista del usuario. Generalmente suelen describirse de forma abstracta

****

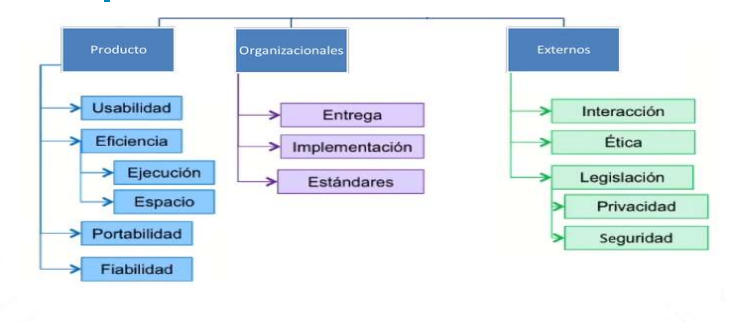
**Requisitos del sistema :** Están especificados desde el punto de vista del sistema ( explican como el sistema debe proporcionar los requerimientos del usuario. Deben ser una especificación completa y consistente de un sistema)

****

**Requisitos NO funcionales**

Los requerimientos no funcionales definen las características o cualidades generales que se esperan de un sistema y establecen restricciones sobre el producto, el proceso de desarrollo de software y establecen restricciones externas que el software debe lograr

Ej : El sistema debe permitir la realización de 1000 transacciones por segundo.

****

**Del producto :** Especifican el comportamiento del producto.

Ej: La tasa de fallos por minuto del sistema no debe ser superior a dos. (Fiabilidad).

**Organizacionales** : Se derivan de políticas y procedimientos existentes en la organización.

Ej: se debe seguir el proceso de calidad que marca el estándar ISO 9001. (Estándar).

**Externos :** se derivan de factores externos al sistema y al proceso de desarrollo, es decir, como el sistema interactúa con otros sistemas y con otras organizaciones.

Ej: Un operador no tendrá acceso a la dirección del cliente. (Privacidad).

**Requisitos de dominio**

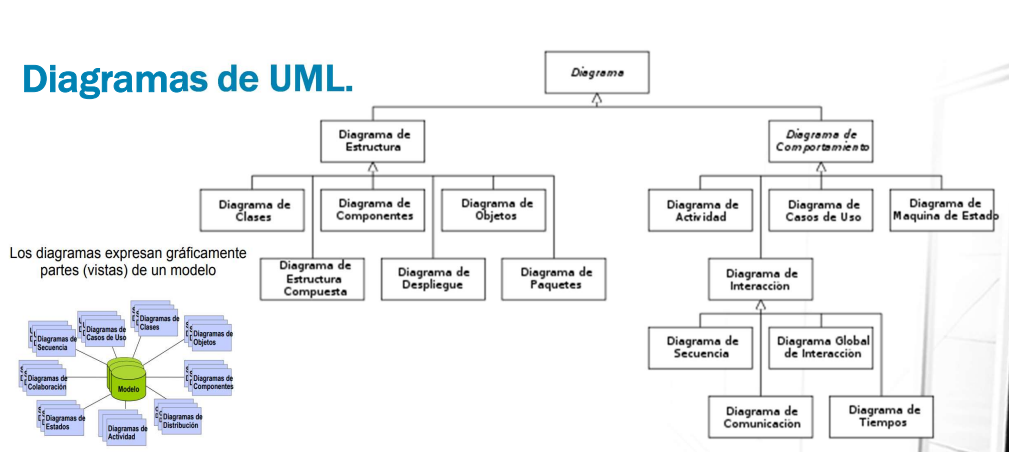
* reflejan características del dominio de la aplicación
* FUNCIONALES O NO FUNCIONALES
* Tienen que ver directamente con “las reglas de juego” de la aplicación que se está construyendo.
* Para los expertos del dominio las especificaciones pueden ser obvias, no así para los desarrolladores.

ej: El alumno no podrá inscribirse a una materia de la cual adeude su correlativa.

