Técnico en

DESARROLLO DE SOFTWARE

Introducción a UML

Introducción a UML

Unidad II

1. Ejemplo de Análisis

Durante esta unidad usaremos el siguiente ejemplo para demostrar las diferentes características de los diagramas.

Un banco le ha dado el proyecto de crear un sistema de cajeros, que están conectados al sistema de red del banco, este sistema está diseñado para mantener la información de las cuentas, así como las transacciones de las cuentas, y las conexiones hacia él.

El sistema de cajeros que usted debe diseñar deberá trabajar únicamente cuando reciba una tarjeta de crédito, una vez que reciba una, debe pedir el PIN o la contraseña, y si es correcta desplegará un menú al usuario, mostrando todos los tipos de transacciones que se pueden realizar: Consultar el Saldo, Realizar Transferencias, Retirar Efectivo, entre otros.

El cajero tiene la capacidad de comunicarse con los servidores del banco, y cuenta con varias funcionalidades.

Para propósitos de seguridad el cajero se comunica con el servidor del banco en muchas partes de las operaciones, por ejemplo cuando se intenta utilizar una tarjeta se debe de verificar que la misma cuenta no se esté usando en otro lugar, esto podría significar que

la tarjeta fue clonada, entonces se deben de bloquear todas las sesiones y deshabilitar el uso de la cuenta hasta que se resuelva el problema.

Los cajeros, luego de cada transacción, entregarán un recibo al usuario.

2. ¿Qué son los Casos de Uso?

Los diagramas de casos de uso dan un resumen de los requisitos de uso de un sistema. Son útiles para las presentaciones a los administradores y / o interesados en el proyecto, pero para el desarrollo real el uso de los casos de uso ofrecen mucho más valor porque describen "la esencia" de las necesidades reales.

Cuando estamos diseñando un sistema, muchas veces lo primero que pensamos es cómo se va a utilizar el nuevo sistema, quien va a ser el que lo utilice, y con qué otros sistemas vamos a interactuar.

Un caso de uso es una herramienta que nos muestra la forma en la que un usuario interactúa con el sistema, entonces un Diagrama de Casos de Uso es una colección de usuarios y casos de uso, y con el diagrama podemos tener una vista general del sistema, cómo funciona y cómo se interactúa con él.

Importancia

Los casos de uso nos proporcionan una herramienta para encontrar los requerimientos de usuario, así como la forma en la que se va a comportar el sistema.

Es importante tener en consideración cómo se va a utilizar el sistema, una cosa es que funcione bien, y otra completamente diferente es que funcione como se necesita. La forma en la que diseñamos un sistema puede no ser la correcta, por eso es importante

involucrar al usuario en el proceso de desarrollo, así podremos tener una idea clara de cómo debe funcionar un sistema.

Introducción a los Casos de Uso

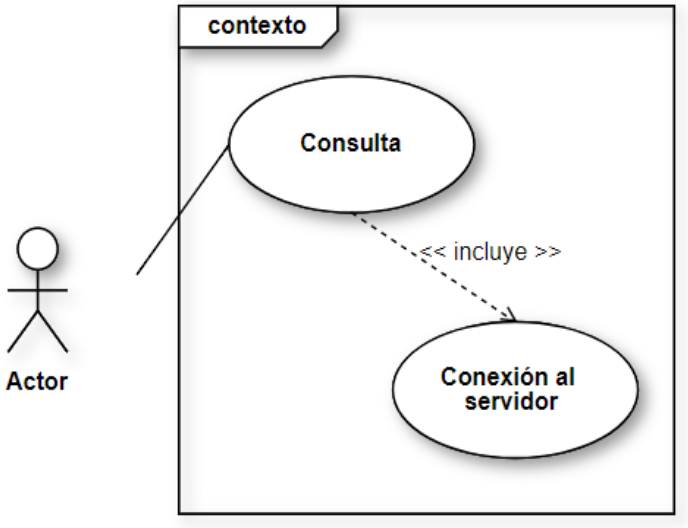
Un caso de uso nos muestra una interacción con una entidad externa al sistema, puede ser un usuario, u otro sistema.

Un caso de uso también puede estar relacionado a otros casos de uso por medio de la reutilización, podemos usar un caso de uso como parte de otro caso de uso, o bien podemos tomar un caso de uso y extender su funcionamiento, agregando pasos y más funciones. A estos tipos de reutilización se les conoce como Inclusión y Extensión respectivamente.

Inclusión

Es cuando usamos los pasos de un caso de uso en otro, un caso de uso utiliza otro caso de uso como parte de su funcionalidad, esto puede ser muy útil a la hora de diseñar para reutilizar código y/o componentes del sistema

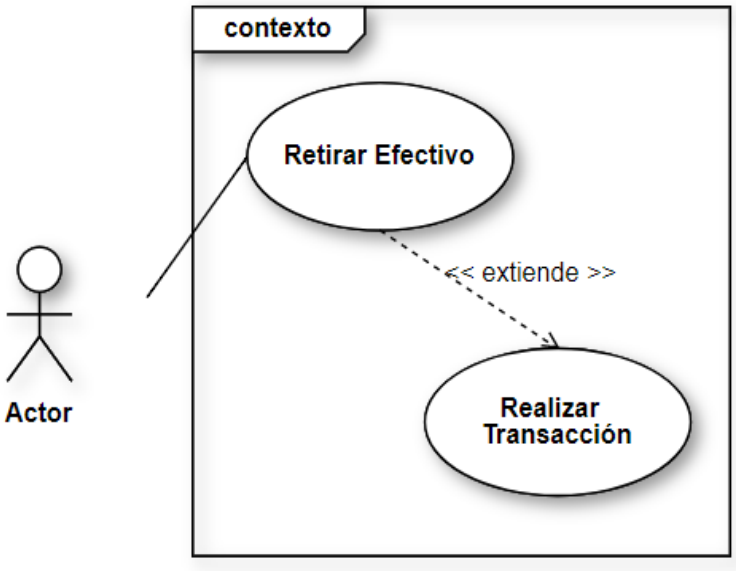
La relación inclusión se indica como una línea discontinua con el estereotipo <<incluye>>



Extensión

Es cuando usamos un caso de uso como base y expandimos su funcionalidad para crear casos de uso más complejos. Una relación extender indica opciones alternativas a un caso de uso.

