Saimazoom

Pareja 13
Iker Manuel Pérez Villa
Lucía Pereira Menchero

Índice

1. Introducción

En el presente documento se describen los requisitos principales del sistema *Saimazoom* y la explicación de la lógica de ello. El objetivo principal de este software es simular un mercado online de venta de productos en el cual se pueda abastecer a varios clientes simultáneamente.

Lo interesante del desarrollo es el uso de las estructuras de colas, en **RabbitMQ** que funciona como middleware de mensajería. Los actores participantes en el sistema son:

- **Controlador**: encargado de controlar los mensajes entrantes y salientes del sistema y de llevar un control de los datos de pedidos, productos y clientes.

- **Cliente**: realiza y gestiona sus pedidos.

- **Robots**: encargados de buscar los productos en el almacén y de poner los pedidos en cinta.

- **Repartidores**: encargados de repartir los pedidos a los clientes.

Todos se conectan mediante el controlador, el elemento central de la estructura, y a raíz de él fluyen todos los mensajes de las colas implementadas.

La estructura del documento se divide en tres secciones; por un lado la *Definición del proyecto*, donde se detalla el funcionamiento interno del sistema (utilización de colas y sus tipos) así como los requisitos funcionales, por otro lado la *Implementación* representada de forma visual mediante distintos diagramas, y por último las *Conclusiones* obtenidas de la práctica, por parte de ambos integrantes de la pareja.

(Para saber como ejecutar los scripts proporcionados en el código y su funcionalidad, mirar la wiki del proyecto).

2. Definición del proyecto

2.1. Objetivos y funcionalidad

El objetivo del proyecto es simplemente el **uso de colas** para la comunicación entre los distintos actores del sistema. En este tipo de sistemas, la funcionalidad de las colas proporciona una gran ventaja de comunicación simultánea y de resguardo de mensajes, ya que a una misma cola pueden llegar mensajes de distintos actores sin pérdida cuando uno de los actores se encuentra saturado o trabajando en otro mensaje.

En la cabeza de la estructura del sistema se encuentra el controlador, quien mantiene comunicación directa con el resto de actores. Los clientes envían mensajes solicitando una acción en el sistema, ya sea realizar, cancelar o listar pedidos.

En caso de realizarse un pedido, el controlador envía mensajes a la cola de los robots, para que los productos de dicho pedido sean buscados en el almacén. Los robots informan al controlador cuando encuentren o no encuentren cada producto. Pueden ocurrir dos situaciones:

- **Todos los productos de un pedido son encontrados**

    1. El controlador envía mensaje a la cola de los repartidores para que el pedido se intente entregar.

    2. El repartidor encargado del pedido informa cuando el pedido empiece a ser repartido y también cuando se intente entregar (ya sea con éxito o no).

    3. Si no se pudo realizar la entrega, el controlador comprueba cuántos intentos de entrega ha habido con ese pedido, y lo vuelve a poner en la cola de mensajes de repartidores si el número de intentos es menor que 3.