**EL Documento formal que contiene los requisitos de un sistema. es el estándar IEEE 830-1998 para el SRS(en inglés) o ERS (Especificación de requisitos de software)**

Lenguaje de Modelado Unificado (UML)

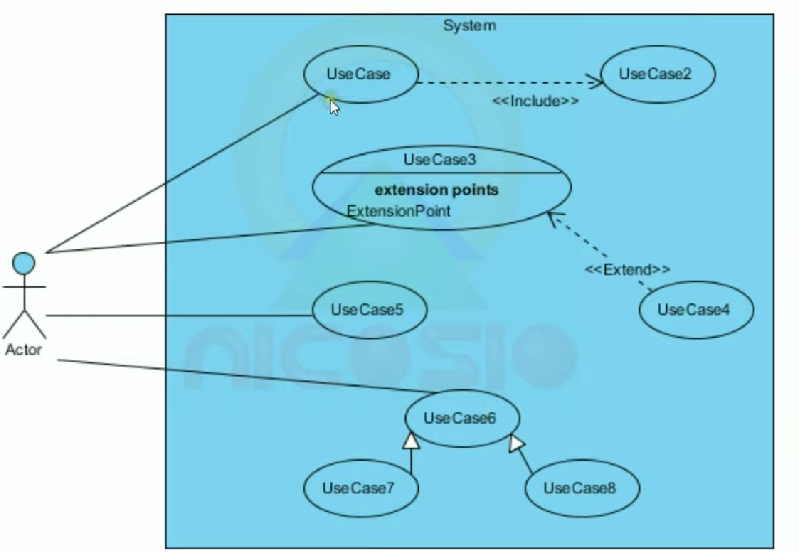
UML da soporte al desarrollo de software pero no especifica qué metodología o proceso usar. UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de los sistemas que se modelan.

****

**Diagrama de Casos de Uso (CU).**

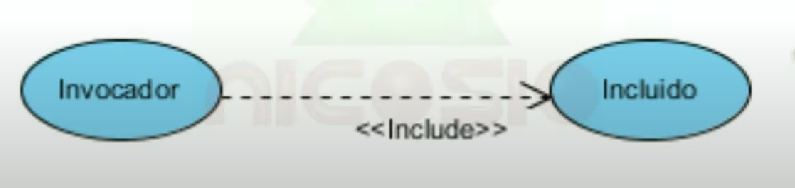
Muestra a los actores involucrados en un sistema, las funciones presentes en el sistema y cómo éstas interactúan con dichos actores. Muestra actores, casos de uso, paquetes de casos de uso y sus relaciones.

