Práctica 2

Saimazoom

Redes de comunicaciones

Lía Adriana Castañeda Rosadio

Vasco Nunes

Introducción

Para la segunda práctica de REDES II, se codificó una simulación de un servicio de delivery. Para esto, se implementaron cuatro actores: **Cliente**, **Controlador**, **Robot** y **Delivery**. El cliente, realiza pedidos al controlador. Este, posteriormente, se los envía al robot y el robot una vez que ha buscado y encontrado los productos en el almacén se los envía a delivery para que realice la entrega. La comunicación mediante todos los actores es mediante colas de la librería ***pika***, cliente de ***RabbitMQ***.

Documentos de Diseño

Diagrama de Estados:

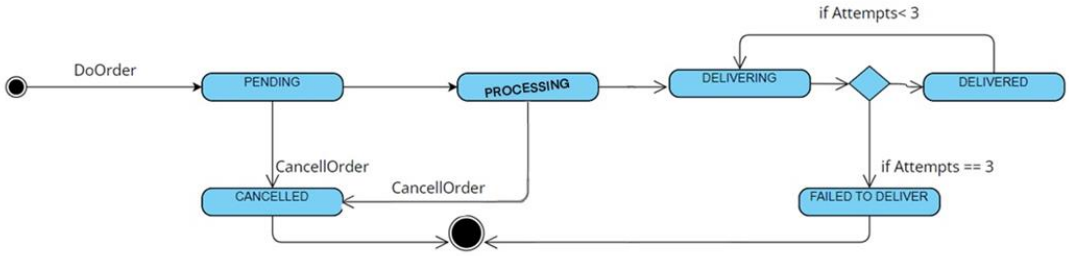
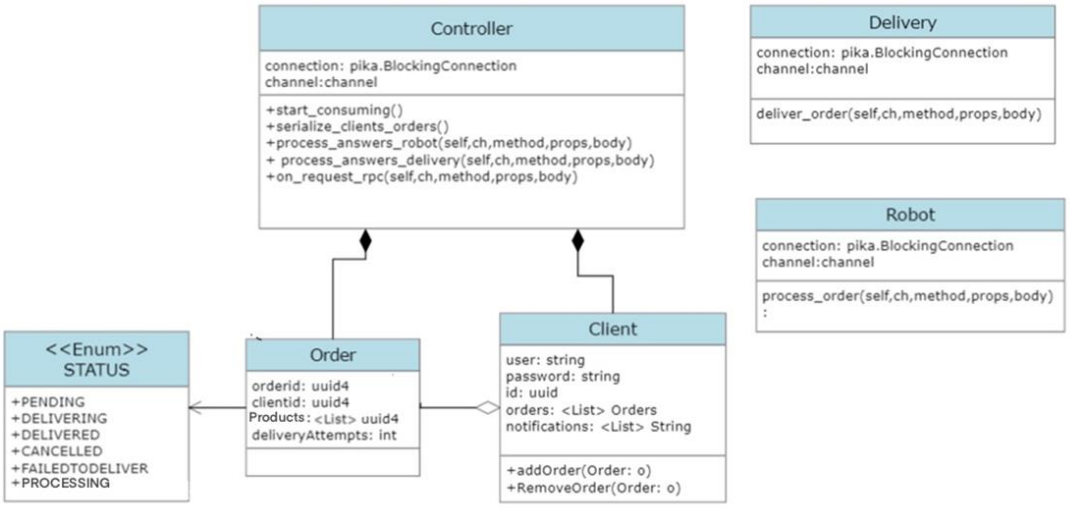


Diagrama de clases:



Casos de Uso

1. Pedido completado hasta el final

Actores involucrados: Controlador, Robot, Delivery, Cliente.

Resumen: El controlador recibe la orden del cliente y procede a enviársela al robot, con cada respuesta de cada actor, el controlador se encarga de cambiar el estado del pedido.

Pre-condiciones: El cliente tendrá que hacer un pedido.

Post-condiciones: El pedido se enviará al robot y luego a delivery, consecuentemente, el controlador recibirá mensajes de los actores encargados de procesar el pedido y este se encargará de modificar su estado.

Curso básico de eventos:

1. El controlador recibe el pedido hecho por el cliente, crea el objeto *Order* y se lo envía al robot
2. El robot recibe el pedido , notifica a controlador que ha recibido el producto mediante una cola exclusiva controlador-robot para que este pueda cambiar su estado a *PROCESSING*. el robot procesa cada producto durante un tiempo aleatorio entre 5 y 10 segundos.
3. Una vez procesados todos los productos, el robot avisa al controlador mediante la cola y el controlador envía la orden a Delivery. Delivery envía un mensaje al controlador mediante una cola exclusiva controlador-delivery para que este pueda cambiar el estado del pedido a *DELIVERING*.
4. El actor Delivery reparte el producto en un tiempo aleatorio, una vez terminado este tiempo, el controlador recibe un mensaje mediante la cola que el pedido ha sido entregado de manera exitosa, el controlador añade esta notificación al cliente y cambia el estado del pedido a *DELIVERED*.

