Eduardo De la Cotera Martínez 1ºDAW

Entornos de desarrollo

Tema 11 TDD

Práctica Diagramas UML Clases

Memoria

https://github.com/Bufoncete/UMLClasesEduardoDLCM

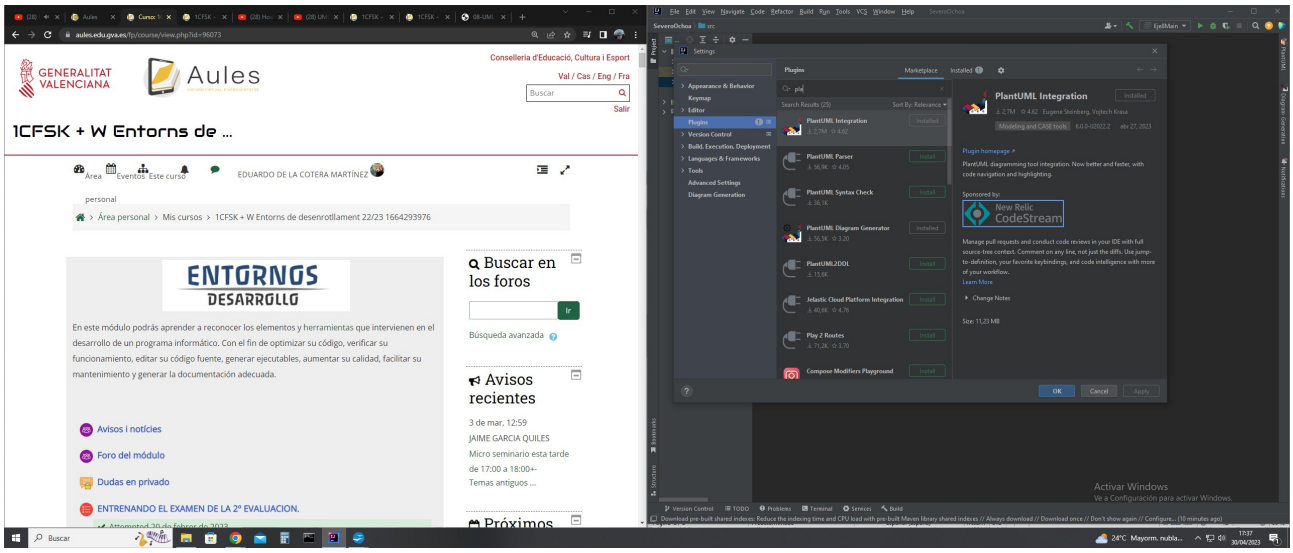
A continuación presento la memora de la práctica por puntos, siguiendo los diferentes pasos del enunciado.

Para acreditar la autenticidad del autor de las capturas, añadiré junto a las mismas una captura de mi cuenta de Aules abierta para evitar confusiones.