Un caso de uso solo se puede extender en puntos específicos, llamados puntos de extensión

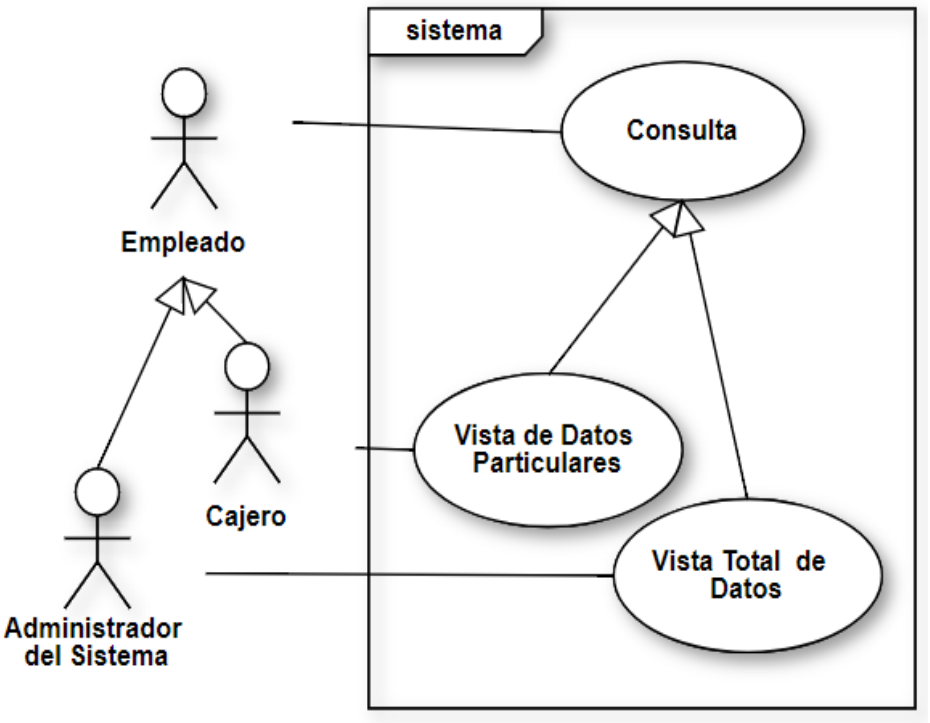


Generalización

La generalización, es cuando un caso de uso hereda el comportamiento y significado de otro, y además agrega su propia funcionalidad. La generalización también se conoce como herencia o especialización. Es el mismo concepto que se vio con los diagramas de clases.

Dicho de otra manera, el caso “hijo” es una especialización del caso “padre”.

Esta relación también puede ser aplicada a actores.



En este ejemplo, vemos especialización de los actores, los empleados se especializan dependiendo de sus funciones, asimismo los casos de uso que cada actor utiliza se especializa de un caso de uso general.

3. Aplicación de los modelos de caso de uso

Al principio de un proyecto, se hacen una serie de entrevistas con los clientes, las personas que nos dan el proyecto, de estas entrevistas obtenemos los requerimientos del sistema, que son la base del análisis, y serán nuestra guía durante el proyecto.

Como siguiente paso, se hacen entrevistas con los usuarios que usaran el sistema, quienes nos dirán como se utilizara el sistema, esta nueva entrevista nos dará lo que se conoce como los requerimientos de usuario, y estos nos revelan los actores y los casos de uso.

Un caso de uso es la representación de un requerimiento de usuario.

Analicemos un caso del ejemplo, tomemos el caso de que un Usuario hará un retiro de efectivo

Dominio del Sistema

En primer lugar siempre se debería realizar un análisis del dominio, hablar con el cliente, entender el sistema, realizar el análisis de clases, ver las relaciones, etc.

En este caso podemos ver tres clases:

- El Usuario: La persona que usará el cajero, tiene una cuenta, una tarjeta, conoce su contraseña y puede hacer uso del cajero.
- El Cajero, en este caso el cajero se puede considerar como una interfaz, lo podemos modelar, pero no será de utilidad en el caso de uso, ya que sirve para comunicar a el usuario con el servidor, sin embargo también es importante tenerlo

en cuenta, ya que a la hora de verificar las interfaces y las conexiones físicas, tenemos que tener en cuenta el cajero.

- El Servidor del Banco: Es el servidor que maneja las transacciones, las verificaciones, etc. Nos comunicaremos con el servidor, así que habrá que considerar una interfaz, en estos casos siempre tenemos que comunicarnos con el cliente y asegurarnos de trabajar bien con la interfaz.

Requerimientos de Usuario

Una vez que comprendemos el dominio del sistema, lo que tenemos que hacer es hablar con las personas que van a usar el sistema.

Los clientes nos pueden dar una buena idea de cómo va a ser el sistema, pero algunas veces los clientes no van a usar el sistema, las personas que usaran la interfaz nos puede decir mucho de cómo debe funcionar, qué esperan de este sistema y que cosas les gustaría que el sistema tenga.

Para el caso de un retiro, supongamos que los usuarios nos dijeron que se espera que la terminal cuente con seguridad, que sea una interfaz simple de usar, y que sea rápida.

Entonces obtenemos los actores

- Usuario: La persona que inicia la acción, maneja las opciones y recibe el dinero, los recibos, etc.
- Servidor: El servidor que debe de estar encargado de las transacciones, manejar la seguridad, etc. en este caso nosotros no nos encargamos de esto, pero si somos responsables de manejar las peticiones y las respuestas.

Con estos actores, podemos ver que se tendrán algunos casos de uso: retiro de efectivo, transferencias, consultas, etc.

Desarrollar un caso de Uso

Tenemos todas las bases para desarrollar el caso de uso,

Entonces, el primer paso para realizar un caso de uso es pensar cómo se ejecutará, debemos planear el comportamiento, y así tendremos una clara idea del proceso de desarrollo que involucra desarrollar este caso. En este punto es también donde podemos ver si podemos hacer uso de las relaciones entre casos de uso y/o actores.

Veamos qué pasos involucraría el caso de uso de un retiro

1. El cajero recibe una tarjeta, y el usuario verifica su identidad
2. El usuario navega el menú y selecciona la transacción.
3. Confirma la transacción y el mensaje se le envía al servidor
4. El servidor contesta, y aprueba la transacción
5. El usuario recibe el dinero y un recibo

A grandes rasgos, esto es lo que sucede en este caso de uso, cuando examinamos los casos de uso, lo normal es que el análisis sea mucho más detallado, ya que nos ayuda a ver los detalles, las posibles conexiones, las dependencias, etc.