- **Hay algún producto no encontrado**

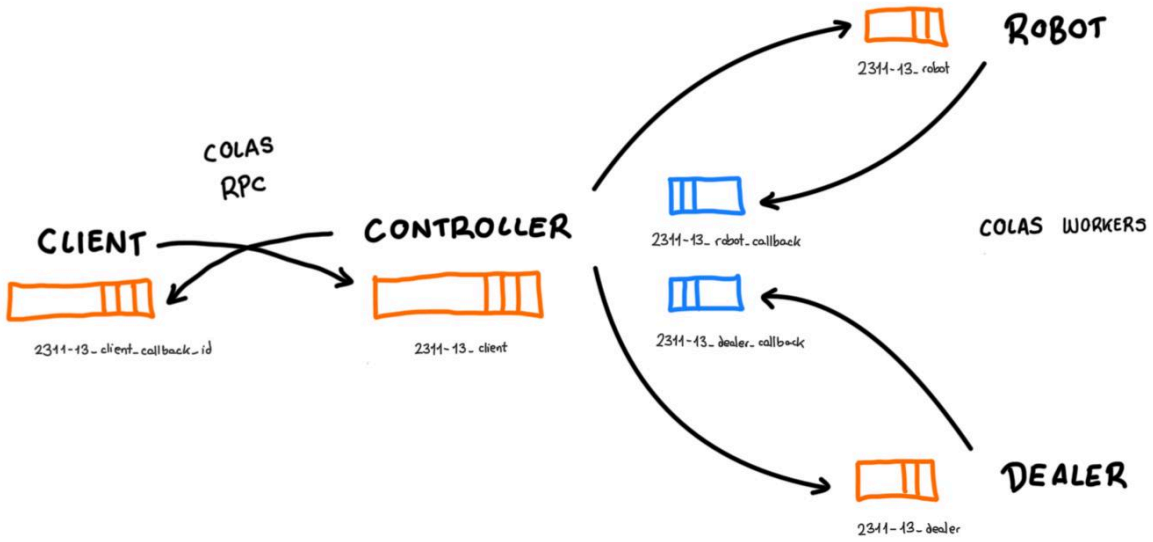
    1. El controlador pone el pedido en estado *incompleto* y no realiza la entrega. (Un pedido también se pone en este estado si no hay stock de alguno de los productos de un pedido).

    2. El cliente podrá visualizar dicho pedido en la sección de sus pedidos, pero no saldrá completo. Además los productos de dicho pedido volverán al almacén por lo que realmente ese pedido no existe ni puede continuar.

En el caso de cancelar un pedido, puede solicitarse dicha acción con cualquier pedido y siempre que se quiera, pero únicamente se cancela con éxito si el pedido aún no ha salido del almacén de Saimazoom, es decir, cuando aún no ha entrado en la cinta de repartición y tampoco cuando se encuentra en entrega. **Un pedido cancelado retiene sus productos, por lo que no son devueltos al almacén**.

2.1.1. Funcionamiento de las colas

Para mayor entendimiento del uso de colas en el sistema implementado, se proporciona a continuación un diagrama hecho a mano de la comunicación entre los actores.



En la imagen anterior encontramos las siguientes colas:

Colas “RPC”

- **2311-13_client y 2311-13_client_callback_id** (Clientes < –> Controlador)
  Los clientes envían mensajes solicitando una acción u otra.

  El controlador envía mensajes con el id del cliente correspondiente (correlation id). De esta manera se redirecciona los mensajes de respuesta a cada cliente, y en caso de que existieran numerosos clientes no habría interferencia ni dificultad para consumir las respuestas del controlador (en un sistema de compra online realista habría muchos más clientes simultáneamente).

Colas “Workers”

- **2311-13_robot** (Controlador --> Robots)
  El controlador envía mensajes para solicitar la búsqueda de productos.

- **2311-13_dealer** (Controlador --> Repartidores)
  El controlador envía mensajes para solicitar el reparto de pedidos.

Colas “Hello World”

- **2311-13_robot_callback**
  Los robots envían mensajes de información de búsqueda de productos.

- **2311-13_dealer_callback**
  Los repartidores envían mensajes de información de entrega de pedidos..

2.2. Requisitos Funcionales

2.2.1. Lógica de clientes

[RF-C-1] El cliente puede registrarse en la aplicación, introduciendo nombre de usuario y clave de acceso. El nombre de usuario debe ser único, por lo que el sistema informa cuando el nombre ya está en uso.

[RF-C-2] El cliente puede iniciar sesión con su nombre de usuario y clave de acceso. En caso de introducirse erróneamente o de no haberse registrado antes, el sistema informa con un mensaje de error.

[RF-C-3] El cliente tiene la posibilidad de realizar un pedido indicando los ids de los productos que quiere que lo compongan. No puede realizar un pedido sin productos.

[RF-C-4] El cliente puede solicitar el listado de sus pedidos. Puede visualizar el id del pedido, los ids y cantidades de los productos que lo componen y el estado del pedido en dicho momento.

[RF-C-5] El cliente, antes de que un pedido salga del almacén, puede cancelarlo indicando el número de id de dicho pedido.

