UML

DIAGRAMAS DE CLASES

CON

PLANTUML

EN

IntelliJ

ENTORNOS DE DESARROLLO
DAW 1W 22/23

ALBERT PÉREZ BALEYTO
https://github.com/AlbertBeto/UMLClasesAlbertPB.git

Se nos presenta un ejercicio en el que una asociación de antiguos alumnos de la **UOC** nos solicita un software para la gestión de asociados, eventos y demás temas relacionados.

Se nos solicita que creemos un diagrama de Clases usando UML del software solicitado, para ello usemos la aplicación **IntelliJ** y el plugin especializado de UML **Plantuml**.

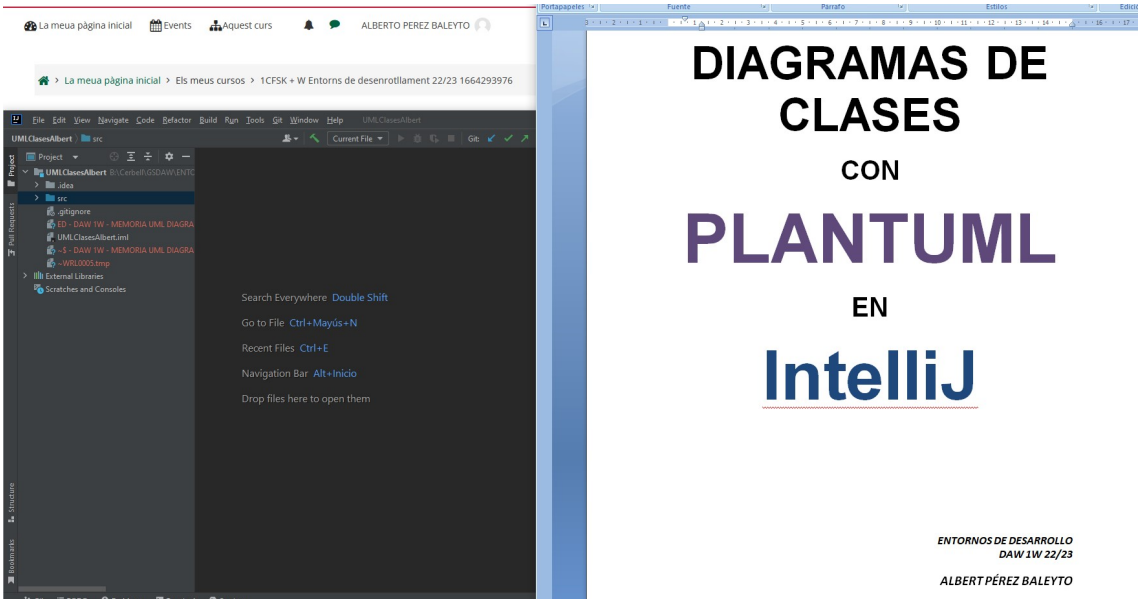
Todo el ejercicio y los pasos realizados deben quedar registrados en una memoria la cual, junto al código escrito, deben estar almacenados en un repositorio de **github**.

PREPARACIÓN

- Creo el repositorio de github.

  https://github.com/AlbertBeto/UMLClasesAlbertPB.git

- Creo el archivo .docx de la memoria.



- Creo un proyecto Java en IntelliJ con el nombre UMLClasesAlbert, incorporo el presente archivo .docx en su raíz y vinculo el proyecto al repositorio.

PLANTEAMIENTO

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.