2. Robot no encuentra producto

Actores involucrados: Controlador, Robot, Delivery, Cliente.

Resumen: El controlador recibe la orden del cliente y procede a enviársela al robot, con cada respuesta de cada actor, el controlador se encarga de cambiar el estado del pedido.

Pre-condiciones: El cliente tendrá que hacer un pedido.

Post-condiciones: El pedido se enviará al robot y el robot a delivery, consecuentemente, el controlador recibirá mensajes de los actores encargados de procesar el pedido.

Curso básico de eventos:

1. El controlador recibe el pedido hecho por el cliente, crea el objeto *Order* y se lo envía al robot
2. El robot recibe el pedido , notifica a controlador que ha recibido el producto mediante una cola exclusiva controlador-robot para que este pueda cambiar su

estado a *PROCESSING*. el robot procesa cada producto durante un tiempo aleatorio entre 5 y 10 segundos. Un producto no es encontrado, entonces el robot procede a comunicárselo al controlador mediante la cola de mensaje controlador-robot que hay, el robot sigue buscando los demás productos en el almacén y hace el mismo procedimiento para cada producto que no encuentre en el almacén.

3. El controlador recibe el mensaje del robot de que no se ha encontrado un producto y procede a avisarle al cliente mediante sus notificaciones (síncrono).
4. Una vez procesados todos los productos, el robot avisa al controlador que ha terminado de buscar los productos mediante la cola, el controlador luego envía la orden a Delivery y este envía un mensaje al controlador mediante una cola exclusiva controlador-delivery cuando esté atendiendo dicha orden para que este pueda cambiar el estado del pedido a *DELIVERING*.
5. El actor Delivery reparte el producto en un tiempo aleatorio, una vez terminado este tiempo, el controlador recibe un mensaje mediante la cola exclusiva controlador-delivery que el pedido ha sido entregado de manera exitosa, el controlador añade esta notificación al cliente y cambia el estado del pedido a *DELIVERED*.

3. pedido que se cancela antes de empezar el reparto

Actores involucrados: Controlador, Robot, Cliente.

Resumen: El controlador recibe la orden del cliente y procede a enviársela al robot. El cliente cancela la orden antes de que esta llegue a repartirse.

Pre-condiciones: El cliente tendrá que hacer un pedido y luego cancelarlo.

Post-condiciones: El pedido se enviará al robot pero este se cancelará mientras esté siendo procesado.

Curso básico de eventos:

1. El controlador recibe el pedido hecho por el cliente, crea el objeto *Order* y se lo envía al robot
2. El robot recibe el pedido , notifica a controlador que ha recibido el producto mediante una cola exclusiva controlador-robot para que este pueda cambiar su estado a *PROCESSING*. el robot procesa cada producto durante un tiempo aleatorio entre 5 y 10 segundos.
3. El cliente envía petición de cancelar pedido a controlador.

4. Controlador verifica que el estado del pedido no sea en reparto y cambia el estado del pedido a CANCELLED.
5. Controlador notifica a cliente que pedido ha sido cancelado mediante la cola RPC, de comunicación asíncrona.
6. Una vez que el robot ha terminado de procesar el pedido, avisa al controlador, pero el controlador no enviará la orden a Delivery, pues ha sido cancelada.

Descripción de los mensajes

En la mayoría de casos los mensajes entre los actores están construidos en forma de string y cada dato del mensaje está separado con un espacio. La primera palabra del string está en mayúsculas y es siempre el identificador de la órden que se solicita para que la función de respuesta sepa qué tiene que hacer.

