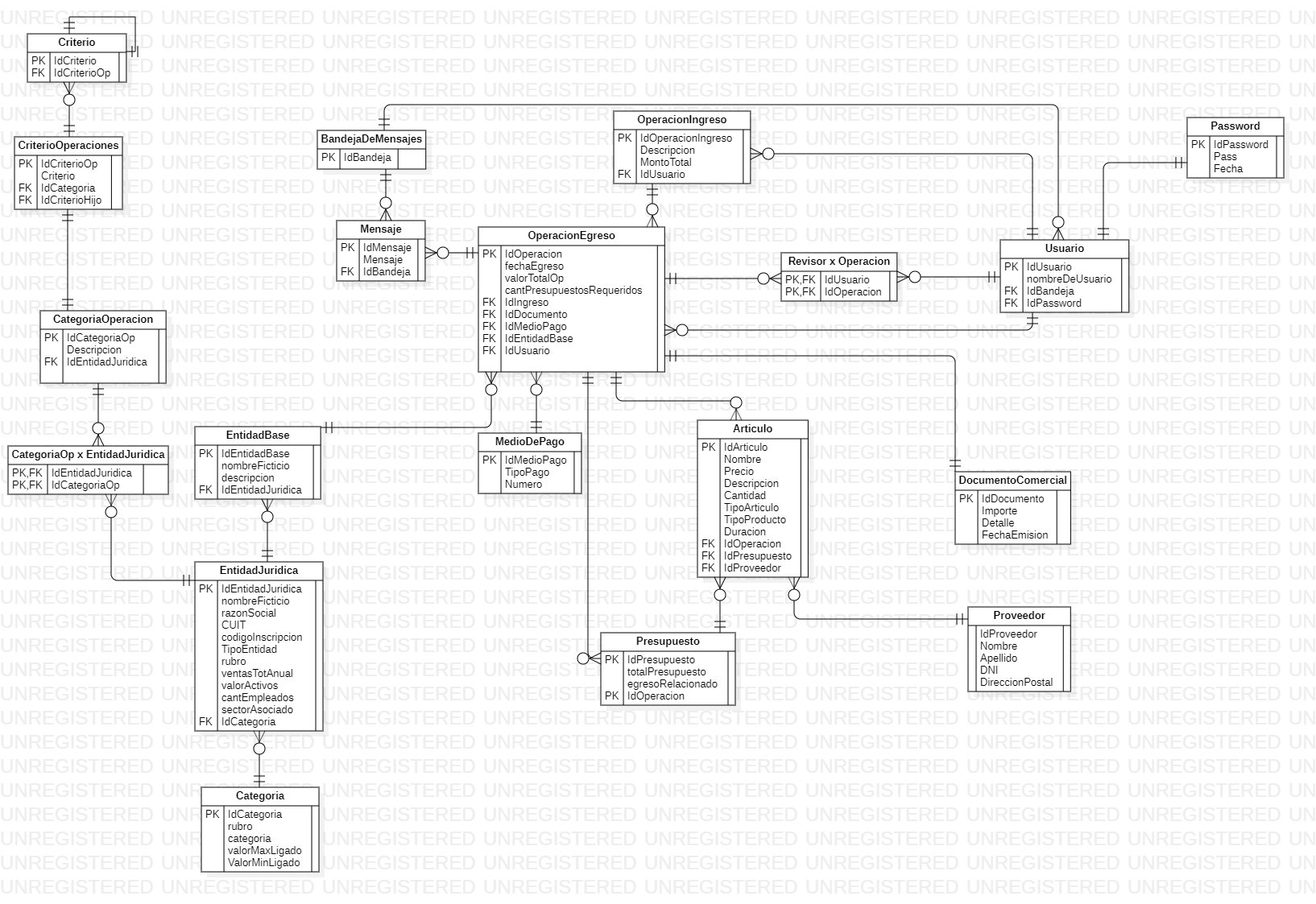
**Documentación sobre decisiones de diseño:**

**Persistencia:**

**Diagrama Entidad Relación:**



**Decisiones de Diseño:**

1. Se mapeo la herencia de las clases Producto y Servicio con Articulo ya que se considero como la estrategia mas eficiente teniendo en cuenta la poca cantidad de atributos de las clases que heredan ya que mejora la performance y las columnas que quedan con NULL son pocas.
2. Se mapeo la herencia de las clases Empresa y OSC con Entidad Juridica como una single table ya que solo queda una cantidad considerable de columnas en NULL si se consulta una OSC, debido a que Empresa tiene varios atributos a persistir, contrario a OSC que tiene solo uno. Esta desventaja nos parecio aceptable en cara a una mejora de performance.
3. Se mapeo la herencia de la clase Criterio con CriterioOperaciones como una Joined table debido a la necesidad de persistir ambas clases por separado ya que CriterioOperaciones contiene Criterios que a su vez heredan de CriterioOperaciones.
4. Diagramamos la relación entre Usuario y OperacionEgreso como una relación doble ya que por un lado un usuario tiene varias Operaciones Egreso creadas y por el otro las Operaciones Egreso tienen una lista de Revisores que también son usuarios que no necesariamente incluyen al usuario que creo la operación. Al mismo tiempo un usuario puede ser Revisor de varias OperacionesEgreso de ahí que esa relacion quede diagramada como un One to Many y un Many to Many.
5. Para la vinculación se tomaron en cuenta los siguientes planteos:
   1. La misma se hace desde el objeto OperacionIngreso ya que un ingreso va a estar asociado a un conjunto de egresos. Para mantener el encapsulamiento, esto se realiza desde adentro del mismo objeto con el método SetEgresosVinculados().
   2. Para vincular se necesitan condiciones que actúan como filtros. Por ejemplo, que los egresos estén dentro de un rango de fecha respecto al ingreso. Por lo tanto, planteamos resolverlo con el patrón Strategy, ya que se expresa que se podría agregar mas condiciones en el futuro.
   3. La vinculación se ejecuta de maneras distintas por lo que se creo la clase MetodoEjecucion la cual se redefine según como se ejecuta el método. Estos métodos no son intercambiables y dentro de los mismos se ejecuta la vinculación de los ingresos.
   4. En caso de querer verificar si un egreso esta vinculado se busca a través de los ingresos.
   5. El usuario es el que ejecuta la vinculación . Pero el usuario de tipo administrador es el único capaz de elegir el método con el que decide hacerlo.
6. La vinculación se hizo en un paquete aparte con la intención de que, en las próximas entregas se utilice Spark y Postman para que sea un componente externo, el cual vinculamos a través de un patron Adapter.