Pero consideremos, en este problema, hay más transacciones, y estas también tendrán pasos similares, este es un buen punto para considerar las relaciones y agregar algunas para simplificar el desarrollo de los casos individuales

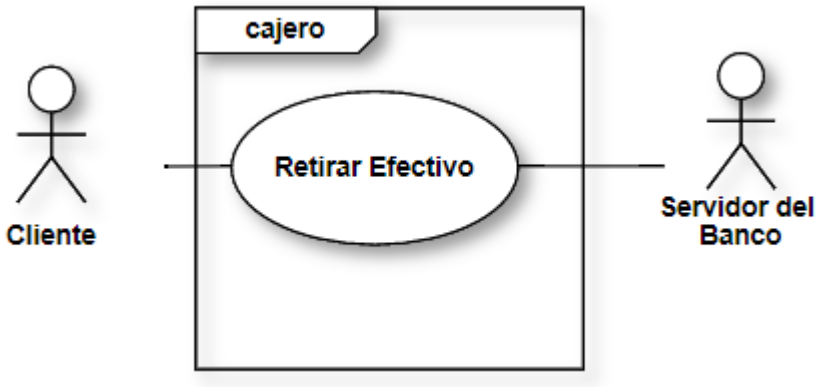
Pensemos, existen varias transacciones, y todas empiezan por interactuar con el usuario, quien ingresa su tarjeta y verificar sus credenciales.

Entonces podemos decir que las transacciones incluyen el caso de verificar datos.

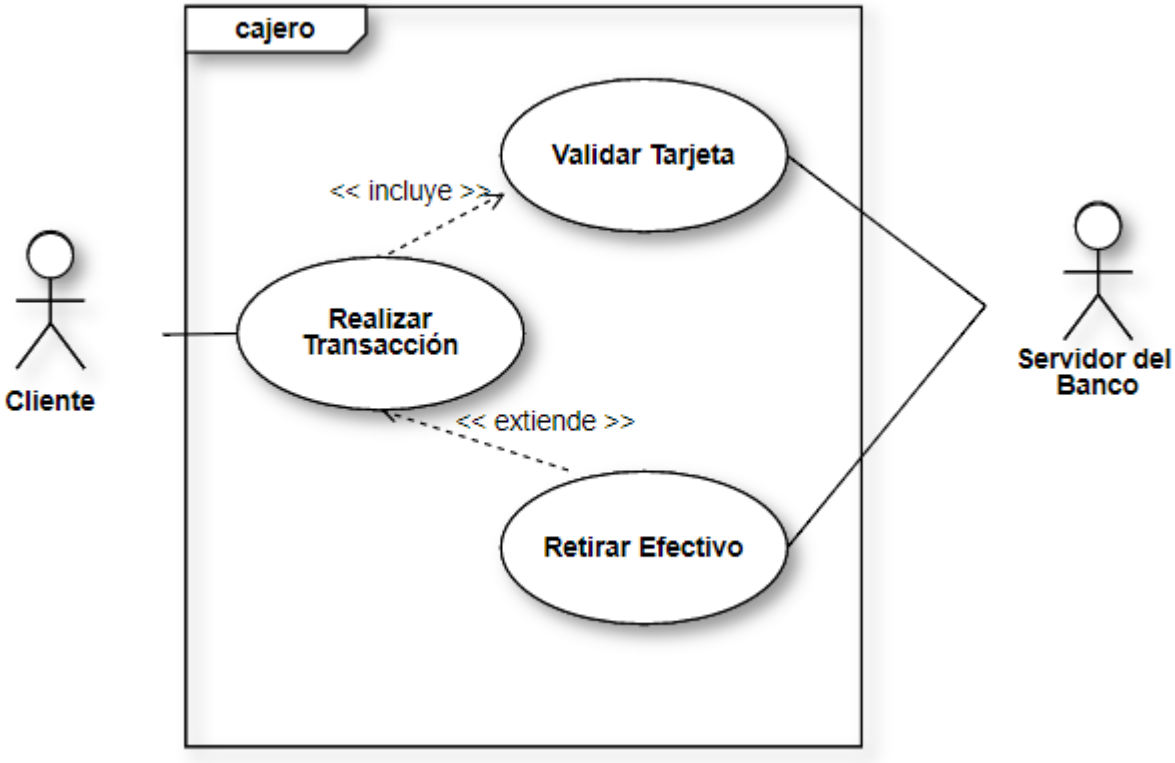
Ahora, sabemos que las transacciones se trabajan en el servidor, y que todas las transacciones inician de la misma manera, se selecciona del menú y se abre una conexión, el resto de la transacción depende del mensaje y de lo que se deba hacer.

En este caso nos conviene hacer un caso de uso general donde las demás transacciones extiendan su funcionalidad.

Entonces, pasamos de



A



Que como podemos ver, es una vista mucho más significativa del sistema

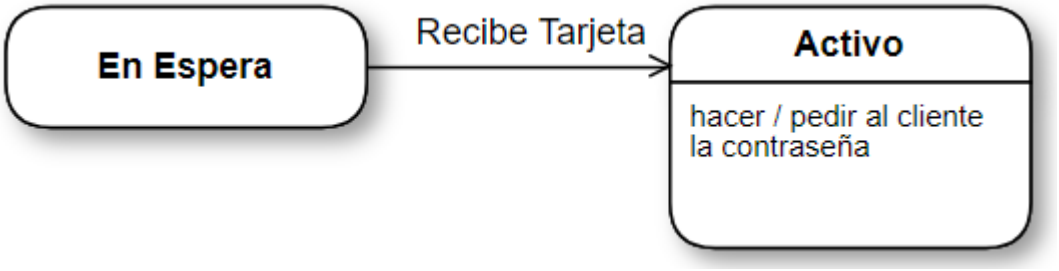
4. ¿Qué son los Diagramas de Estados?

Mientras un sistema interactúa con los actores, este cambia para ajustarse a las interacciones, y así poder responder a las necesidades.

El diagrama de estados es la herramienta para modelar estos cambios.

