P2 - Saimazoom

Sitio: MOODLE DE GRADO 2022/23
Curso: REDES DE COMUNICACIONES II
Libro: P2 - Saimazoom

Imprimido por: Pablo Sanchez Fernandez del Pozo
Día: jueves, 16 de marzo de 2023, 15:13

Descripción

Objetivo

En esta práctica el objetivo es implementar un sistema distribuido sencillo usando RabbitMQ (colas de mensajes). Hay que implementar toda la cadena logística de un marketplace llamado Saimazoom. En este escenario existen los siguientes actores:

- **Cliente**, que realiza y gestiona pedidos de productos.
- **Controlador** central, que gestiona todo el proceso.
- **Robots**, que se encargan de buscar los productos en el almacén y colocarlos en las cintas transportadoras.
- **Repartidores**, encargados de transportar el producto a la casa del cliente.

Para ello, será necesario implementar las operaciones habituales en este tipo de escenarios, que se realizarán en el siguiente orden: creación/consulta/cancelación de un pedido. La cancelación del pedido, solo se permite antes que el pedido haya empezado a ser tramitado por un repartidor. Se entrega un documento de requisitos, a completar, para definir de forma más clara las funcionalidades necesarias.

Para poder llevar a cabo la práctica, deberás comenzar realizando una serie de decisiones de diseño (documentándolas adecuadamente a través de un diagrama de clases, diagrama de estados) que incluirán, como mínimo, las siguientes:

- Diagrama de estados en los que puede estar un pedido en cada momento
- Diagrama de clases
- Formato y sintaxis de los mensajes (RFC)

Consejos de desarrollo

- Hay que entender bien qué es necesario implementar y qué no. Por ejemplo, no hace falta llevar un registro de los productos que hay en el almacen, podemos simular que un robot no encuentra un producto con una probabilidad.
- Para facilitar el desarrollo, puede utilizarse el siguiente truco: crear los distintos actores, en lugar de como programas independientes, como hilos dentro del proceso principal (controlador). En cualquier caso, antes de entregar la práctica debe revertirse el cambio.
- Siempre es buena idea atacar el desarrollo dividiendo el problema en partes, para aislar y reducir las posibles fuentes de errores. Así, para desarrollar y probar el repartidor, por ejemplo, se pueden generar e insertar en la cola mensajes de prueba, de forma que no haga falta que esté ya implementado ningún otro módulo.

Calidad del código

- El código fuente debe estar correctamente comentado
- Se valorará que el sistema se organice como un paquete de Python
- Se deben implementar un número suficiente de tests unitarios. Idealmente, se recomienda utilizar una metodología de desarrollo TDD (Test-Driven Development)

Tabla de contenidos

1. Funcionalidad

Para explicar mejor la funcionalidad a desarrollar la dividimos entre los siguientes apartados:

- Controlador
- Actores externos: cliente, robot y repartidor
- Pedidos
- Colas de mensajes
- Mensajes

1.1. Controlador

El controlador es el principal actor del sistema y se encarga de gestionar los pedidos y la comunicación con el resto de actores. Son necesarias varias colas para dicha comunicación, como por ejemplo la cola para el trabajo de robots o de los repartidores, pero seguramente hagan falta más. Es necesario diseñar el controlador definiendo qué colas son necesarias y cómo será la interacción con el resto de actores del sistema.

Respecto a los pedidos, el controlador tiene que tener un registro de qué pedidos han sido realizados y llevar un control de su estado. También hay que tener un registro de los clientes y de qué pedidos ha realizado cada uno. Si un cliente no registrado intenta hacer un pedido, el sistema debe contestar con un mensaje indicando que no está registrado. Sin embargo, NO hace falta modelar los productos, no hay que tener un registro de productos válidos.

Por último, es necesario que el sistema persista su estado en todo momento, para lo que pueden utilizarse un par de opciones: i) estado en ficheros (utilizando la librería Pickle, o formato CSV) o,l ii) usando una base de datos al lanzar el controlador; en este caso también hay que guardar los datos antes de terminar la ejecución.

1.2. Actores Externos

Los siguientes actores están involucrados en el sistema de Saimazoom.

Clientes

Los clientes tienen que registrarse en el servidor y, una vez registrados, pueden hacer pedidos, ver sus pedidos y cancelar un pedido. Es necesario también crear una interfaz para línea de comandos (no gráfica) para los clientes, para poder simular el comportamiento de uno desde la terminal.

Robots

Los robots tienen que escuchar peticiones que el servidor coloca en una cola y simular que se realiza un trabajo. Este trabajo sería el de mover el producto del almacén a la cintra transportadora y lo simulamos con una espera aleatoria de entre 5 y 10 segundos. Además, simularemos que el robot encuentra el producto en el almacén con una probabilidad configurable (*p_almacen*). En caso de éxito hay que confirmar al controlador la finalización del trabajo, y en caso de no encontrar el producto solicitado hay que enviar un mensaje al controlador comunicando este problema.

