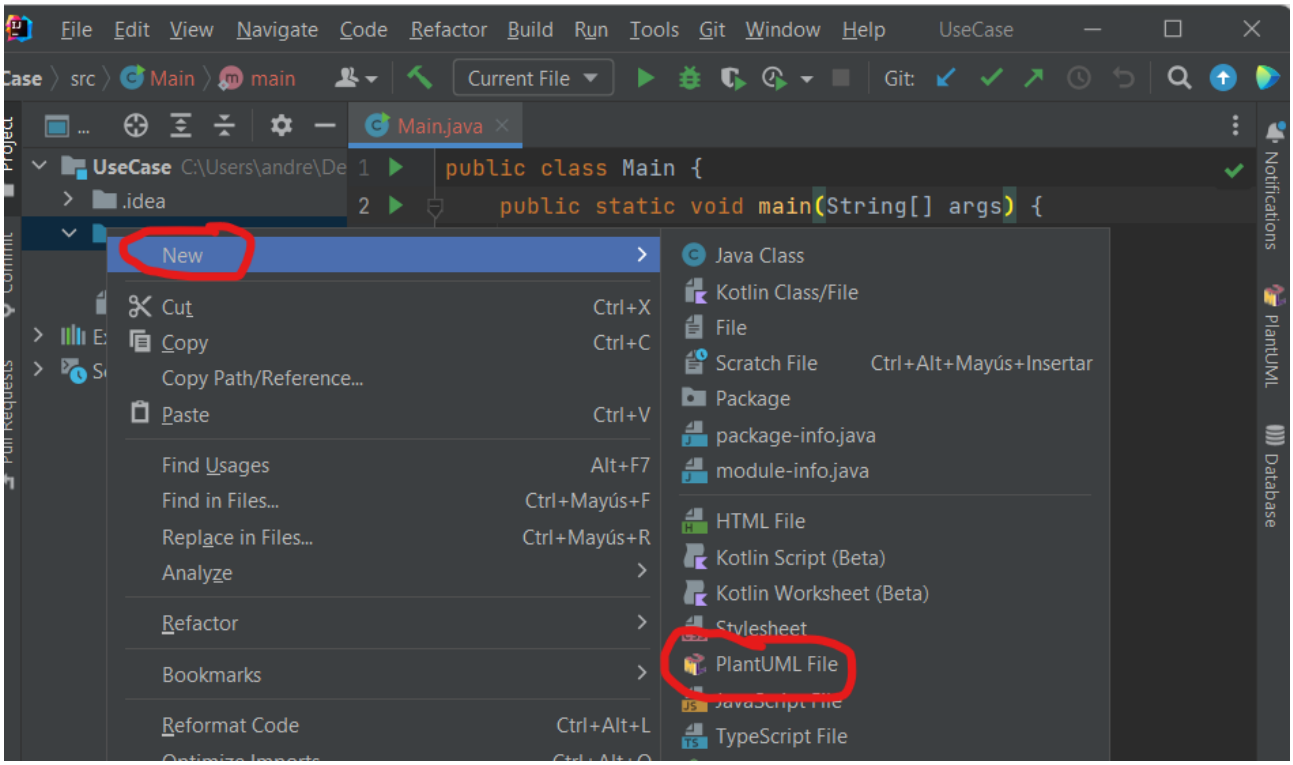
Andrés Penalva González 12/05/2023

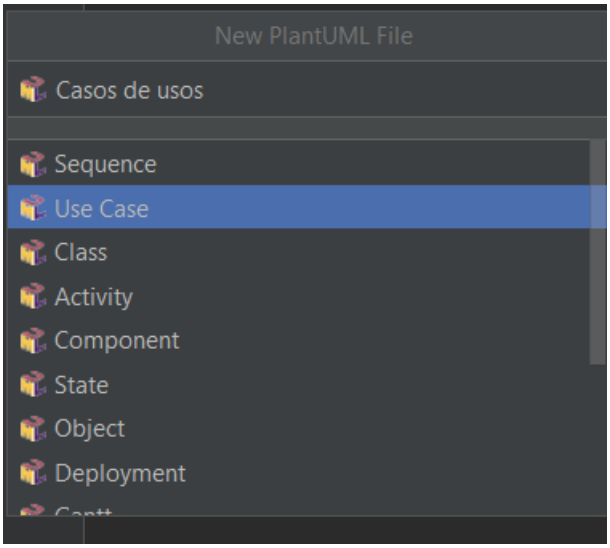
Vamos a crear una guia de cómo se utiliza los casos de usos.

