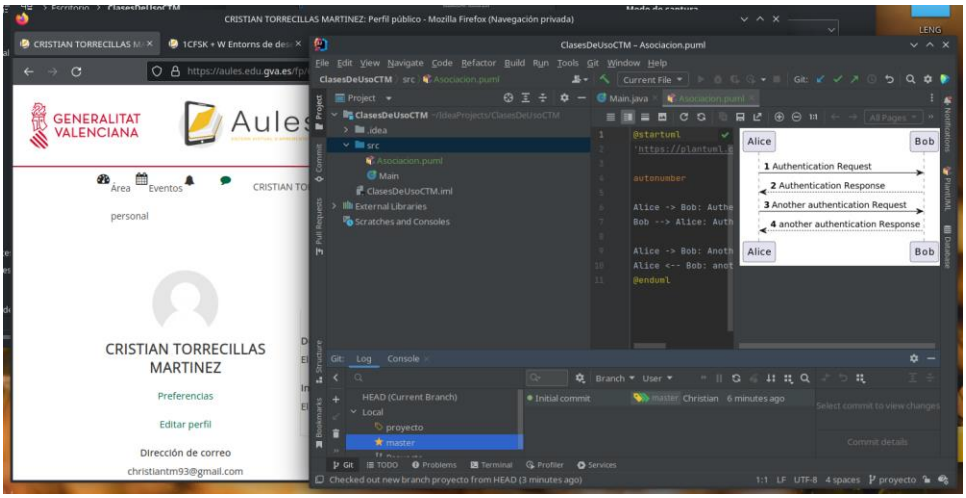
Diagramas de casos de uso

Christian Torrecillas 1K DAM

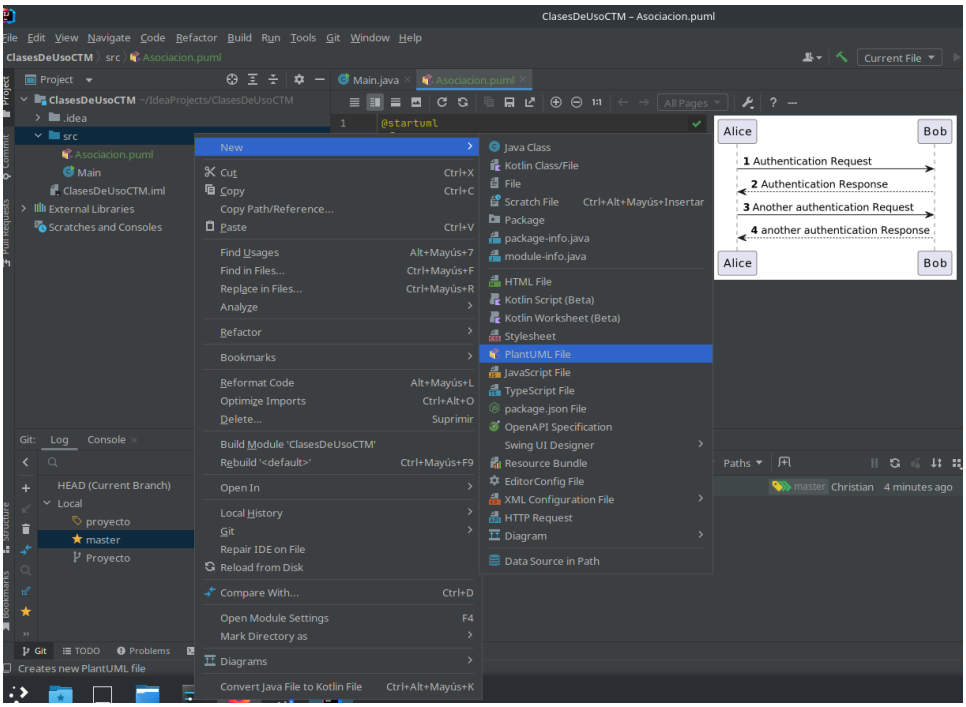
https://github.com/riekk/ClasesDeUsoICTM

En esta práctica vamos a realizar una pequeña introducción a los diagramas de uso, sus elementos, sus relaciones y unos ejemplos para ver como quedarían en un proyecto con varios elementos.

Para comenzar debemos crear un nuevo proyecto y tener instalada la herramienta PlantUML en nuestro IDE.



Una vez disponemos de ella deberemos crear un nuevo archivo UML para poder hacer uso de sus características.