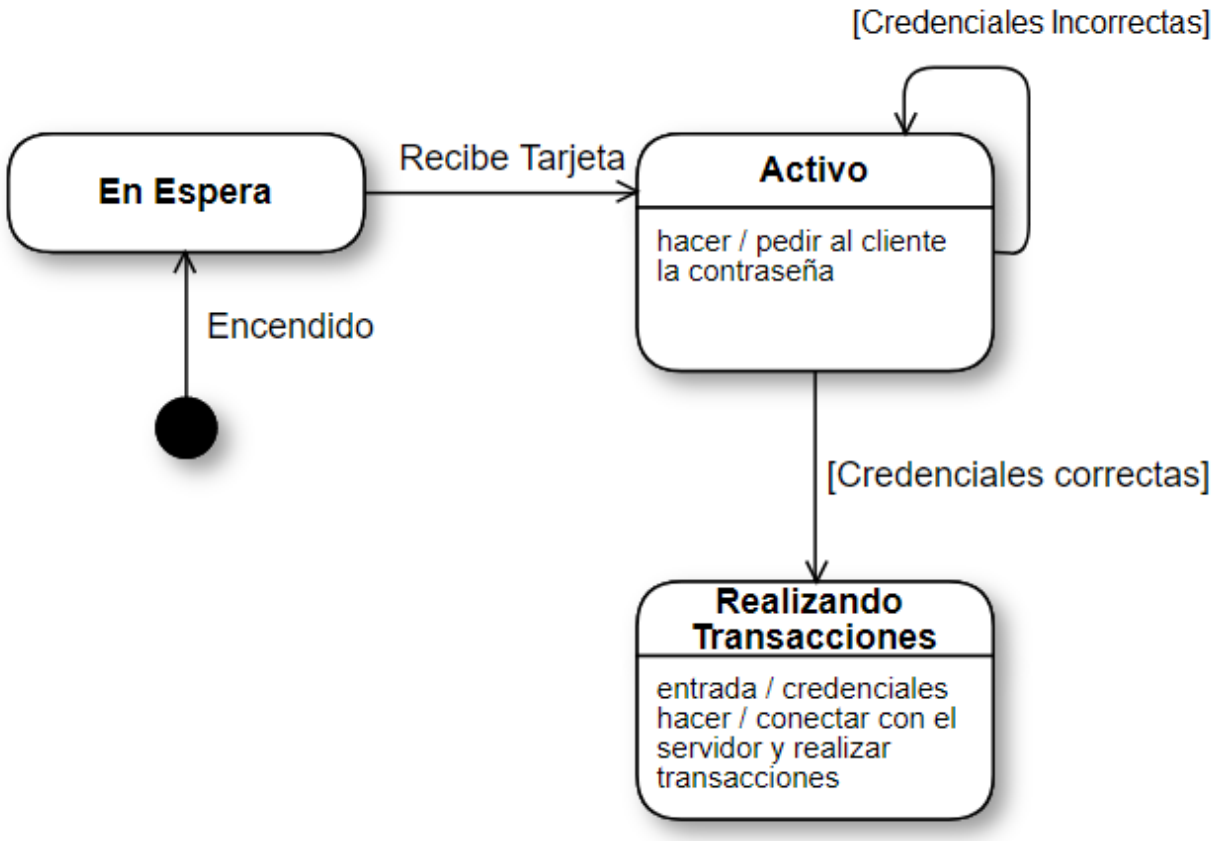
En el ejemplo de análisis de la unidad: el cajero se encuentra en un estado de espera hasta que llega un cliente e ingresa su tarjeta, entonces el cajero pasa a un estado activo donde interactúa con el cliente.

Esto podríamos representarlo como



Condiciones

Muchas veces las transiciones dependen de alguna condición, para representar esto se escribe la transición entre corchetes []



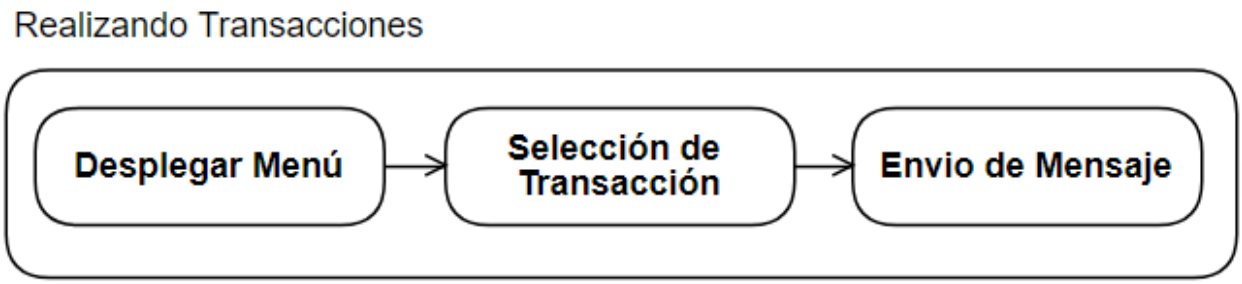
Veamos aquí el cajero solo debería permitir una transacción si las credenciales son correctas, de lo contrario debería volver a pedir la contraseña

Subestados

A veces un estado atraviesa distintos estados mientras ejecuta su funcionalidad, estos cambios de estados que ocurren dentro de otros estados, se conocen como subestados.

Subestados Secuenciales

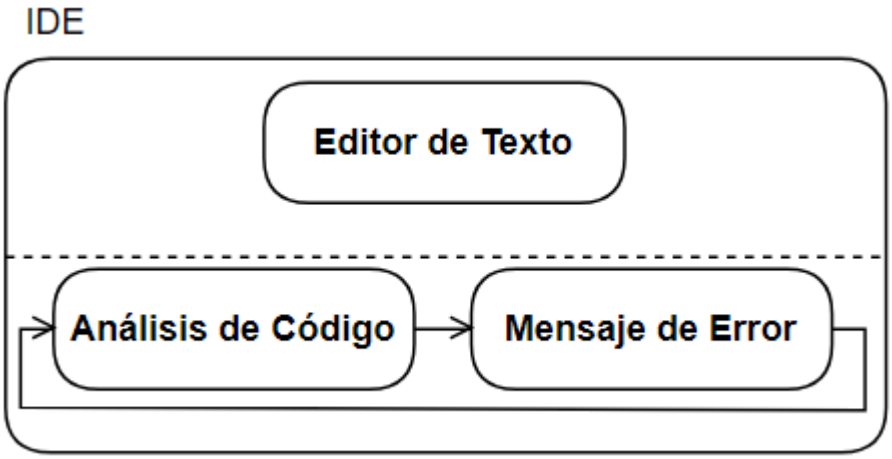
Veamos nuestro ejemplo, el estado ***Realizando Transacciones*** es cuando se ve el menú, el usuario selecciona su transacción y la transacción se envía al servidor



Subestados Concurrentes

Veamos un caso algo diferente, usted está desarrollando un proyecto, para eso hace uso de un IDE, con muchas funcionalidades, una de ellas es la de análisis de código, lo que permite que el IDE análice el código mientras usted lo escribe, y si hay algún error semántico, o algún error en los parámetros el IDE se lo hace saber por medio de un mensaje de error.

El análisis de código se realiza mientras usted escribe el programa.