Lo primero que todo, será crear el proyecto en nuestro IDE y crear un archivo *'.uml'* para poder trabajar con ello. Los pasos son los siguientes:

En la carpeta 'src' hacemos click derecho → 'new' → 'PlantUML file'

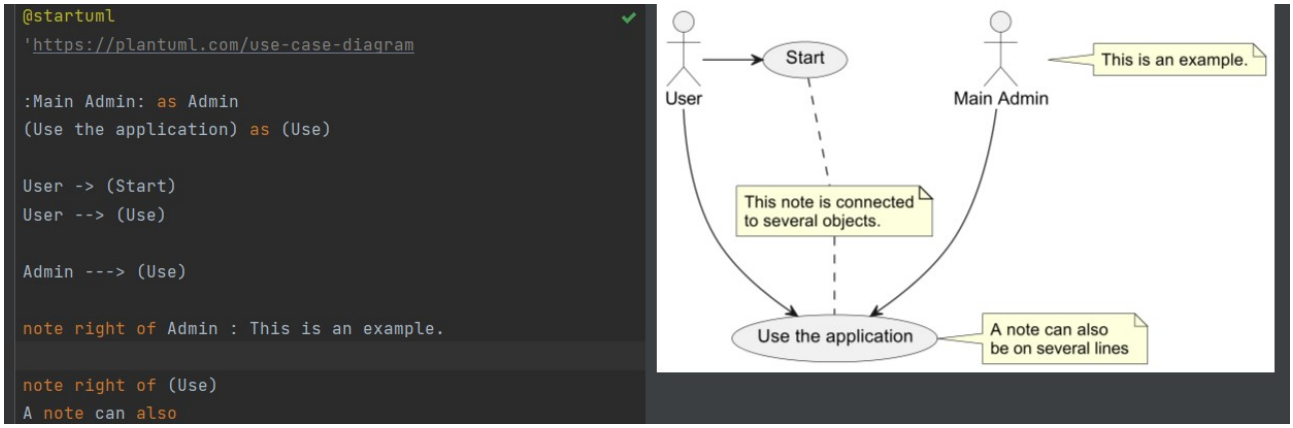


Ahora, pondremos le pondremos el nombre que queramos y le damos a la opción 'use case'.



Aceptamos y ya tendremos nuestro '*Use case*' creado.

Vemos también que el propio IDE nos abre la ventana con un pequeño esquema.



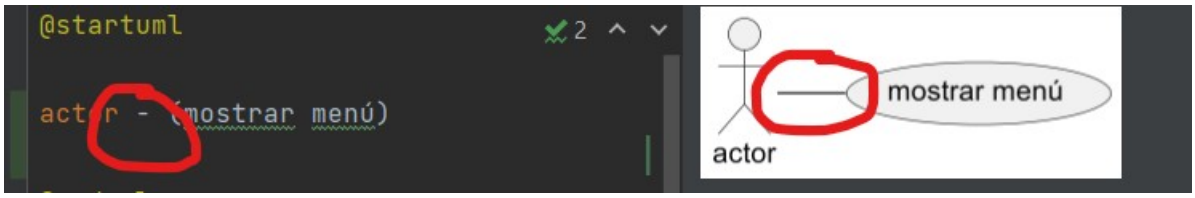
Ahora ya podremos trabajar en él.

Empezaremos conociendo los distintos elementos que lo forman:

- **actor**: son personas o procesos automáticos que necesitan interactuar con el sistema. Se deben identificar sus papeles en el sistema. En el diagrama, se representan del siguiente modo:

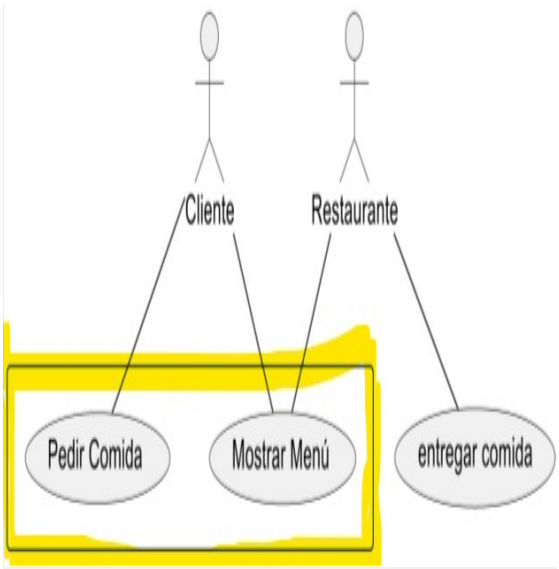
- **Caso de uso:** corresponden a acciones generales que pueden realizar los actores. Se representan mediante el uso de elipses y suelen ser verbos.

