FACULTAD DE INGENIERÍA

UNIVERSIDAD DE BUENOS AIRES

TALLER DE PROGRAMACIÓN III
(75.61)

2DO CUATRIMESTRE 2015

EJERCICIO DE COLAS RABBITMQ

ATENCIÓN DE PEDIDOS

FLORENCIA ALVAREZ ETCHEVERRY

05 Octubre 2015

ÍNDICE

OBJETIVOS

El sistema debe poder aceptar pedidos de compra de distintos tipos de productos. Cada cliente realiza un pedido con una identificación del mismo, el tipo de producto a comprar y la cantidad del mismo. Se debe mantener el stock para cada tipo de producto, y todo pedido de compra debe ser almacenado como una entrada de Log con fines de auditoría, por lo que se almacenarán con el horario en que se realizó. Se controlarán los estados de cada pedido por su identificación, pudiendo ser “Recibido”, “Aceptado”, “Rechazado” o “Entregado”.

Además de aceptar pedidos de compras el sistema aceptará modificaciones de stock por medio de un Administrador del sistema, consultas del estado de los pedidos por parte de los clientes (con su identificación de pedido) y la notificación de la entrega de un pedido, por parte de empleados.

Como requerimientos no funcionales del sistema, se pide que se contemple la existencia de un gran stock de productos pero que se puedan catalogar en unos pocos tipos, que la aplicación debe estar preparada para una gran cantidad de usuarios concurrentes, y que se pueda permitir la distribución de sus componentes y la comunicación de información mediante colas persistentes.

VISTA DE CASOS DE USO

A continuación se muestra el diagrama con los casos de uso que se identifican en el sistema que resuelven los requerimientos funcionales.

