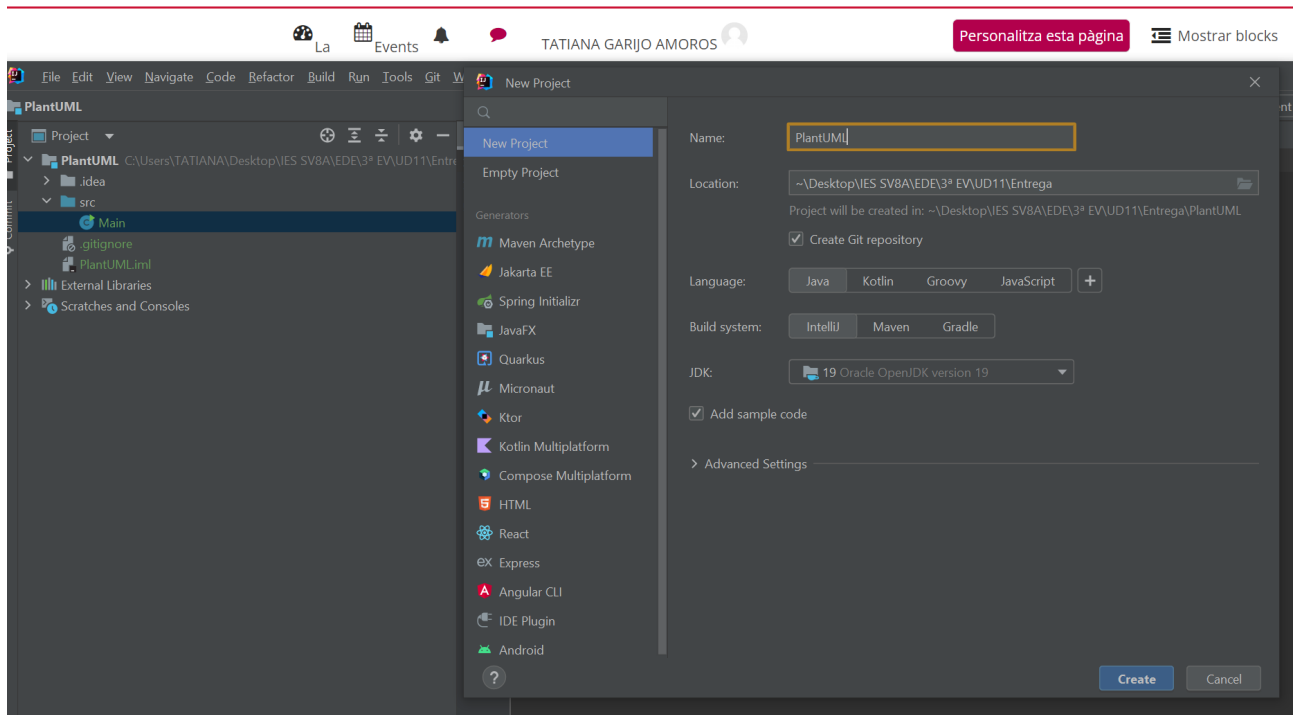


PRÁCTICA Diagramas UML Clases de uso

Repositorio GitHub

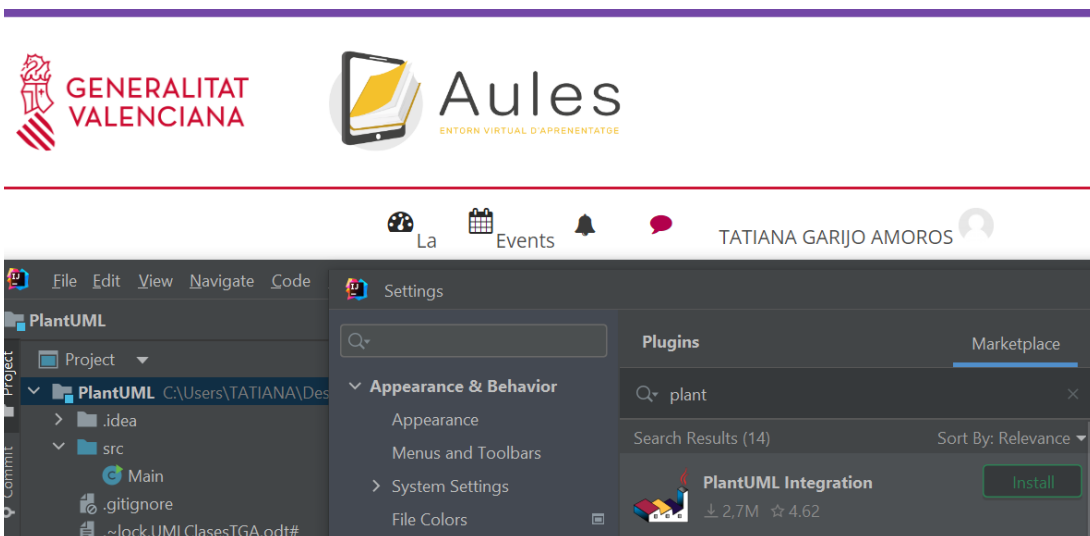
<https://github.com/tatianagarijo/UMLClasesUsoTatianaGarijo.git>



- New Project IntelliJ
- Abrir IntelliJ
- Seleccionar la opción New Project
- Configurar las características del proyecto
 - En mi caso he seleccionado
 - Nombre del proyecto
 - Lenguaje: Java
 - Build System: IntelliJ
 - version 19 JDK
 - Create Git repository

• PlantUML

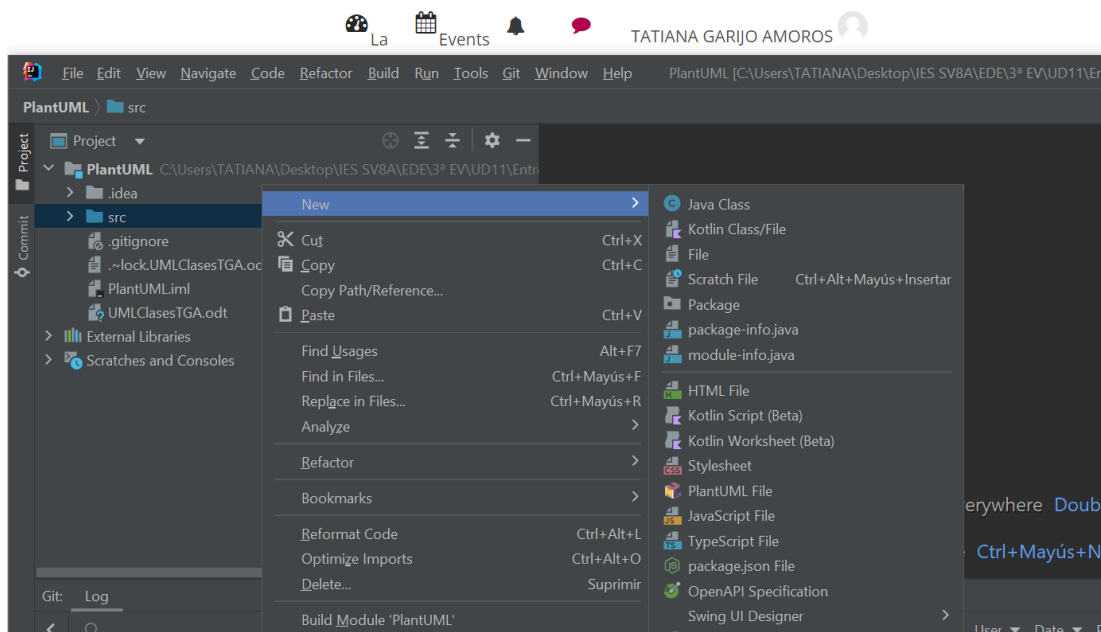
- PlantUML es una herramienta que nos permite crear diagramas tipo UML.
- Instalamos el plugin PlantUML en IntelliJ.
 - En la web oficial <https://plantuml.com/es/> podemos encontrar toda la información del plugin.
 - Navegamos por el menú File --> Settings --> Plugins
 - Buscamos el plugin PlantUML haciendo uso del buscador, una vez localizado hacemos clic sobre el botón **install**.



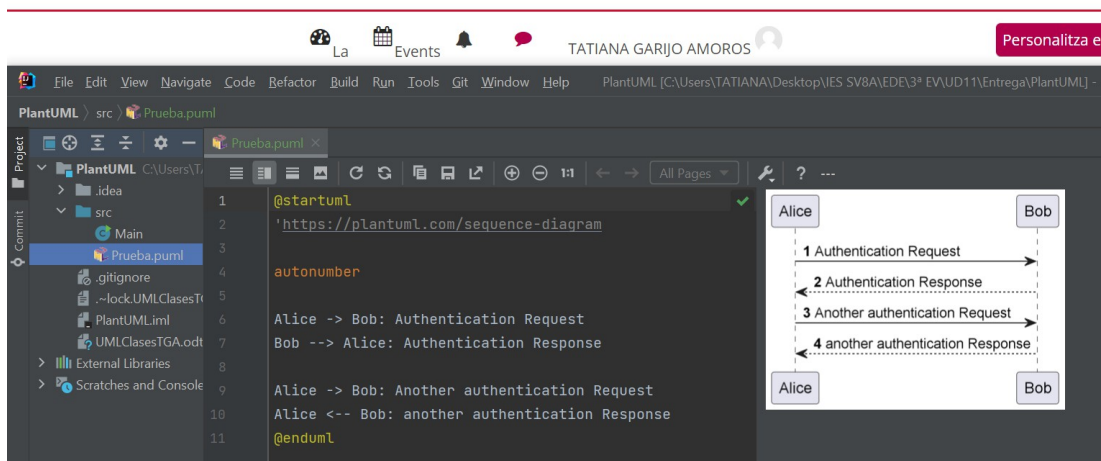
- El IDE nos pide reiniciar para cargar el plugin, hacemos clic sobre el botón **Restart IDE**



- Hacemos clic botón derecho sobre la carpeta src → new → PlantUML File → Use Case



- Le damos el nombre de nuestra elección y nos aparece la siguiente ventana



- En el panel central entre las instrucciones `@startuml` `@enduml` iremos escribiendo el código necesario para crear el diagrama, en el panel derecho irá apareciendo el diagrama según el código que vayamos generando.
- Para crear los diagramas hacemos uso de la información que nos facilita la web de PlantUML (<https://plantuml.com/es/class-diagram>), copiamos los fragmentos de código necesarios para crear el diagrama requerido.

- Replicando las fases del ejemplo resuelto

- Elementos del diagrama:

- **Actor:** Los actores 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:



```
@startuml
    :Actor:
@enduml
```

- **Caso de uso:** Los casos de uso se representan mediante elipses y corresponden a acciones generales del sistema. Una forma de reconocerlos es que suelen ser verbos en la descripción del caso de uso.

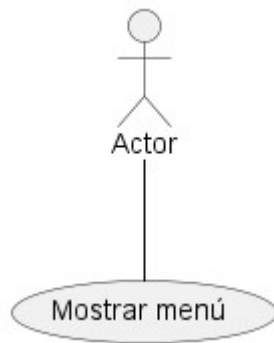


```
@startuml
    usecase (Mostrar menú)
@enduml
```

- **Asociación:** La interacción entre los actores y los casos de uso del sistema se representan por una línea recta que une a ambos.



```
@startuml
    :Actor: -- (Mostrar menú)
@enduml
```



```

@startuml
    left to right direction
    :Actor: -- (Mostrar menú)
@enduml
  
```

- **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. Dentro de ella se incluyen los casos de uso soportados por el sistema.



```

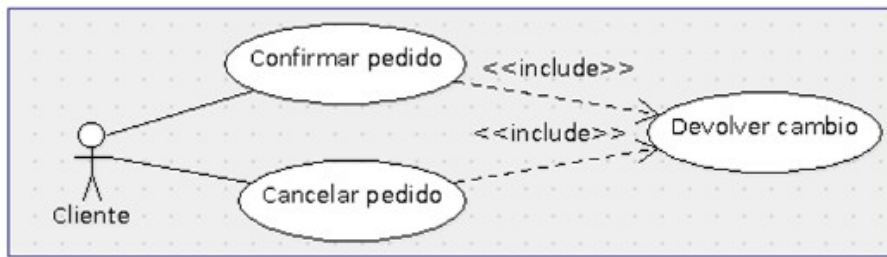
@startuml
    left to right direction

    :Actor: as act
    rectangle {
        usecase (Mostrar menú) as UC1
        usecase (Pedir comida) as UC2
    }
    usecase (Entregar comida) as UC3
    :Restaurante: as rest
    act -- UC1
    act -- UC2
    UC1 -- rest
    UC3 -- rest
@enduml
  
```

- **Inclusión:** 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»

Los casos de uso incluidos se pueden compartir, así evitamos repetirlos.

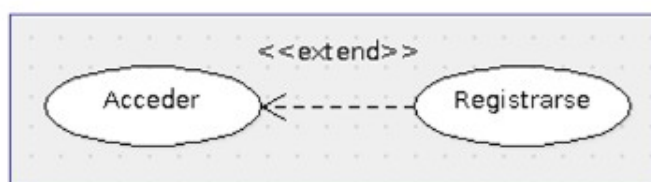
También se pueden utilizar para estructurar el diagrama en varios niveles de detalle, pero no conviene abusar de ellos (ver recomendaciones más abajo).



```

@startuml
left to right direction
:Cliente: as cli
usecase (Confirmar pedido) as UC1
usecase (Cancelar pedido) as UC2
usecase (Devolver cambio) as UC3
cli -- UC1
cli -- UC2
UC1 ..> UC3 : <<include>>
UC2 ..> UC3 : <<include>>
@enduml
  
```

- **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».

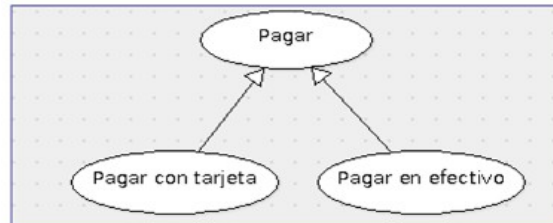


```

@startuml
left to right direction
usecase (Acceder) as UC1
usecase (Registrarse) as UC2
UC1 <.. UC2 : <<extend>>
@enduml
  
```

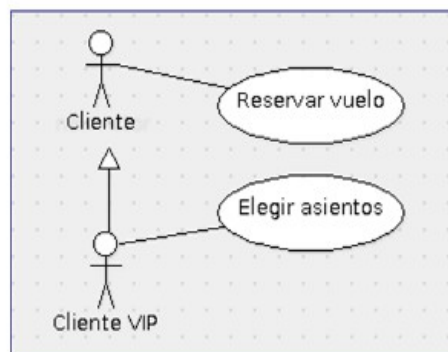
** En la web de PlantUML (<https://plantuml.com/es/use-case-diagram>) la extensión la representa con este tipo de flecha: <|--

- **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 acabada en punta triangular hueca que apunta al caso más general.



```
@startuml
usecase (Pagar) as UC1
usecase (Pagar con tarjetas) as UC2
usecase (Pagar en efectivo) as UC3
UC1 <|-- UC2
UC1 <|-- UC3
@enduml
```

También se puede utilizar con actores:



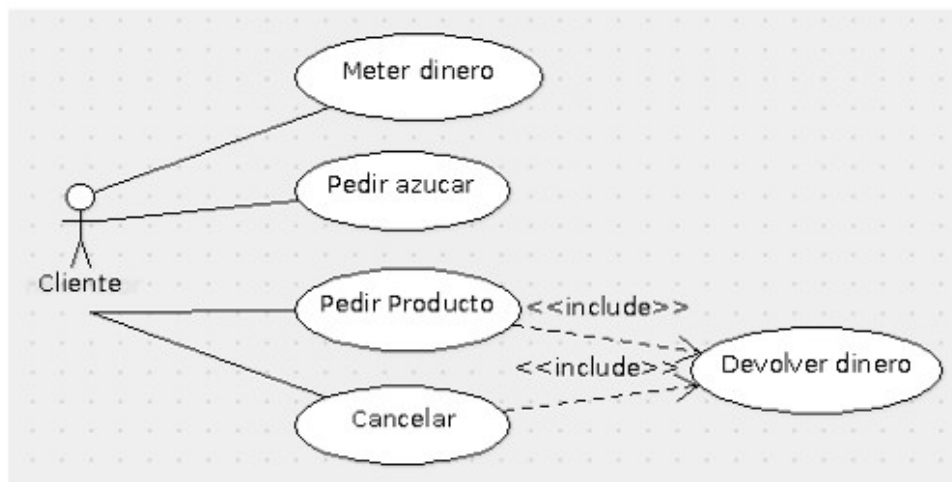
```
@startuml
left to right direction
:Cliente: as cli
usecase (Reservar vuelo) as UC1
usecase (Elegir asientos) as UC2
:Cliente VIP: as vip
cli <|-- vip
cli -- UC1
vip -- UC2
@enduml
```

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.



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

```

@startuml
left to right direction
:Cliente: as cli
usecase (Meter dinero) as UC1
usecase (Pedir azúcar) as UC2
usecase (Pedir Producto) as UC3
usecase (Cancelar) as UC4
usecase (Devolver dinero) as UC5
cli --- UC1
cli --- UC2
cli --- UC3
cli --- UC4
UC3 ..> UC5: <<include>>
UC4 ..> UC5: <<include>>
@enduml
  
```


Ejemplo: Tienda en Internet

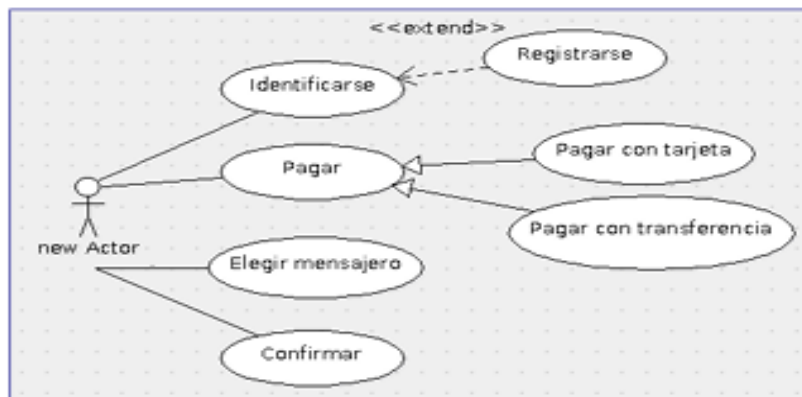
Queremos modelar el sistema de pago en una tienda web.

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.



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”.

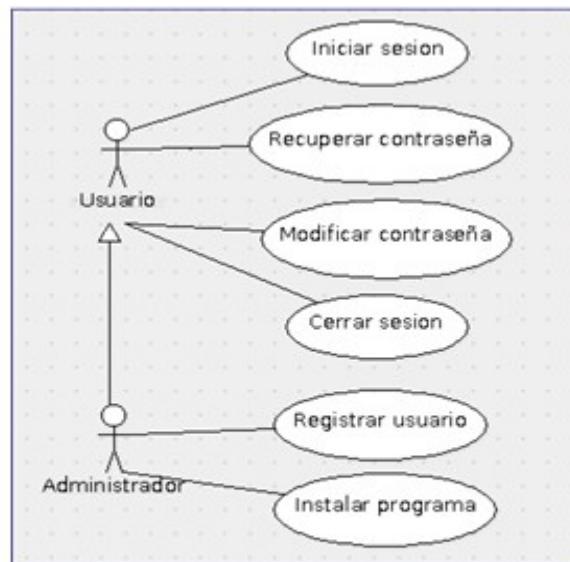
```

@startuml
left to right direction
: new Actor: as act
usecase (Identificarse) as UC1
usecase (Pagar) as UC2
usecase (Elegir mensajero) as UC3
usecase (Confirmar) as UC4
usecase (Registrarse) as UC5
act -- UC1
act -- UC2
act -- UC3
act -- UC4
UC1 <.. UC5: <<extend>>
usecase (Pagar con tarjeta) as UC6
usecase (Pagar con transferencia) as UC7
UC2 <|-- UC6
UC2 <|-- UC7
@enduml
  
```

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.



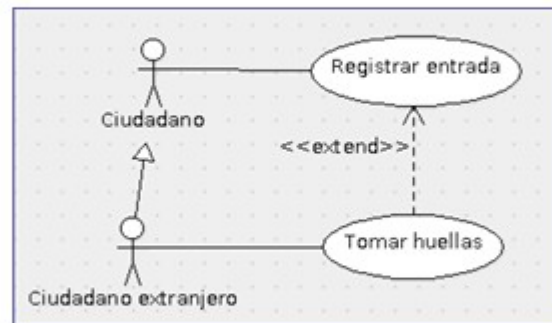
En este ejemplo nuevamente se utiliza la generalización, esta vez entre actores, para indicar que los administradores son un tipo específico de usuario.

```

@startuml
left to right direction
:Usuario: as usu
:Administrador: as admin
usecase (Iniciar sesión) as UC1
usecase (Recuperar contraseña) as UC2
usecase (Modificar contraseña) as UC3
usecase (Cerrar sesión) as UC4
usecase (Registrar usuario) as UC5
usecase (Instalar programa) as UC6
usu <|-- admin
usu -- UC1
usu -- UC2
usu -- UC3
usu -- UC4
admin -- UC5
admin -- UC6
@enduml
  
```

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.



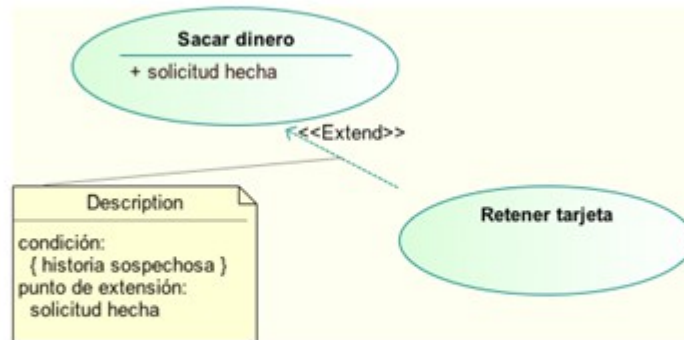
Nuevamente utilizamos la relación «*extend*» para indicar que la toma de huellas se realiza como parte del registro de entrada cuando se da la condición de que el ciudadano es extranjero, lo que expresamos añadiendo el actor “ciudadano extranjero” que interactúa con este caso de uso.

```

@startuml
left to right direction
:Ciudadano: as ciu
:Ciudadano Extranjero: as ext
usecase (Registrar entrada) as UC1
usecase (Tomar huellas) as UC2
ciu <|-- ext
ciu -- UC1
UC2 .left.> UC1: <<extend>>
ext -- UC2
@enduml
  
```

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:

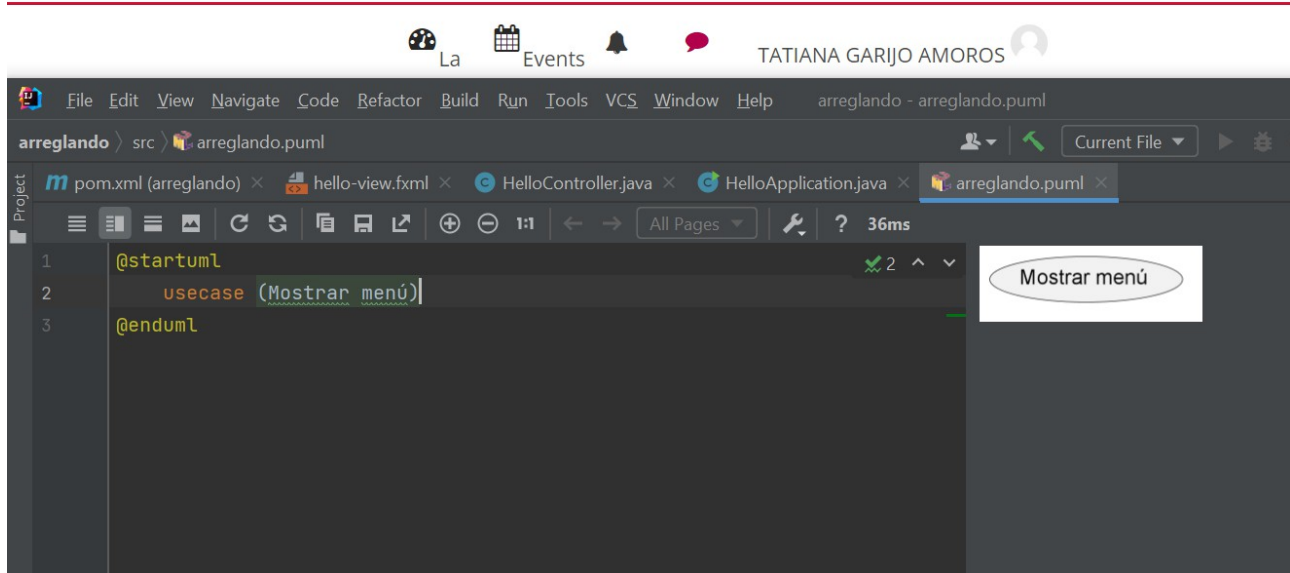
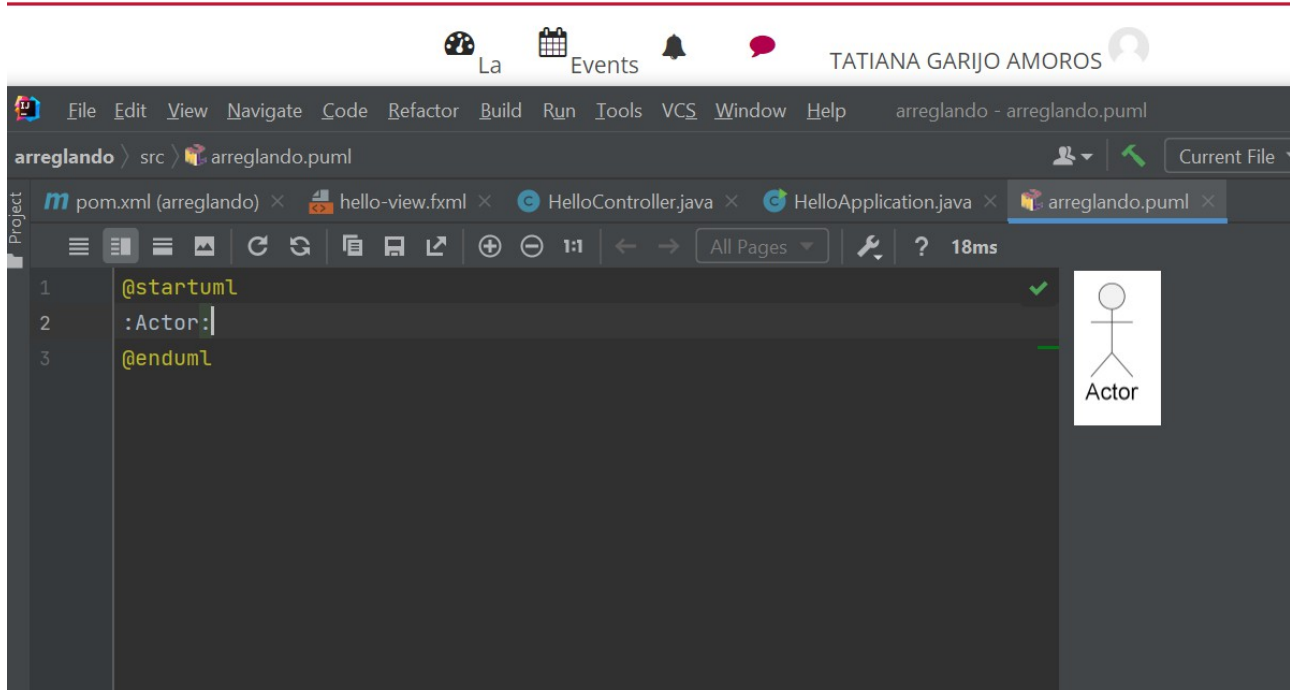


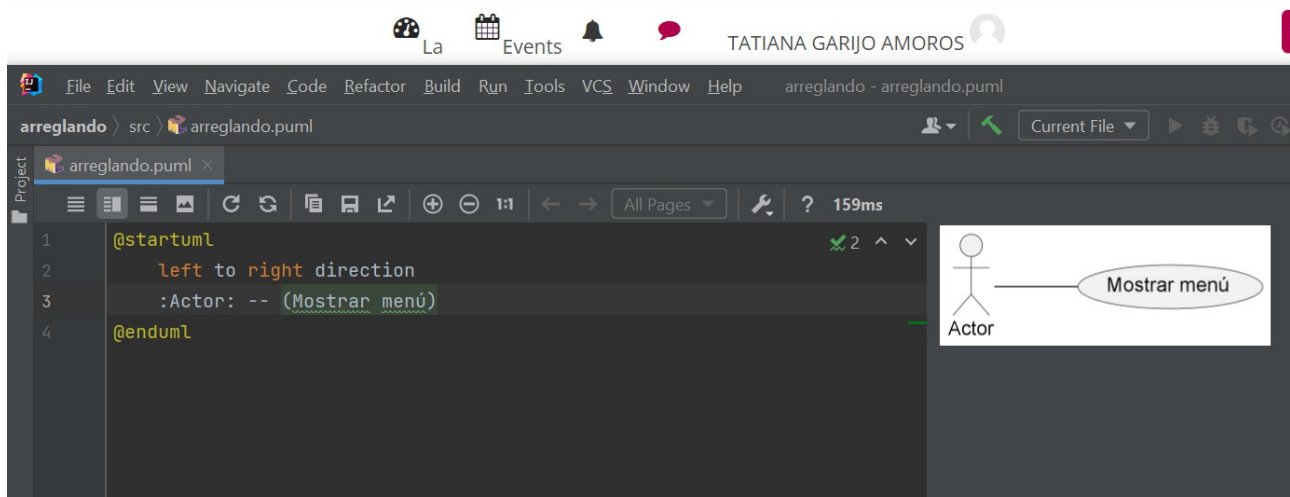
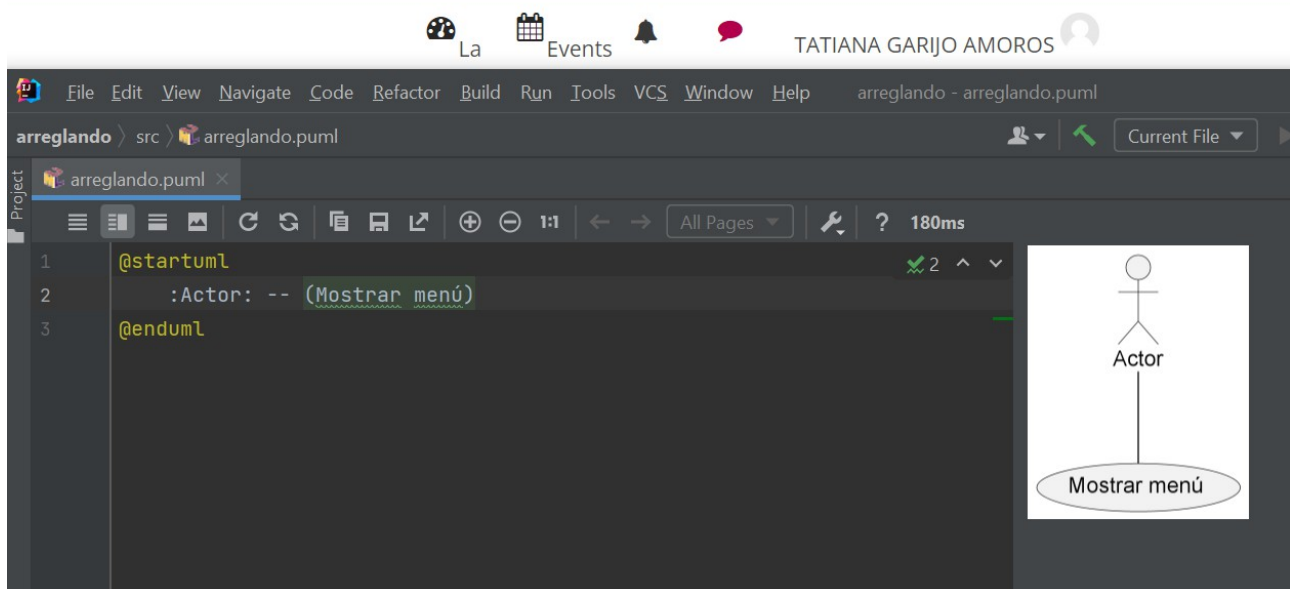
En el caso base (*Sacar dinero*) hemos añadido un punto de extensión *solicitud hecha*, que es el momento en el que se inserta el comportamiento de *Retener tarjeta*, pero sólo cuando se dé la condición *historia sospechosa*.

En Modelio, los puntos de extensión están en el apartado *Nodes* → *Extension Point*. A la flecha le hemos añadido una nota (*Common* → *Note*) indicando la condición entre llaves y el punto de extensión.

```
@startuml
skinparam usecase {
  BackgroundColor #CFF8DC
  BorderColor #4CF885
  ArrowColor #4CF885
}
left to right direction
usecase UC1 as "**Sacar dinero**"
--
+solicitud hecha"
usecase "**Retener tarjeta**" as UC2
UC1 <.. UC2 : <<Extend>>
note left of UC2
  Descripción
  --
  condición:
  { historia sospechosa}
  punto de extensión:
  solicitud hecha
end note
@enduml
```

**Dentro del proyecto he incluido una carpeta “images” donde he ido guardando los resultados de los diagramas





La Events TATIANA GARIJO AMOROS Person

arreglando - arreglando.puml

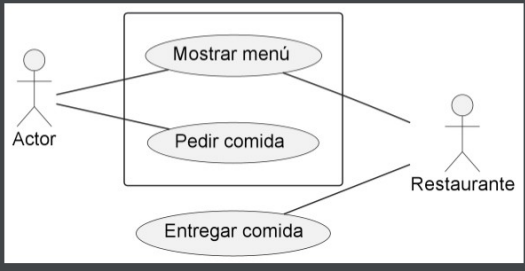
arreglando > src > arreglando.puml

Project arreglando.puml

```

1  @startuml
2      left to right direction
3      :Actor: as act
4      rectangle {
5          usecase (Mostrar menú) as UC1
6          usecase (Pedir comida) as UC2
7      }
8      usecase (Entregar comida) as UC3
9      :Restaurante: as rest
10     act -- UC1
11     act -- UC2
12     UC1 -- rest
13     UC3 -- rest
14 @enduml

```



La Events TATIANA GARIJO AMOROS Personalitza est

arreglando - arreglando.puml

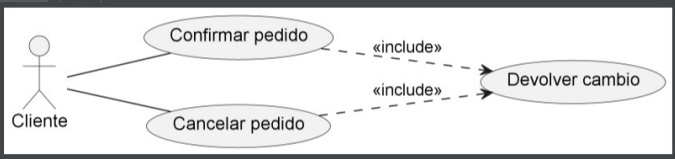
arreglando > src > arreglando.puml

Project arreglando.puml

```

1  @startuml
2      left to right direction
3      :Cliente: as cli
4      usecase (Confirmar pedido) as UC1
5      usecase (Cancelar pedido) as UC2
6      usecase (Devolver cambio) as UC3
7      cli -- UC1
8      cli -- UC2
9      UC1 ..> UC3 : <<include>>
10     UC2 ..> UC3 : <<include>>
11 @enduml

```





La Events TATIANA GARIJO AMOROS

arreglando - arreglando.puml

arreglando > src > arreglando.puml

Project arreglando.puml x

```
1 @startuml
2   left to right direction
3   usecase (Acceder) as UC1
4   usecase (Registrarse) as UC2
5   UC1 <.. UC2 : <<extend>>
6 @enduml
```

Acceder ← «extend» → Registrarse



La Events TATIANA GARIJO AMOROS

arreglando - arreglando.puml

arreglando > src > arreglando.puml

Project arreglando.puml x

```
1 @startuml
2   usecase (Pagar) as UC1
3   usecase (Pagar con tarjetas) as UC2
4   usecase (Pagar en efectivo) as UC3
5   UC1 <|-- UC2
6   UC1 <|-- UC3
7 @enduml
```

Pagar

Pagar con tarjetas

Pagar en efectivo



La Events TATIANA GARIJO AMOROS

arreglando - arreglando.puml

arreglando > src > arreglando.puml

Project arreglando.puml

```

1 @startuml
2   left to right direction
3   :Cliente: as cli
4   usecase (Reservar vuelo) as UC1
5   usecase (Elegir asientos) as UC2
6   :Cliente VIP: as vip
7   cli <- vip
8   cli -- UC1
9   vip -- UC2
10 @enduml

```

128ms



La Events TATIANA GARIJO AMOROS Personalitza est

arreglando - arreglando.puml

arreglando > src > arreglando.puml

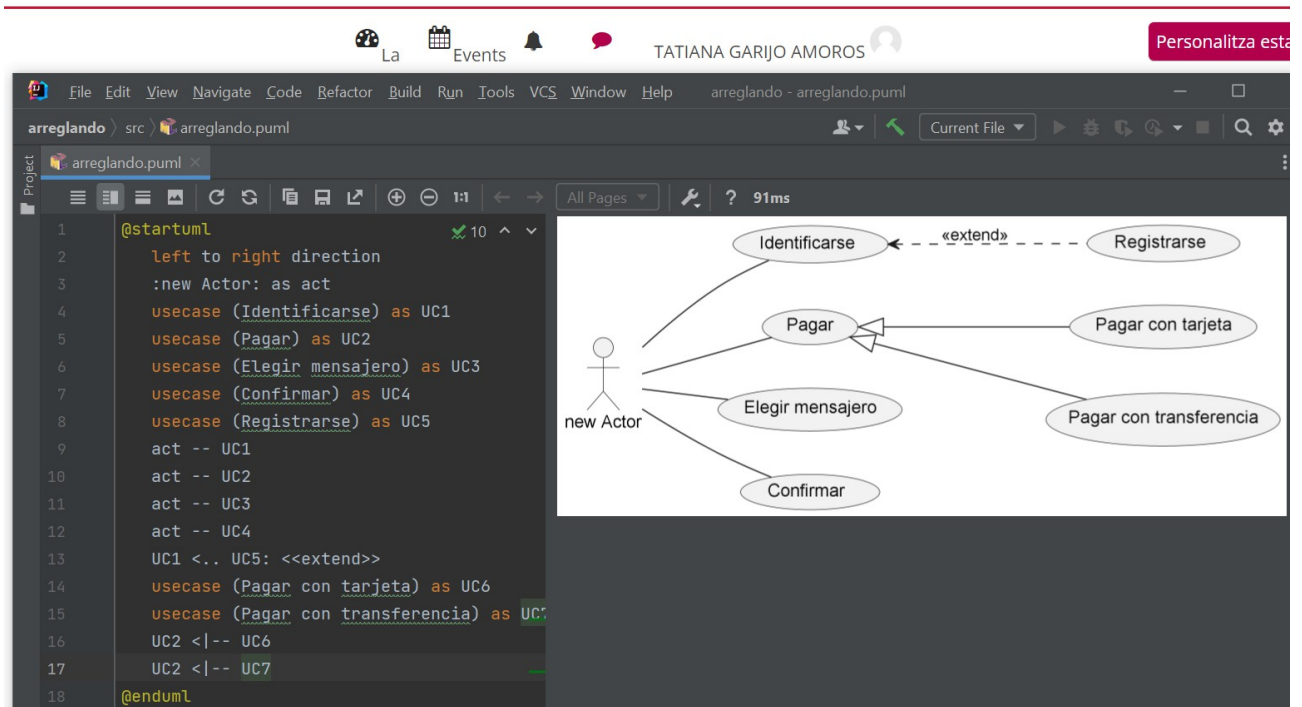
Project arreglando.puml

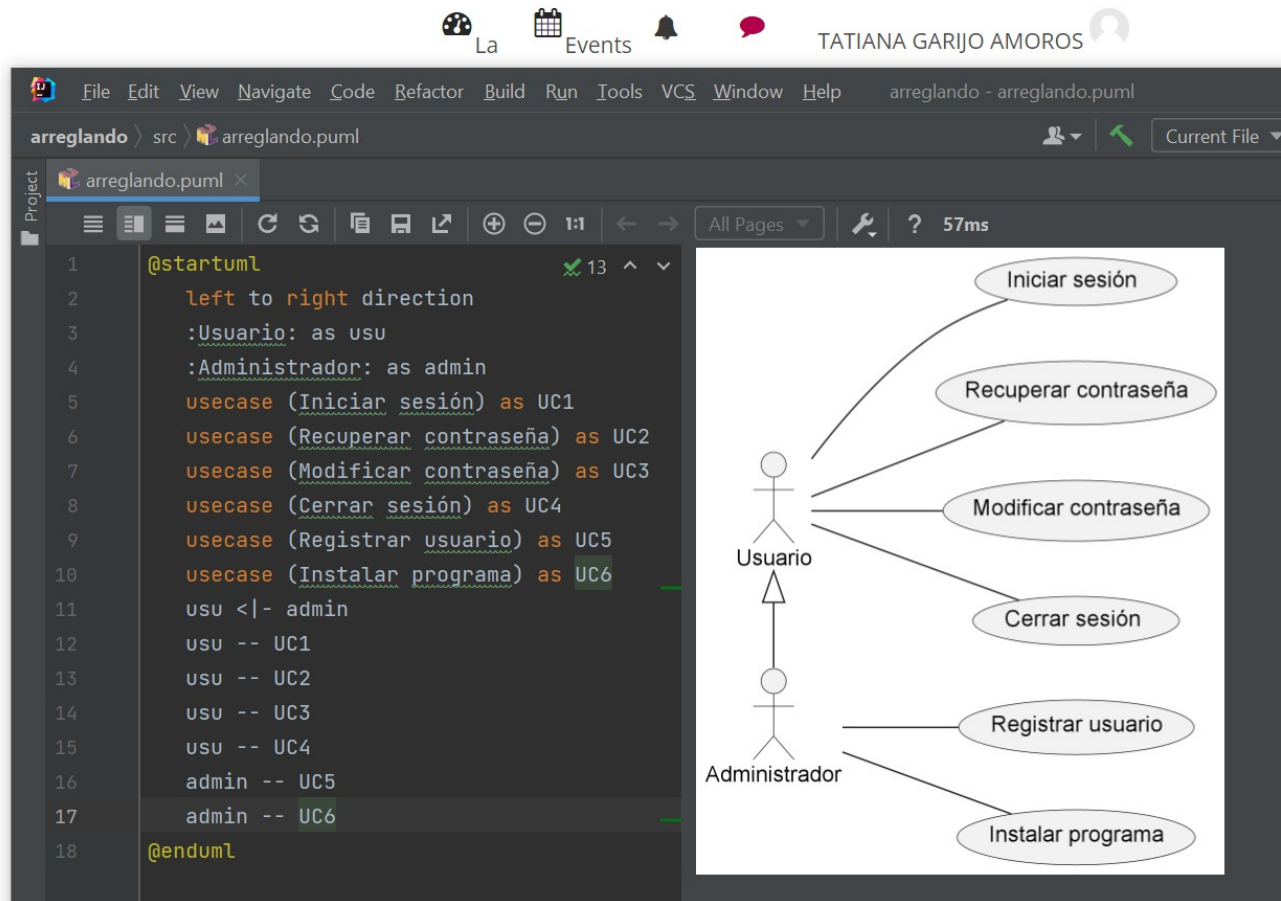
```

1 @startuml
2   left to right direction
3   :Cliente: as cli
4   usecase (Meter dinero) as UC1
5   usecase (Pedir azúcar) as UC2
6   usecase (Pedir Producto) as UC3
7   usecase (Cancelar) as UC4
8   usecase (Devolver dinero) as UC5
9   cli --- UC1
10  cli --- UC2
11  cli --- UC3
12  cli --- UC4
13  UC3 ..> UC5: «include»
14  UC4 ..> UC5: «include»
15 @enduml

```

65ms







La Events TATIANA GARIJO AMOROS

arreglando - arreglando.puml

arreglando > src > arreglando.puml

Project arreglando.puml

```
1 @startuml
2   left to right direction
3   :Ciudadano: as ciu
4   :Ciudadano Extranjero: as ext
5   usecase (Registrar entrada) as UC1
6   usecase (Tomar huellas) as UC2
7   ciu <|-- ext
8   ciu -- UC1
9   UC2 .left> UC1: <<extend>>
10  ext -- UC2
11 @enduml
```



La Events TATIANA GARIJO AMOROS

arreglando - arreglando.puml

arreglando > src > arreglando.puml

Project arreglando.puml

```
1 @startuml
2   skinparam usecase {
3     BackgroundColor #CFF8DC
4     BorderColor #4CF885
5     ArrowColor #4CF885
6   }
7   left to right direction
8   usecase UC1 as "**Sacar dinero**"
9   --
10  +solicitud hecha"
11  usecase "**Retener tarjeta**" as UC2
12  UC1 <.. UC2 : <<Extend>>
13  note left of UC2
14  Descripción
15  --
16  condición:
17  { historia sospechosa}
18  punto de extensión:
19  solicitud hecha
20  end note
21 @enduml
```

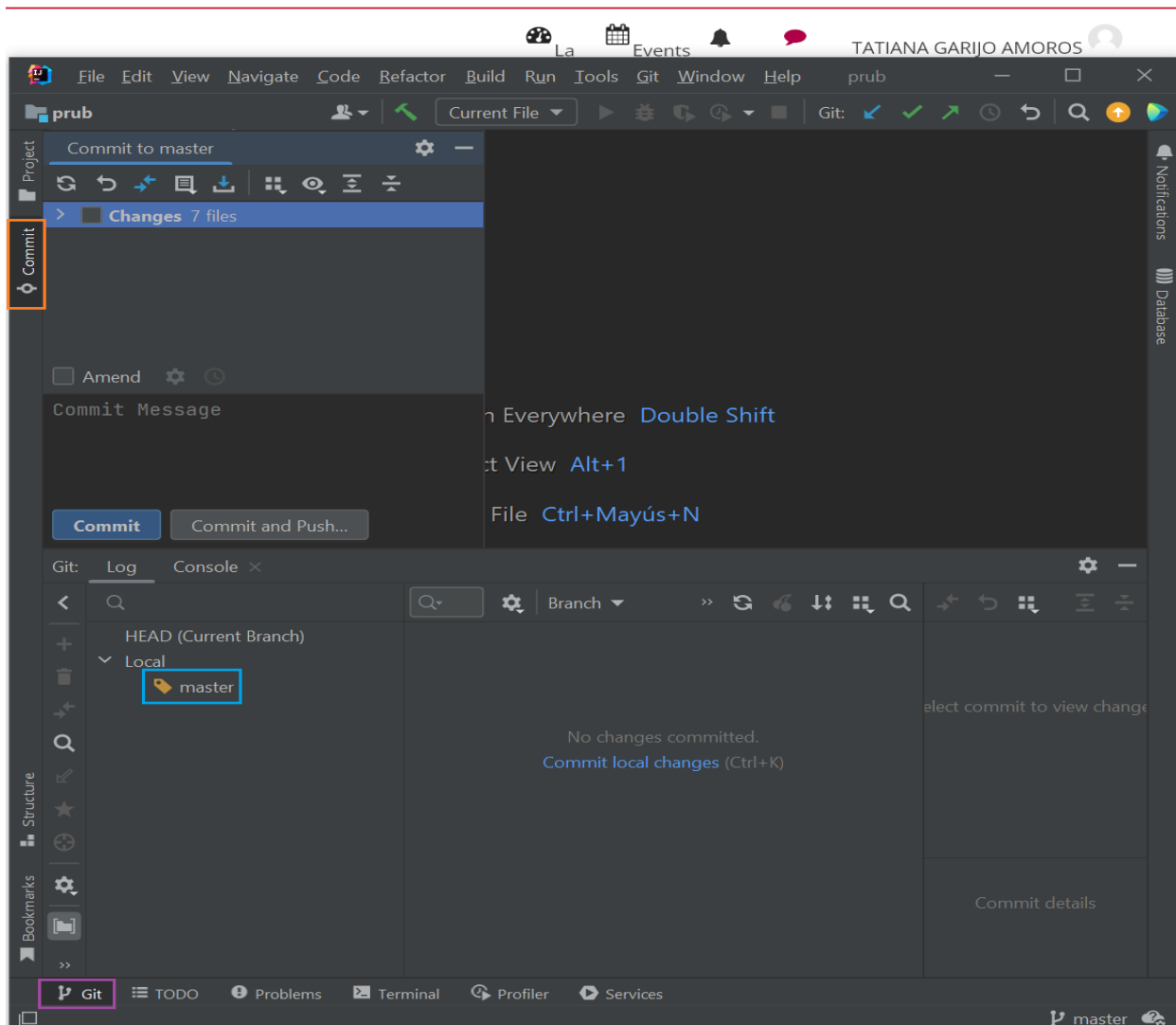
Sacar dinero
+solicitud hecha

«Extend»

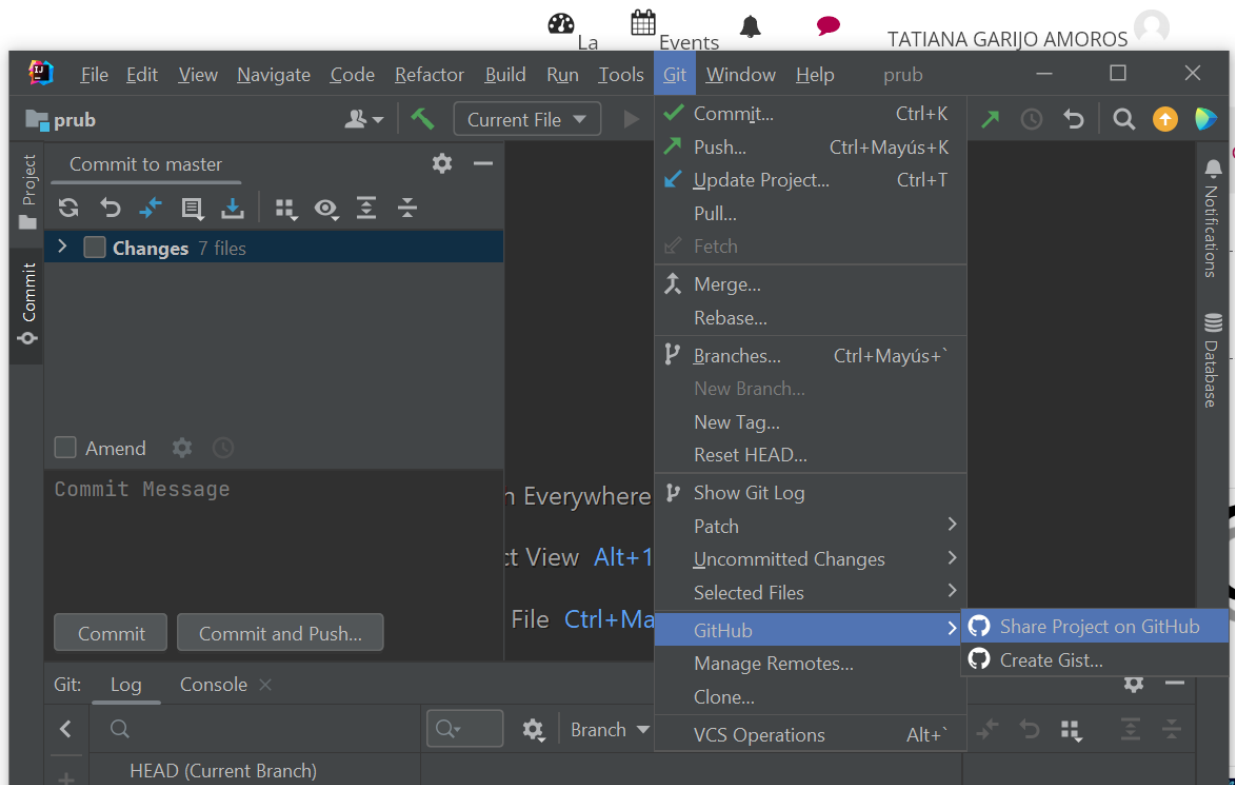
Retener tarjeta

Descripción
condición:
{ historia sospechosa}
punto de extensión:
solicitud hecha

- **VSC**
- Al crear el proyecto he seleccionado la opción de crear un repositorio Git
- Durante el proceso realizar commit, haciendo uso del **panel lateral izquierdo 'commit'**
 - Seleccionar los archivos modificados y los que se vayan a incluir en el commit
 - Añadir mensaje del commit
 - Hacer clic sobre el botón commit
-



- Subir repositorio a Github



- Hacer clic sobre el menú superior 'Git' → GitHub → Share Project on GitHub

- Dar nombre al repositorio y hacer clic sobre el botón 'Share'

