**Lógica del Sistema**

Voy a tener una clase denomida ComfyChair, que es la encargada de crear las Conferencias y los Usuarios.

La clase Conferencias es la encargada de crear las Sesiones y llevar un listado de las sesiones.

La clase Sesiones es la encargada de recibir los artículos enviados por los autores.

La Sesión se crea en el estado ‘Recepción’.

El Autor crea un artículo. El artículo se crea en un estado ‘enProceso’, quiere decir que no está aprobado, ni rechazado aún.

La Sesión valida que el artículo sea apto de recibirlo.

Validaciones:

\*Que no se haya definido un Autor para recibir las notificaciones.

\*Si la fecha actual es mayor al deadline de la recepción, se rechaza.

\*Si la Sesión es Regular y el tipo de artículo no es Regular, se rechaza.

\*Si la Sesión es Posters y el tipo de artículo no es Posters, se rechaza.

\*Si la Sesión es Workshop y el tipo de artículo no es ni Posters, ni Regular, se rechaza.

\*Si el tipo de artículo es Regular y no tiene Abstract o es superior a 300 caracteres o si no tiene Título o si no tiene Archivo adjunto, se rechaza.

\*Si el tipo de artículo es Posters y no tiene Título o si no tiene Archivo adjunto, o si no tiene los Fuentes, se rechaza.

\*Si no se definen los autores del artículo, se rechaza.

Si pasa todas las validaciones se crea un archivo por cada sesión y el nombre del archivo es el nombre del tema de la sesión .json donde almaceno todos los datos del artículo. Como no trabajo con js, ni con apis encontré esta manera de almacenar los datos de los artículos. Si bien a los objetos no se les define un id, a modo de que se me haga más sencillo de buscarlo, rescatarlo, realizar bidds; al artículo le asigné un atributo id.

Los artículos que no pasan las validaciones, no se graban en estos archivos.

Cuando un artículo no pasa las validaciones, la sesión manda un mensaje al autor designado para recibir las notificaciones, avisando porque no pasó la revisión.

Una vez pasada el deadline de recepción, la Sesión cambia su estado a ‘Bidding’.

Definí una clase abstracta denominada Usuarios.

La clase Autores hereda de Usuarios y es ella quien crea los artículos, envía a la sesión el mismo y recibe las notificaciones enviadas por la sesión.

Como el artículo que no pasó las validaciones nunca se llegó a grabar en el archivo, el autor deberá volver a pasar por el proceso de crear artículo.

La Sesión mantiene un listado de todos los artículos que pasaron las validaciones.

Los revisores pueden expresar su interés sobre un artículo. Pueden modificar también su interés.

El Chairs tiene acceso al listado de Bidds por artículos.

El encargado de cambiar el estado de la Sesión es el Chairs.

El chair es el encargado de asignar el articulo a un revisor.

El revisor puntúa el articulo con un puntaje entre -3 y 3 y además envía un comentario sobre el mismo.

En la sesión se define el tipo de evaluación que se va a usar para la selección de artículos.

El tipo de evaluación para la selección se puede modificar.

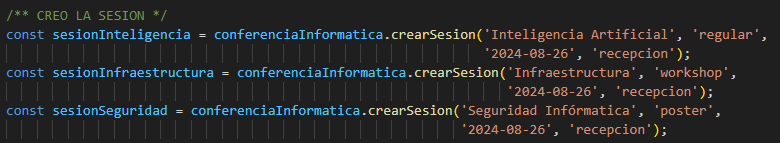
Si se trata de un workshop, se define el tipo de evaluación por tipo de articulo.

**Que tener en cuenta para testear**

**Playground**

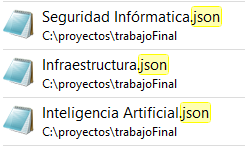
Etapa de Recepción

\*Verificar que la fecha actual no sea superior a la fecha del deadline de recepción.



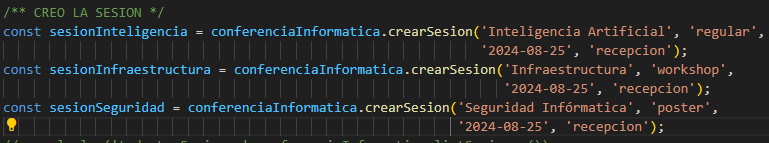
\*Para que no genere muchas leyendas en la terminal de pruebas, sugiero comentar todo lo que tenga que ver con las etapas posteriores.

**\***Todos los artículos que se envían y pasan las verificaciones, se graban en estos archivos.



Etapa de Bidding

\*Se debe tener en cuenta que ya haya pasado el deadline de recepción de artículos.



