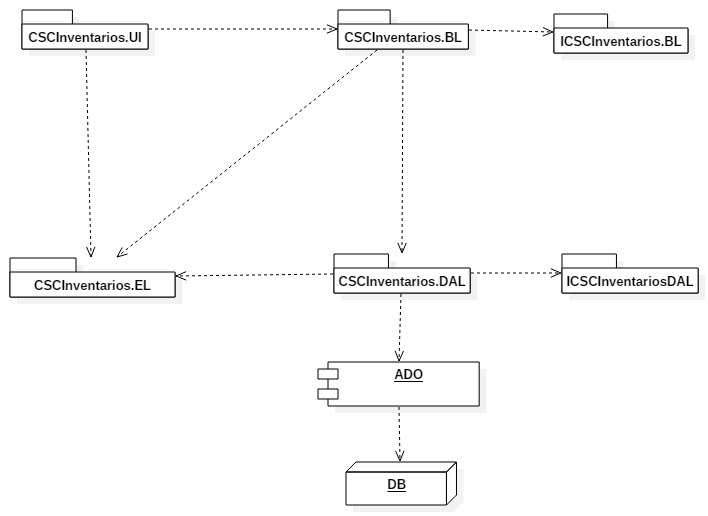
**Estructura de Diseño**

Describir el diseño del más alto nivel. Con un diagrama que muestra una arquitectura en capas, donde tendremos la capa de interface de usuario “UI” que consumirá a la capa de negocio “BL” y a la capa de dominio; la capa de negocio “BL” consumirá a la capa de acceso a datos “DA” y a la capa de dominio; la capa de acceso a datos “DA” usara ADO o ODBC para conectarse a la base de datos “DB” donde se almacenaran todos los registros la base de datos puede ser SQL o ACCESS; además la capa de negocio “BL” implantara las intereses que se encontraran en la capa “IBL, la capa de acceso a datos “DA” implantara las intereses que se encontraran en la capa “IDA”.

****

**Patrones**

**GRASP: Alta cohesión y bajo acoplamiento**

**Cohesión**: La cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes. (Una clase debe de hacer lo que respecta a su entidad, y no hacer acciones que involucren a otra clase ó entidad).

**Acoplamiento**: El acoplamiento es la medida en que los cambios de un componente tiende a necesitar cambios de otro componente. (Cuando modificamos los atributos de una clase, se tienen que modificar los atributos de otra clase.)

Esto permitirá en el desarrollo tener un mejor control del código

**Inversión de Control “IoC” Inyección de dependencias**

Como lo mencioné anteriormente, una de las grandes ventajas es el bajo acoplamiento entre los componentes, lo cual es una gran ayuda sobretodo en la mantenibilidad del software.

La Inyección de Dependencias es un patrón de diseño ampliamente conocido, por ende es fácil de adaptar en múltiples lenguajes de programación, y son soluciones ya probadas a problemas recurrentes en el desarrollo de software.

Facilidad para pruebas, ya que al tener componentes más desacoplados estos son más independientes, y como consecuencia se hace más fácil la implementación de prácticas recomendables de usar como TDD (Test Driven Development).

El software se hace más mantenible a medida que va creciendo, ya que si se implementa una buena arquitectura con DI, la responsabilidad de cada uno de los componentes será muy clara y un cambio podrá ser más fácil de implementar.

Al usar DI debemos pensar un poco más y planear mejor las dependencias de una clase, ya que la idea es utilizar solo las que sean necesarias.

Consiste en inyectar comportamientos a componentes.

**Singleton**

Este patrón se usara para conectarse a la base de datos ya que nos permite una sola instancia por cada sesión.