[RF-C-6] Para poder realizar un pedido se debe conocer los ids de los productos y su disponibilidad, por ello el cliente puede solicitar la lista de productos Saimazoom pudiendo visualizar tanto la cantidad disponible como una pequeña descripción de cada uno.

2.2.2. Lógica de robots

[RF-RB-1] El robot recibe un mensaje que indica la búsqueda de un producto asociado a un pedido. Se encarga de buscarlo en el almacén durante 5-10 segundos.

[RF-RB-2] El robot, una vez haya realizado la búsqueda con probabilidad *p_almacen*, informa al controlador de si ha sido capaz de encontrarlo o no.

2.2.3. Lógica de repartidores

[RF-RP-1] El repartidor recibe un mensaje que indica la entrega de un pedido. Se encarga de entregarlo en 10 segundos informando al controlador que el pedido ya está en camino a entregarse.

[RF-RP-2] El repartidor, una vez haya realizado la entrega con probabilidad *p_entrega*, informa al controlador de si ha sido capaz de entregarlo o no.

2.2.4. Lógica del controlador

[RF-CD-1] El controlador lleva un registro de los elementos necesarios a guardar en el sistema: clientes, pedidos y productos.

[RF-CD-2] El controlador es capaz de guardar los datos del sistema en un fichero *data*.

[RF-CD-3] El controlador es capaz de salvar los datos del sistema del fichero *data* en caso de que exista y sinó, crea uno nuevo donde se guardarán los datos al finalizar su ejecución.

[RF-CD-4] El controlador cambia el estado de los pedidos según en qué circunstancias se encuentre:

- IN_WAREHOUSE (en almacén): cuando recién se crea el pedido.
- INCOMPLETE (incompleto): cuando un producto del pedido no tiene stock suficiente o bien, cuando un robot no encuentra el producto en el almacén.
- CANCELLED (cancelado): cuando un cliente cancela el pedido antes de salir del almacén.
- ON_CONVEYOR (en cinta): cuando manda el pedido a los robots para que busquen los productos correspondientes.
- ON_DELIVERY (en entrega): cuando un repartidor ha cogido el pedido para entregarlo.
- DELIVERED (entrega): cuando un repartidor entrega el pedido con éxito.
- FAILED_DELIVERY (entrega fallida): cuando un repartidor no es capaz de entregar un pedido con éxito tras intentarlo 3 veces.

3. Implementación

A raíz de las ideas iniciales de los diagramas y los tipos de colas, empezamos a implementar el código. En nuestro caso, hemos dividido el código de la siguiente manera:

- Carpeta *classes*: donde se encuentran las distintas clases/tipos enumerados que componen el sistema.

- Scripts en carpeta principal: encargados de la ejecución base y prueba del sistema.

Nuestra herramienta principal es el editor de código Visual Studio y el lenguaje empleado es Python. Para los diagramas utilizamos la extensión de UMLet. Los diagramas fueron actualizados a medida que agregamos funcionalidad en el código.

Para la implementación del código de las colas nos basamos en el tutorial proporcionado en la práctica *https://www.rabbitmq.com/tutorials*. Asimismo, para probar el funcionamiento de éstas localmente, generamos un contenedor Docker con la imagen de RabbitMQ ejecutando el comando:

```
> docker run -d --name rabbitmq-container -p 5672:5672 -p
15672:15672 rabbitmq:latest
```

Todas las funciones del código se encuentran documentadas para un perfecto entendimiento del funcionamiento del sistema. Las colas se inicializan en la función *init()* de cada uno de los actores.

3.1. Diagrama de clases

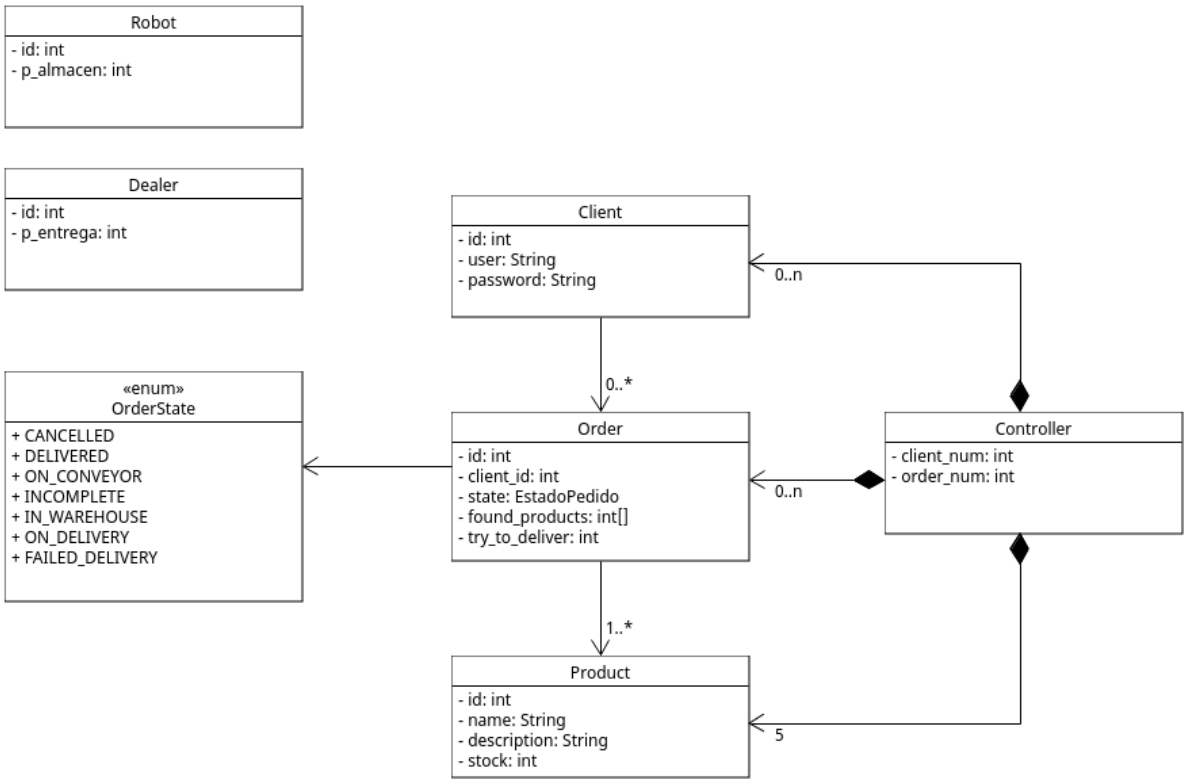


Figura 1: Diagrama de Clases

3.2. Diagrama de Casos de Uso

Nota: Para los diagramas de casos de uso, asumimos que todos los casos de uso del cliente, a excepción de “register” necesitan de “log in”, es decir, existiría una flecha “include” desde cada uno hacia “log in”. Esto es para facilitar la comprensión del diagrama y que sea más visible.

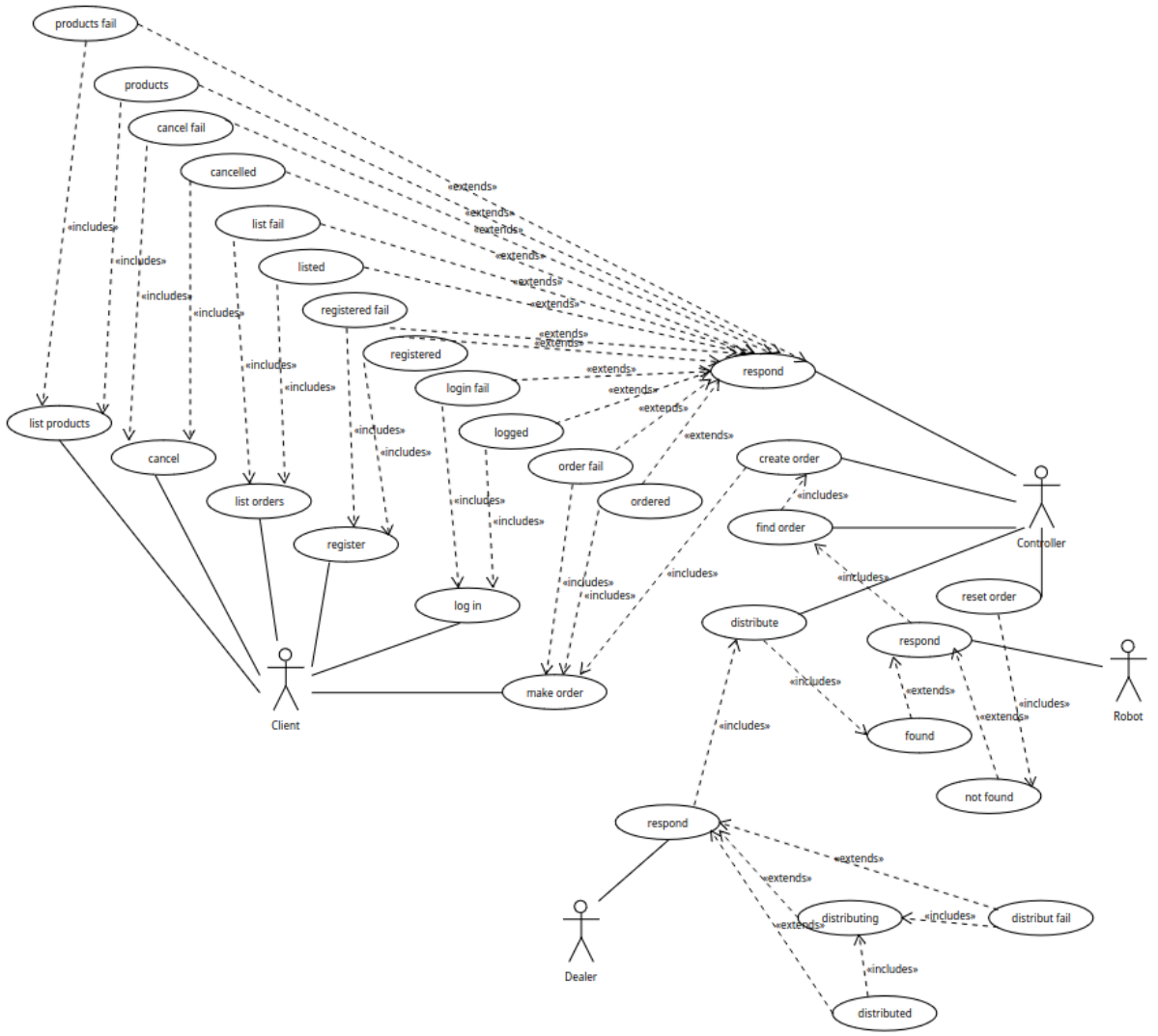


Figura 2: Diagrama de Casos de Uso general

3.2.1. Pedido completado hasta el final

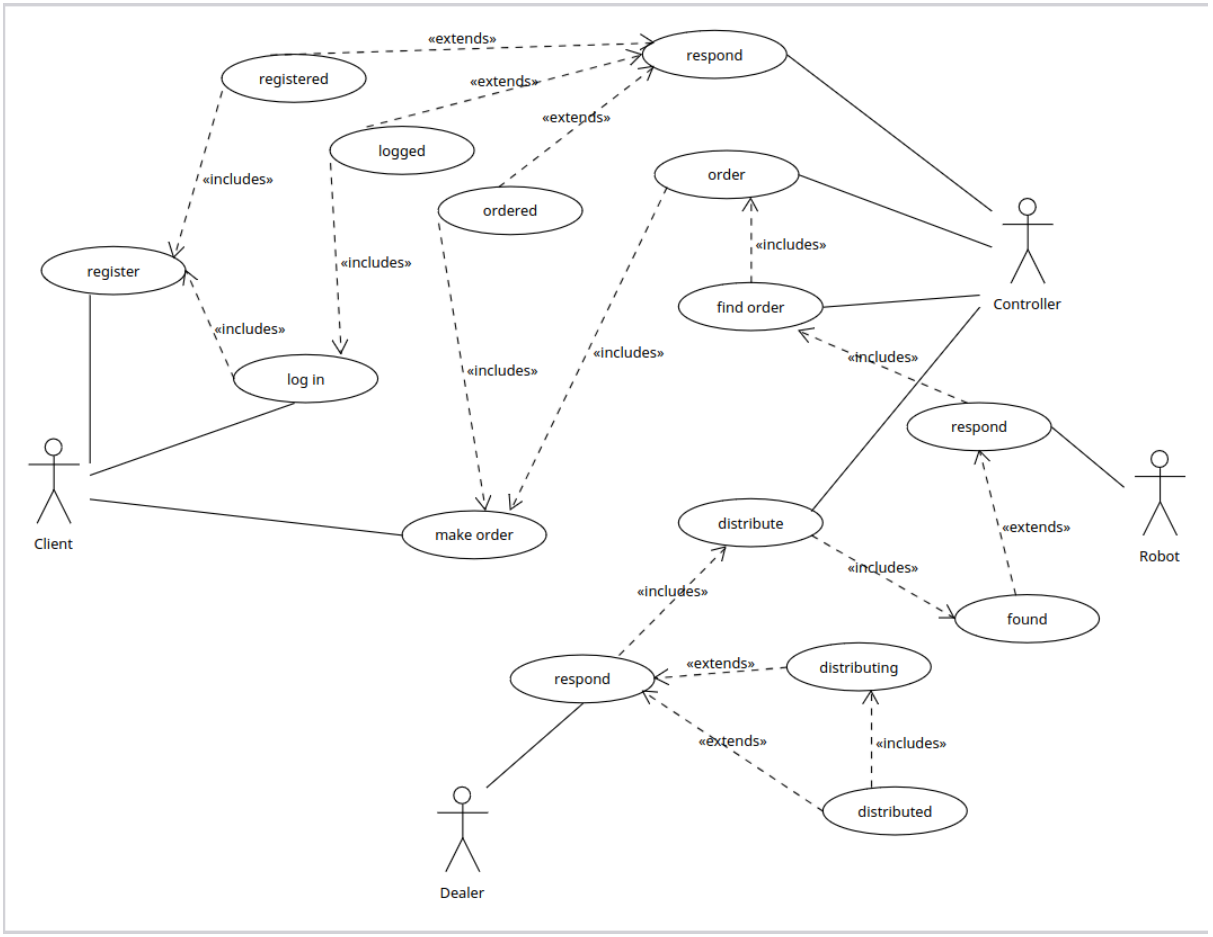


Figura 3: Diagrama de Caso de Uso 1

3.2.2. Pedido en el que el robot no encuentra el producto

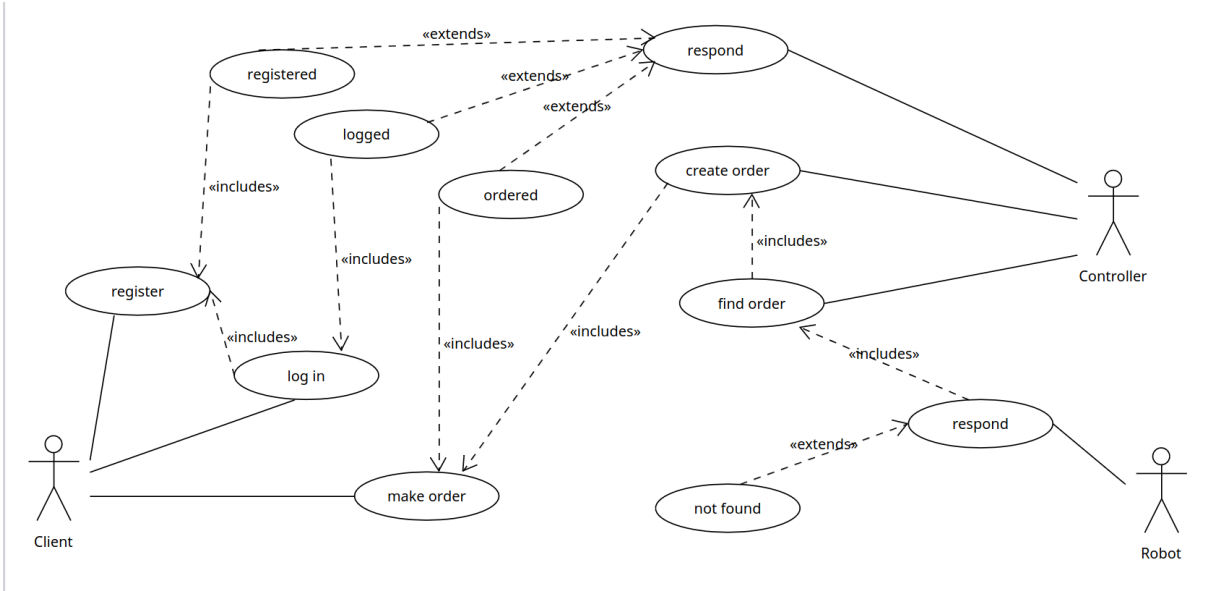


Figura 4: Diagrama de Caso de Uso 2

3.2.3. Pedido que se cancela antes de empezar el reparto

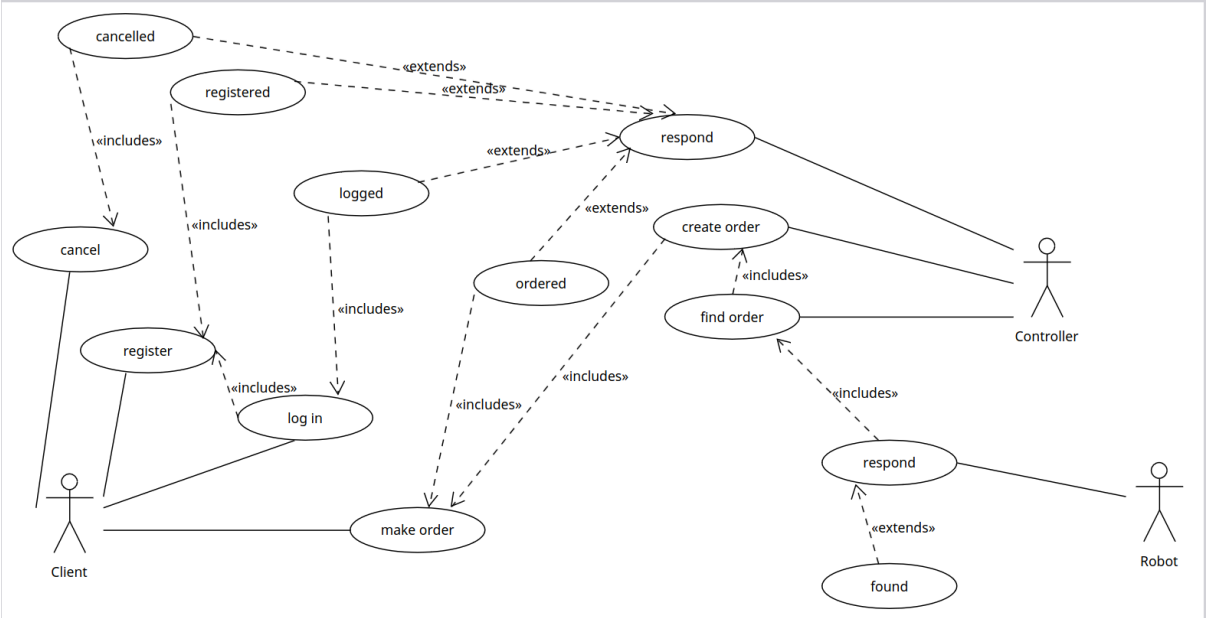


Figura 5: Diagrama de Caso de Uso 3

3.3. Diagrama de estados de un pedido

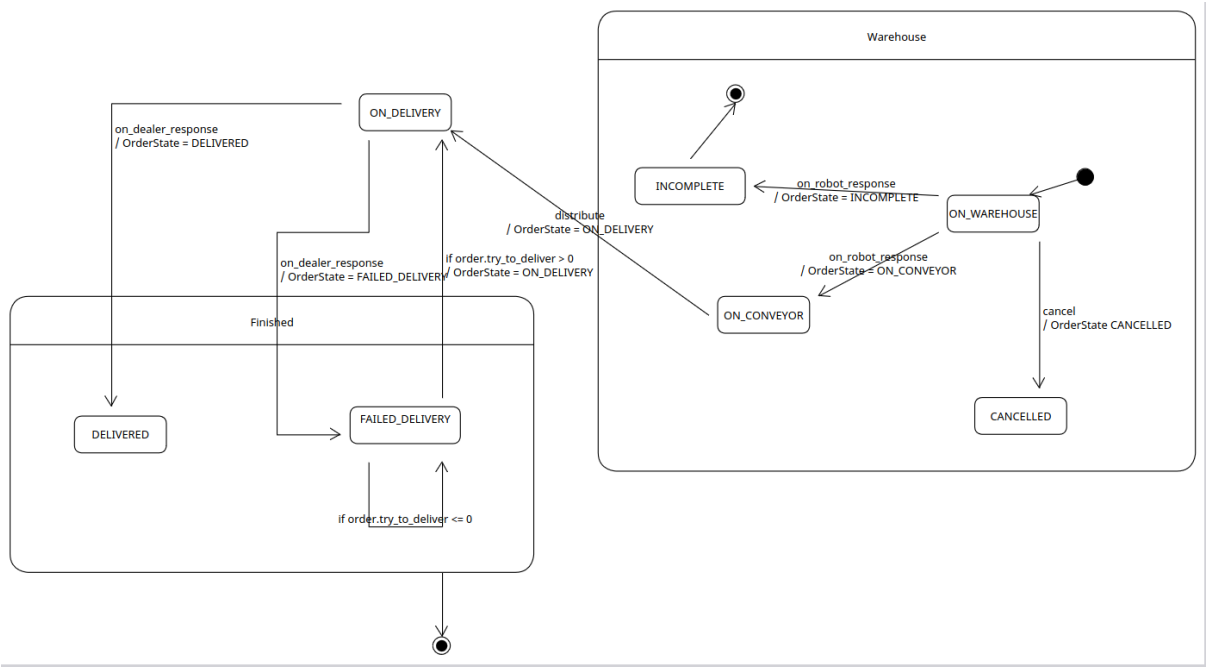


Figura 6: Diagrama de estados de un pedido

4. Conclusiones

En este proyecto hemos diseñado un sistema software que simula la actividad de un mercado online, donde los actores principales se comunican mediante un formato específico de mensajes para solicitar unas acciones diferentes entre ellos. Para dicha comunicacion se han utilizado distintos tipos de colas según las necesidades y decisiones de nuestra implementación. La estructura del código de las colas fue sencillo de utilizar gracias al tutorial de RabbitMQ proporcionado, al igual que la funcionalidad a términos generales gracias al diseño de unos diagramas iniciales.

Hemos aprendido a utilizar colas de mensajes y manejar la funcionalidad de un actor tras recibir un mensaje específico. Asimismo hemos generado unos scripts en formade test para la funcionalidad de cada actor, por lo que la implementación se basaba principalmente en que los tests básicos funcionaran.

En cuanto a dificultades durante la práctica únicsmente podemos hablar del tiempo requerido para llevarla a cabo, ya que la funcionalidad no ha resultado compleja. Al tener que llevar a cabo un proyecto grande, debe de acompañar al código una documentación impecable y unos requisitos, objetivos y pautas de funcionamiento bien definidos, por lo que también hay que tener en cuenta eso para la estimación del tiempo de trabajo.

Todo lo aprendido en colas de mensajes es aplicable a muchos otros sistemas donde se requiere una comunicación simultánea entre elementos sin permanecer bloqueados esperando a una respuesta.