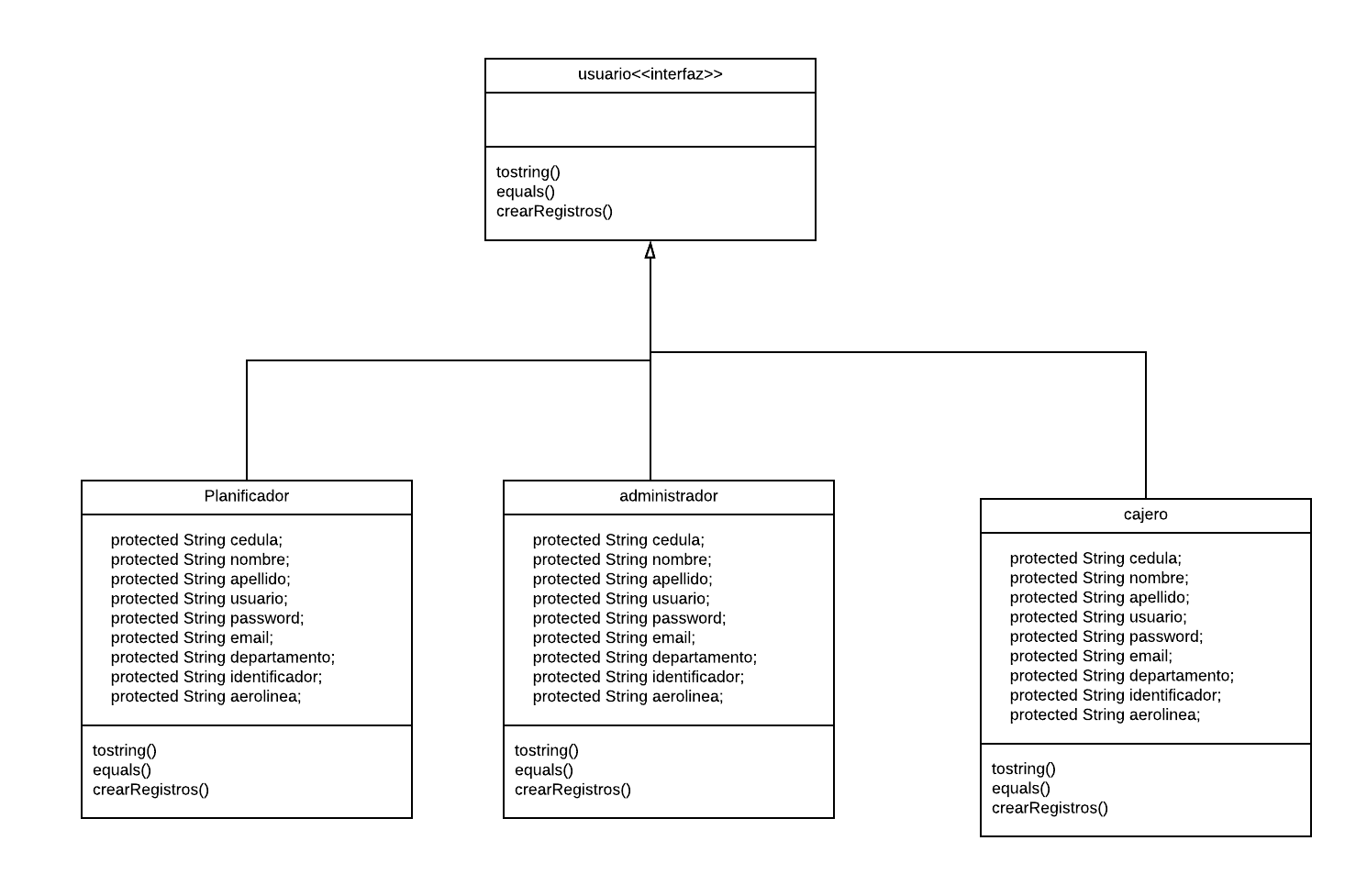
**Repositorio** https://github.com/dsjsanchez/sistemaAeropuerto