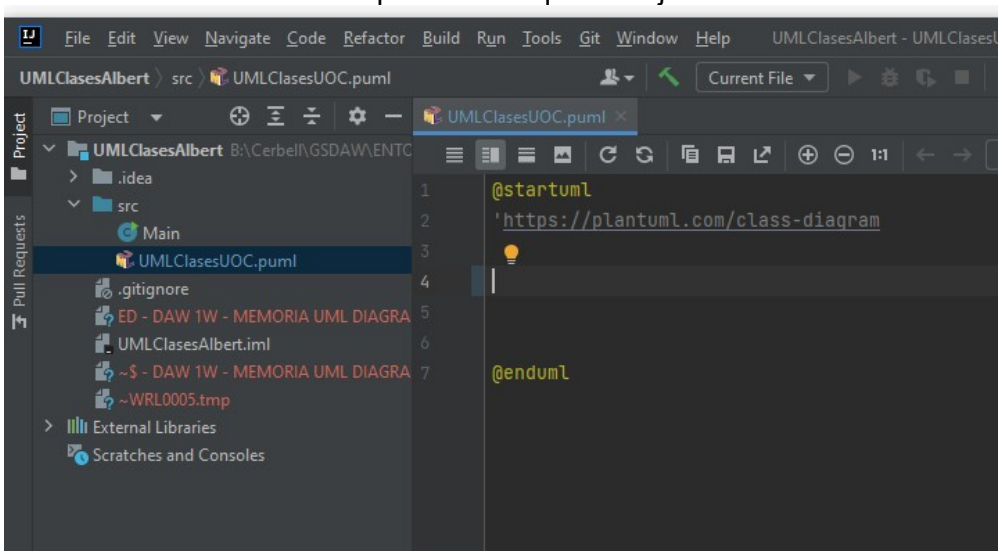
En la convocatoria, también aparece información sobre el lugar que en muchos casos se repite, por lo que nos han dicho que quieren almacenar los datos para futuros usos.

EJECUCIÓN

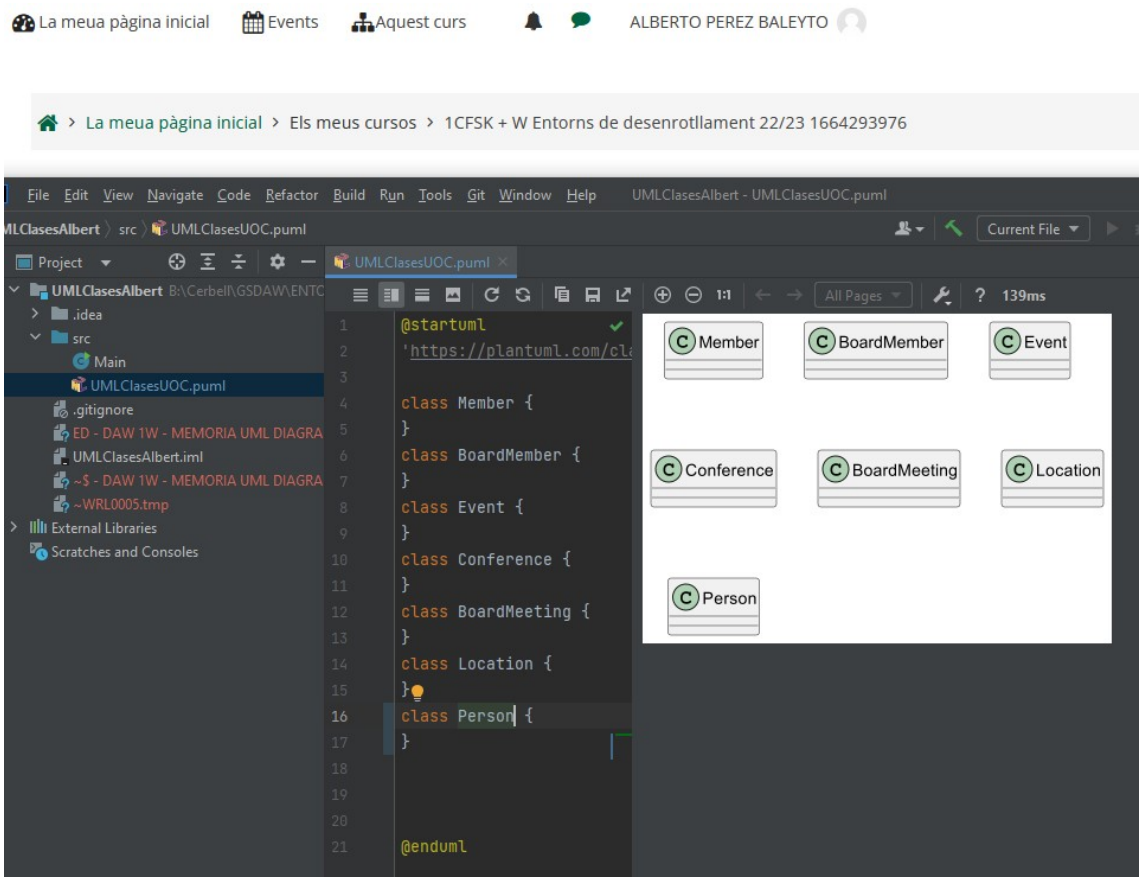
1) **Identificación de las clases**

   • 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.
   Creo en el IntelliJ el archivo .puml con el que trabajaremos el UML.

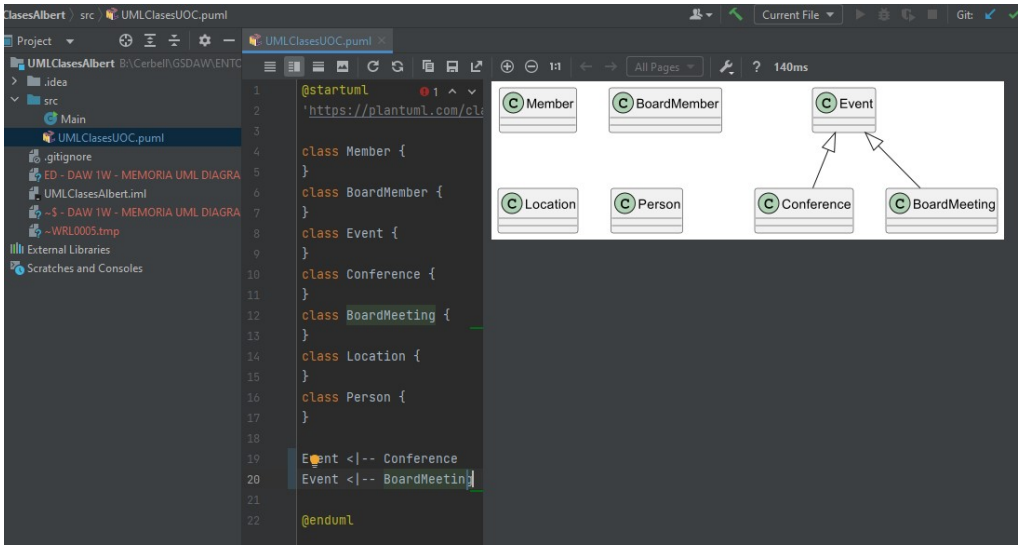


Procedo a crear las clases.

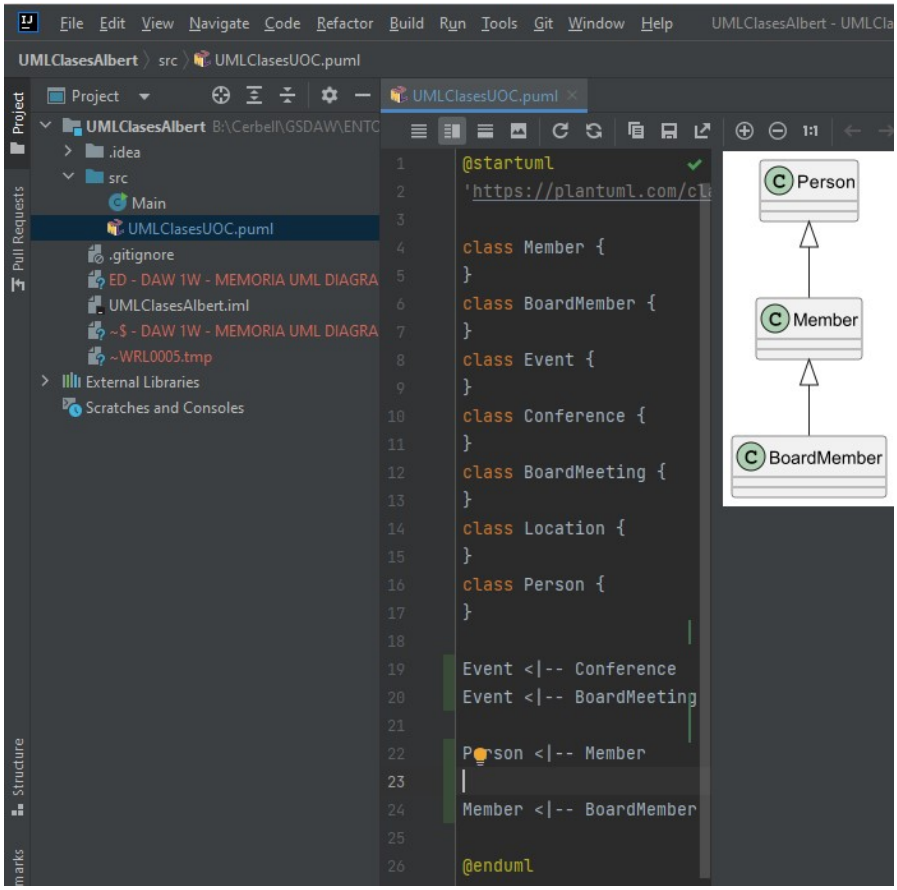


2) Creación del modelo de datos

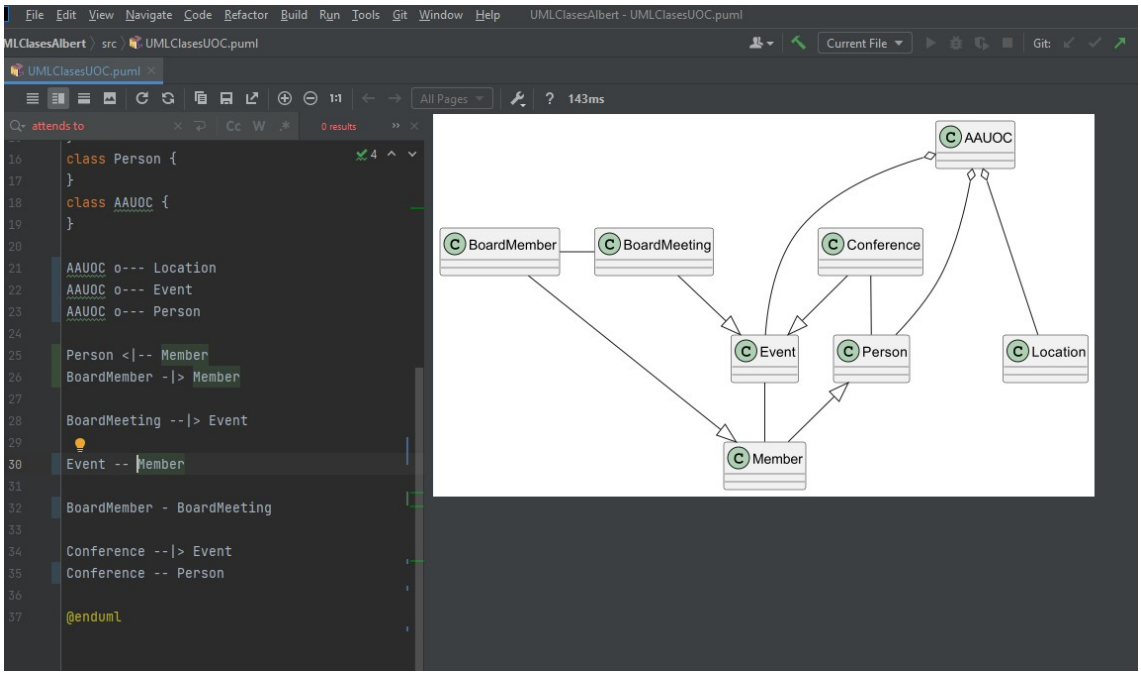
Para crear el modelo de datos, se ha detectado la existencia de una jerarquía de herencia cuya superclase es el evento y, según la descripción del problema, tiene únicamente dos subclases, que son las conferencias y las reuniones de la junta directiva. En este problema, se ha descartado la inclusión 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:

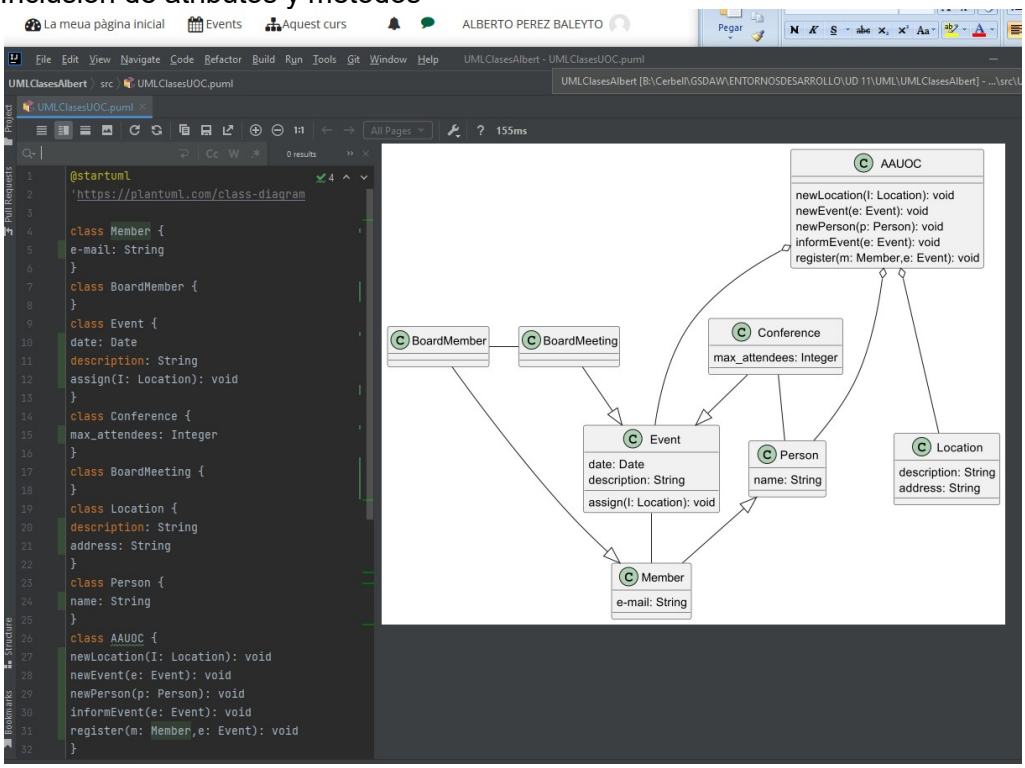


3) Inclusión de atributos y métodos

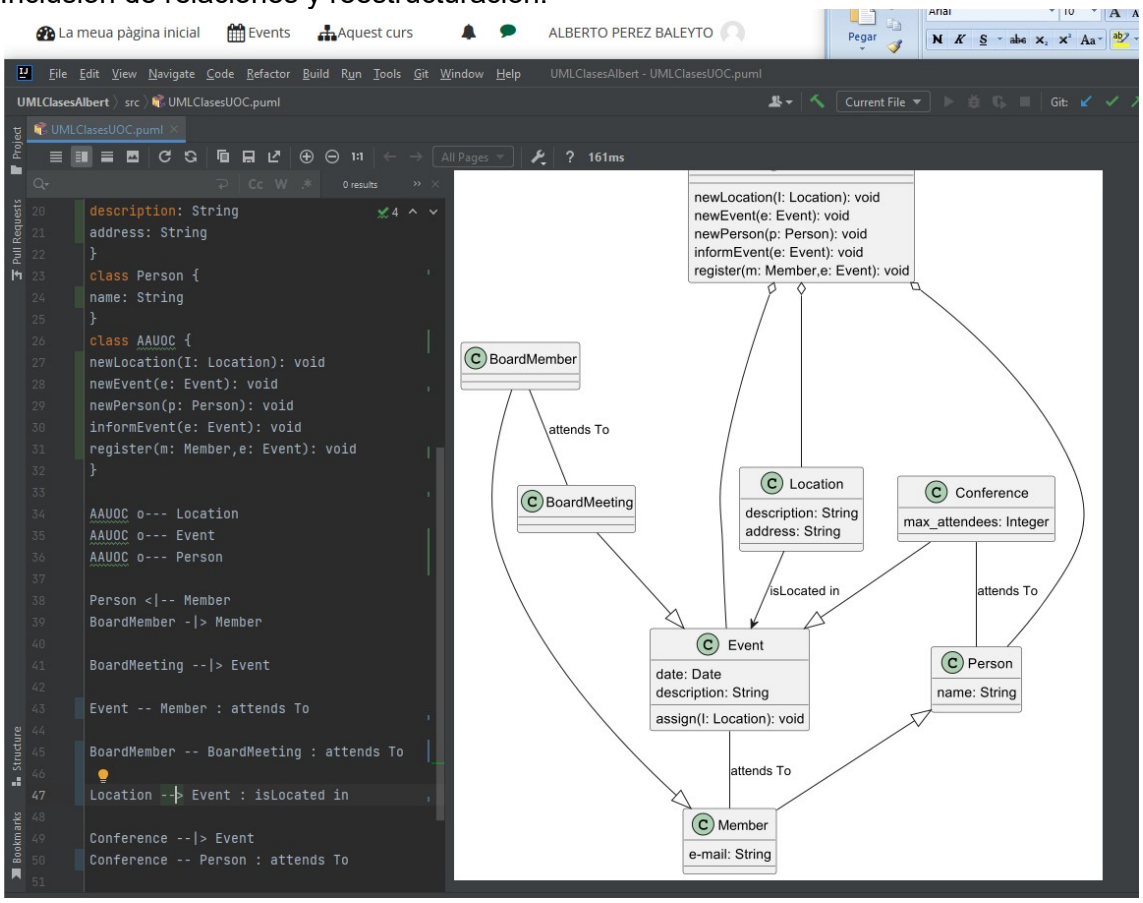
Una vez creado el modelo de datos, completamos las clases con sus atributos y los métodos más relevantes (quedan fuera de este diagrama los métodos getters y setters, así como los métodos constructores de cada clase).

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).

Inclusión de atributos y métodos

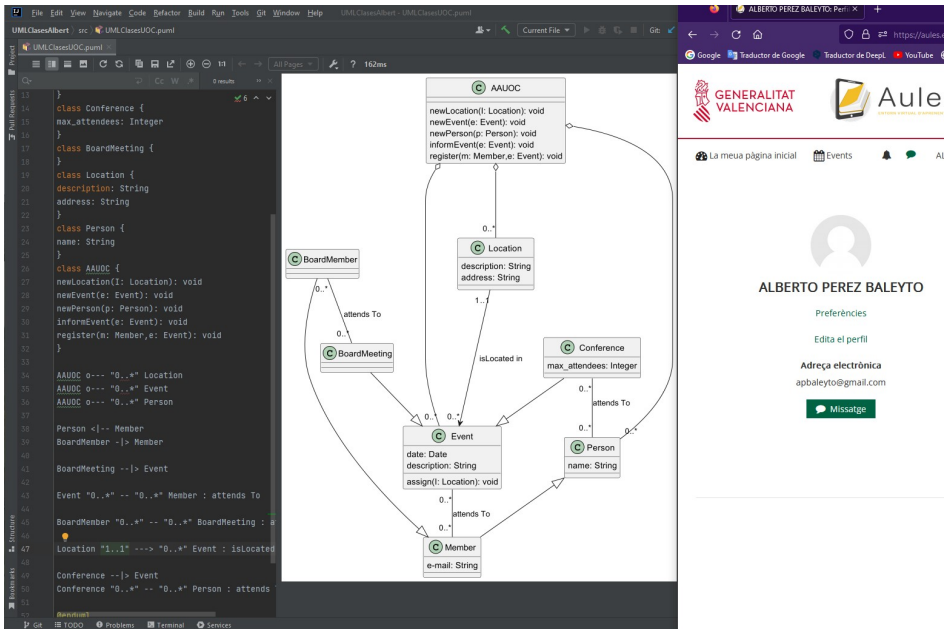


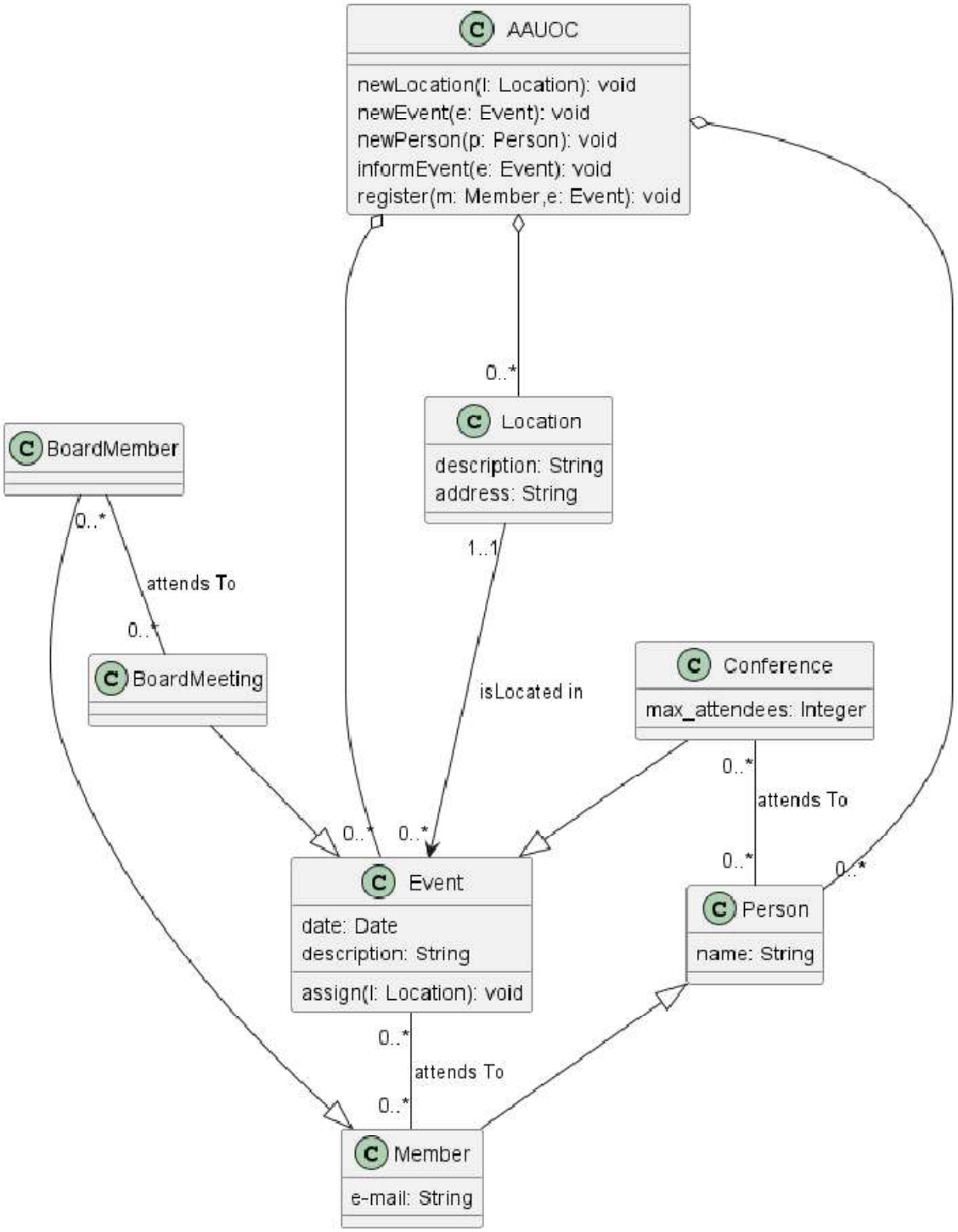
Inclusión de relaciones y reestructuración.



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.





CONCLUSIONES

PantUML funciona muy bien excepto en acomodar la estructura. En mi experiencia no es intuitivo ni agil ya que no he podido montar la estructura como yo he querido si no que he tenido que adaptarme a su sistema de incluir mas guiones en el codigo para los niveles o grados de separación aunque con practica seguro que podría acomodar mucho mejor la información en las estructuras.