Esto se conoce como un estado concurrente, cuando varios estados se ejecutan al mismo tiempo

Estados Historicos

En ocasiones el estado de un sistema se interrumpe, trabajemos el caso cuando una computadora entra en estado de suspensión, la computadora no se ha apagado, en cambio se interrumpió su estado de ejecución. Sin embargo la computadora al reiniciarse no inicia como si se encendiera, se reinicia justo como la dejamos,

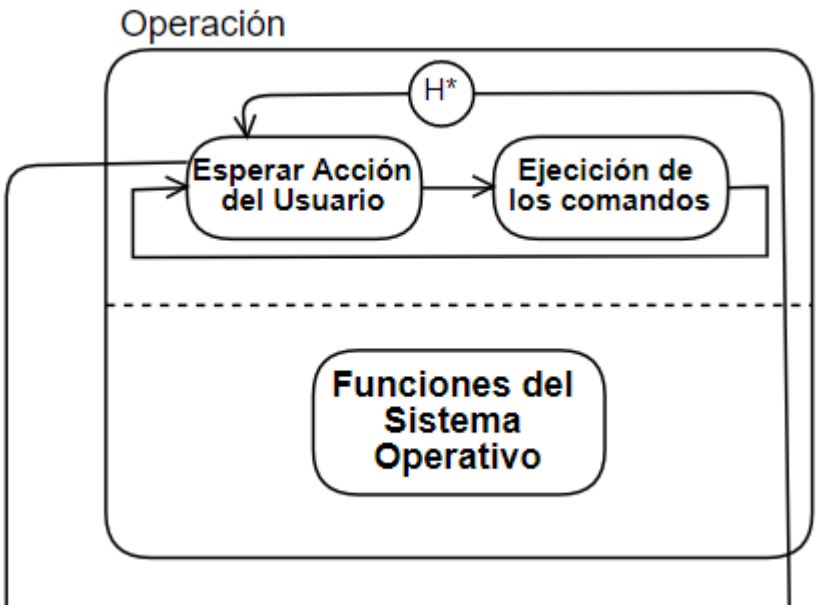
Esto es lo que conocemos como *Estados Históricos*, son estados que recuerdan el punto donde se quedaron cuando el estado trasciende fuera del estado compuesto.

Existen dos tipos de *Estados Históricos,* los superficiales, y los profundos.

Para explicar la diferencia hay que entender que los subestados que componen un estado, pueden estar al mismo tiempo compuestos de más subestados, en el ejemplo anterior había 2 niveles de estados, pero puede llegarse más profundo.

Ahora, un historial superficial recuerda el estado al que apunta, el estado de los subestados que se pudieran encontrar en su interior no son recordados, causando que el estado se reinicie, también se puede ver como una entrada al subestado.

Un historial profundo si recuerda el estado de todos los niveles de los subestados, restaurando el estado a la situación exacta en la que se encontraba al ser interrumpido.



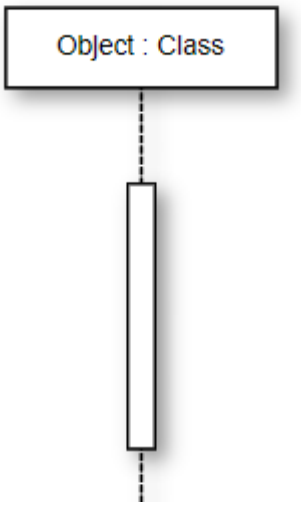
Este es una vista muy simplificada del estado de operación de un computador, cuando el computador, cambia de estado, por el salvapantalla, o por que entro en estado de suspensión para ahorrar batería, cuando vuelva a funcionar, volverá al punto exacto en el que se quedó, por lo tanto es un estado histórico *profundo.*

5. ¿Qué son los Diagramas de Secuencia?

Un diagrama de secuencia es una representación de las interacciones a través del tiempo de los objetos. Con un diagrama de secuencia podemos ver el comportamiento de un caso de uso a través del tiempo, desde que inicia, como se va pasando el control, el tipo de mensaje, etc.

Objetos

Se colocan en la parte superior de un diagrama de secuencia.



Línea de Vida

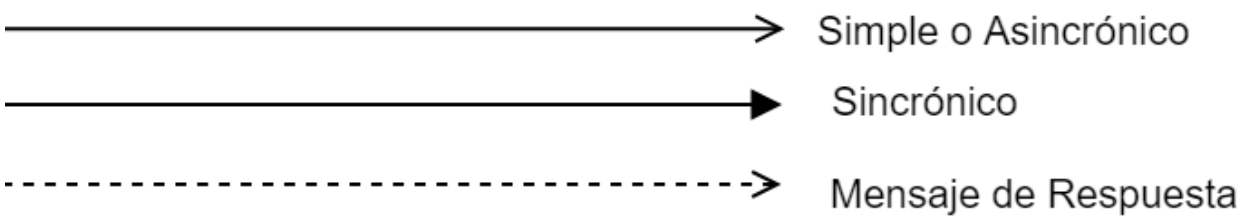
Es la línea punteada que se coloca debajo de un objeto, representa el tiempo transcurrido en un diagrama de secuencias

Tiempo de Activación

Es un rectángulo que se coloca sobre la línea de vida de un objeto, representa el tiempo de ejecución de una acción determinada.

Mensajes

Flechas que van de un objeto a otro, dependiendo del tipo de flecha depende el tipo de mensaje, aquí veremos el extremo de la flecha, pero hay que recordar que las flechas pueden significar otras cosas en otros diagramas, además de que si una flecha es solida o punteada, también puede cambiar el significado.



- Sincrónico: El objeto queda a la espera de la respuesta a un mensaje antes de continuar con su trabajo.

- Asincrónico: Un objeto no espera una respuesta al mensaje para continuar.
- Mensaje de Respuesta: Cuando queremos mostrar que un mensaje esta contestando a una petición, o bien es la respuesta a una solicitud, se usara una flecha punteada, el extremo puede también ser sincrónico o asincrónico

Diagramas de Secuencias de Instancia y Genéricos

Muchas veces utilizaremos los diagramas de secuencias para representar los casos de uso, y los casos de uso suelen incluir comprobaciones, caminos alternos y opciones, estos casos los podemos representar de dos formas con un diagrama de secuencia.