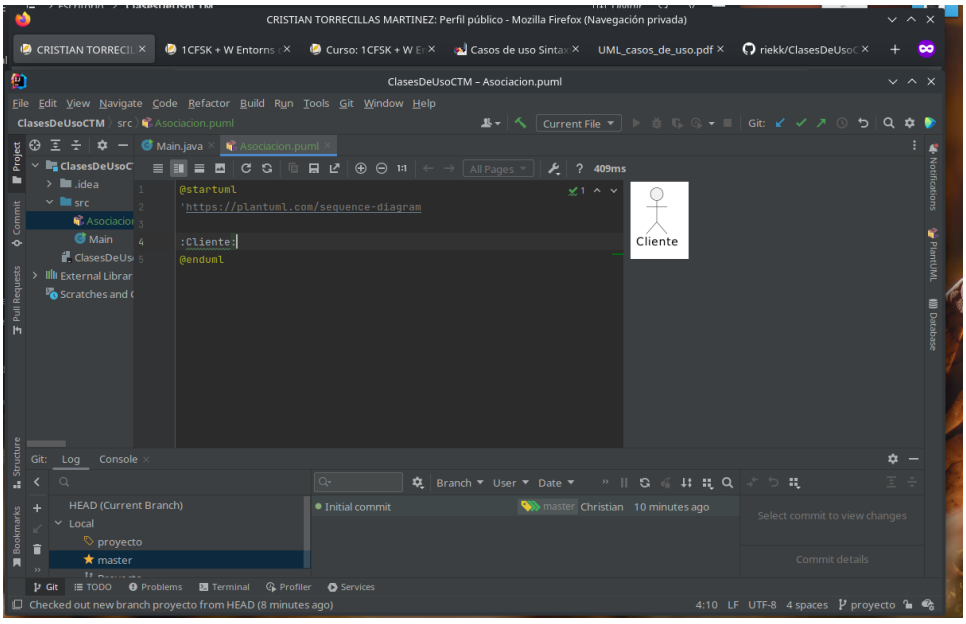


Lo primero que disponemos son los actores, en este caso un cliente. Podemos crearlo de dos formas:

:Cliente:

actor :ClienteVIP: as VIP

Si añadimos el ‘as’ estamos estableciendo un alias a nuestro actor. Con esto conseguimos poder utilizarlo de forma mas sencilla.

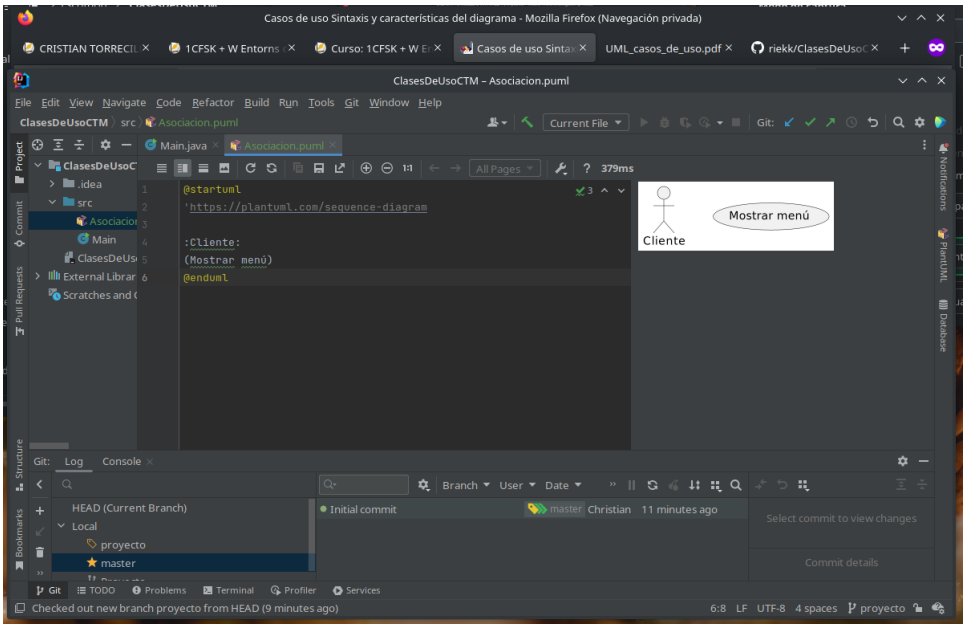


Lo siguiente que disponemos es de los casos de uso. También tenemos dos formas para crearlos:

usecase c1 as "Mostrar menú"

(Mostrar menú)

Estas corresponden a acciones que llevaran a cabo nuestros actores, cuando las relacionemos.