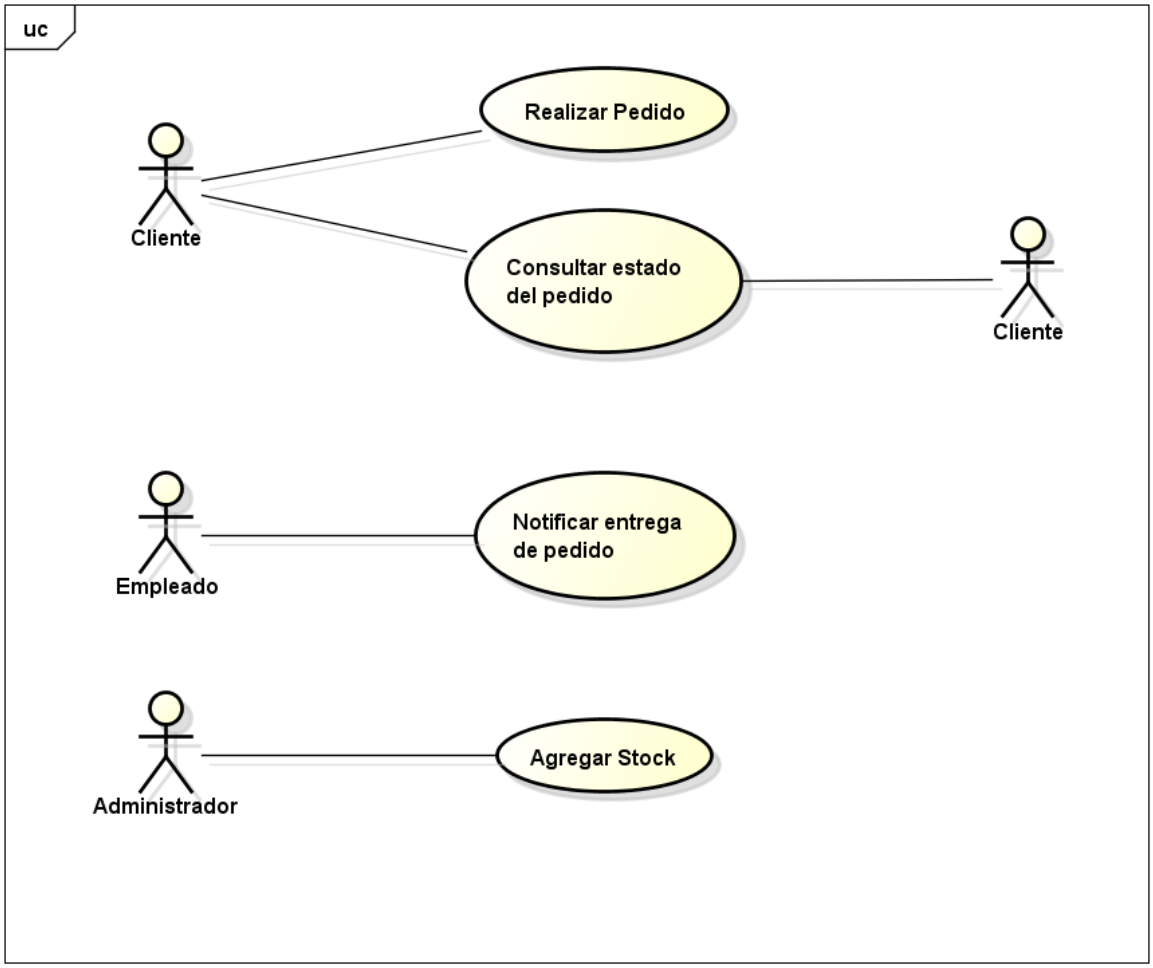

Figura 1: Diagrama de Casos de Uso.

Se describen también los casos de uso modelados para este sistema.

Caso de uso: Realizar Pedido	
Descripción:	El cliente realiza un nuevo pedido con un ID, para un tipo de producto y una cantidad a comprar; se recibe y se controla si se puede aceptar o no.
Actores:	Cliente
Precondiciones:	Se tiene un ID único para el pedido.
Flujo Principal:	1. Crea el mensaje a enviar con el ID, tipo de producto y cantidad a pedir.

	2. Envía el mensaje por la cola de “nuevas órdenes”. 3. Se guarda la orden y los datos en el archivo Log. 4. Se guarda el identificador y se marca como “Recibida”. 5. Se envía para controlar si el stock es suficiente. 6. Se resta el stock para ese producto y se guarda el nuevo stock. 7. Se marca como “Aceptada”.
Postcondiciones:	El nuevo pedido ya fue ingresado al sistema.

Caso de uso: Consultar estado del pedido	
Actores:	Cliente
Precondiciones:	Se tiene el ID único del pedido ya ingresado al sistema.
Flujo Principal:	1. Envía el mensaje con el ID por la cola de “Queries”. 2. Se consulta en el archivo de órdenes el estado para ese ID. 3. Se devuelve el resultado al cliente.
Postcondiciones:	El cliente recibió el estado del pedido.

Caso de uso: Notificar entrega de pedido	
Actores:	Empleado
Precondiciones:	Se tiene el ID único del pedido aceptado ya ingresado al sistema.
Flujo Principal:	1. Envía el mensaje con el ID del pedido a la cola de “cambiar estado”. 2. Se cambia el estado “Aceptado” a “Entregado”.
Postcondiciones:	El pedido se encuentra Entregado.

Caso de uso: Agregar Stock	
Actores:	Administrador
Precondiciones:	Se tiene el tipo de producto y la cantidad a agregar.
Flujo Principal:	1. Envía el mensaje con el tipo de producto y la cantidad a agregar por la cola de “agregar stock”. 2. Se suma el stock para ese producto y se almacena.
Postcondiciones:	Para el tipo de producto dado, se agregó el stock por la cantidad pedida.

VISTA LÓGICA

MENSAJES

Para modelar y encapsular el envío de información a través de las colas, se crearon clases para los distintos tipos de mensajes que se envían. En el diagrama siguiente se muestra la jerarquía creada para los mensajes.