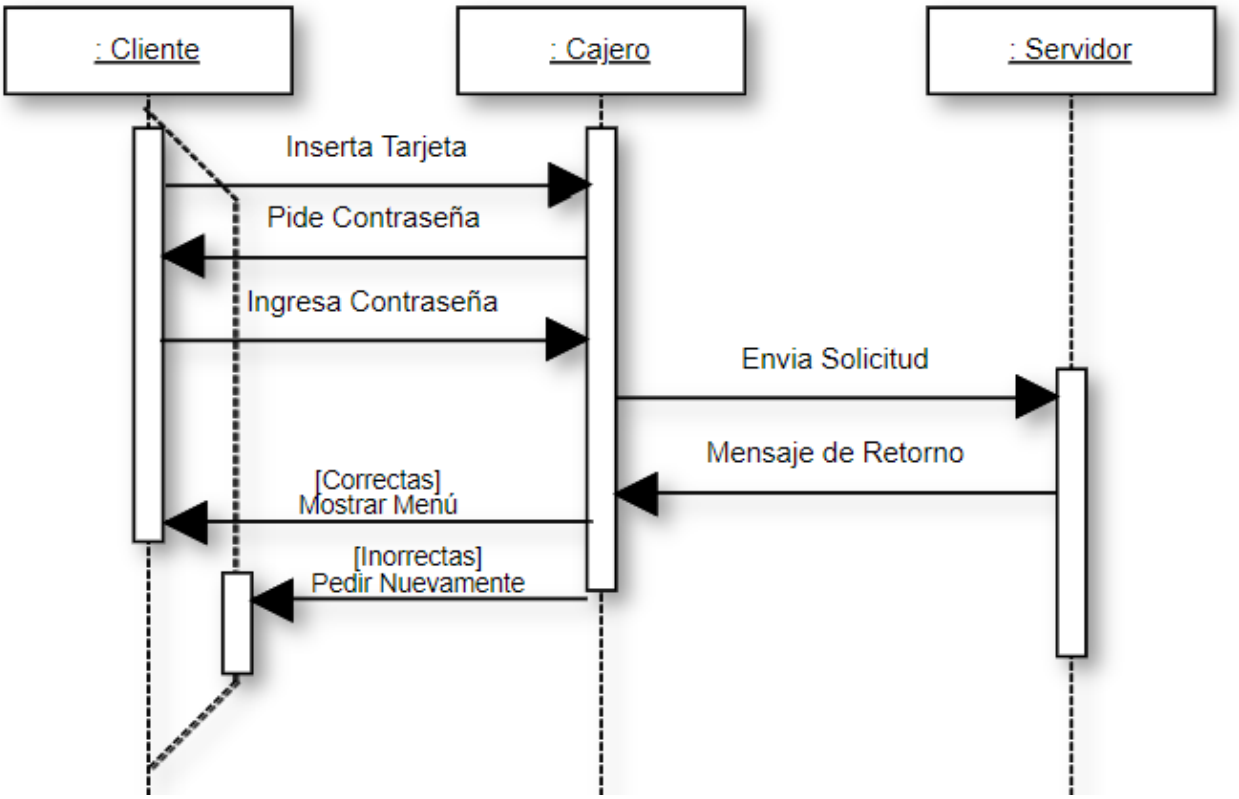
Diagrama de Secuencia de Instancia

Este diagrama representa una opción o un escenario, regresemos al ejemplo dado al principio de la unidad, el cajero no dejará que el usuario haga ninguna transacción si sus credenciales no son correctas, aquí podemos ver que existen dos escenarios, las credenciales son válidas, entonces se puede hacer una transacción, y las credenciales no son válidas, y se solicitan nuevamente.

Si usáramos diagramas de secuencia de instancia tendríamos que hacer dos diagramas, uno representando el caso donde las credenciales son aceptadas, y otro donde no lo son

Diagrama de Secuencia Genérico

Un diagrama de secuencia genérico es un diagrama donde representamos todos los posibles escenarios de un caso de uso, esto lo hacemos por medio de líneas de vida que se dividen de una principal, y cada condición que causará una división colocamos la causa entre corchetes

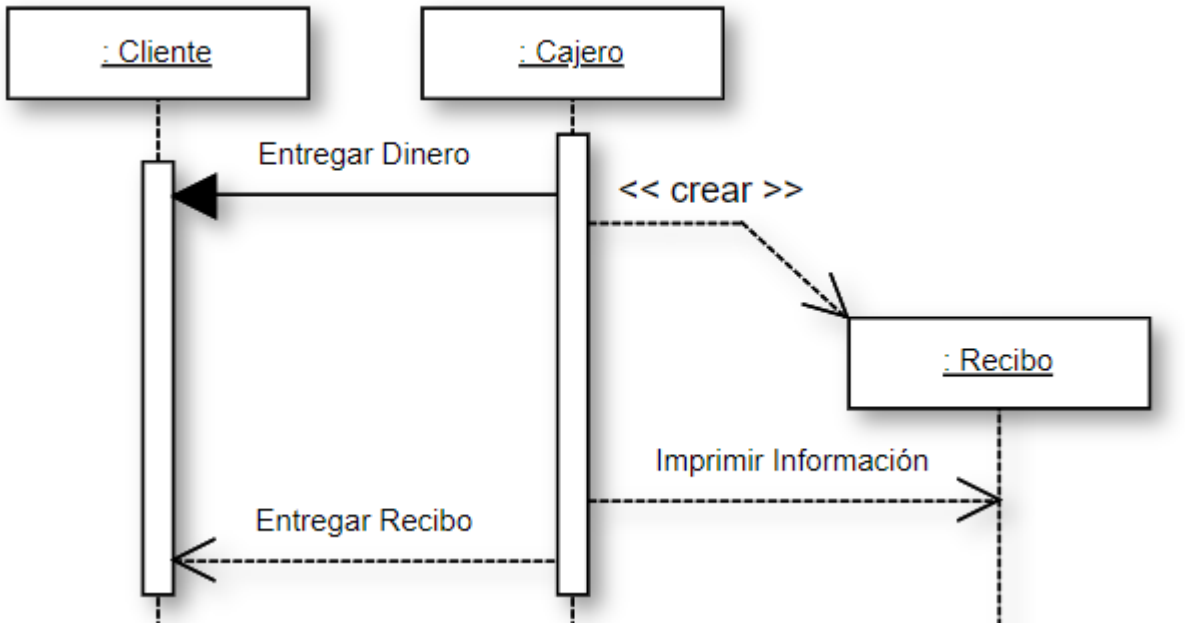


Vemos aquí el proceso de validación de la tarjeta del usuario.

Creación de un objeto

Muchas veces, durante una secuencia, se crearán objetos en nuestros casos de uso, un documento, una propuesta, un servidor, etc.

Por ejemplo, al final de un retiro, nuestro ejemplo entregará un recibo al cliente



Podemos ver aquí cómo se crea el recibo, y ahora también podemos interactuar con él.