- **Asociación:** es la interacción de los actores y los casos de uso del sistema. Se representa con una línea recta que une a ambos.



- **Sistema:** El sistema es el software que vamos a desarrollar. Puede ser un pequeño componente cuyos actores son otros componentes, o puede ser una aplicación completa. Se representa como una caja rectangular usando en el código *'rectangle {}'*, donde dentro de las llaves se incluyen los casos de uso soportados por el sistema.

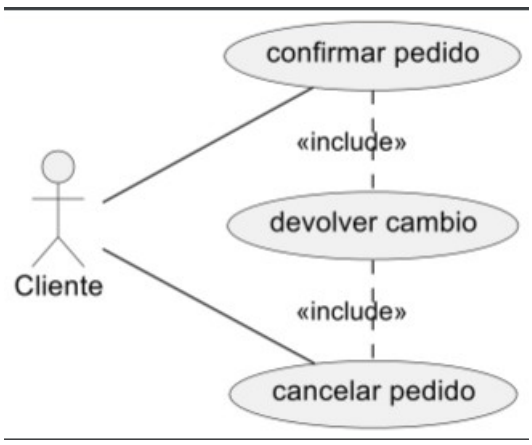
```
@startuml
rectangle {
    usecase "Mostrar Menú" as menu
    usecase "Pedir Comida" as pedido

}
  actor Restaurante
  actor Cliente
  Restaurante -- menu
  Cliente -- menu
  Cliente - pedido
  Restaurante -- (entregar comida)
@enduml
```



- **Inlcusión:** se utiliza cuando el comportamiento de un caso de uso se incluye dentro del comportamiento de otro. Se representa con una flecha de trazo discontinuo desde el caso que incluye hasta el caso incluido, con el estereotipo «include» o «use».

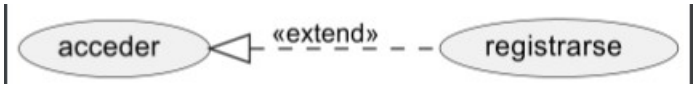
```
@startuml
left to right direction
Cliente -- (confirmar pedido)
Cliente -- (cancelar pedido)
(devolver cambio) as devolucion
devolucion .(cancelar pedido) :<<include>>
(confirmar pedido) . devolucion :<<include>>
@enduml
```

Con la directiva 'left to right direction' conseguimos que la representación se haga de izquierda a derecha y no de arriba a bajo.



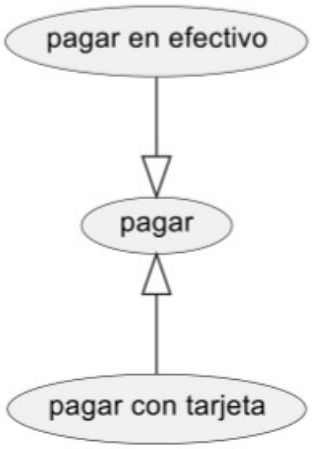
- **Extensión:** se utiliza cuando un caso además aporta un comportamiento adicional en determinadas circunstancias o cuando se cumple cierta condición. Se representa con una flecha de trazo discontinuo que apunta al caso que queremos extender, y el estereotipo «extend»

```
@startuml
left to right direction
(acceder)<|.. (registrarse) : <<extend>>
@enduml
```



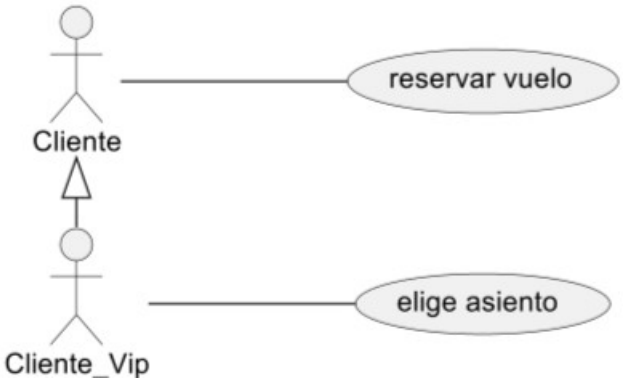
- **Generalización:** se utiliza para expresar que un caso de uso especializado es una forma particular de conseguir los objetivos de otro caso de uso más general. Se representa como una flecha continua acabado en punta triangular hueca que apunta al caso más general.

```
@startuml
left to right direction
(pagar) <|-- (pagar con tarjeta)
(pagar en efectivo)--|> (pagar)
@enduml
```