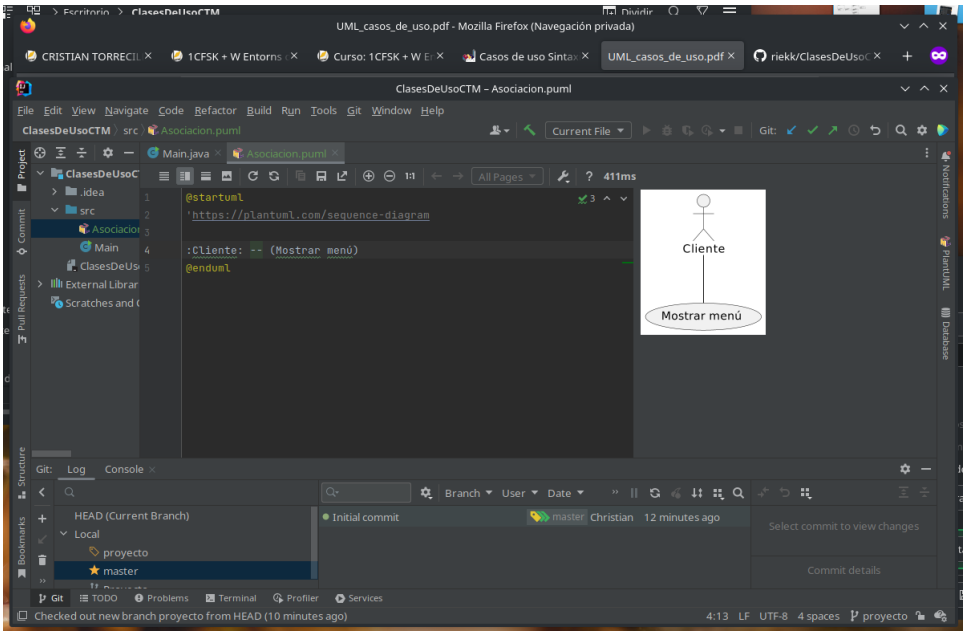


Para relacionar nuestros actores con nuestros casos de uso los uniremos de la siguiente manera:

:Cliente: -- (Mostrar menú)

En la parte de la derecha que nos muestra nuestro diagrama de uso vemos que ahora aparece una linea que une nuestro actor y nuestro uso de caso.

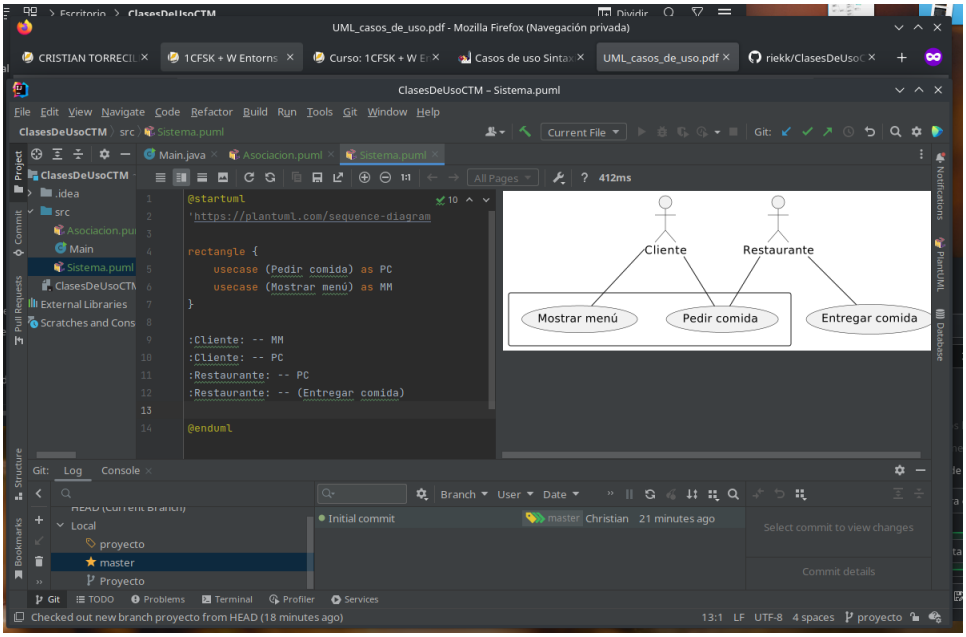
Esta es la manera más sencilla de relacionar los elementos en nuestro diagrama de casos de usos y es conocida como **asociación.**



El siguiente elemento es el **Sistema** sirve para separar indicar cuales son los casos de usos soportados por un sistema y se representa usando un rectángulo para encajonar los casos de usos.