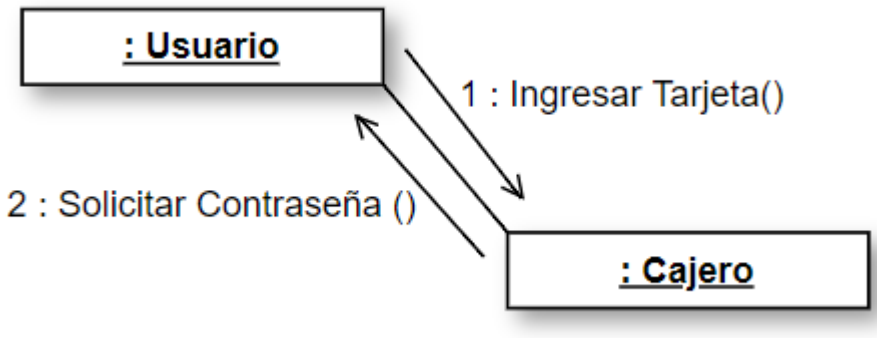
6. ¿Qué son los Diagramas de Comunicación?

El diagrama de comunicación, anteriormente llamado diagrama de colaboración, describe las interacciones entre los objetos en términos de mensajes secuenciados. Los diagramas de comunicación representan una combinación de información tomada de los diagramas de clases, Objetos y de casos de uso, describiendo el comportamiento, tanto de la estructura estática, como de la estructura dinámica de un sistema.

El diagrama de comunicación es semánticamente equivalente al diagrama de secuencias, esto quiere decir que la misma información está presente en ambos diagramas y se podría inferir uno del otro. La diferencia entre estos diagramas es que el diagrama de secuencias muestra la sucesión de las interacciones de los objetos y el diagrama de comunicación representa el contexto y la organización general de los objetos.

Un diagrama de comunicación es una extensión de uno de los objetos, que muestra las relaciones entre objetos y agrega los mensajes que los objetos intercambian entre sí.

Un mensaje se representa con una flecha cerca de la línea de asociación entre dos objetos, una etiqueta cerca de la flecha muestra el tipo de mensaje, un mensaje por lo general indica al receptor que ejecute una de sus operaciones, el mensaje se finaliza con un par de paréntesis dentro de los cuales se colocan los parámetros necesarios para la operación, se agrega una cifra al inicio del mensaje para representar la secuencia de los mensajes, la cifra y el mensaje se separan por dos puntos (:).

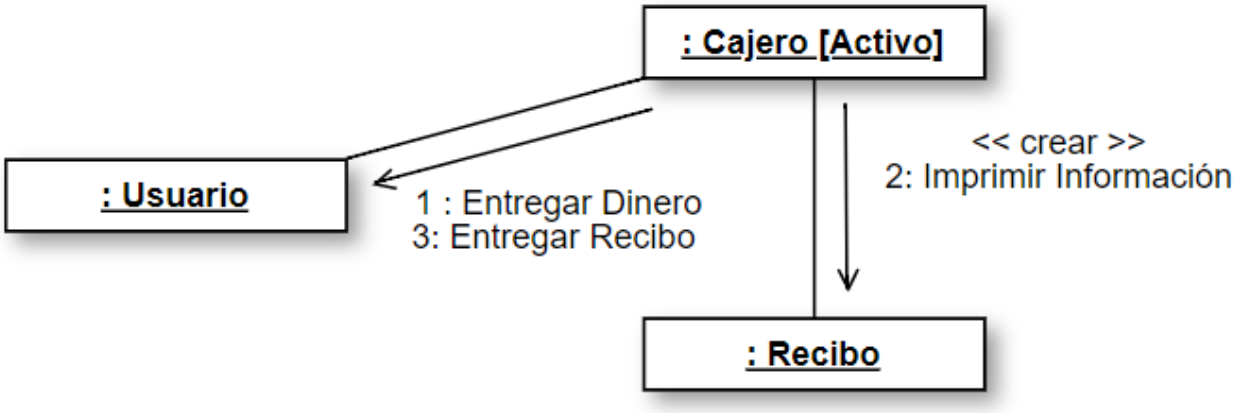


Aquí vemos la primera interacción entre un usuario y un cajero

Creación de Objetos

Para representar la creación de un objeto en el diagrama de comunicación, se agrega a la flecha del mensaje el estereotipo “<<Crear>>”

Recordemos el diagrama de secuencias, donde mostramos la parte final de un retiro de efectivo

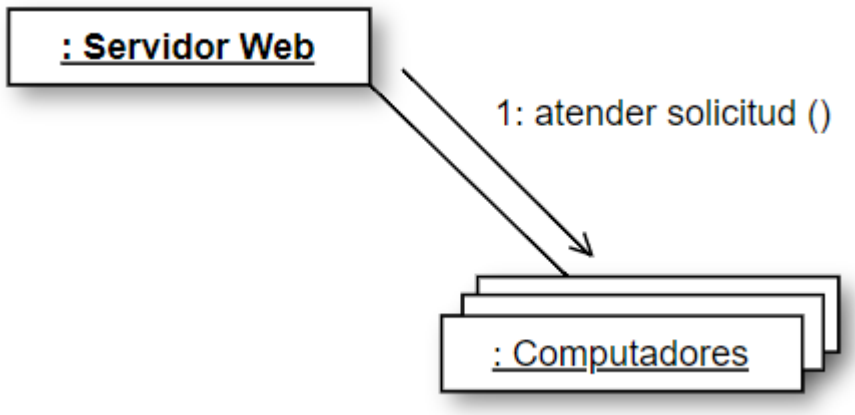


Así se vería como un diagrama de comunicación

Varios Receptores

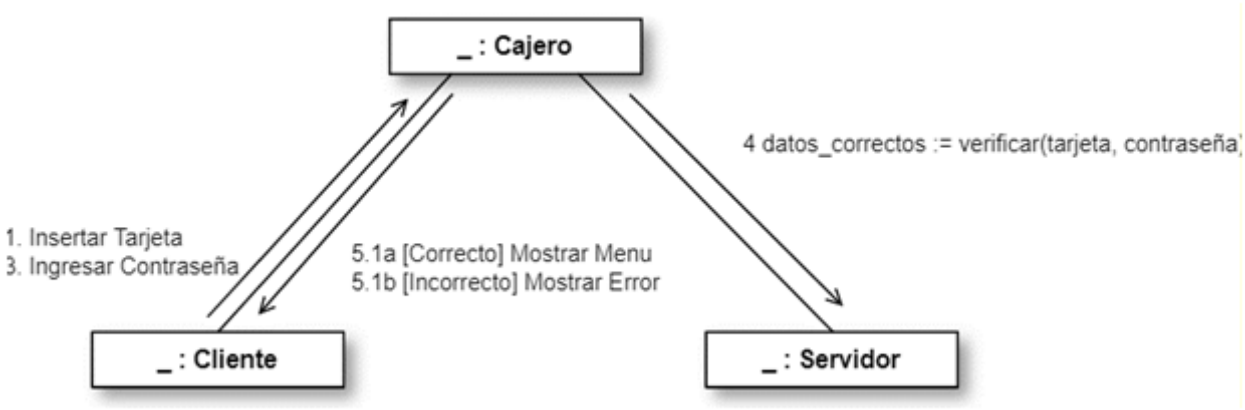
A veces un objeto puede enviar un mensaje a varios receptores, para indicar eso se utiliza una pila de objetos.

