UML

Sesión conceptual 1

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercitación

- Reflexión

Inicio

15 minutos

{desafío} latam_

Objetivo

- Implementar una aplicación básica de consola utilizando las buenas prácticas y convenciones para resolver un problema de baja complejidad acorde al lenguaje Java.

Desarrollo

150 minutos

{desafío} latam_

/* Introducción a UML */

{desafío}
latam_

Introducción

- Los escenarios eran representados de forma muy rústica.
- Modelos que se construían; pero no se documentaron.
- Ivar Jacobson, uno de los tres amigos, elevó la visibilidad del caso de uso (su nombre para un escenario). a tal punto, que lo convirtió en un elemento primario de la planificación y el desarrollo de proyectos de software.

Algunas definiciones

- ¿Qué es un modelo?
- ¿Qué es UML?
- UML es un **lenguaje** y no una metodología.

Un poco de historia

- Muchos diseños para la programación procedural.
- El paradigma de la orientación a objetos.
- Smalltalk, C++ , Java, etc.
- Muchas formas de modelar este “nuevo” paradigma.
- Los tres amigos.

A la hora de implementar este lenguaje, ¿Qué tan rigurosos debemos ser?

- El propósito de UML es ser una herramienta.
- Para entender un problema, sin dar tanta importancia a los detalles.
- Basta con que expliquen el problema de una forma global.
- Usar lo mínimo que nos aporte una mejor interpretación del problema que se desea resolver.
- Diferir mucho la estructura oficial, puede resultar en que no se comprenda lo que se desee expresar.

Antes de usar UML, ¿Qué nos debemos preguntar?

- ¿Cuál será el beneficio de utilizarlo?
- ¿Cómo su uso ayudará al momento de implementar el código?

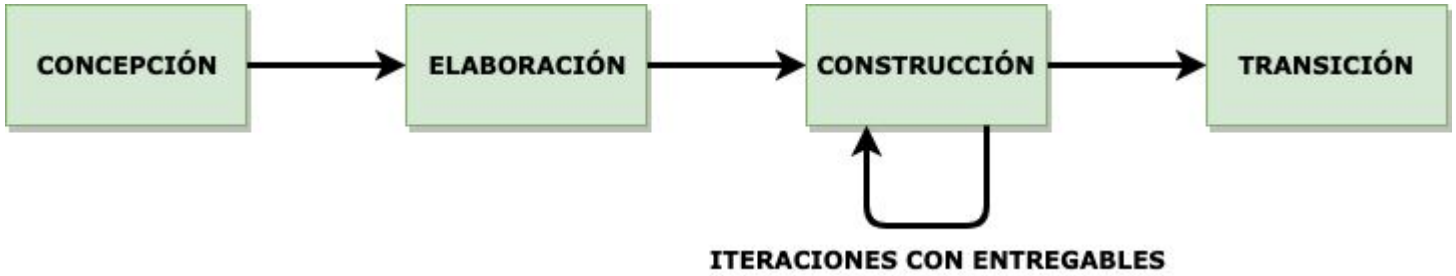
UML y los paradigmas de programación

"Los lenguajes de objetos, permiten ventajas pero no las proporcionan. Para aprovechar estas ventajas hay que realizar el infame cambio de paradigma. (¡Sólo asegúrese de estar sentado al momento de hacerlo!)" - Tom Hadfield.

- Programación Imperativa.
- Programación Orientada a Objetos.
- Programación Declarativa.
- Programación Funcional.
- UML aún nos entrega una interfaz entre lo que es abstracto y lo que podríamos llamar tangible.

El proceso de desarrollo de software

- El proceso de desarrollo de software, en su nivel más alto
- ¿Cuándo se debe usar el desarrollo iterativo?



La práctica de la ingeniería de software 1.

- Entender el problema (Comunicación y análisis).
- **Planear la solución (Modelado y diseño del software).**
- Ejecutar el plan (generación del código).
- Examinar la exactitud del resultado (probar y asegurar la calidad).

La práctica de la ingeniería de software 2

- ¿Ha visto antes, problemas similares?
- ¿Hay patrones reconocibles en una solución potencial?
- ¿Hay algún software existente que implemente los datos, funciones y características que se requieren?
- ¿Ha resuelto un problema similar?
- ¿Son reutilizables los elementos de la solución?
- ¿El problema, se puede dividir en problemas más pequeños?
- ¿Hay soluciones evidentes para estos?