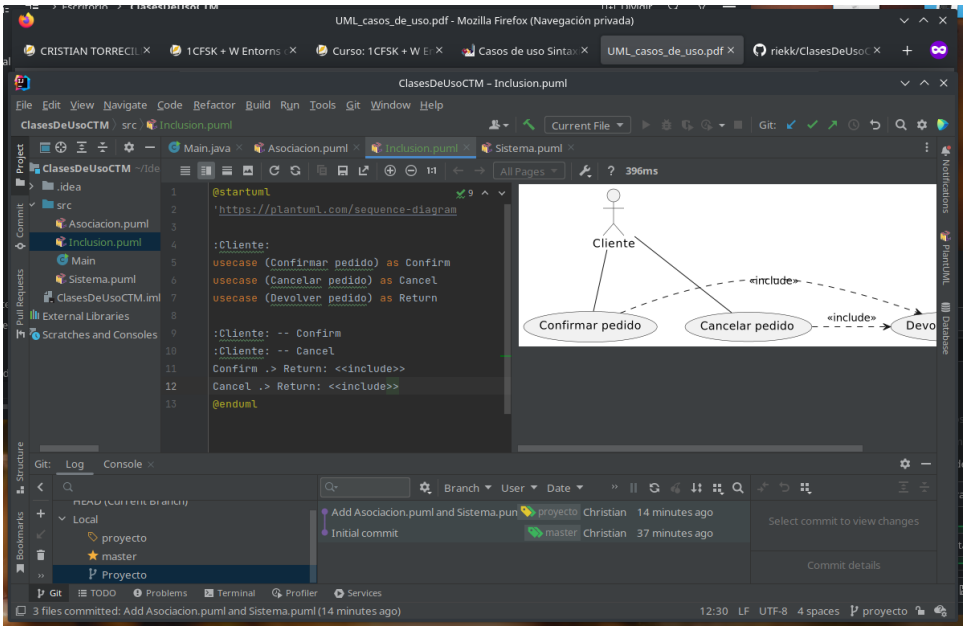
Representándolo en código de la siguiente manera:

```
rectangle {
 usecase (Pedir comida) as PC
 usecase (Mostrar menú) as MM
}
```



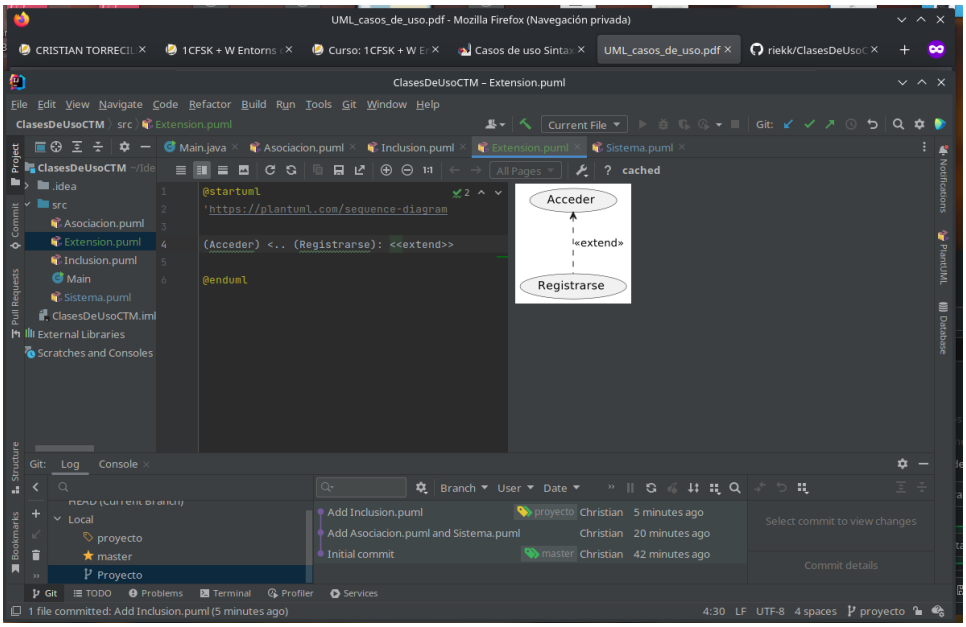
Ahora vamos a comentar la **inclusión** sirve para indicar que el comportamiento de uno de nuestros casos de uso se incluye dentro del comportamiento de otro. Se representa con una línea discontinua y una flecha al caso de uso que incluye al anterior y añadiendo una etiqueta **<<include**>> en la relación. La forma para representarlo es la siguiente:

```
Confirm .> Return: <<include>>
```