Mensajes entre cliente-Controlador (cola RPC):

- **Sentido Cliente --> Controlador**

  REGISTER usuario password

  LOGIN usuario password

  DOORODER usuario numeroProductos

  VIEWORDERS usuario

  CANCELORDER orderid

  SEENOTIFICATIONS usuario

- **Sentido Controlador --> cliente**

  LOGEDIN

  ERRORLOGIN

  REGISTERED

  ORDERCREATED

  ORDERS: ...List Orderids

  Order with id: orderid was cancelled

  Order with id: orderid cannot be cancelled

  NOTIFICATIONS: ...List Notifications

Mensajes entre Controlador-Robot:

- **Sentido Controlador --> Robot (cola robot_queue)**

  orderid ...List Productids

- **Sentido Robot --> Controlador (cola controller_queue_robot)**

  PROCESSING orderid

  NOTFOUND orderid productid

  PROCESSED orderid

Mensajes entre Controlador-Delivery:

- Sentido Controlador --> Delivery (cola delivery_queue)

  orderid

- **Sentido Delivery --> Controlador (cola controller_queue_delivery)**

  DELIVERING orderid

  NOTDELIVERED orderid

  DELIVERED orderid

Desarrollo técnico

Durante el desarrollo de la práctica se tomaron diversas decisiones con respecto al diseño, implementación, entre otros.

Cliente

Un cliente tiene que registrarse y luego loggearse previamente. Una vez loggeado el cliente tendrá las siguientes opciones:

- Make Order: Como no hace falta modelar productos, basta con el cliente ingrese el número de productos en su pedido e internamente el controller le asignará un id (uuid4) a cada producto.
- See Order
- Cancell Order
- See Notifications : Cada vez que el pedido cambie de estado o no se encuentre un producto en el almacén se añadirá un aviso en las notificaciones del cliente y este podrá verlo en sus notificaciones.

Colas de mensajes

Todo el sistema cuenta con 5 colas en total:

- **RPC_queue:**

  Utilizada por el cliente y controlador. Es una cola que permite comunicación en doble sentido e identificar cada petición con un correlation id, es por ello que se ha escogido este modelo de cola.

- **Controller_queue_robot**

  Utilizada por el controlador y robot. La cola permite que el robot le notifique al controlador que ha empezado a procesar una orden, que no ha encontrado un producto o que ha terminado de procesar una orden.

- **controller_queue_delivery**

  Utilizada por el controlador y delivery. La cola permite que el delivery le notifique una vez que ha recibido una orden para ser entregada, que no ha podido entregar una orden o que ha entregado con éxito una orden.

- **robot_queue**

  Utilizada por el controlador y robot. La cola permite que el controlador le envíe al robot la orden de procesar un producto.

- **delivery_queue**

  Utilizada por el controlador y delivery. La cola permite que el controlador le la orden a delivery de entregar un producto.

Importante: Tenemos en cuenta que la cola controller_queue_delivery y controller_queue_robot pudieron haberse implementado en una sola, sin embargo decidimos separarlas, para que la carga sea menos y se divida en dos colas.

Serialización

Las órdenes y clientes del controlador se persisten utilizando la librería pickle, que debe ser previamente instalado en el entorno de Python.

Ejecución de la práctica

La práctica cuenta con una variable global (hostname) en todos los actores donde se puede modificar el nombre de servidor host de las colas utilizadas a local host o redes.ii.uam.es.

La práctica se puede práctica se puede probar a través de las siguientes formas:

- Lanzando todos los actores y commandline_client.py: se tiene que ejecutar launch_robot.py, launch_controller.py y launch_delivery.py y finalmente commandline_client.py. Se podrá ingresar órdenes, ver órdenes, cancelar órdenes y ver notificaciones por linea de comandos. Previamente el cliente deberá registrarse y luego loggearse.

- Lanzando todos los actores y launch_client.py: se tiene que ejecutar launch_robot.py, launch_controller.py y launch_delivery.py y finalmente launch_client.py. Launch_client crea 1 cliente y lanza 3 órdenes con diferente número de productos, se hacen 10 sleeps de 15 segundos y luego de cada uno se envían mensajes de petición a la cola RPC para mostrar notificaciones y estado de los pedidos.

- Lanzando el script de bash: Se ejecuta todo el sistema ejecutando ./launch_system.sh. Este script lanza un controlador, dos robots, dos deliverys y el launch_client.py en terminales separadas (procesos separados). Cada terminal lleva el nombre del actor correspondiente.

Importante: El comando de cancelar una orden podrá probarse únicamente ejecutando commandline_client.py, pues se necesita visualizar las órdenes (petición de View Orders a cola RPC de controlador) para poder saber el id (uuid4) del pedido a cancelar.

Conclusión

En esta práctica aprendimos a trabajar con distintos tipos de colas del servidor RabbitMQ. El sistema implementado funciona y si lanza más de un robot y delivery, el tiempo de procesamiento es aún mejor. Dentro de las dificultades encontradas está que al principio no sabíamos cómo utilizar esta librería por lo que tuvimos que leer tutoriales. Otra dificultad encontrada, fue el desconocimiento de cómo se utilizaba la librería pickle y nos tomó un poco de tiempo conseguir que los clientes y órdenes se serializaran correctamente.