Un poco acerca de requerimientos.

- Son descripciones explícitas del comportamiento que debe tener el sistema y qué información debe manejar.
- Requerimientos funcionales (qué hacer).
  - El sistema deberá ser capaz de generar el reporte anual.
  - Mantendrá los datos de ventas actualizados.
- Requerimientos no funcionales (cómo hacerlo).
  - Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador.
  - El tiempo de aprendizaje del sistema por usuario; deberá ser menor a 4 horas.

Tipos de diagramas en UML

Tipos Estructurales

- **Diagrama de clases.**
- Diagrama de paquetes.
- Diagrama de objetos.
- Diagrama de componentes.
- Diagrama de estructura compuesta.
- Diagrama de despliegue.

Tipos comportamiento.

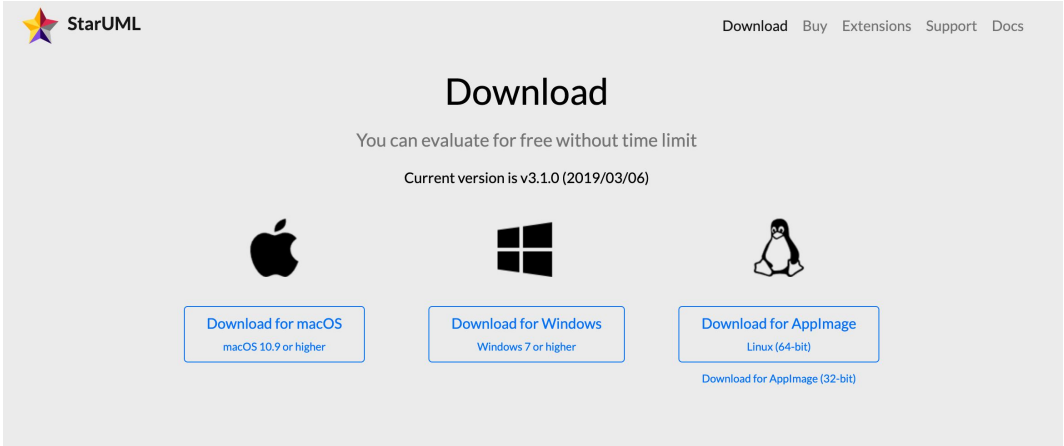
- Diagrama de actividad.
- **Diagrama de secuencia.**
- **Diagrama de casos de uso.**
- Diagrama de estado.
- Diagrama de comunicación.
- Diagrama de interacción.
- Diagrama de sincronización.