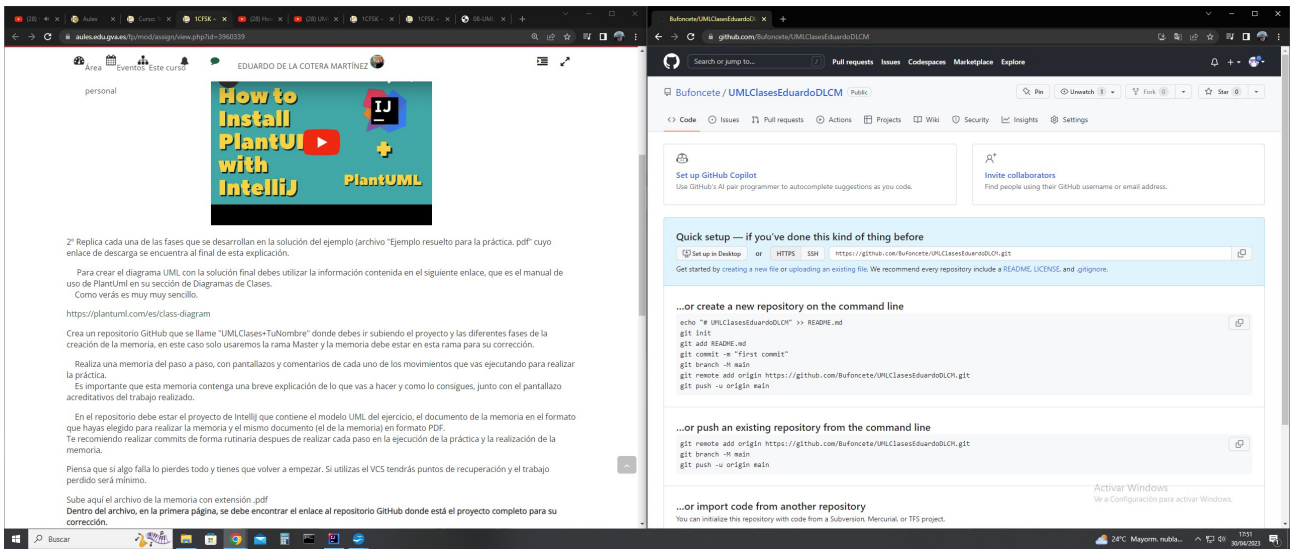
1. Instalar PlanUml en IntelliJ

En mi caso ya lo tengo instalado de otra práctica.

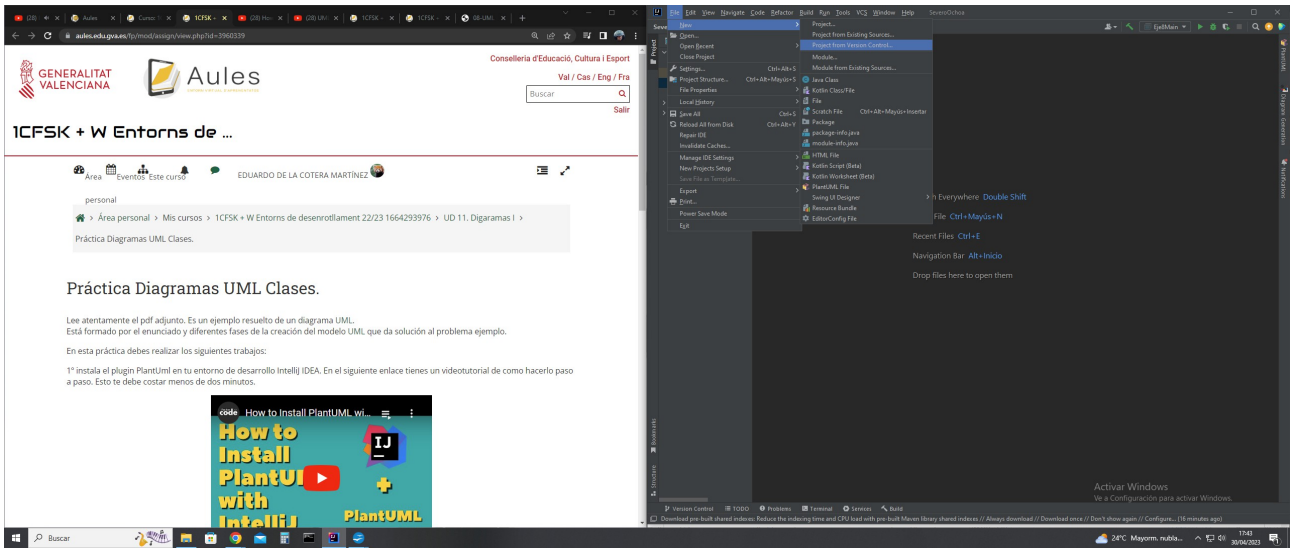
Capturo pantalla de esto.