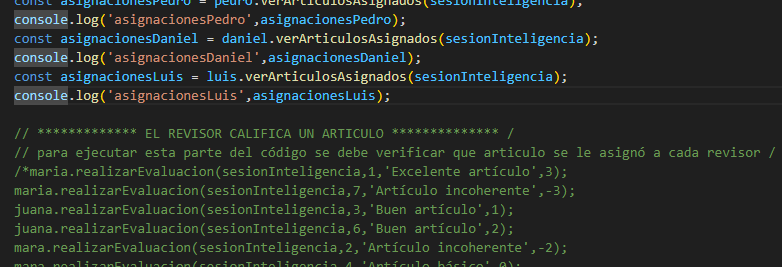
\*Para que no genere muchas leyendas en la terminal de pruebas, sugiero comentar la Etapa de Recepción y las etapas posteriores al Bidding.

Etapa de Asignación

\*Comentar la Etapa de Recepción y las posteriores a la etapa de Asignación.

\*Si se cambió algo de las preferencias del revisor, se debe tener en cuenta que para poder ejecutar el bloque donde EL REVISOR CALIFICA UN ARTICULO, primero se debe verificar los console.log para ver a que revisor se le asignó efectivamente el artículo. Y luego modificar en las sentencias de abajo el ID del artículo para poder evaluar correctamente.

\*El sistema no deja evaluar si ese artículo no se le asignó a ese revisor.



Etapa de Selección

\*Comentar la Etapa de Recepción.

**Test**

Algunos test creé con mock y otros no.

EL SISTEMA

**Clases:**

* ComfyChair: Representa la empresa. En ella concentré la creación de usuarios y conferencias.
* Conferencias: Representa las conferencias. Es la encargada de gestionar las sesiones.
* Sesiones: Representa la sesión. Es la encargada de gestionar los artículos, su validación, obtención, visualización, su asignación, su evaluación y las estrategias de selección.
* Usuarios: Representa a los usuarios y es una clase abstracta.
* Chairs: Representa a los usuarios que son organizadores de una sesión. Es la clase encargada de cambiar el estado de la sesión, de verificar los bidds y de asignar revisores a los artículos.
* Revisores: Representa a los usuarios que son revisores de una sesión. Es la clase encargada de gestionar los intereses y la evaluación de los revisores.
* Autores: Representa a los usuarios que son autores de una sesión. Es la clase encargada de gestionar los artículos y manejar las notificaciones cuando los artículos no pasan las validaciones.
* Articulos: Representa los artículos. Mantiene los intereses de los revisores.
* Estrategias: Representa las estrategias que se pueden utilizar para el proceso de selección. Es una clase abstracta.
* Estrategia Porcentaje de Aceptados: Representa la estrategia de porcentaje de aceptados y es una subclase de Estrategias.
* Estrategia Puntaje Mínimo: Representa la estrategia de puntaje mínimo y es una subclase de Estrategias.

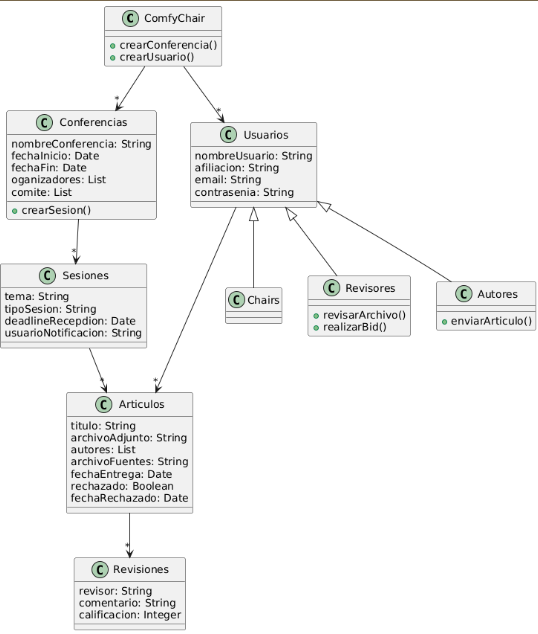
**Relaciones:**

Asociación: La mayoría de las relaciones son de asociación. Por ej: Una Empresa tiene Conferencias y tiene Usuarios. Una Conferencia tiene Sesiones, etc.