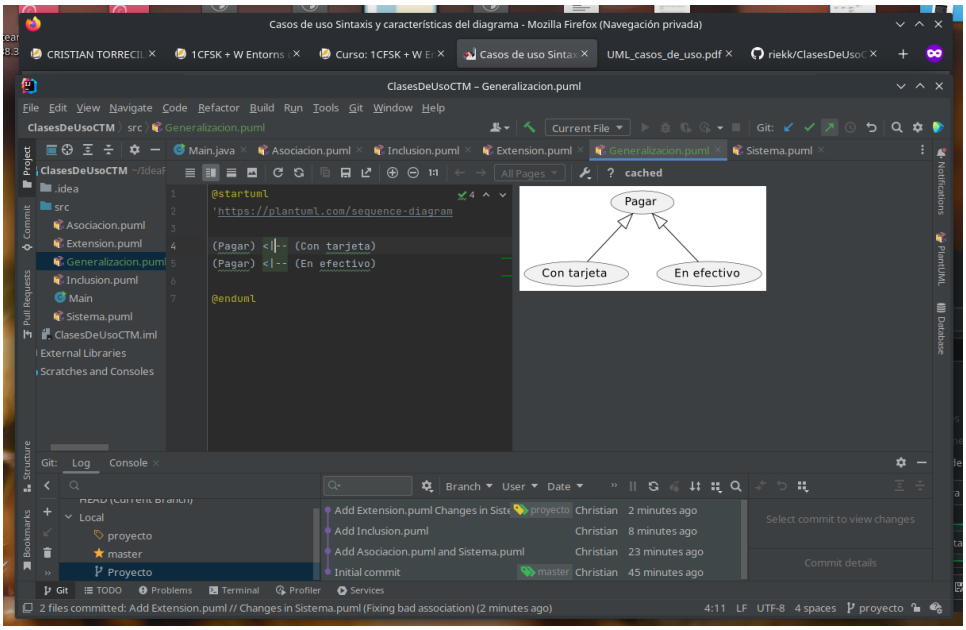
Ahora vamos a utilizar la Extensión, se utiliza para indicar que un caso de uso viene de otro, pero además tiene un comportamiento adicional sobre el que ha heredado. Se representa con una línea discontinua y una flecha indicando el caso de uso que hereda de quien, añadiendo además una etiqueta a la relación que indique **<<extends>> .** Se representa de igual forma que la inclusión pero cambiando el valor de la etiqueta.

```
(Acceder) <.. (Registrarse): <<extend>>
```



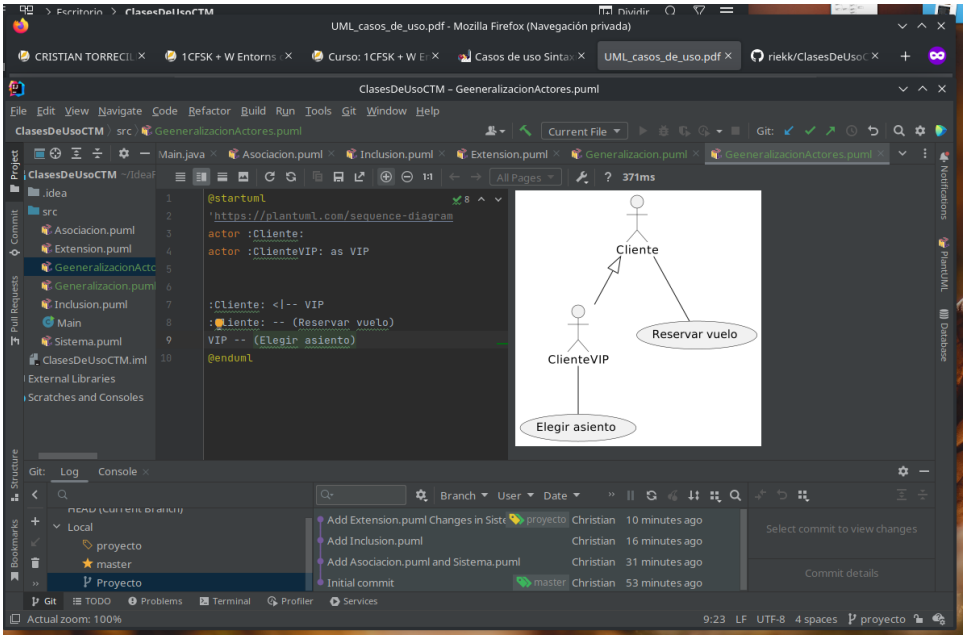
La **generalización** se utiliza para indicar que los casos de usos especializados a la hora de indicarlos de forma mas específica. Se representa con una flecha continua y una punta triangular hueca apuntando al caso que se considera mas general.

```
(Pagar) <|-- (Con tarjeta)
(Pagar) <|-- (En efectivo)
```



También puedes realizar generalizaciones con actores, quedando de la siguiente manera.