Repartidores

Los repartidores entran en acción una vez que los robots han terminado su trabajo, y se encargan de hacer llegar el pedido al domicilio del cliente. Cada repartidor intenta entregar el paquete tres veces, cada uno con una probabilidad de éxito configurable (*p_entrega*), y manteniendo siempre al controlador actualizado con el estado del pedido. Cada intento de entrega se simula con una espera aleatoria de entre 10 y 20 segundos.

1.3. Pedidos

Un pedido tiene que estar definido por, al menos, los siguientes atributos:

- El ID del cliente que realiza el pedido.
- Los IDs de los productos que lo conforman.
- El estado de dicho pedido en función de en qué etapa del proceso de envío se encuentra (ej.: en el almacén, en la cinta...)

Es necesario definir qué posibles estados puede tener un pedido y realizar un diagrama de estados que muestre las transiciones entre ellos. La cancelación de un pedido solo es posible mientras el mismo se encuentre aún dentro de las instalaciones de Saimazoom, es decir, antes de que comience a ser procesado por un repartidor. Los productos de un pedido cancelado vuelven inmediamente al almacén.

1.4. Colas de mensajes

Para la implementación de este sistema es necesario utilizar colas de mensajes; para ello, utilizaremos RabbitMQ: https://www.rabbitmq.com/ .

Se recomienda leer los tutoriales de https://www.rabbitmq.com/getstarted.html , y decidir qué esquemas (work queues, publish/subscribe, ...) son los que tenemos que aplicar en esta práctica.

Para las pruebas se puede lanzar un servidor de RabbitMQ en local con la imagen de docker que ofrece la página (https://registry.hub.docker.com/_/rabbitmq/). Sin embargo, para la entrega se debe usar el servidor que será desplegado en TODO.

Para que en el servidor compartido no existan conflictos en los nombres de las colas de mensajes, poned a todas como prefijo GRUPO-PAREJA_ y después el nombre de la cola, por ejemplo: 2321-01_robots.

También, como habrá problemas durante el desarrollo, añadir los siguientes argumentos a `queue_declare`:

- `durable=False` : para que las colas no se guarden tras el reinicio del servidor
- `auto_delete=True` : para que la cola se borre después de que los consumidores se desconecten

1.5. Mensajes

Para realizar esta práctica es necesario definir los mensajes (la sintaxis) que utlizarán cada uno de los actores.

Por ejemplo, para el registro de un cliente los mensajes podrían ser los siguientes (pero se puede cambiar):

Mensaje de registro:

```
REGISTER client_id
```

Mensaje de respuesta (exitosa):

```
REGISTERED client
```

Otro ejemplo, los mensajes para que un robot mueva un producto (asociado a un pedido) del almacén a la cinta transportadora:

Mensaje de registro:

```
MOVE order_id
```

Mensaje de respuesta (exitosa):

```
MOVED order_id
```

Para completar la práctica es necesario hacer un documento (tipo RFC) explicando todos los mensajes definidos.

2. Entregables

Además del código es necesario entregar diversos documentos y diagramas para esta práctica.

Código

Centrándonos en la parte de código, es necesario entregar:

Obligatorio

- La implementación de todos los actores involucrados: controlador, cliente, robot y repartidor
- Un script de python que lance el controlador (launch_controller.py)
- Un script de python que lance un cliente y simule una serie de acciones (launch_client.py)
- Un script de python para simular un cliente desde línea de comandos (commandline_client.py)
- Un script de python que lance un robot (launch_robot.py)
- Un script de python que lance un repartidor (launch_delivery.py)
- Scripts de bash que simulen situaciones "reales" en los que se utilicen el resto de scripts de Python, que se deben ejecutar desde distintos procesos.

Recomendado implementar

- Scripts en python que simule situaciones "reales" lanzando actores en distintos hilos
- Test unitarios para cada uno de los actores

Otros (No código)

Es necesario entregar los siguientes documentos:

- Diagrama de clases
- Diagrama de estados de un pedido
- Tres casos de uso (pedido completado hasta el final, pedido en el que el robot no encuentra el producto, pedido que se cancela antes de empezar el reparto) desde el punto de vista del controlador
- Descripción de los mensajes (sintaxis y formato)
- Documento de requisitos completado incluyendo los diagramas anteriores y descripción de mensajes en la parte de implementación de la solución.

3. Recursos y librerías

Como es habitual en estas prácticas, puede utilizarse cualquier librería para la realización de las mismas, pero se recomiendan, al menos, las siguientes:

- Librería pika, para utilizar RabbitMQ en Python
- Librería uuid, para generar IDs de forma automática
- Librería argparse, para el análisis de la línea de comandos (para el cliente)

Se aporta también una plantilla a rellenar del documento de requisitos para Saimazoom.