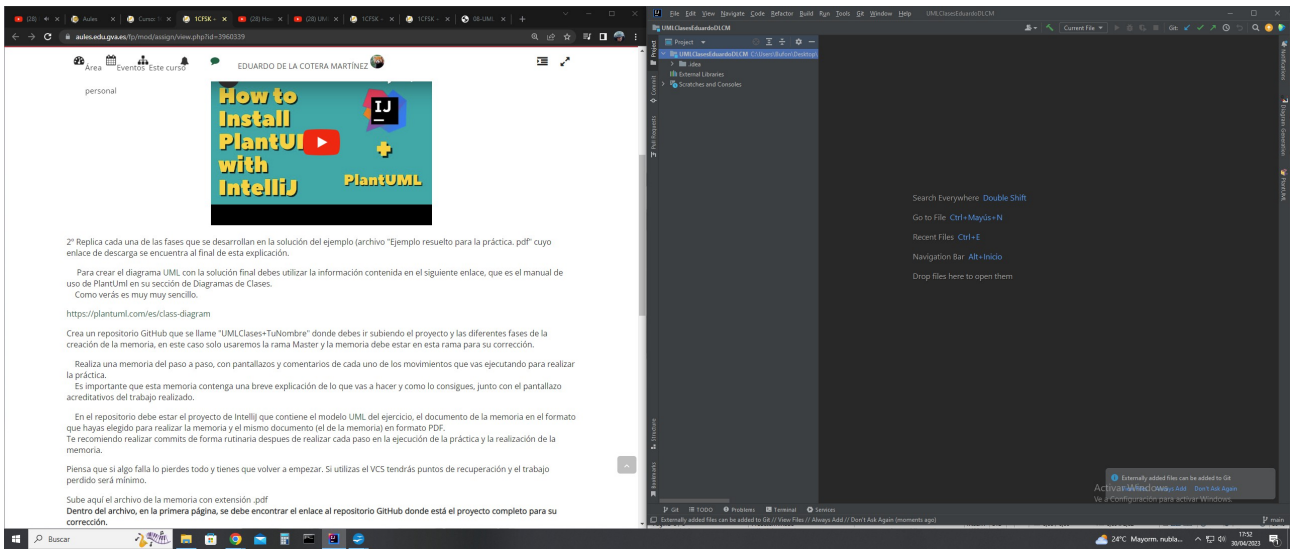


2. Creo un repositorio en GitHub.



A partir del repositorio remoto lo clonamos en IntelliJ





Ya podemos trabajar desde IntelliJ, hacer commits y desarrollar los requerimientos del ejercicio.

3,Comienzo del ejercicio

La Asociación de Antiguos Alumnos de la UOC nos ha pedido si podemos ayudarles a confeccionar un programa que les permita gestionar a sus asociados, eventos y demás elementos relacionados.

Los asociados se pueden dividir en miembros numerarios y en miembros de la junta directiva, que es elegida por votación en una asamblea general cada cuatro años. La única diferencia entre ellos es que los miembros de la junta directiva son convocados a las reuniones de junta y los demás miembros no, pero el resto de actividades que se realizan están abiertas a todos los miembros de la asociación.

La convocatoria de un evento se realiza por correo electrónico a todos los miembros activos en el momento del envío, recibiendo un enlace para aceptar su participación. En todos los eventos, la aceptación de los asistentes se realiza por orden de llegada ya que, en algún caso, se puede dar que el número de asistentes sea limitado, como en las conferencias.

En la convocatoria, también aparece información sobre el lugar que en muchos casos se repite, por lo que nos han dicho que quieren almacenar los datos para futuros usos.

Solución:

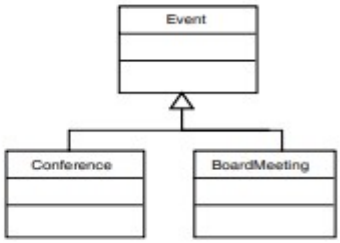
1) Identificación de las clases
• Miembro (o miembro numerario) (Member)
• Miembro de la junta directiva (BoardMember)
• Evento (Event)
• Conferencia (Conference)
• Reunión de la junta directiva (BoardMeeting)
• Localización (Location)

Adicionalmente, se ha añadido la clase Persona (Person) para poder identificar también a los conferenciantes, ya que podría darse el caso de que éstos no fueran miembros de la asociación.

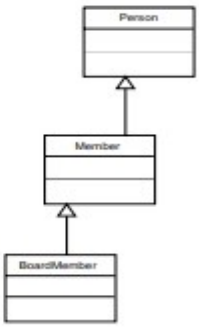
//La primera parte del ejercicio ya la tenemos resuelta en el enunciado.

2) Creación del modelo de datos

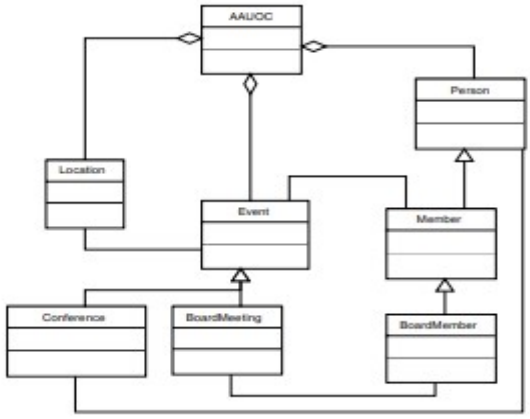
Para crear el modelo de datos, se ha detectado la existencia de una jerarquía de herencia cuya superclase es el evento y, según la descripción del problema, tiene únicamente dos subclases, que son las conferencias y las reuniones de la Practica UD 11. Diagramas UML, Clases. junta directiva. En este problema, se ha descartado la inclusión de una clase que represente a los eventos con restricciones en el número de asistentes. En caso de ampliarse la tipología de eventos, se debería considerar dicho punto.