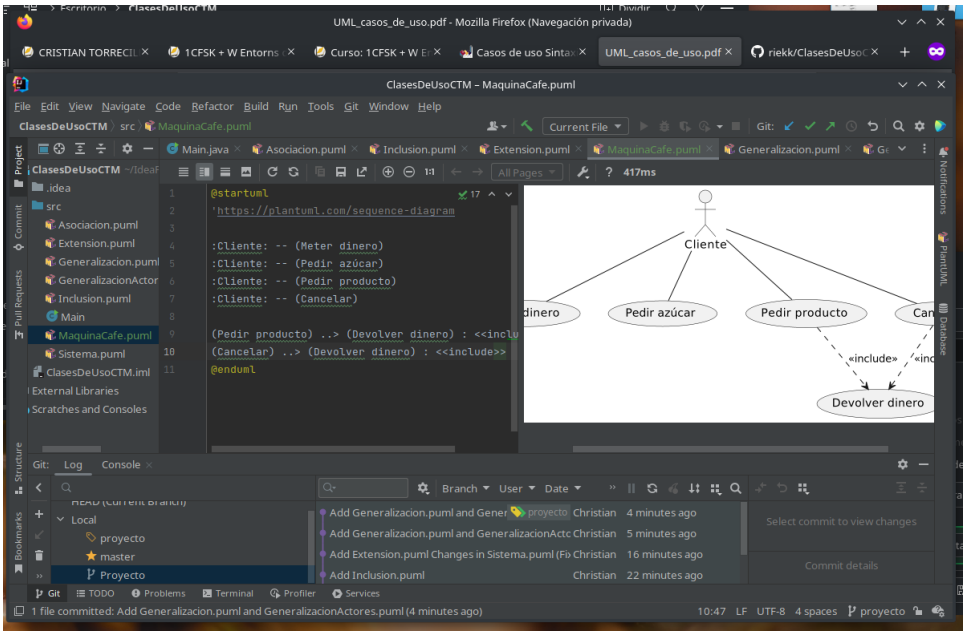
```
actor :Cliente:
actor :ClienteVIP: as VIP


:Cliente: <|-- VIP
:Cliente: -- (Reservar vuelo)
VIP -- (Elegir asiento)
```



Ahora vamos a poner el caso de una máquina de café en la que encontramos en la que podemos realizar varias acciones, pero a la hora de pedir un producto o cancelar la compra estas acciones tienen incluido el caso de devolver el dinero por si el cliente no quiere nada, o si le sobra dinero tras la compra.

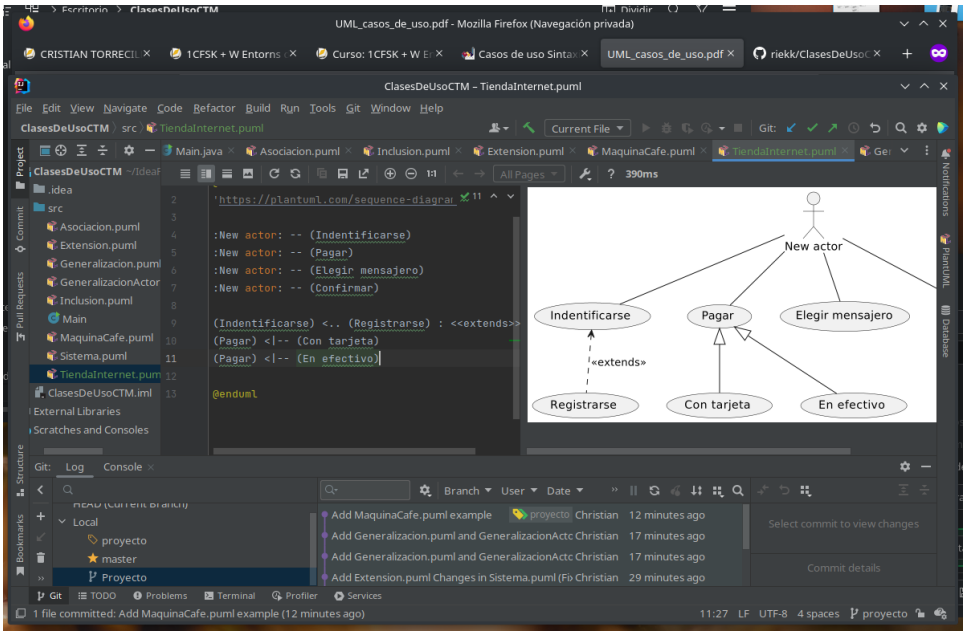
Esto ejemplifica mejor como seria el diagrama de casos de uso de un sistema más avanzado a las simples relaciones que hemos visto anteriormente.

```
:Cliente: -- (Meter dinero)
:Cliente: -- (Pedir azúcar)
:Cliente: -- (Pedir producto)
:Cliente: -- (Cancelar)

(Pedir producto) ..> (Devolver dinero) : <<include>>
(Cancelar) ..> (Devolver dinero) : <<include>>
```