Quiz

{desafío}
latam_

/* Diagrama de clases */

Uso de la herramienta StarUml



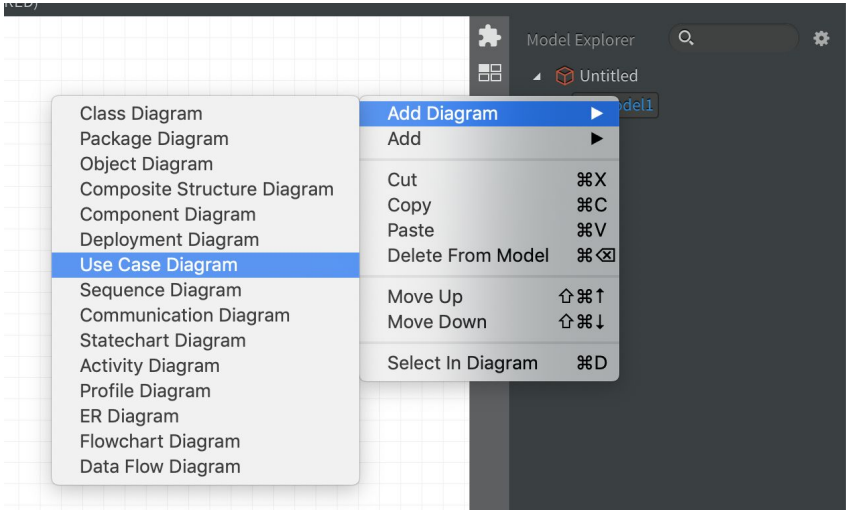
Release Notes

- http://staruml.io/download

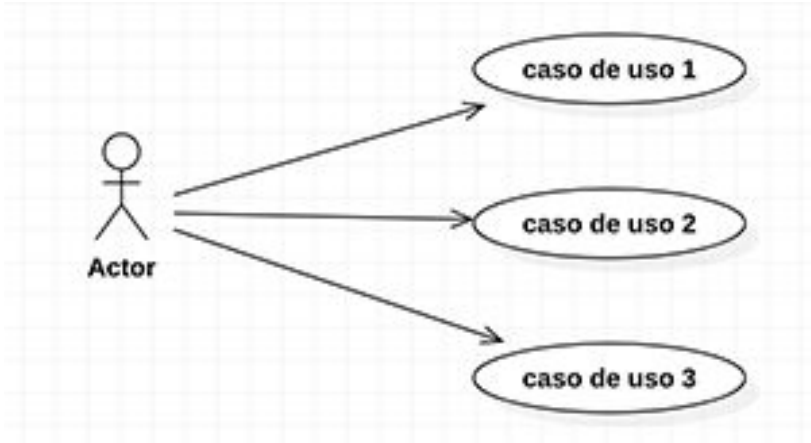
{desafío} latam_

Diagramas de casos de uso (Cuentan una historia)

- Model -> add diagram -> Use Case Diagram.



- Una representación de un **diagrama** de casos de uso sería el siguiente



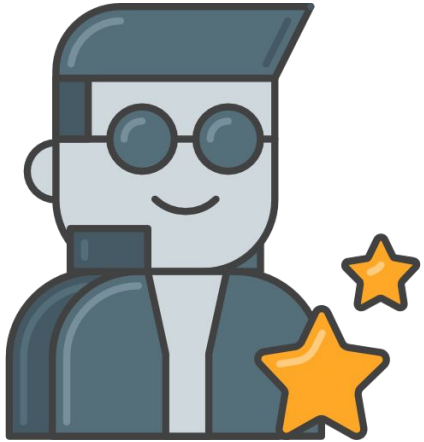
Casos de uso (CU)

- Interacción típica entre un usuario y un sistema de software.
- Capta alguna función visible para el usuario.
- Puede ser pequeño o grande.
- Logra un objetivo discreto para el usuario.

- Usualmente, el caso de uso se extrae de las **interacciones**, que los potenciales usuarios del sistema tengan con la aplicación que se desee construir.
- Cada una de estas, se debe abordar de forma discreta, darle un nombre y escribir una breve descripción.
- **No hay que detallar tan profundamente esta interacción,** todo esto dependiendo de la cantidad de ramificaciones, de las que esté compuesto el caso de uso, se podrá más adelante, obtener mayores detalles que **pueden resultar en nuevos casos de uso.**
- **Objetivos del usuario versus, interacciones con el sistema.**

Actores

- Empleamos el término actor para llamar así al usuario, cuando desempeña ese papel con respecto al sistema.
- No es necesario que los actores sean seres humanos



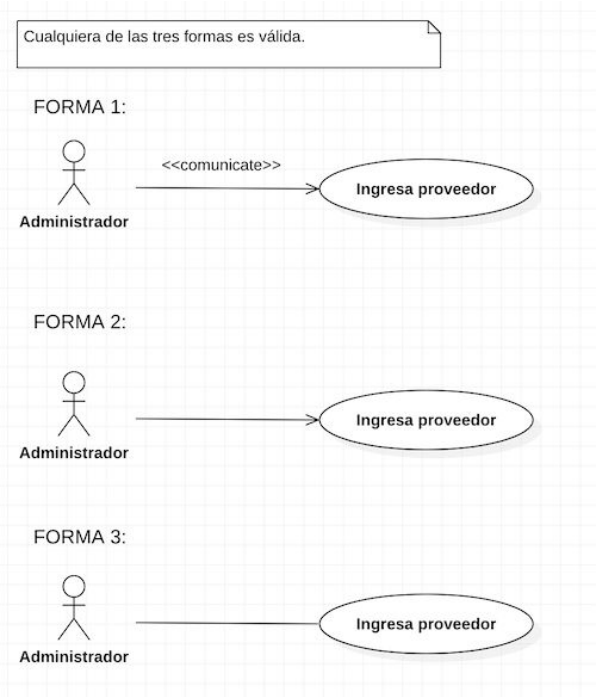
Nombre del actor

Relaciones

- UML, define relaciones de estereotipos y generalización. Con estas relaciones, podemos ver gráficamente el cómo interactúan los casos de uso y los actores, para una mejor comprensión del escenario.