Esta generalización también se puede usar con actores

```
left to right direction
Cliente <|- Cliente_Vip
Cliente -- (reservar vuelo)
Cliente_Vip -- (elige asiento)
@enduml
```



En ocasiones, es díficil saber qué relación debemos usar de las tres. En ese caso, una ayuda sería la siguiente:
- La **inclusión** equivale a “copiar y pegar”, de modo que un caso de uso está dentro de otro.
- La **extensión** se utiliza cuando un caso añade funcionalidad a otro dependiendo de alguna condición. Sería el equivalente a introducir una condición “if” en algún punto de A, de forma que si se cumple la condición, se ejecuta B antes de proseguir con A.
- La **generalización** sirve para indicar que varios casos tienen el mismo comportamiento pero lo llevan a cabo de modo distinto. Es la equivalencia a la *‘herencia’* en POO, donde las clases hijas pueden rescribir parte del código de la clase padre.

Además, deberemos tener en cuenta las siguientes recomendaciones:

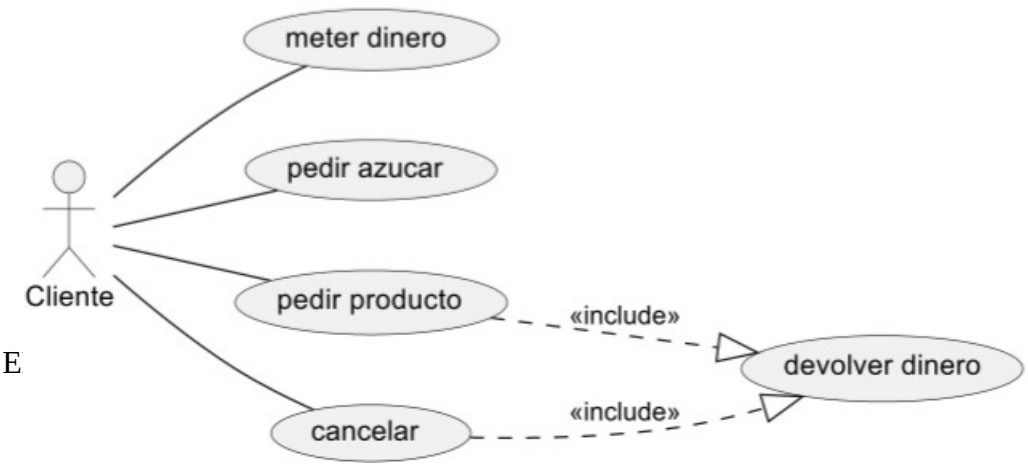
- Los casos de uso describen las interacciones más importantes con el sistema, no su funcionamiento interno.
- Los casos de uso deben ser lo más simples posibles, para poder entenderlo fácilmente.
- Hay que tener en cuenta que el diagrama de casos de usos se elabora durante la fase de especificación de requisitos y que éstos son cambiantes. Por ello, no conviene profundizar mucho en detalles porque sería tiempo perdido si éstos cambiasen.
- En cualquier caso, es preferible acompañar el diagrama con una buena descripción narrativa del caso de uso.

==========

Ejemplo: La máquina de café

Supongamos que se requiere desarrollar el control de una máquina de entrega de café automática. La máquina debe permitir a una persona introducir dinero, escoger uno de los productos de acuerdo a su precio, escoger un nivel de azúcar y entregar el producto y las vueltas. El usuario puede en cualquier momento antes de escoger el azúcar cancelar la operación, mediante un botón existente para este objetivo.

```
@startuml
left to right direction
Cliente -- (meter dinero)
Cliente -- (pedir azucar)
Cliente -- (pedir producto)
Cliente -- (cancelar)
(cancelar)..|>(devolver dinero):<<include>>
(pedir producto)..|>(devolver dinero) : <<include>>
@enduml
```