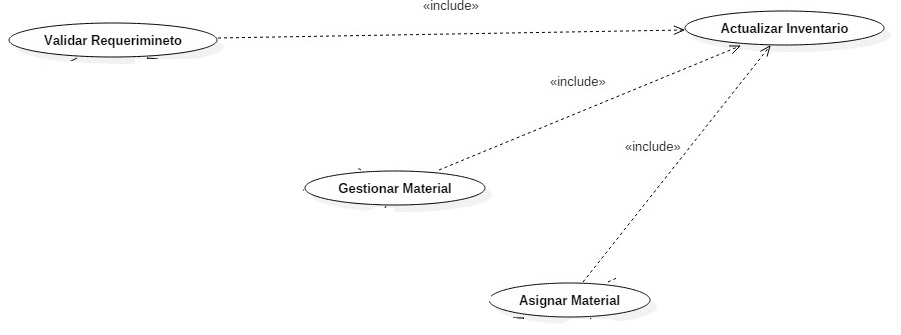
**Subsistemas**

El sistema no contara con sub sistemas debido a que está orientado a una pequeña área de control y asignación de materiales

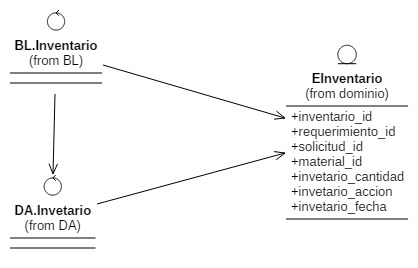
**Realización de Requerimientos**

**Realización Actualizar Inventario**

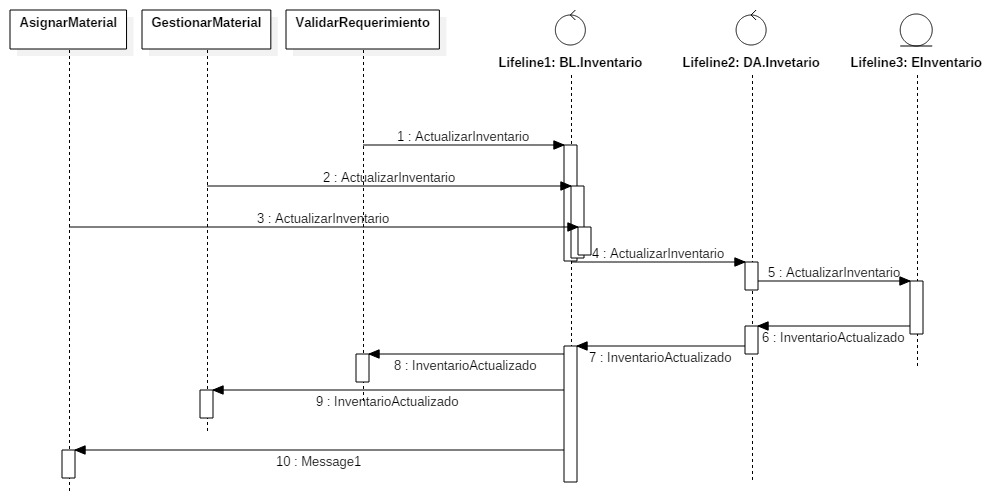
**Vista de los participantes**

****

**Diagrama de clases**

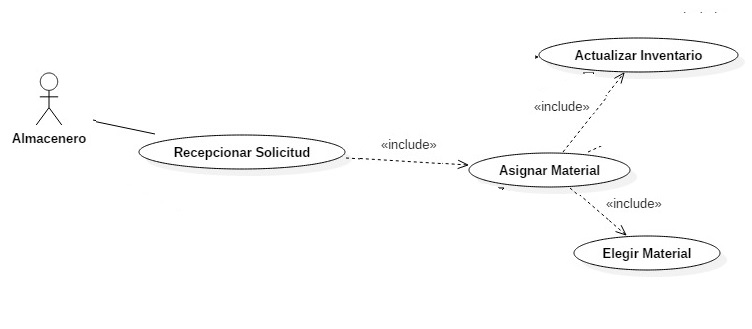
****

**Diagrama de secuencia**

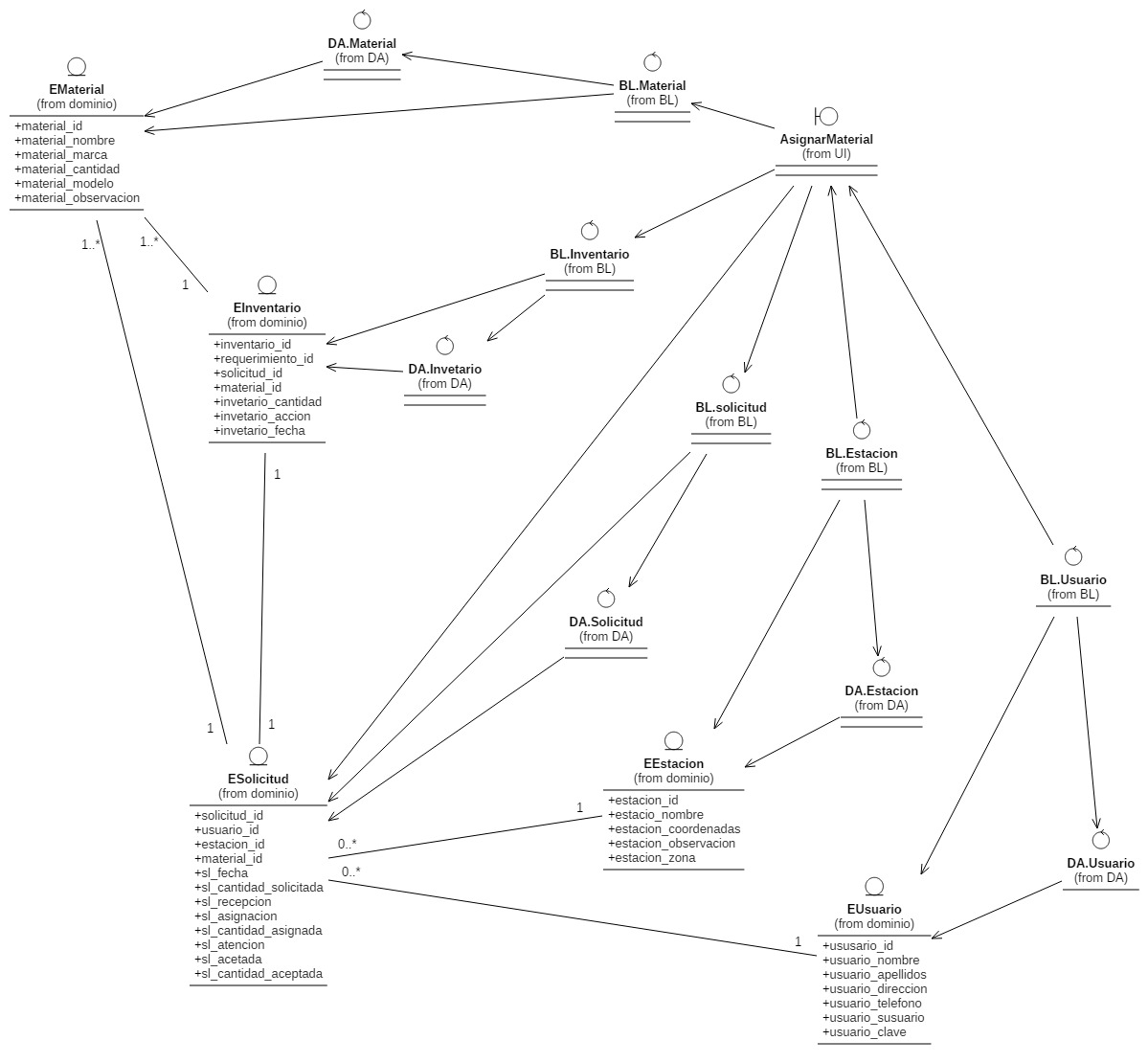
****

**Realización Asignar Material**

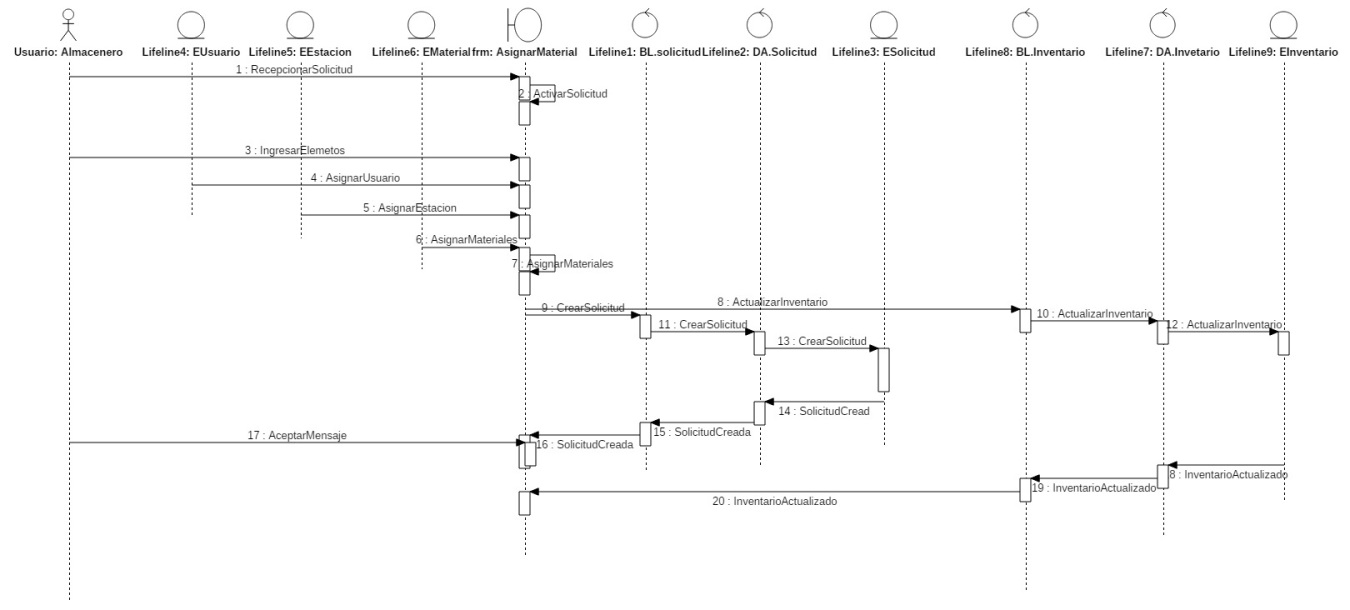