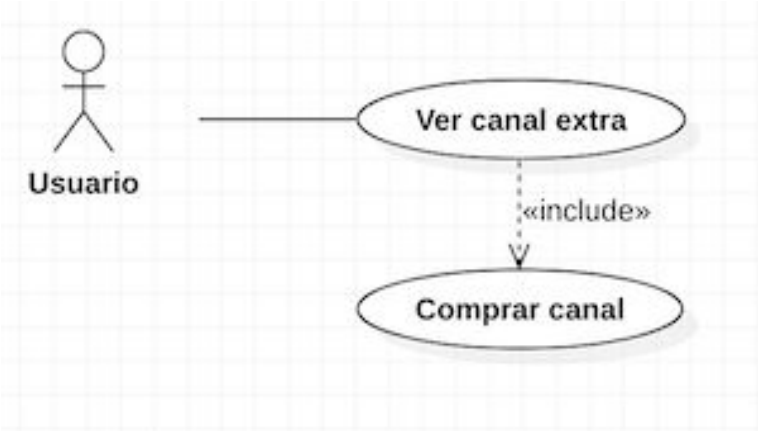
Estereotipo: <<comunicate>>

- Esta relación es la que más veremos en los CU, como estereotipo se representa por <<communicate>>; pero generalmente este estereotipo no va escrito. Es una relación de asociación que nos muestra la interacción entre un actor y el caso de uso.



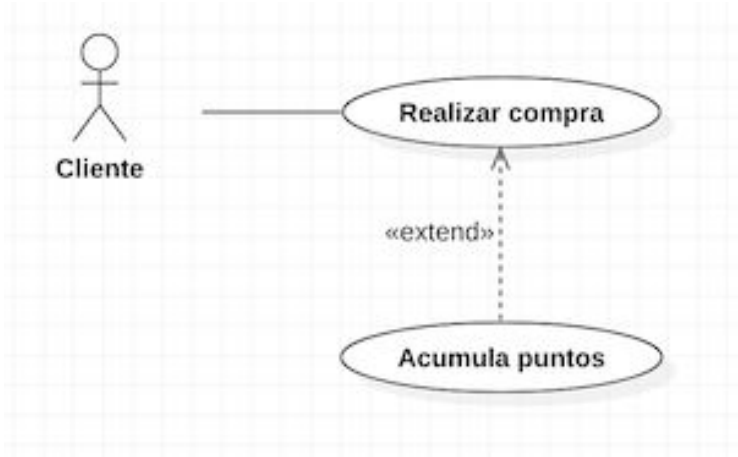
Estereotipo: <<include>>

- En términos muy simples, cuando relacionamos dos casos de uso con un “include”, estamos diciendo que el primero (el caso de uso base) incluye al segundo (el caso de uso incluido). Es decir, el segundo es parte esencial del primero. Sin el segundo, el primero no podría funcionar bien; pues no podría cumplir su objetivo.



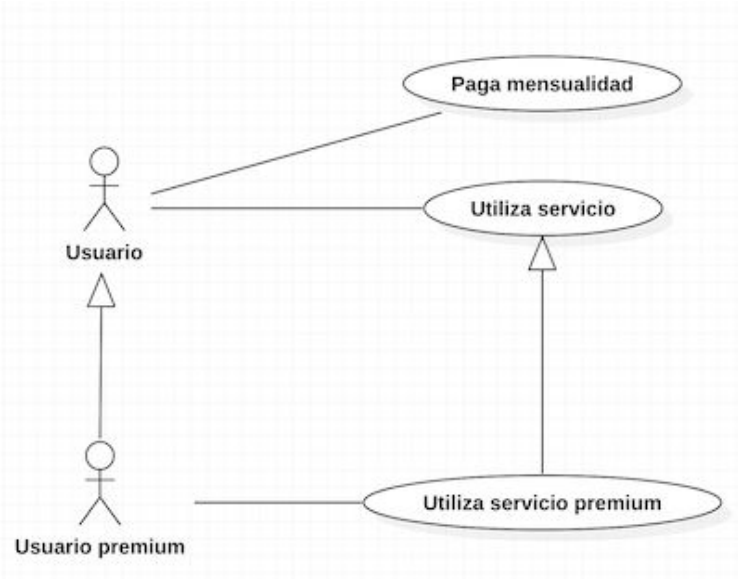
Estereotipo: <<extend>>

- Un caso de uso puede tener una extensión **que no sea indispensable**, pero sería bueno para la comprensión de lo que se desea desarrollar, que esta extensión sea expresada en el diagrama, es en este caso (no abusar), en donde haremos esta relación.