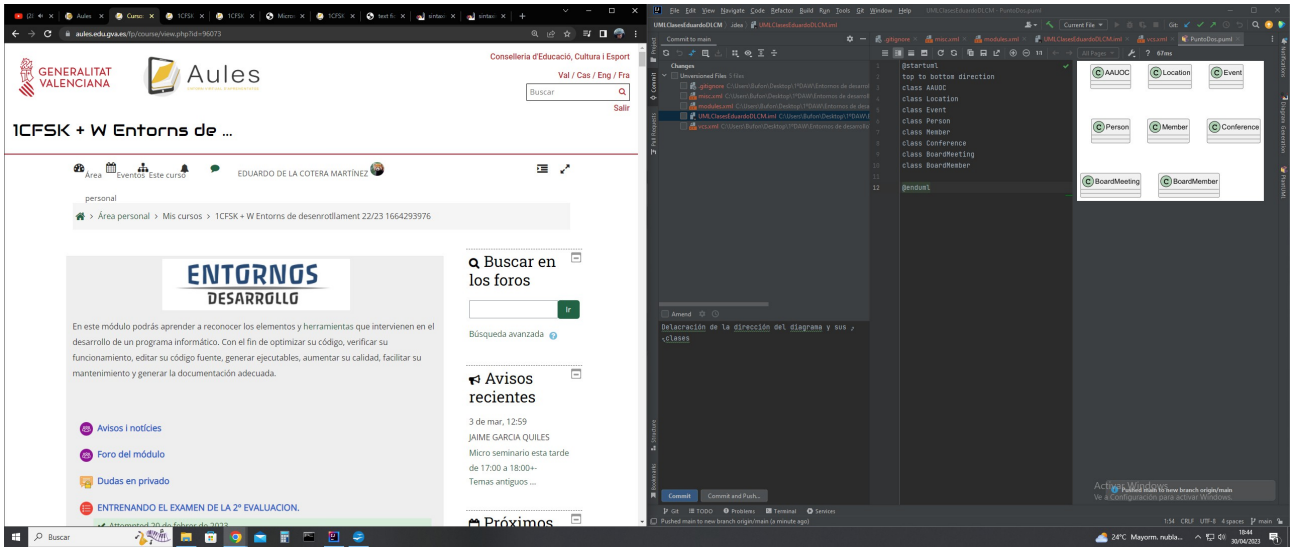
Al mismo tiempo, existe la siguiente jerarquía entre los miembros de la asociación:



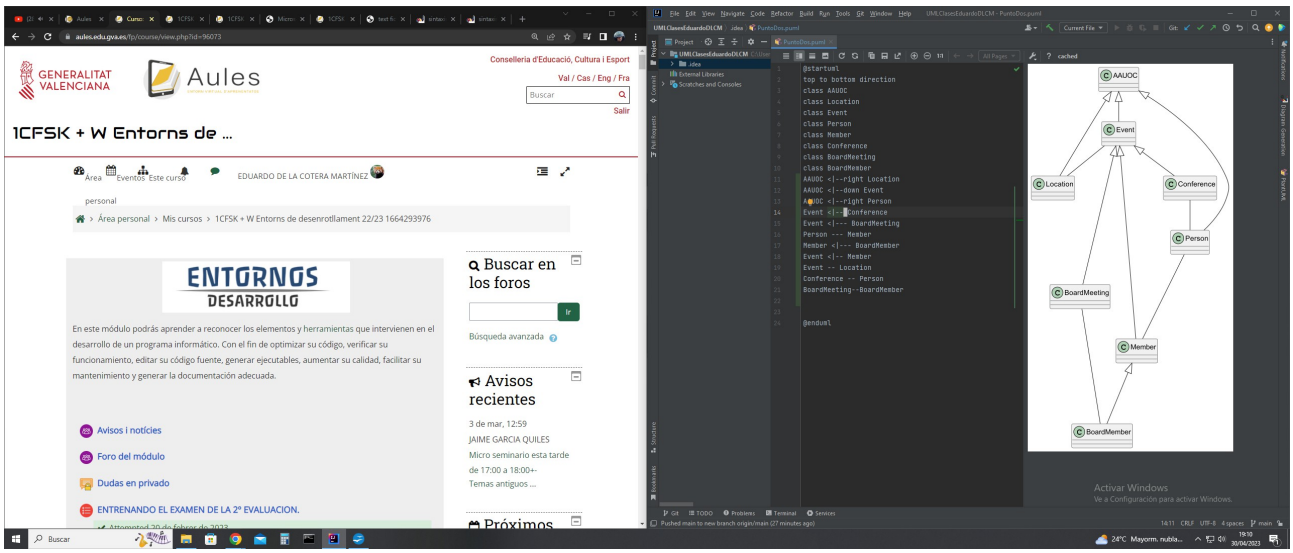
Además, tenemos las clases Localización (*Location*) y Asociación (AAUOC), que se relacionan con el resto de clases del siguiente modo:



Declaración de la dirección del diagrama y sus clases



Se establecen las relaciones entre las clases siguiendo el esquema proporcionado por el enunciado.



//Código

```
@startuml
top to bottom direction
class AAUOC
class Location
class Event
class Person
class Member
class Conference
class BoardMeeting
class BoardMember
AAUOC <|--right Location
AAUOC <|--down Event
AAUOC <|--right Person
Event <|-- Conference
Event <|--- BoardMeeting
Person --- Member
Member <|--- BoardMember
Event <|-- Member
Event -- Location
Conference -- Person
BoardMeeting--BoardMember
@enduml
```

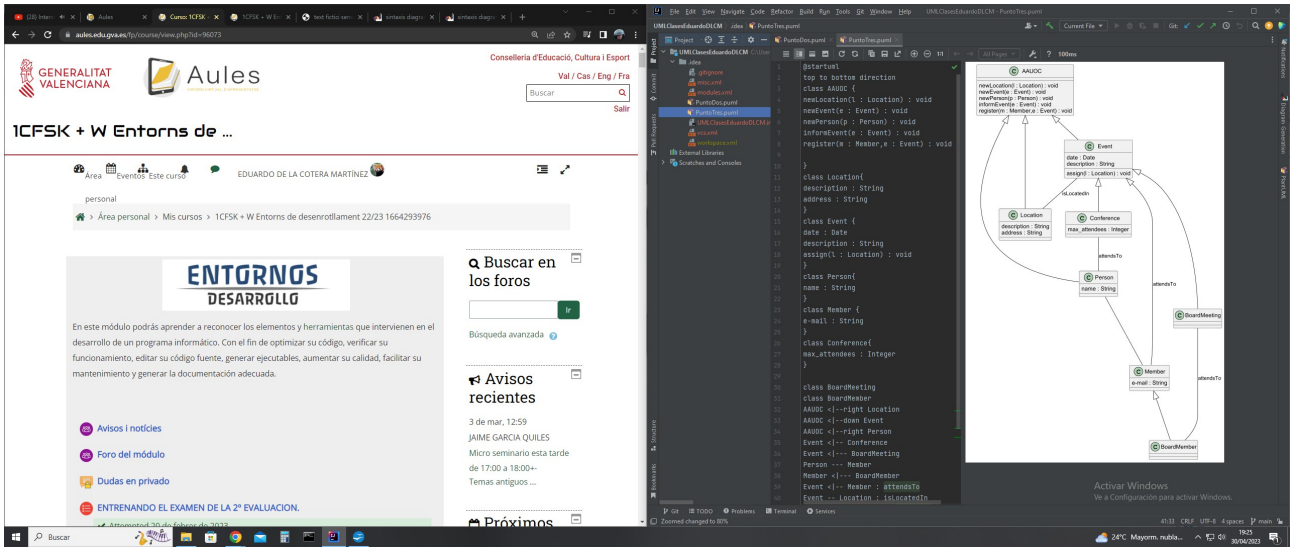
3) Inclusión de atributos y métodos

Una vez creado el modelo de datos, completamos las clases con sus atributos y los métodos más relevantes (quedan fuera de este diagrama los métodos getters y setters, así como los métodos constructores de cada clase).

La asociación necesitará un conjunto de métodos para añadir nuevos eventos, personas y localizaciones al sistema (métodos newX de la clase AAUOC), así como también un método para informar a los miembros de la convocatoria de un evento (método informEvent). Al mismo tiempo, se dice que los usuarios necesitarán confirmar la asistencia a los eventos (método register, que deberá almacenar los asistentes por orden y controlar el número

máximo de éstos si fuera necesario).

Añado los datos del enunciado en nuevo documento UML.



//Código completo

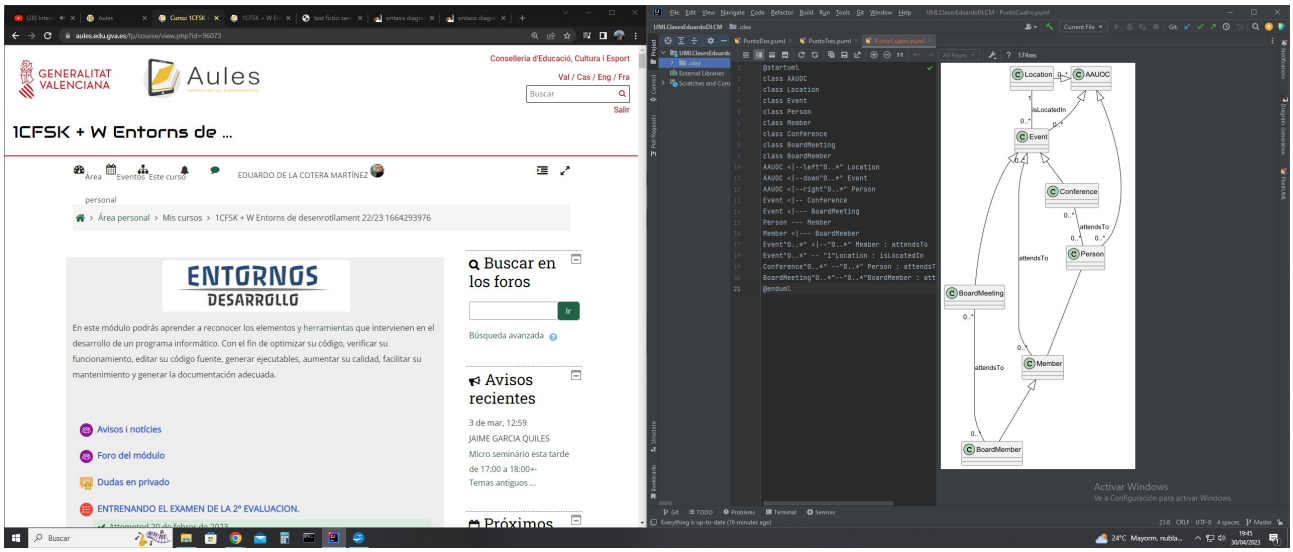
```
@startuml
top to bottom direction
class AAUOC {
newLocation(l : Location) : void
newEvent(e : Event) : void
newPerson(p : Person) : void
informEvent(e : Event) : void
register(m : Member,e : Event) : void
}
class Location{
description : String
address : String
}
class Event {
date : Date
description : String
assign(l : Location) : void
}
class Person{
name : String
}
class Member {
e-mail : String
}
class Conference{
max_attendees : Integer
}
class BoardMeeting
class BoardMember
AAUOC <|--right Location
AAUOC <|--down Event
AAUOC <|--right Person
Event <|-- Conference
Event <|--- BoardMeeting
Person --- Member
Member <|--- BoardMember
Event <|-- Member : attendsTo
```

```
Event -- Location : isLocatedIn
Conference -- Person : attendsTo
BoardMeeting--BoardMember : attendsTo
@enduml
```

4) Inclusión de la cardinalidad y navegabilidad de las relaciones

En el siguiente diagrama, se han eliminado los métodos e incluido las navegabilidades (en este caso, todas son bidireccionales, debido a que no se nos ha comunicado ningún tipo de relación y/o acceso a la información de una clase a otra), y las cardinalidades de dichas relaciones.

Creo el tercer documento para realizar el UML de las relaciones de las clases junto sus cardinalidades.



//Código

```
@startuml
class AAUOC
class Location
class Event
class Person
class Member
class Conference
class BoardMeeting
class BoardMember
AAUOC <|--left"0..*" Location
AAUOC <|--down"0..*" Event
AAUOC <|--right"0..*" Person
Event <|-- Conference
Event <|--- BoardMeeting
Person --- Member
Member <|--- BoardMember
Event"0..*" <|--"0..*" Member : attendsTo
Event"0..*" -- "1"Location : isLocatedIn
Conference"0..*" --"0..*" Person : attendsTo
BoardMeeting"0..*"--"0..*"BoardMember : attendsTo
@enduml
```

Último commit del ejercicio