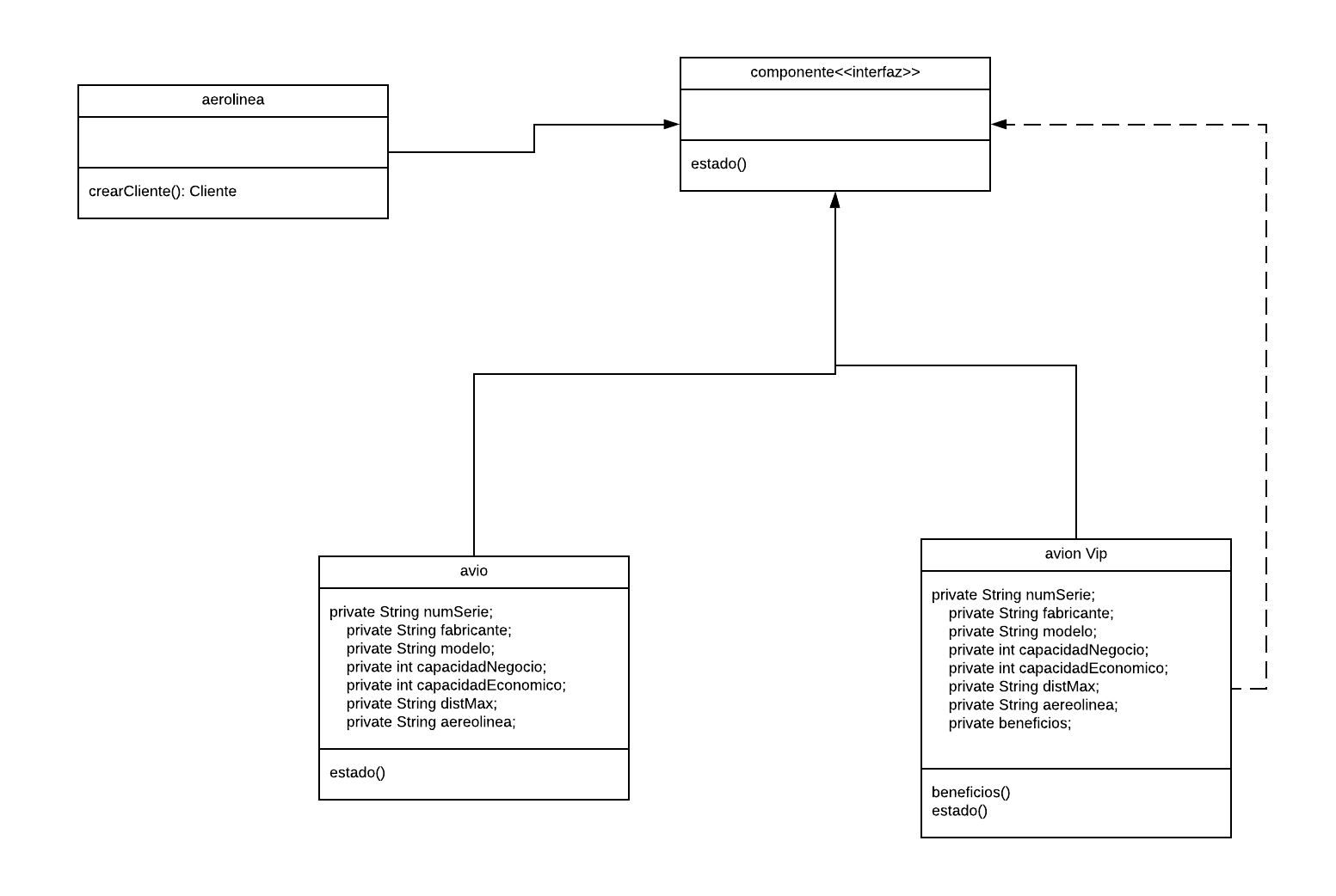
**Factory method**

En la clase usuario sería mucho mejor representarla como interface para así trabajar con las subclases administrador, cajero, y planificador sin tener que implementar todos los atributos mostrados y sin tener un constructor muy grande innecesariamente, también sería muy útil en el caso de que se desea añadir más usuarios como sistema de seguridad, limpieza etc. Esto ayudar con el principio de responsabilidad única y el principio de abierto o cerrado.



**Composite**

En la clase aerolíneas puede tener diferentes tipos de aviones como aviones exclusivos que tienen mejor acceso que los aviones e pasajeros y por eso tener una interfaz con exclusividad sería muy bueno



**Chain of responsability**

En la clase usuario también su puede implementar chain of responsability ya que el administrador planificador y cajero tienen una misma función que es agregar pero hay una especie de jerarquía en la cual el cajero puede agregar clientes, el planificador cajero o clientes y aerolíneas y el admin puede agregar todos en el cual este patrón ayudaría mucho en controlar el orden de ingresos de solicitudes mejora el open close principle y el principio de responsabilidad única porque puedes desacoplar clases pero puede tener algunas solicitudes sin ser gestionadas