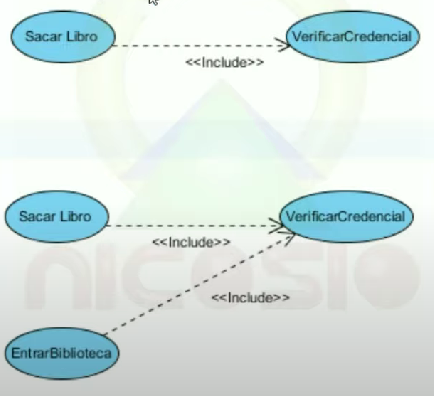
****

****

**<<incluide>>**

**DEPENDE si o si de otro atributo para existir (siempre se coloca a la derecha del invocador)**

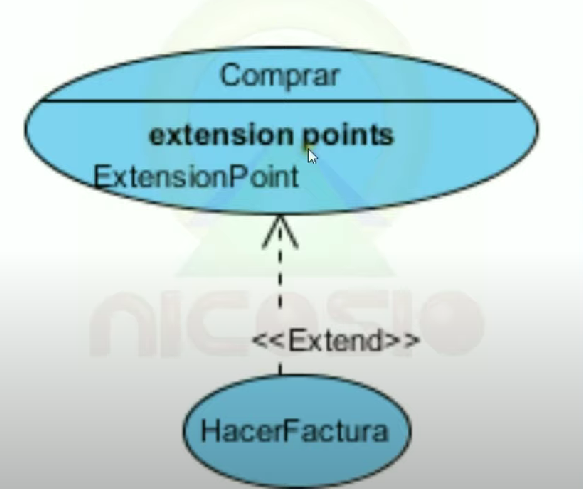
****

****

**Sacar libro <DEPENDE> --> de verificarcredencial**

**<<Extend>>**

**Es un caso de uso que extiende la funcionalidad de otro ( ES OPCIONAL )**

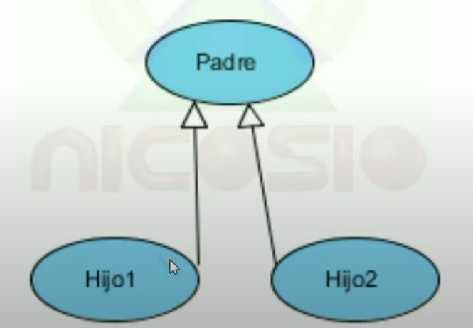
****

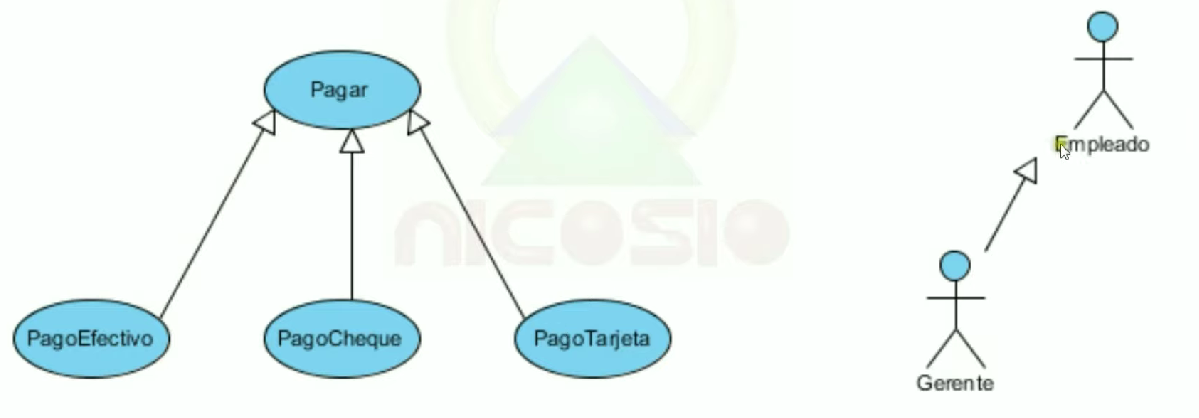
**Hacer factura <(OPCIONAL)> --> comprar**