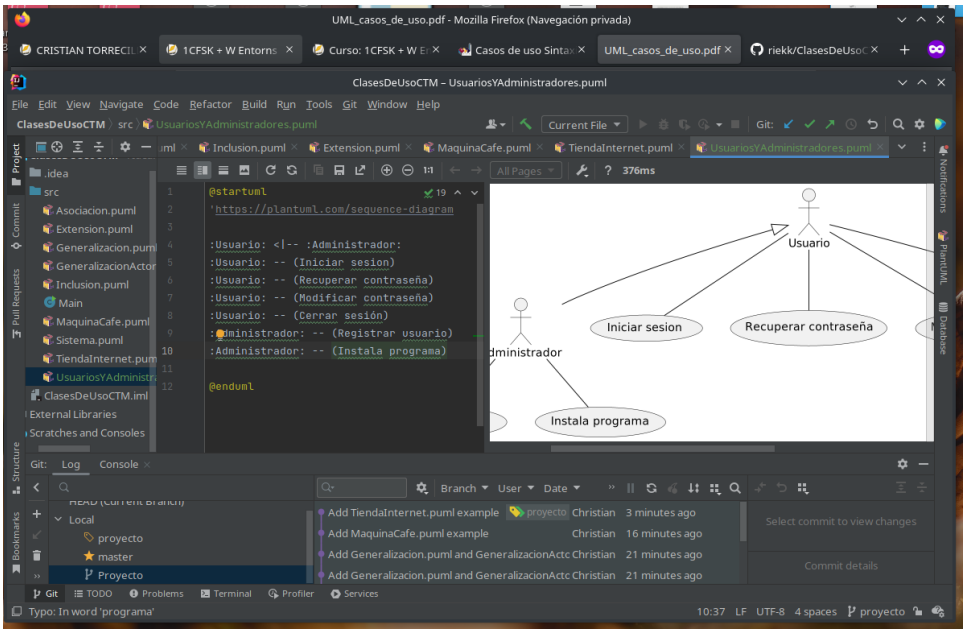
En el caso de una tienda online disponemos de más acciones como identificarse y en caso de no estarlo registrarse para ser identificado con lo que conlleva que es una extensión de la primera. Una generalización a la hora de pagar, ya que se hace un pago, pero puede ser en efectivo o con tarjeta. Y otras asociaciones sencillas.

```
:New actor: -- (Indentificarse)
:New actor: -- (Pagar)
:New actor: -- (Elegir mensajero)
:New actor: -- (Confirmar)

(Indentificarse) <.. (Registrarse) : <<extends>>
(Pagar) <|-- (Con tarjeta)
(Pagar) <|-- (En efectivo)
```



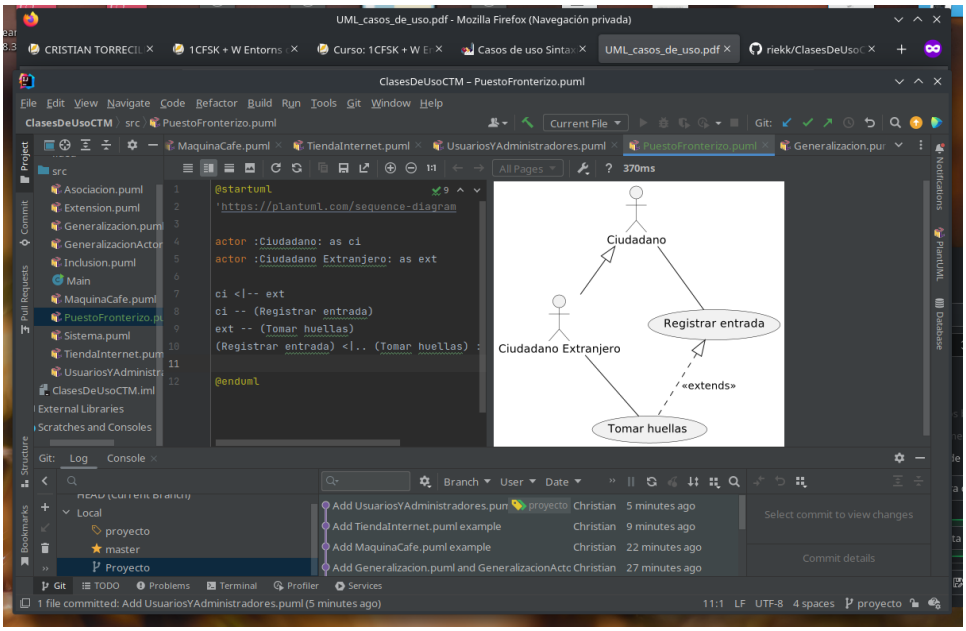
Ahora vamos a ver el ejemplo de usuarios y administradores de un sistema. Como vemos un Administrador es una extensión de un usuario normal, pero a la hora de realizar sus funciones han sido modificadas y dispone de dos casos de uso más que el usuario normal. Esto nos indica que además de tener la parte que ha heredado del usuario tiene acciones específicas propias.