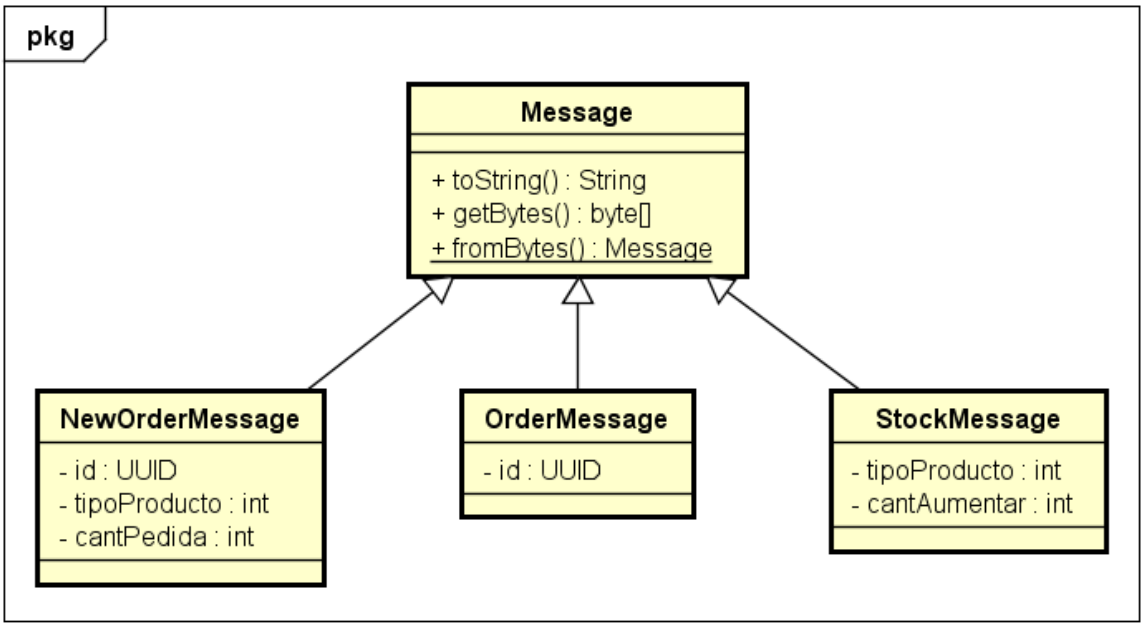

Figura 2: Diagrama de Clases, Mensajes.

La clase Message generaliza el comportamiento de las 3 clases concretas, es decir, las acciones de serializar y deserializar los mensajes en bytes. Entre las clases hijas, la diferencia radica en la información (atributos) que poseen, ya que en el sistema depende de la situación el tipo de mensaje que se necesite utilizar.

COLAS Y PROCESOS

Para encapsular las acciones que realiza las colas de RabbitMQ se creó una clase Queue con los métodos que se pueden realizar en las colas. Estas colas son de algún tipo de mensaje particular, ya que por una cola determinada siempre se manda el mismo tipo de mensaje. A su vez, como la cola debe procesar por cada mensaje que se lee de la misma, posee un atributo de tipo QueueProcesser, para que se pueda llamar al process() correspondiente. Por este motivo, todas las clases que deben realizar una acción al leer de la cola, implementan esta inferfaz.

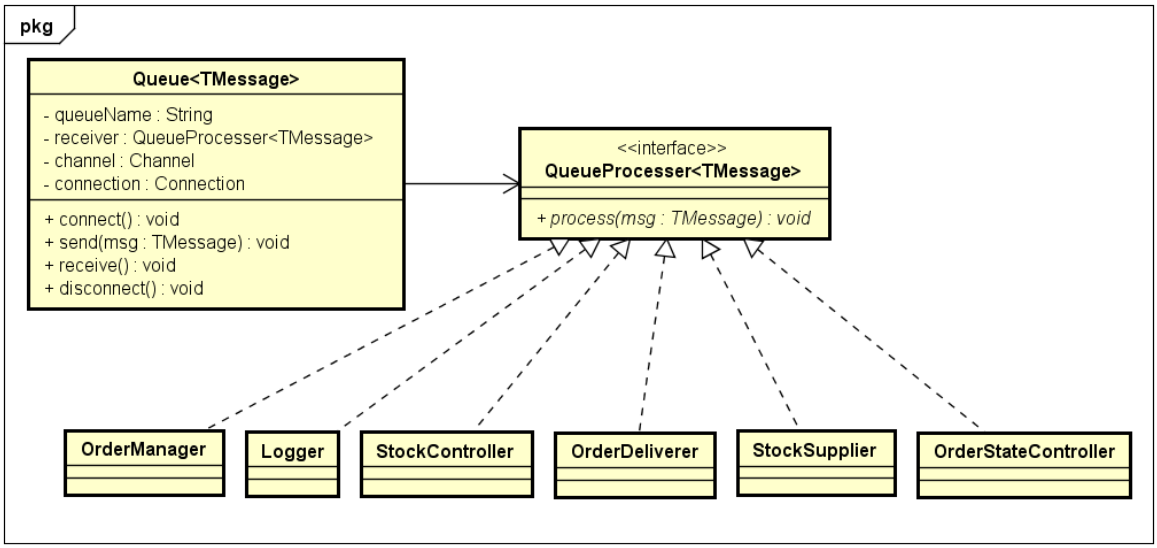

Figura 3: Diagrama de Clases, Colas y Procesos.

La clase OrderManager es el proceso que se encarga de leer de una cola los mensajes de nuevas órdenes, reenviarlos a la cola para ser loggeado, guardar la orden como “Recibida” en el archivo, y finalmente reenviando el mensaje a StockController para que se verifique el stock.

La clase Logger es el proceso que lee de una cola todas las órdenes que van llegando y las almacena en un archivo de log.

La clase StockController es el proceso que se encarga de leer de una cola las nuevas órdenes, verificar si para ese tipo de producto es suficiente el stock, y modificar el estado de la orden a “Aceptada” o “Rechazada” según corresponda.

La clase OrderDeliverer es el proceso que lee de una cola donde el Empleado coloca las órdenes que fueron entregadas, por lo que modifica el estado a “Entregada” en el archivo de órdenes.

La clase StockSupplier es el proceso que lee de una cola donde el Administrador coloca el mensaje con el tipo y la cantidad de stock que quiere aumentar para un producto, y lo actualiza en el archivo de stocks.

La clase OrderStateController es el proceso que se encarga de leer de una cola las consultas que un cliente realiza para un determinado pedido. Consulta el archivo de órdenes y le comunica al cliente cuál es el estado actual de la misma.

Manejo de Pedidos

En el siguiente diagrama se muestra parte del sistema donde se realiza y controla el acceso a los archivos de órdenes.

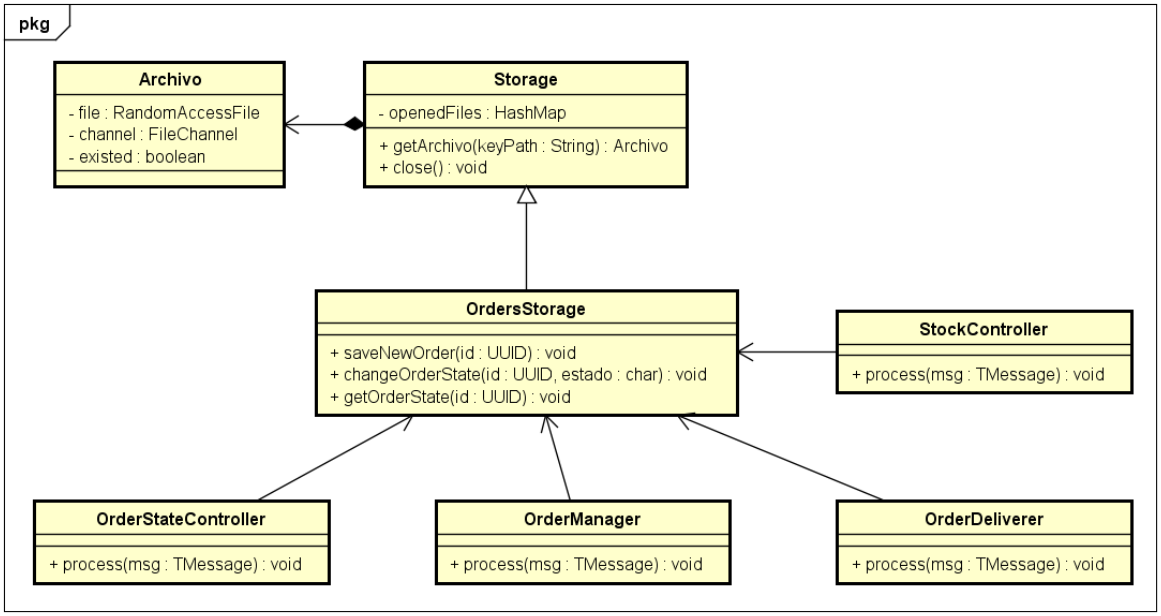

Figura 4: Diagrama de Clases, Orders Management.

Los procesos utilizan la clase OrdersStorage para modificar los datos de las órdenes en los archivos. Esta clase, posee un HashMap por el cual puede obtener cada archivo de acuerdo al nombre del mismo. Los nombres de los archivos se obtienen y generan a partir de los IDs de los pedidos, por lo que en cada función se recibe, y se puede acceder al archivo abierto.

Manejo de Stock

Análogamente, el manejo de los stocks en archivos, también depende de cada invocación a un método, en este caso, al tipo de producto. Por lo que se utilizó la jerarquía para generalizar un Storage con un HashMap de archivos abiertos, a los que se puede acceder fácilmente en cada llamada de los procesos que necesiten modificar el stock.

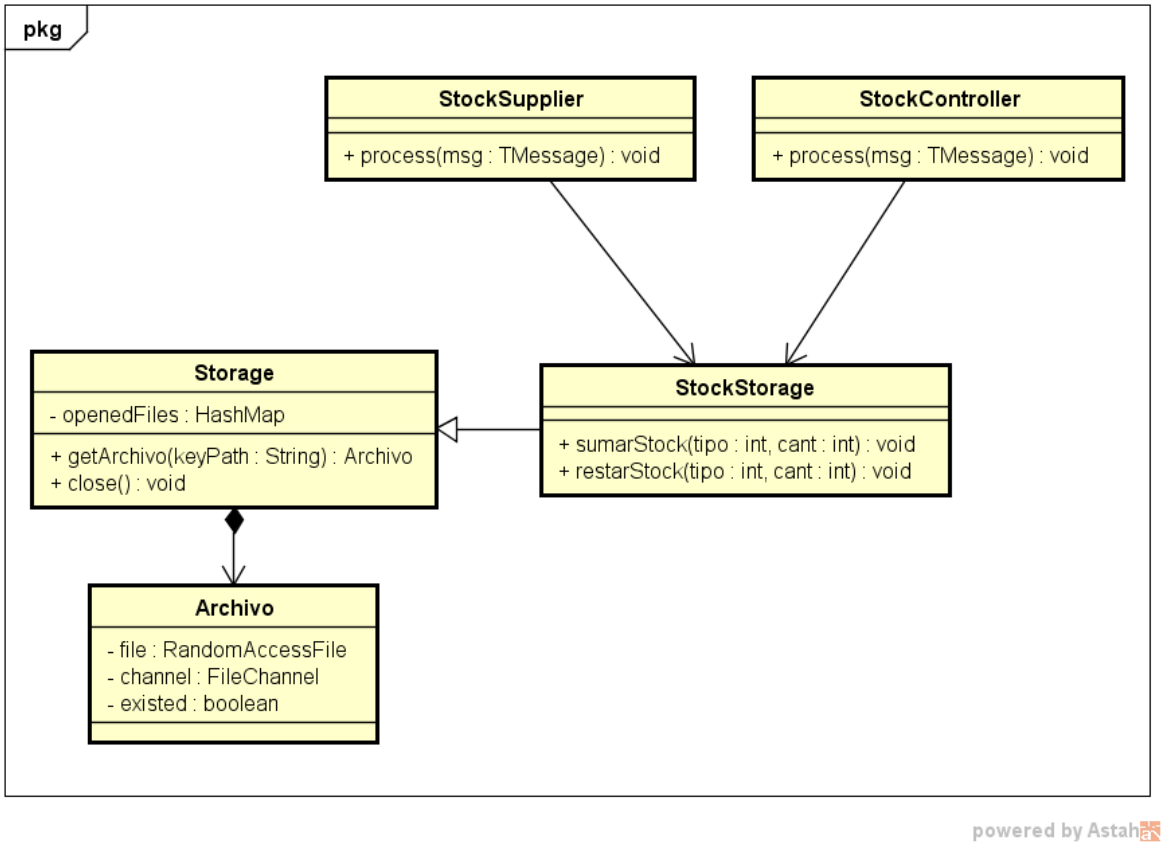

Figura 5: Diagrama de Clases, Stock Management.

VISTA DE PROCESOS

A través de diagramas de secuencia, se puede modelar el flujo de la información y los datos en el sistema. Como en este sistema, las comunicaciones entre procesos se realizan a través de colas, fue diagramado de la siguiente manera: Las flechas "→" indican que el mensaje es pasado a través de una cola. El otro tipo indican invocación a métodos de objetos.

En el siguiente diagrama se muestra la secuencia entre objetos del sistema cuando un cliente realiza una nueva orden.

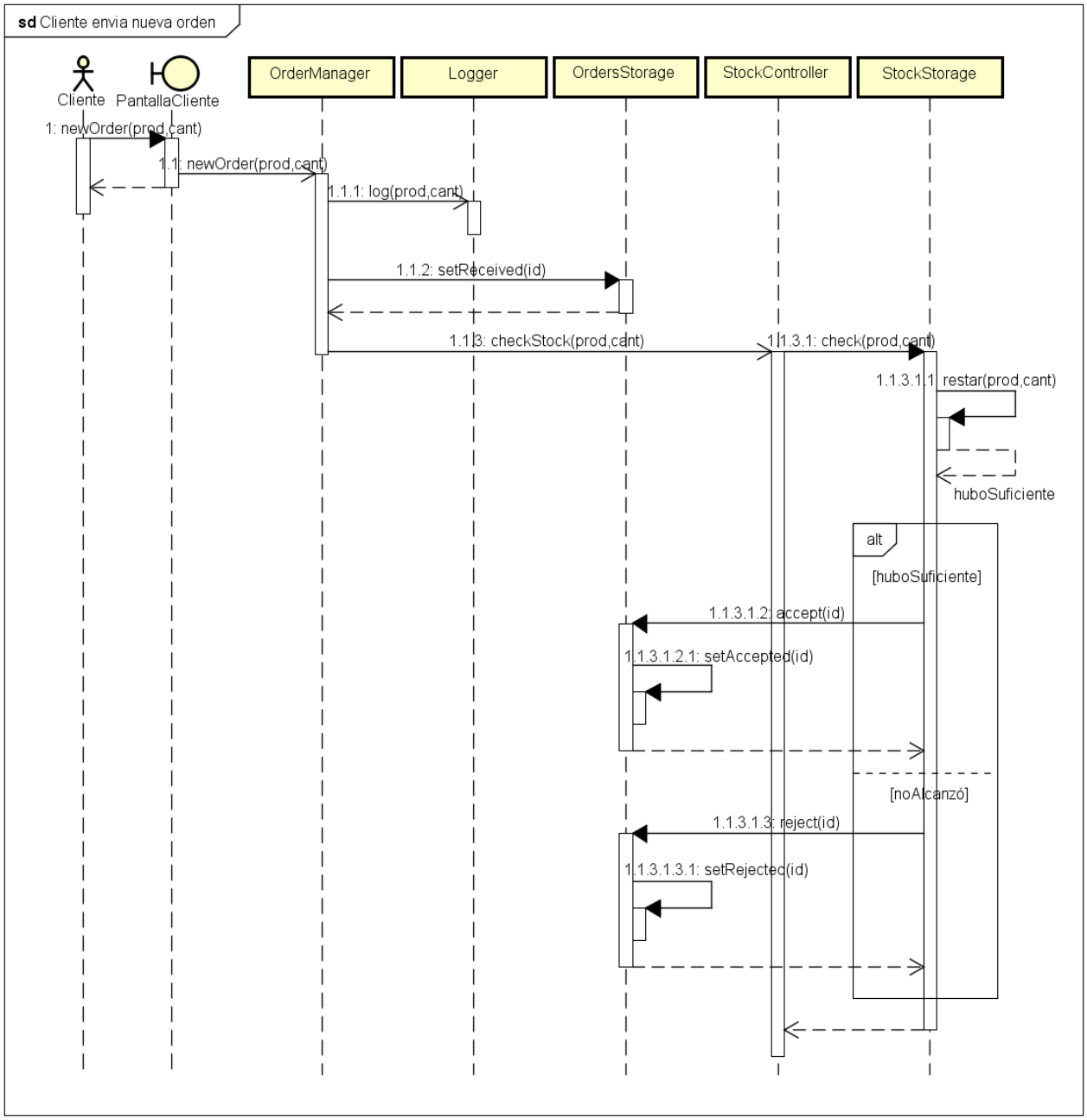

Figura 6: Diagrama de Secuencias, nueva orden.

En el siguiente diagrama de secuencias se describen las acciones en el sistema realizadas cuando un cliente consulta el estado de una orden dada.

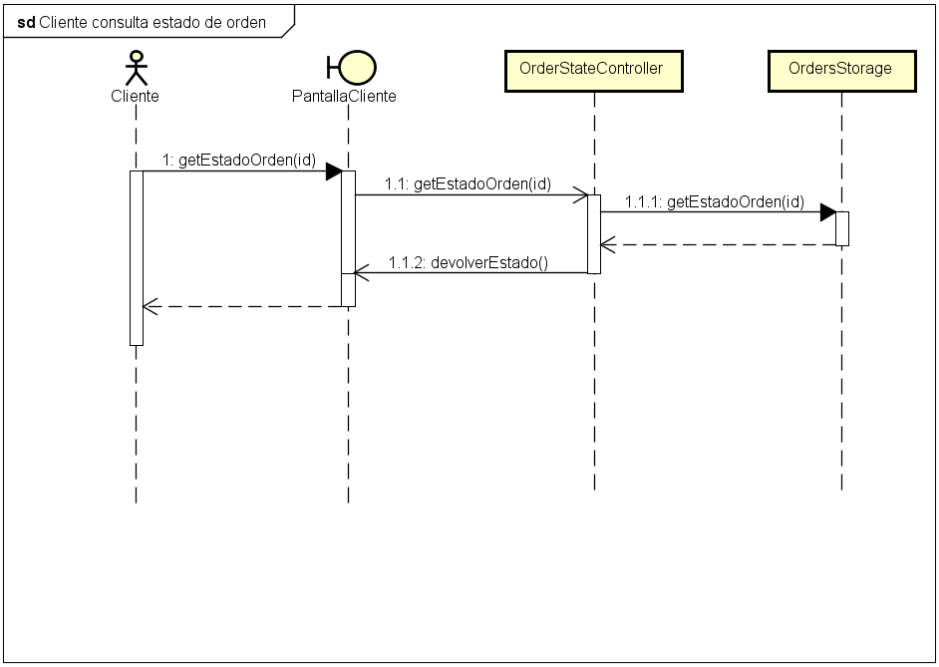

Figura 7: Diagrama de Secuencia, consulta estado.

El siguiente diagrama muestra la secuencia a realizar cuando un empleado notifica la entrega de un producto.

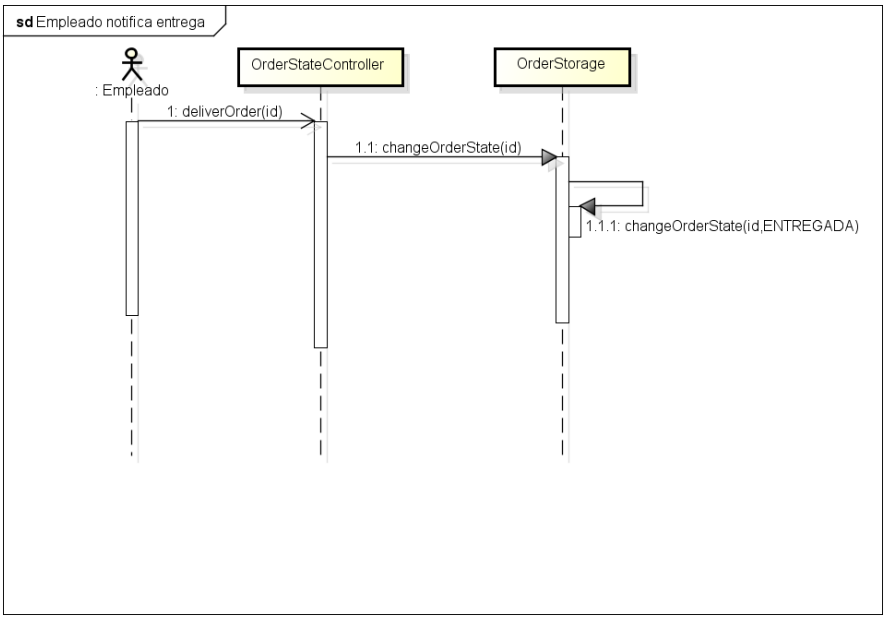

Figura 8: Diagrama de Secuencia, entrega producto.

El siguiente diagrama muestra la secuencia a realizar cuando un administrado aumenta el stock de un determinado producto.

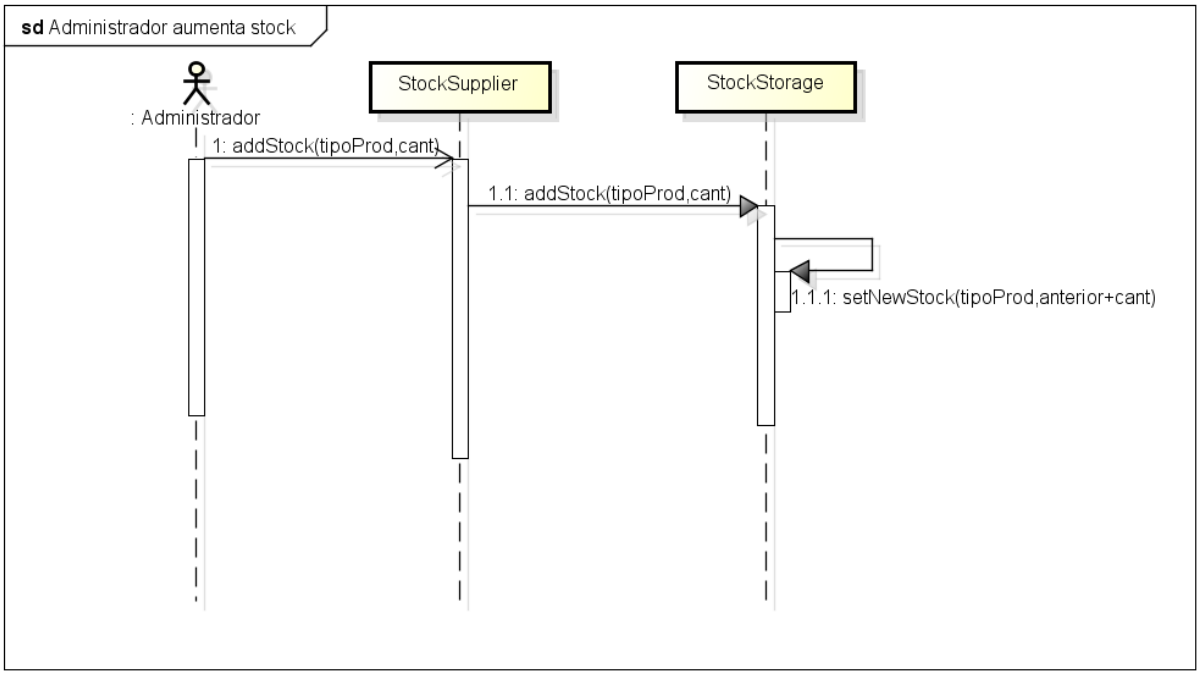

Figura 9: Diagrama de Secuencia, agregar stock.

VISTA DE DESPLIEGUE

Mediante el siguiente diagrama de robustez se puede visualizar la arquitectura y las comunicaciones entre los distintos objetos en el sistema.

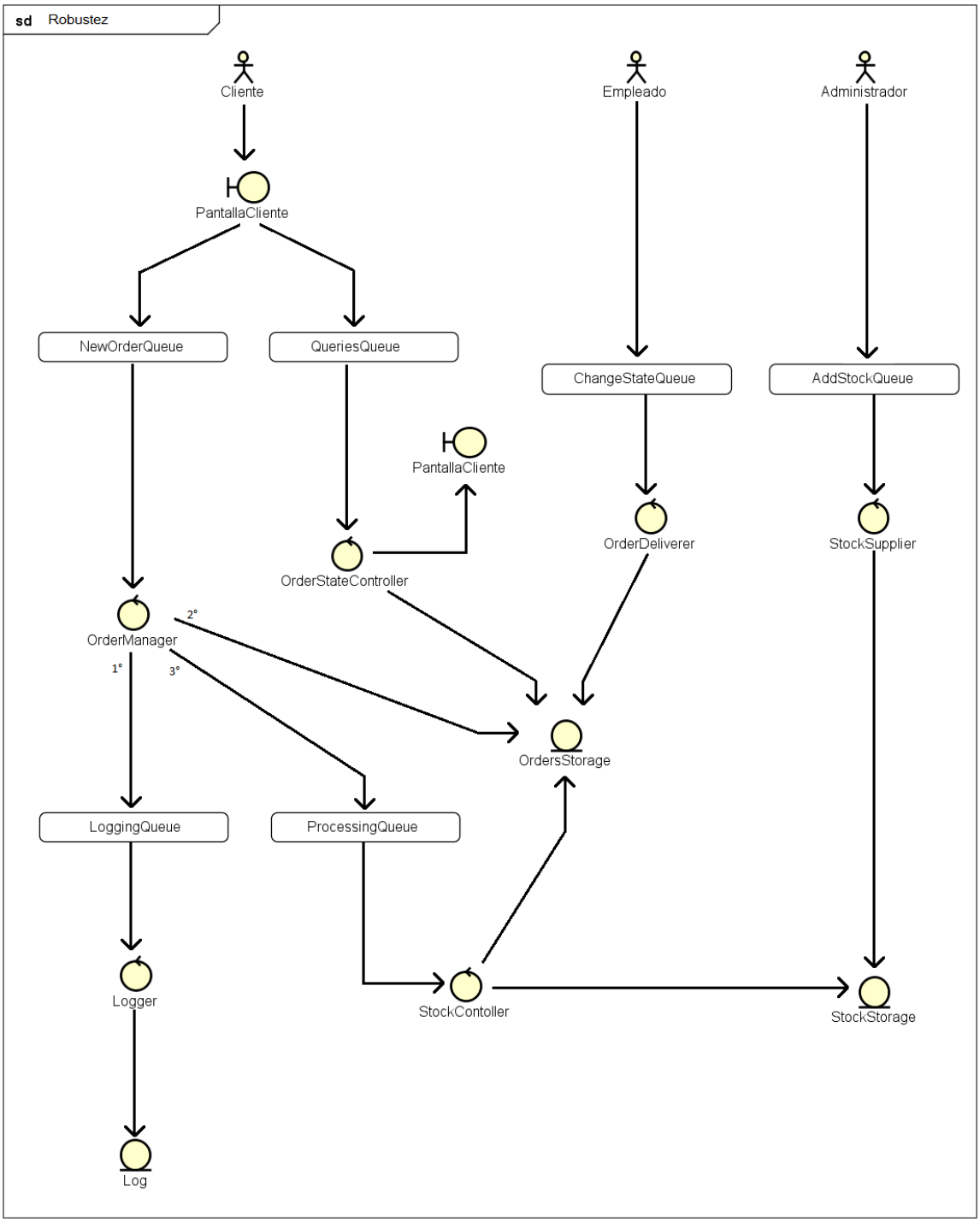

Figura 10: Diagrama de Robustez