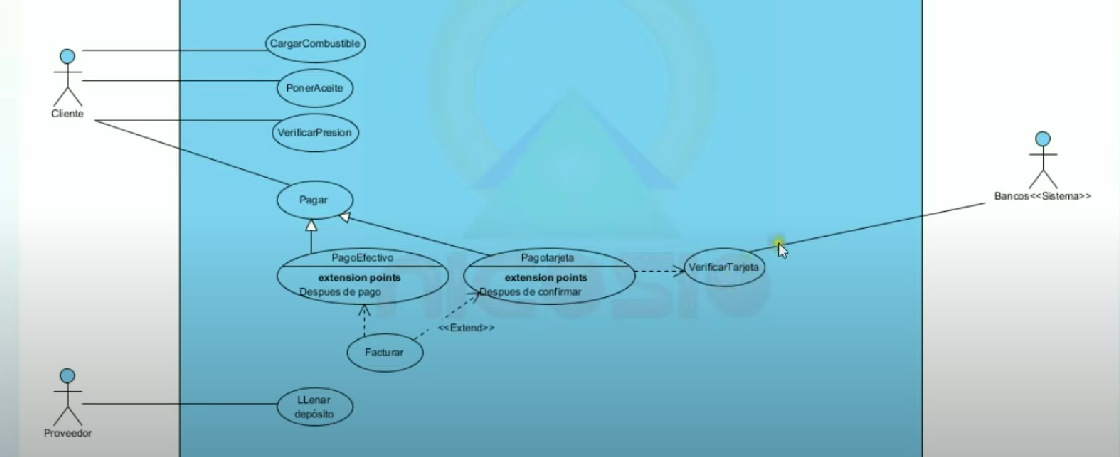
**Generalización (Similar a herencia)**

**El hijo tiene cosas en común con el padre , pero adiciona su propio comportamiento o atributos**

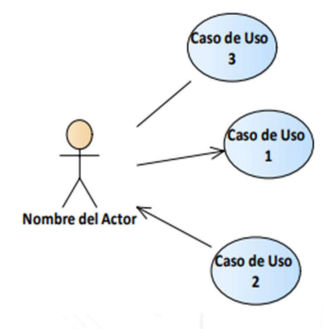
**hijo --> padre**

****

****

**Ejemplo** ****

**Asociación**

****

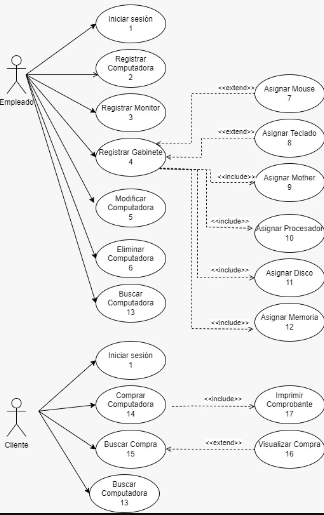
1: esta navegabilidad se utiliza para indicar que es el actor el que inicia la relación de interacción con el caso de uso.

2: esta navegabilidad se utiliza para indicar que es el Caso de Uso el que inicia la relación de interacción con el Actor.

3. Se utiliza cuando no está definido aún quien de los dos iniciar la relación de interacción.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso: Comprar Computadora** | **Nro: 3** | **Versión: 1.0** |
| **Actor principal: Cliente** | **Actor secundario:** | |
| **Objetivo: Comprar una computadora** | | |
| **Pre-condición: Que haya stock disponible, debe iniciar sesión** | | |
| **Pos-condición: Entrega de comprobante** | | |

|  |
| --- |
| **Flujo Básico** |
| 1.Cliente: Buscar la computadora por nombre.  2.Sistema: Buscar el producto , verificar que haya stock  3.Cliente: Realizar compra  4.Sistema: Verifica saldo del comprador  5.Sistema: Crea el comprobante y lo entrega al cliente.  6.Sistema: Disminuye el stock  7: Sistema : envia el producto  8:Cliente : recibe el producto  9.Fin de caso de uso |
| **Flujos Alternativos** |
| 2:Mostrar mensaje de error que no hay Stock  4:mostrar mensaje que no pudo realizar la compra por falta de saldo |
| **Observaciones** |
| Al usuario se le permite buscar su compra por su identificador para visualizarla e imprimir el comprobante |

****

**Diagrama de clases**

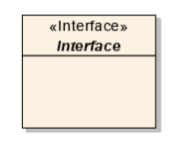
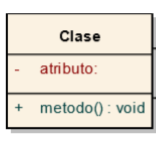
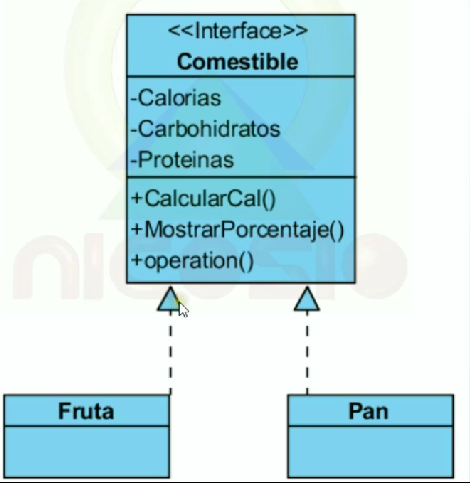
****

**Publica +**

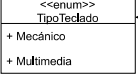
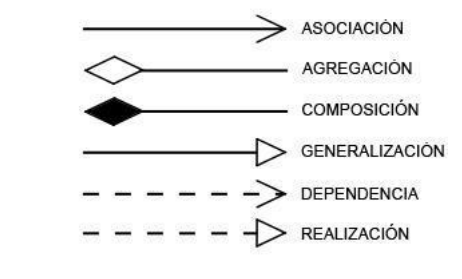
**Privada –**