:Usuario: <|-- :Administrador:
:Usuario: -- (Iniciar sesion)
:Usuario: -- (Recuperar contraseña)
:Usuario: -- (Modificar contraseña)
:Usuario: -- (Cerrar sesión)
:Administrador: -- (Registrar usuario)
:Administrador: -- (Instala programa)



En este ejemplo disponemos de un puesto fronterizo en el cual un ciudadano simplemente registra su entrada, pero un ciudadano extranjero tiene que tomar las huellas como parte de una extensión doble entre ciudadano normal y extranjero y registrar la entrada y tomar las huellas.

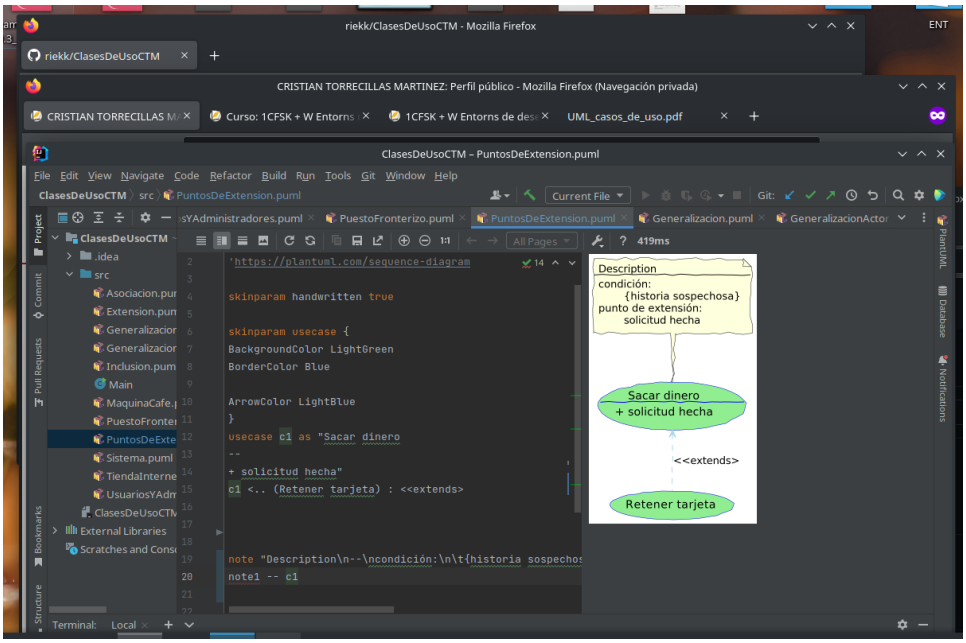
```
actor :Ciudadano: as ci
actor :Ciudadano Extranjero: as ext

ci <|-- ext
ci -- (Registrar entrada)
ext -- (Tomar huellas)
(Registrar entrada) <|.. (Tomar huellas) : <<extends>>
```



En los puntos de extensión podemos agregar información adicional sobre cuando se realizan las extensiones y las condiciones por la que se realiza la extensión.

También podemos añadir un poco de css mediante skinparam para conseguir que nuestros diseños se vean así.

```
skinparam handwritten true

skinparam usecase {
BackgroundColor LightGreen
BorderColor Blue

ArrowColor LightBlue
}
usecase c1 as "Sacar dinero
--
+ solicitud hecha"
c1 <.. (Retener tarjeta) : <<extends>>



note "Description\n--\ncondición:\n\t{historia sospechosa}\npunto de extensión:\n\tsolicitud hecha" as note1
note1 -- c1
```


riekk/ClasesDeUsoCTM - Mozilla Firefox
riekk/ClasesDeUsoCTM
CRISTIAN TORRECILLAS MARTINEZ: Perfil público - Mozilla Firefox (Navegación privada)
Curso: 1CFSK + W Entorns
1CFSK + W Entorns de des
UML_casos_de_uso.pdf
ClasesDeUsoCTM – PuntosDeExtension.puml
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
ClasesDeUsoCTM src PuntosDeExtension.puml
Current File
PuestoFronterizo.puml
PuntosDeExtension.puml
Generalizacion.puml
All Pages
419ms
'https://plantuml.com/sequence-diagram
skinparam handwritten true
skinparam usecase {
BackgroundColor LightGreen
BorderColor Blue
ArrowColor LightBlue
}
usecase c1 as "Sacar dinero
+ solicitud hecha"
c1 <.. (Retener tarjeta) : <<extends>
note "Description\n--\ncondición:\n\t{historia sospechos
note1 -- c1
Description
condición:
{historia sospechosa}
punto de extensión:
solicitud hecha
Sacar dinero
+ solicitud hecha
<<extends>>
Retener tarjeta
Terminal: Local