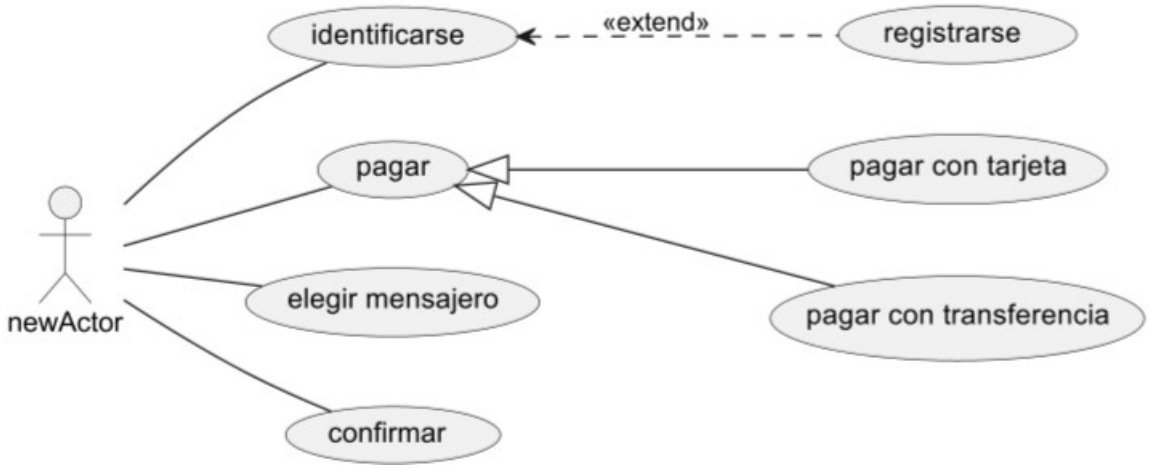


El diagrama hace uso de la relación «include» para reutilizar el caso de uso “Devolver dinero”

Ejemplo: Tienda en Internet

Queremos modelar el sistema de pago en una tienda web. El cliente debe identificarse mediante su dirección de correo. Si es un nuevo cliente se le debe registrar en el sistema previamente, pidiéndole los datos personales. Una vez identificado al cliente, éste podrá elegir el medio de pago: por transferencia bancaria o con tarjeta de crédito. Según el medio de pago se le solicitarán unos datos u otros. El cliente también deberá elegir el método de envío. Finalmente se le mostrarán todos los datos del pedido para pedirle que confirme.

```
@startuml
left to right direction
newActor -- (identificarse)
newActor -- (pagar)
newActor -- (elegir mensajero)
newActor -- (confirmar)
(pagar) <|--(pagar con tarjeta)
(pagar) <|-- (pagar con transferencia)
(identificarse)<.. (registrarse): <<extend>>
@enduml
```

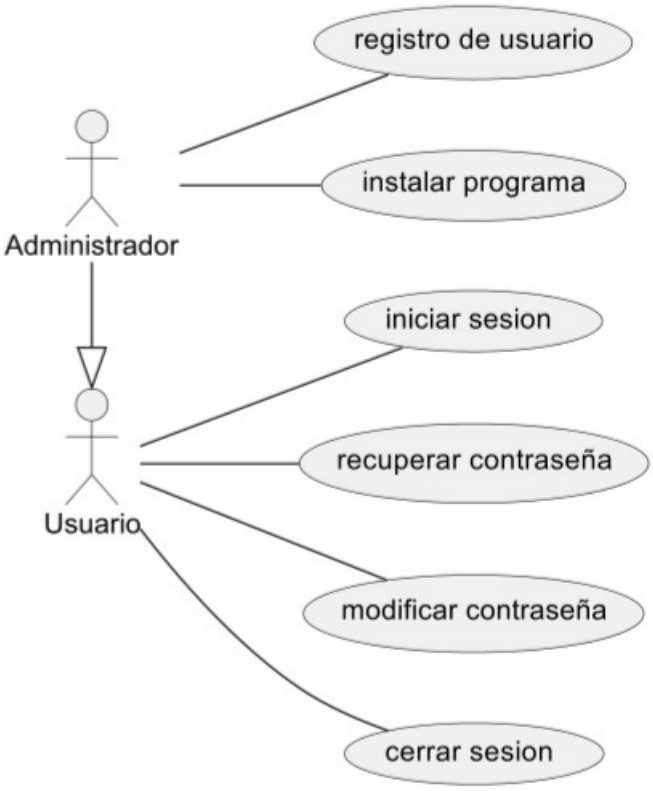


En este diagrama se puede observar la descomposición del caso general “Pagar” en los casos específicos “Pagar con tarjeta” y “Pagar con transferencia” mediante una generalización. El caso “Registrarse” extiende a “Identificarse” porque está sujeto a la condición “si es un nuevo cliente”.