**Vista de los participantes**

****

**Diagrama de clases**

****

**Diagrama de secuencia**

****

**Realización Atender Solicitud**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Cerrar Sesión**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Elegir Estación**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Elegir Material**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Enviar Solicitud**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Gestionar Estación**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Gestionar Material**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Gestionar Usuario**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Imprimir Requerimiento**

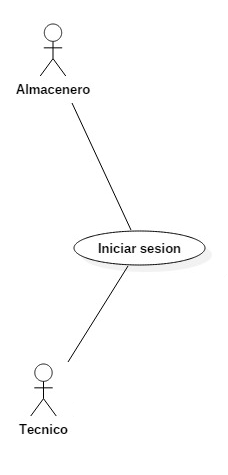
**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Iniciar sesión**

**Vista de los participantes**



**Diagrama de clases**

**Diagrama de secuencia**

**Realización Recepcionar Material**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Recepcionar Solicitud**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Requerir Material**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Gestionar Estación**

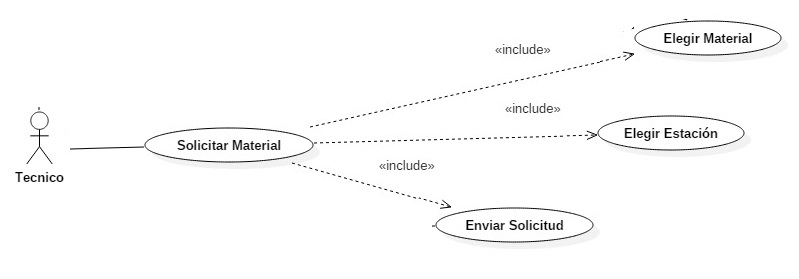
**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Solicitar Material**

**Vista de los participantes**

****

**Diagrama de clases**

**Diagrama de secuencia**

**Realización Validar Requerimiento**

**Vista de los participantes**

**Diagrama de clases**

**Diagrama de secuencia**