**Protegida #**

**Paquete ~**

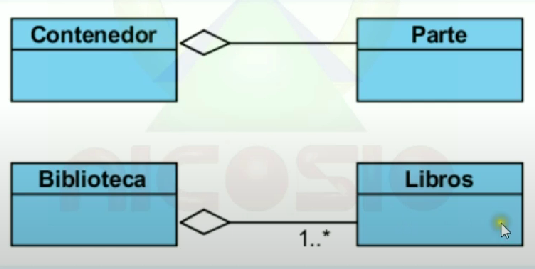
**** ****

**Relaciones**

****

**Agregacion**

**tienen existencias por separados , si uno deja de existir el otro puede seguir existiendo**

****

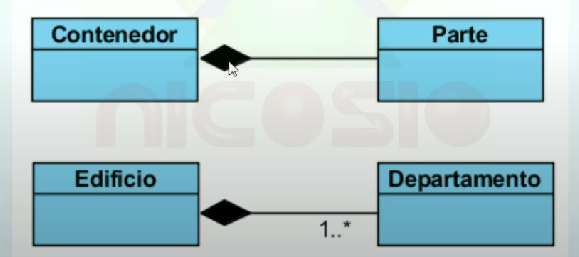
**\*es parte de**

**\*esta hecho de**

**Composición**

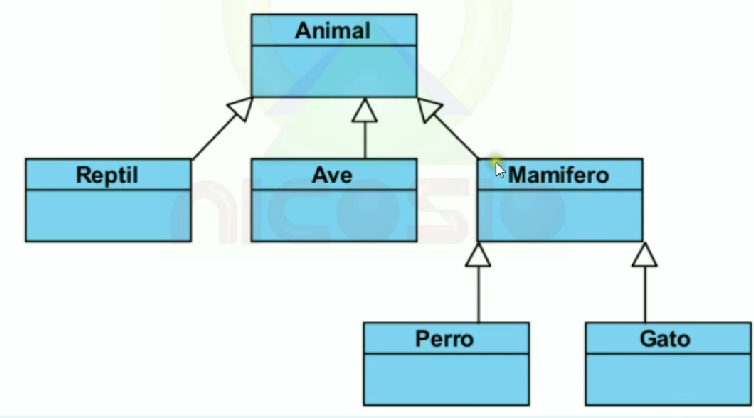
**\*Si el contendor se destruye , se destruyen las otras partes (Depende si o si del contenedor para existir)**

**\*En la composición la parte puede pertenecer a un solo contenedor**

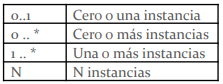
****

**Generalización(herencia)**

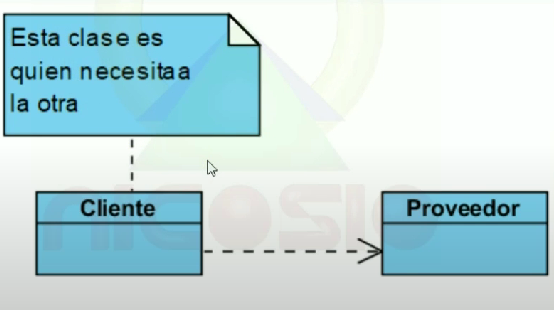
**-Es un**

****

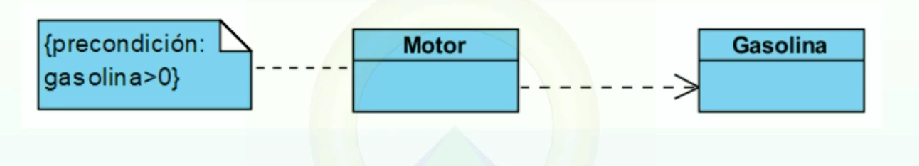
**Multiplicidad(cardinalidad)**

****

**Notas**

****

**Restricciones**

****