Ejemplo: Usuarios y administradores

Queremos modelar un sistema en el que hay usuarios. Los usuarios pueden iniciar sesión, modificar su contraseña, recuperar su contraseña y cerrar sesión. Los administradores tienen los mismos permisos que los usuarios, pero además, pueden registrar usuarios e instalar programas.

```
@startuml
left to right direction
Usuario -- (iniciar sesion)
Usuario -- (recuperar contraseña)
Usuario -- (modificar contraseña)
Usuario -- (cerrar sesion)
Administrador -- (registro de usuario)
Administrador -- (instalar programa)
Administrador -|> Usuario
@enduml
```

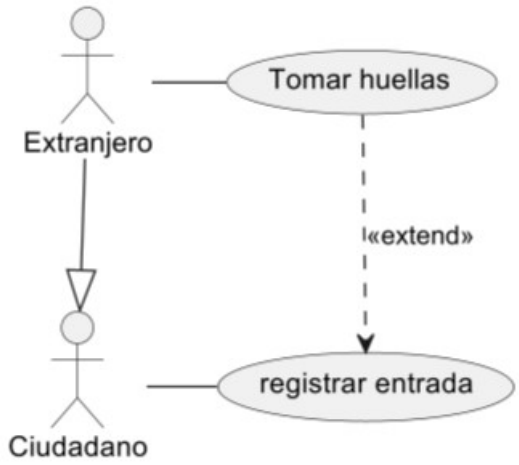


En este ejemplo nuevamente se utiliza la generalización, esta vez entre actores, para indicar que los administradores son un tipo específico de usuario

Ejemplo: Puesto fronterizo

En la frontera de un país se registran todos los ciudadanos que entran. Además, en caso de que el ciudadano sea extranjero, se le toma la huella dactilar.

```
@startuml
Ciudadano - (registrar entrada)
Extranjero- (Tomar huellas)
Extranjero --|> Ciudadano
(Tomar huellas)..> (registrar entrada): <<extend>>
@enduml
```

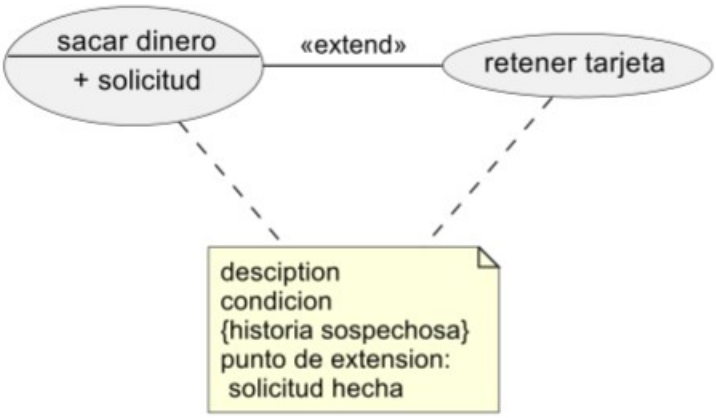


Nuevamente utilizamos la relación «extend» para indicar que la toma de huellas se realiza como parte del registro de entrada cuando se da la condición de que el ciudadano es extranjero, lo que expresamos añadiendo el actor “ciudadano extranjero” que interactúa con este caso de uso.

Puntos de extensión

Los puntos de extensión se utilizan en las relaciones de extensión para indicar en qué punto del caso base se inserta el comportamiento del caso extendido, y bajo qué condición. Veámoslo con un ejemplo:

```
@startuml
usecase UC1 as "sacar dinero
--
+ solicitud"

note "desciption\ncondicion\n{historia sospechosa}\npunto de extension:\n solicitud hecha" as n1

usecase UC2 as "retener tarjeta"
UC1-UC2:  <<extend>>
UC1 .. n1
UC2 .. n1
@enduml
```



En el caso base (Sacar dinero) hemos añadido un punto de extensión solicitud hecha, que es el momento en el que se inserta el comportamiento de Retener tarjeta, pero sólo cuando se dé la condición historia sospechosa.