Por ejemplo pensemos en un servidor web, que contesta la misma solicitud a muchos computadores



Condiciones

Dentro de los mensajes que intercambia los objetos se pueden representar condiciones “si” o “If” y ciclos “mientras” o “while”. Estas condiciones se representan con un par de corchetes [ ] que encierran la condición deseada, si una condición debe aplicar a objetos múltiples se antepone un asterisco “ * ” a la condición.

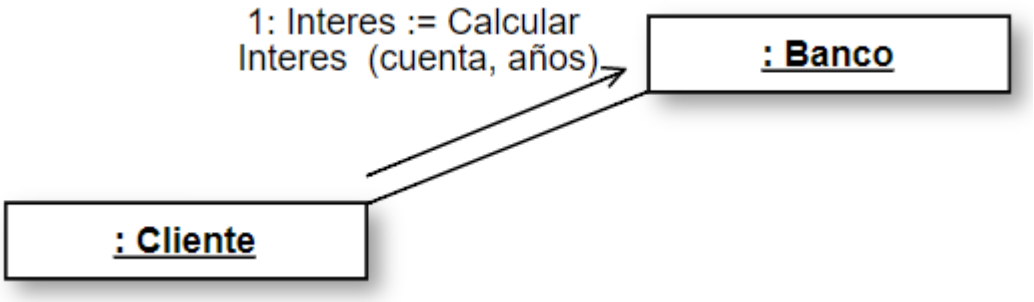


Veamos el caso del ingreso de la contraseña

Representación de resultados

Un objeto puede enviar un mensaje que solicite al receptor a realizar un cálculo y devuelve el resultado, El mensaje se escribiría entonces “NombreValor:=Operacion(parametros)”

Consideremos este caso: un cliente verifica el estado de su cuenta de ahorros, y quiere saber cuánto interés acumulará en cierta cantidad de años, afortunadamente la página del banco le permite hacer esto, ingresando el número de cuenta y el tiempo que quiere calcular

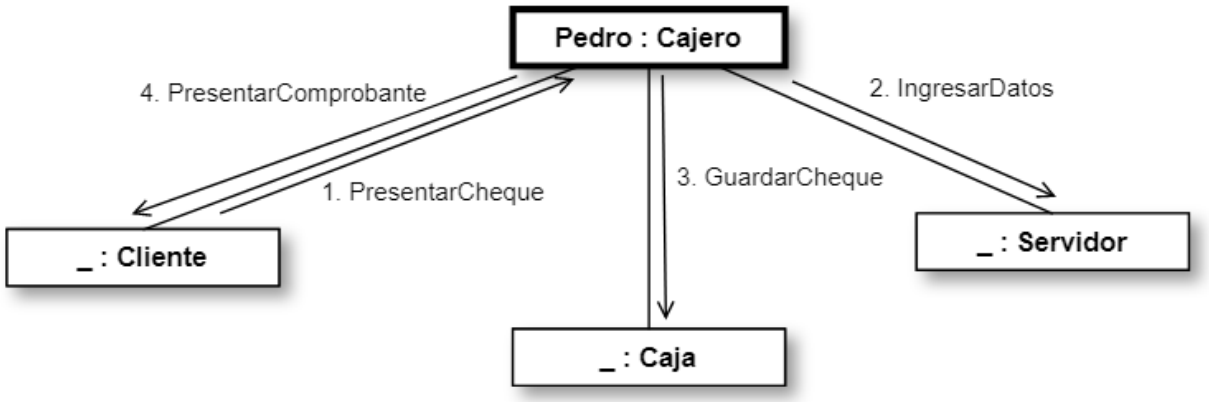


Objetos activos

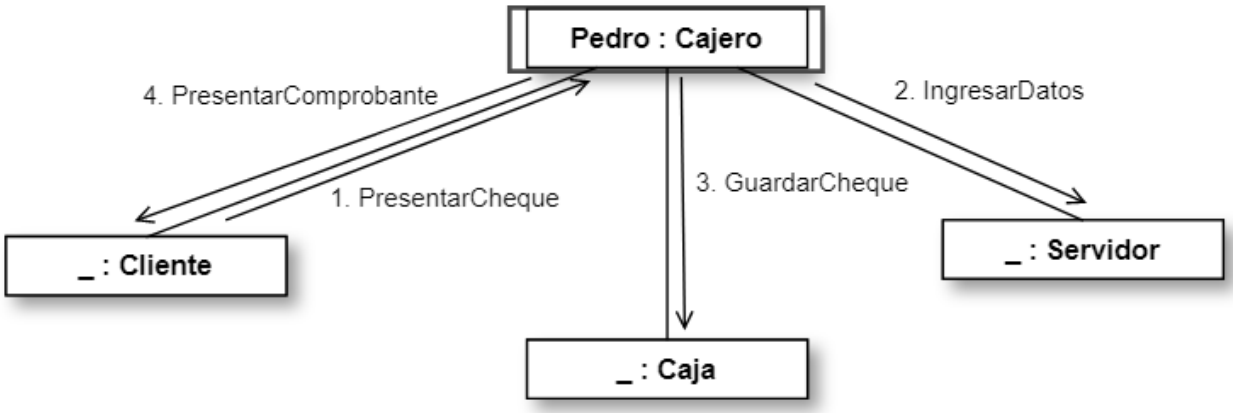
Los Objetos activos son aquellos que controlan el flujo en una secuencia, este puede enviar mensajes a los objetos pasivos o interactuar con otros objetos activos.

Por ejemplo, cuando alguien decide depositar un cheque, el cajero que lo atiende es el encargado de manejar la transacción e interactuar con las demás clases, verifica los datos, envía solicitudes, imprime recibos, etc.

En la versión de UML 1 se presenta un objeto activo con un rectángulo con el borde grueso



En UML 2 un objeto activo se presenta con un rectángulo con líneas dobles a los lados


Pedro : Cajero
4. PresentarComprobante
1. PresentarCheque
2. IngresarDatos
3. GuardarCheque
_ : Cliente
_ : Caja
_ : Servidor

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.