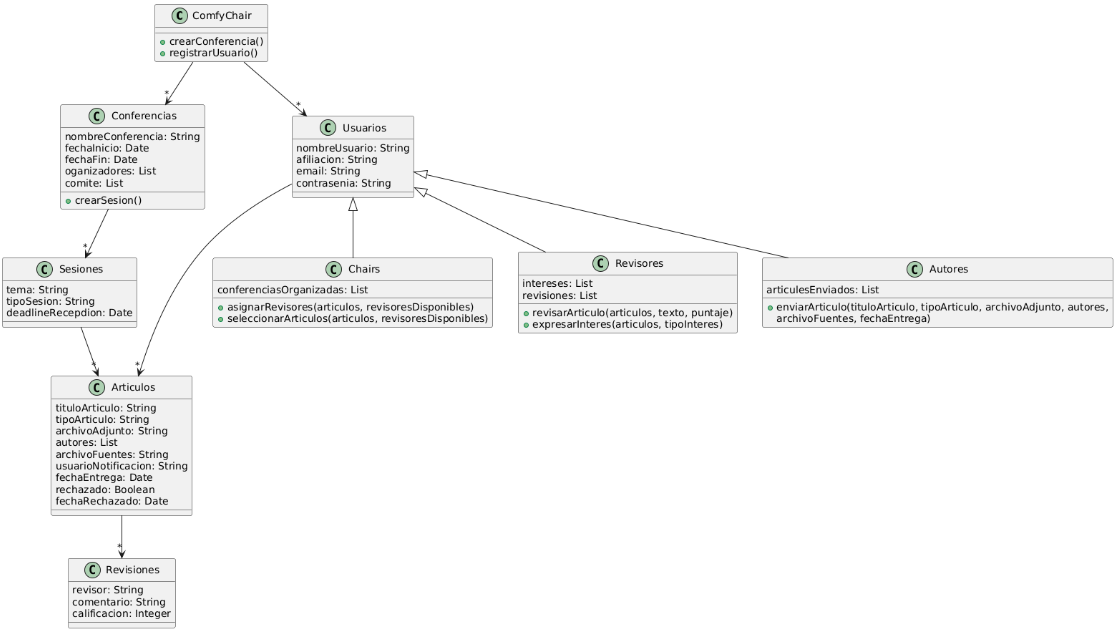
Herencia: Usuarios es la clase abstracta y Chairs, Revisores y Autores heredan los atributos y métodos de Usuarios. Cada subclase define sus propios comportamientos y atributos.

Patrón Strategy: Estrategias es la clase abstracta y sus subclases Porcentaje de Aceptados y Puntaje Mínimo son los que tienen el comportamiento que se pueden instanciar. Cada subclase define un tipo de selección distinta de artículos.

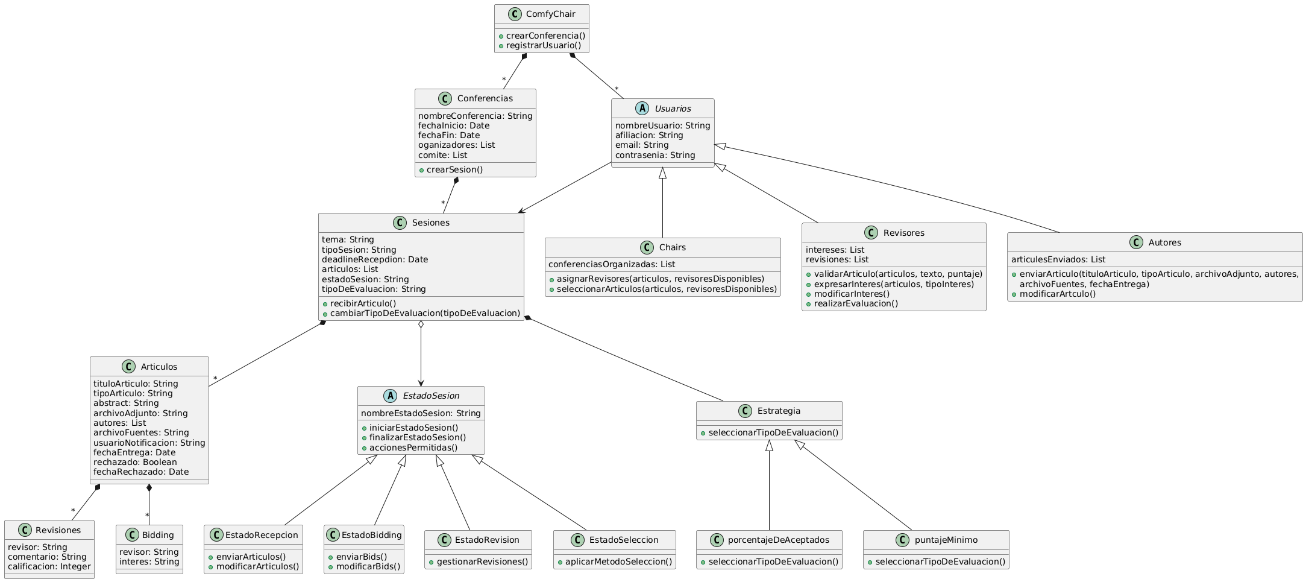
**Diagrama UML:**

1era Aproximación a lo q sería el diagrama UML de ComfyChair

2da Aproximación



3era Aproximación



4ta Aproximación