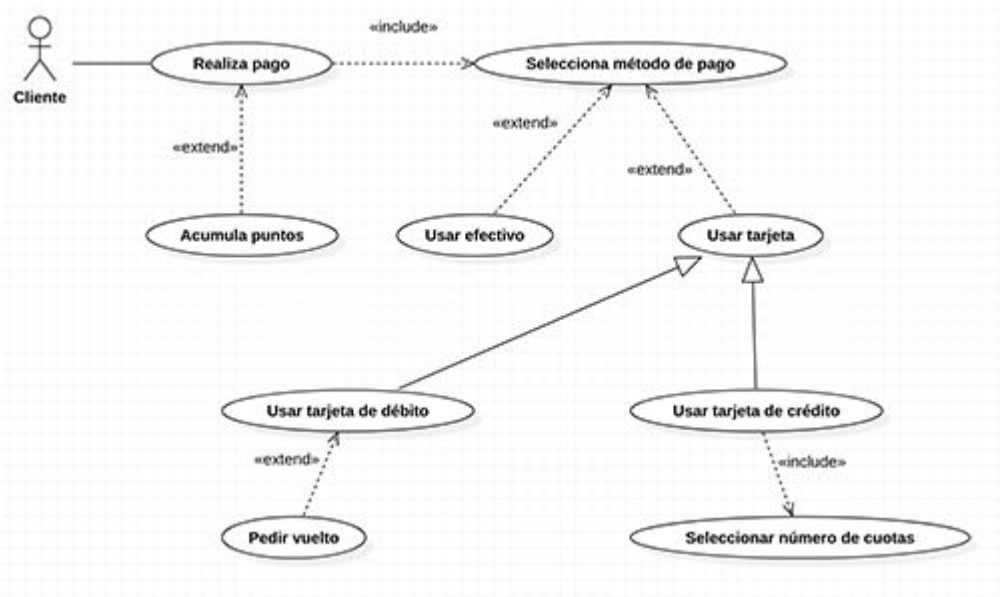


Generalización

- Cuando hablamos de generalización, esta vez **sí que nos estamos refiriendo, a algo muy parecido de lo que hace la herencia** en la orientación a objetos, pero esta vez en el contexto de comprender el escenario.

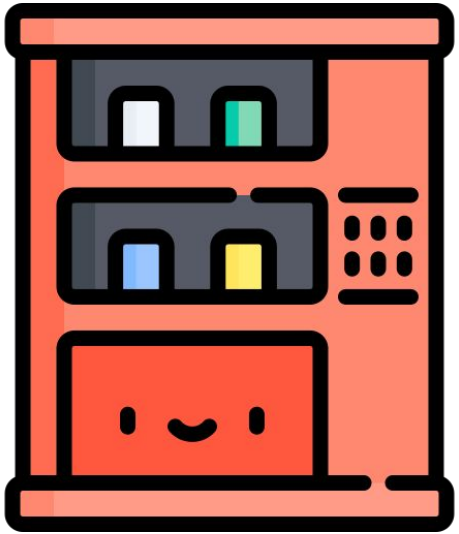


Evitar abusos en la granularidad



Ejemplos 1

- Diseñar un diagrama de casos de uso, que exprese el escenario de una máquina expendedora de bebidas.



Ejemplos 2

- Diseñar un diagrama de casos de uso, que exprese el escenario que responde a los siguientes requerimientos:

a. Un juego de teléfono móvil donde participan dos jugadores cada uno con su propia terminal.
b. Cuando dos jugadores desean jugar, uno de ellos crea una nueva partida y el otro se conecta.
c. El objetivo del juego es manejar una nave y disparar al contrario. Si uno de los dos jugadores acierta, la partida termina.
d. Si uno de los dos jugadores deja la partida (o se pierde la conexión) la partida termina.

Ejemplos 3

- Diseñar un diagrama de casos de uso, que exprese el escenario que responde a un sistema de ventas de entradas online.

Quiz

{desafío}
latam_

Cierre

15 minutos

{desafío} latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales