# REDES DE COMUNICACIÓN II
# PRÁCTICA 2

***AUTORES***
***CARLOS GARCÍA SANTA***

GRUPO2323: PAREJA 04

INGENIERÍA INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD AUTÓNOMA DE MADRID

28/06/2024

# Índice



# Introducción

El objetivo de Saimazoom es el de crear un sistema para la gestión de pedidos online. Este sistema debe incluir a los 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

El sistema debe de gestionar las interacciones entre todos estos actores, para las comunicaciones correspondientes se emplea una cola de mensajes.

## 1. Definición del Proyecto: Sistema Saimazoom

El sistema Saimazoom es una plataforma integral que gestiona pedidos de productos solicitados por los clientes. La secuencia de operaciones comienza cuando un cliente realiza un pedido. Posteriormente, el controlador central notifica a un robot para mover el producto del almacén a la cinta transportadora. Una vez el producto está en la cinta, el controlador informa a un repartidor, quien se encarga de entregar el producto en la residencia del cliente.

### 1.1 Objetivos y Funcionalidad

Los objetivos principales del sistema Saimazoom son:

- **Gestión de Pedidos de Clientes**: Los clientes tienen la capacidad de hacer pedidos, consultarlos y cancelarlos según sea necesario.
- **Gestión de Robots**: Los robots reciben órdenes de transportar productos desde el almacén hasta la cinta transportadora.
- **Gestión de Repartidores**: Los repartidores son responsables de llevar los productos desde la cinta transportadora hasta la casa de los clientes.
- **Control Central**: El controlador central supervisa todos los productos, clientes, robots y repartidores. También se encarga de registrar y monitorizar el estado de los pedidos en relación con las actividades de los demás actores.
- **Comunicaciones**: Se mantiene una comunicación fluida entre el controlador y los demás actores involucrados en el proceso.

Para alcanzar estos objetivos, se desarrollarán las siguientes funcionalidades esenciales:

1. **Registro de Cliente**: Proceso mediante el cual un cliente se registra proporcionando un identificador único.
2. **Registro de Pedido**: El controlador central registra cada pedido en su base de datos asignando un identificador de cliente y de producto, y establece el estado del pedido.
3. **Recepción de Pedidos de Clientes**: Se deben recibir y registrar los pedidos que los clientes realizan, asociados a un producto específico.
4. **Asignación de Trabajo a Robots**: Se asignan tareas a los robots para el transporte de productos correspondientes a los pedidos.

5. **Asignación de Trabajo a Repartidores**: Se asignan tareas a los repartidores para la entrega de productos a los clientes.

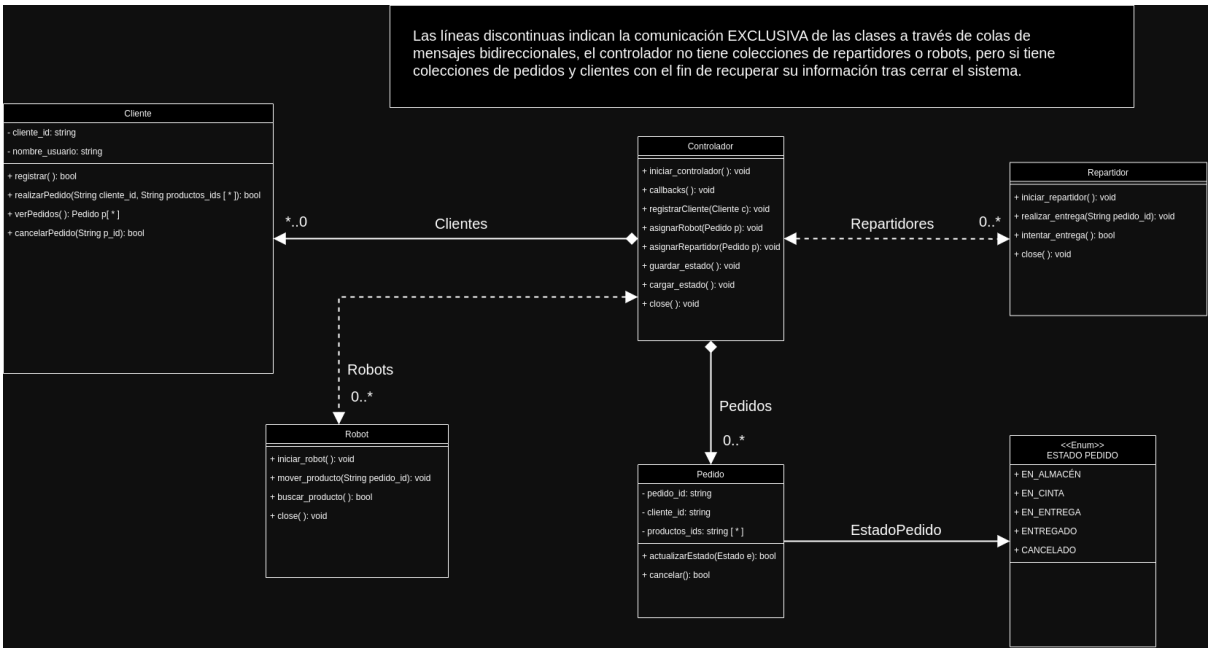
### 1.2 Requisitos Funcionales

Los requisitos funcionales se dividen en los siguientes aspectos relacionados con la lógica de clientes:

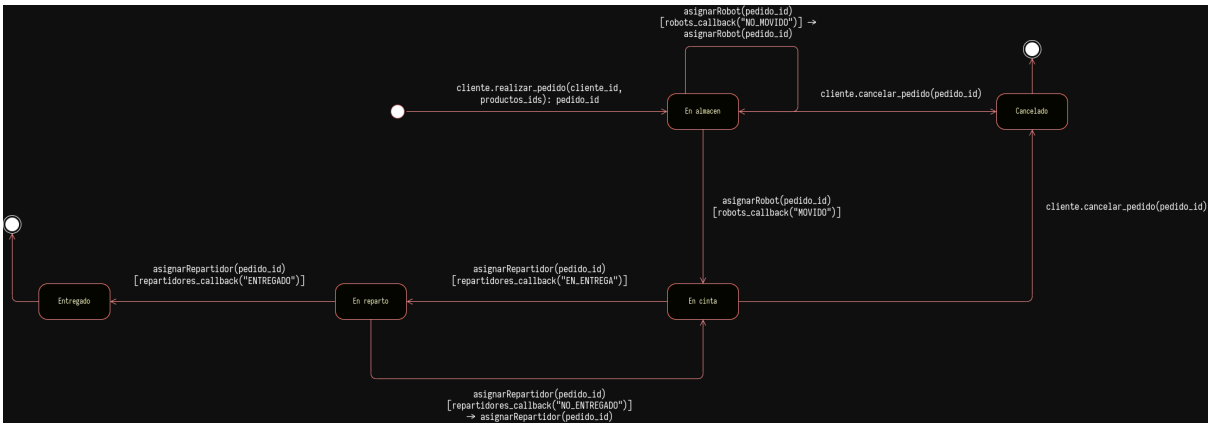
- **LoCI1**: Registro en la aplicación con confirmación de recepción.
- **LoCI2**: Realización de un pedido incluyendo solicitud de un producto.
- **LoCI3**: Consulta de la lista de pedidos realizados que muestra el identificador del producto y el estado del pedido.
- **LoCI4**: Solicitud para la cancelación de un pedido.

Estos elementos conforman la estructura básica y las operaciones clave del sistema Saimazoom, asegurando un flujo de trabajo eficiente y organizado desde la solicitud inicial del cliente hasta la entrega final del producto.

### 1.3 Diagrama de clases



### 1.4 Diagrama de estados




### 1.5 Casos de uso

**Caso de Uso 1: Pedido Completado hasta el Final**

**Descripción:** Este caso de uso sigue el flujo completo de un pedido desde la creación hasta la entrega exitosa al cliente.

**Precondiciones:** El cliente debe estar registrado previamente.

**Flujo de Eventos:**

1. **Creación del Pedido:**
   - El cliente envía una solicitud de pedido (`REALIZAR_PEDIDO cliente_id: x, productos_ids: [1, 2]`) al controlador.
   - El controlador registra el pedido y responde al cliente que lo ha realizado con el id del pedido creado (`PEDIDO_REALIZADO pedido_id`).
   - El controlador envía una tarea a la cola de robots para recoger el producto (`MUEVE pedido_id`).
2. **Recogida por Robot:**
   - Un robot recoge la tarea de la cola, intenta encontrar el producto en el almacén y tiene éxito (si un número aleatorio entre 0 y 100 es menor que `P_ALMACEN`).

    - El robot informa al controlador que el producto ha sido movido (`MOVIDO pedido_id`).
3. **Preparación para Envío:**
    - El controlador actualiza el estado del pedido a `En cinta`.
    - El controlador envía una tarea `ENTREGA pedido_id` a la cola de repartidores para entregar el pedido .
4. **Entrega por Repartidor:**
    - Un repartidor recoge la tarea y envía a la cola del controlador un mensaje `EN_ENTREGA pedido_id` para actualizar el estado del pedido a `EN_ENTREGA`, intenta entregar el paquete y tiene éxito en uno de sus tres intentos (si un número aleatorio entre 0 y 100 es menor que `P_ENTREGA`).
    - El repartidor informa al controlador que el pedido ha sido entregado (`ENTREGADO pedido_id`).
5. **Confirmación de Recepción:**
    - El controlador actualiza el estado del pedido a "entregado".

**Postcondiciones:**

- El pedido está completado.

**Caso de Uso 2: Pedido en el que el Robot no Encuentra el Producto**

**Descripción:** Este caso de uso describe la situación en la que un robot no puede encontrar el producto solicitado en el almacén.

**Flujo de Eventos:**

1. **Creación del Pedido:**
    - Igual al caso de uso 1, el cliente realiza un pedido y el controlador lo procesa.
2. **Fallo en la Recogida por Robot:**
    - El robot intenta encontrar el producto pero falla (probabilidad `P_ALMACEN`).
    - El robot informa al controlador que el producto no está disponible (`NO_MOVIDO pedido_id`).
3. **Reasignación de pedido a robot:**
    - El controlador reasigna el pedido a cualquier robot que esté disponible, volviendo a enviar un mensaje a la cola de robots.
4. **Continuación de flujo normal:**
    - Tras la reasignación del pedido si el robot consigue encontrar el producto, se sigue el flujo del caso de uso 1 desde el paso 2.

**Postcondiciones:**

- El pedido está completado.

**Caso de Uso 3: Pedido Cancelado antes de Empezar el Reparto**

**Descripción:** Este caso de uso ocurre cuando un cliente decide cancelar un pedido antes de que este haya salido para la entrega.

**Flujo de Eventos:**

1. **Cancelación del Pedido:**
    - Antes de que el pedido sea asignado a un repartidor, es decir, si el pedido se encuentra en la cinta o en el almacen, el cliente solicita la cancelación (`CANCELAR pedido_id`).
    - El controlador verifica que el pedido aún está en el almacén o en la cinta y procede a cancelarlo.
2. **Actualización y Notificación:**
    - El controlador actualiza el estado del pedido a "cancelado".

**Postcondiciones:**

- El pedido está cancelado y el estado es actualizado.
- El pedido puede volver a solicitarse.

## 2. Implementación

Las colas se han implementado directamente en las clases de tal manera que se encapsula el funcionamiento del sistema dentro de las clases, esto tiene ventajas y desventajas: entre las ventajas destaca la facilidad de uso ya que la abstracción hace muy sencilla la programación del sistema en los scripts y en un escenario real proporcionaría una API con la que desarrollar rápidamente el sistema, sin embargo, una desventaja importante sería la flexibilidad, ya que si se quisiese cambiar el sistema de comunicación de colas por otro, habría que modificar el comportamiento de las clases y la refactorización sería costosa en cuanto a tiempo.
Por otro lado, como indica el enunciado no se ha implementado un registro de productos, con lo cual no ha sido necesaria implementar la clase Producto y simplemente se ha simulado el procesamiento de los productos, además por simplicidad no se ha tenido en cuenta el número de productos de un pedido cuando se ha realiza la búsqueda de los mismos.
Para la persistencia de datos se ha utilizado pickle para guardar y cargar el estado de los clientes y los pedidos, que son los objetos de los que el controlador guarda el estado, el sistema no es muy complejo y no es un requisito obligatorio usar una base de datos, con lo cual se ha utilizado el sistema más sencillo.

### 2.1 Definición de mensajes

#### 2.1.1. Formato de los Mensajes

Los mensajes en el sistema Saimazoom siguen un formato con una cabecera constituida por la acción a realizar y luego campos clave-valor para facilitar la lectura y el procesamiento por parte de los actores del sistema. Los mensajes son enviados en texto plano con campos separados por comas, donde cada campo tiene un identificador único seguido por su valor correspondiente.

#### 2.1.2. Descripción de Mensajes

**Mensajes de Cliente a Controlador**

- **REGISTRAR**
  - **Sintaxis**: `REGISTRAR nombre_usuario: <nombre_usuario>`
  - **Descripción**: Utilizado por un cliente para registrarse en el sistema. El cliente envía su nombre de usuario deseado al controlador.
- **REALIZAR_PEDIDO**
  - **Formato**: `REALIZAR_PEDIDO cliente_id: <id>, productos_ids: [<id1>, <id2>, ...]`
  - **Descripción**: Un cliente envía este mensaje para solicitar la creación de un pedido. Incluye el identificador del cliente y la lista de identificadores de productos.
- **VER_PEDIDOS**
  - **Sintaxis**: `VER_PEDIDOS <lista_de_pedidos>`
  - **Descripción**: Permite a un cliente ver detalles de sus pedidos pasados y actuales. La lista puede estar vacía, solicitando así todos los pedidos del cliente.
- **CANCELAR_PEDIDO**
  - **Formato**: `CANCELAR_PEDIDO pedido_id: <id>`
  - **Descripción**: Solicita la cancelación de un pedido antes de que este haya sido asignado a un repartidor.

**Mensajes de Controlador a Cliente**

- **REGISTRADO**
  - **Sintaxis**: `REGISTRADO nombre_usuario: <nombre_usuario> cliente_id: <cliente_id>`
  - **Descripción**: Respuesta del controlador confirmando el registro exitoso del cliente, incluyendo el nombre de usuario y un identificador único asignado al cliente.

- **PEDIDO_REALIZADO**
  - **Formato**: `PEDIDO_REALIZADO pedido_id: <id>`
  - **Descripción**: El controlador envía este mensaje para confirmar que un pedido ha sido registrado exitosamente con el identificador asignado.
- **MOSTRAR_PEDIDOS**
  - **Sintaxis**: `MOSTRAR_PEDIDOS [<detalles_pedido_1>, <detalles_pedido_2>, ...]`
  - **Descripción**: Respuesta del controlador que lista los pedidos del cliente. Cada elemento de la lista contiene detalles de un pedido individual, incluyendo identificadores de productos y estados actuales.
- **PEDIDO_CANCELADO**
  - **Sintaxis**: `PEDIDO_CANCELADO <pedido_id>`
  - **Descripción**: Confirma la cancelación exitosa de un pedido.

**Mensajes de Controlador a Robot**

- **MUEVE**
  - **Formato**: `MUEVE pedido_id: <id>`
  - **Descripción**: Mensaje enviado para indicar a un robot que mueva un producto asociado a un pedido.

**Mensajes de Robot a Controlador**

- **MOVIDO**
  - **Formato**: `MOVIDO pedido_id: <id>`
  - **Descripción**: Confirma que el producto ha sido movido al destino especificado.
- **NO_MOVIDO**
  - **Formato**: `NO_MOVIDO pedido_id: <id>`
  - **Descripción**: Indica que el producto no fue encontrado o no pudo ser movido.

**Mensajes de Controlador a Repartidor**

- **ENTREGA**
  - **Formato**: `ENTREGA pedido_id: <id>`
  - **Descripción**: Asigna un pedido a un repartidor para su entrega.

**Mensajes de Repartidor a Controlador**

- **EN_ENTREGA**
  - **Formato**: `EN_ENTREGA pedido_id: <id>`

    - **Descripción**: Indica que el repartidor ha empezado el proceso de entrega.
- **ENTREGADO**
    - **Formato**: `ENTREGADO pedido_id: <id>`
    - **Descripción**: Confirma que el pedido ha sido entregado al cliente.
- **NO_ENTREGADO**
    - **Formato**: `NO_ENTREGADO pedido_id: <id>`
    - **Descripción**: Establece que el pedido no ha podido ser entregado al cliente.

## 3. Conclusiones

El uso de colas de mensajes en el sistema Saimazoom, a través de RabbitMQ, proporciona beneficios significativos para la gestión de procesos distribuidos. Al implementar colas, se logra un desacoplamiento efectivo entre los componentes del sistema, lo que permite que cada parte funcione independientemente sin afectar a las demás. Esto mejora la fiabilidad y la capacidad de recuperación del sistema, pues los mensajes se mantienen en la cola hasta ser procesados, reduciendo el riesgo de pérdida de datos en caso de fallos. Además, las colas facilitan la escalabilidad del sistema al permitir ajustar fácilmente el número de consumidores que procesan tareas durante picos de demanda. Finalmente, la naturaleza asincrónica de las colas mejora el rendimiento general al permitir que múltiples procesos se ejecuten simultáneamente, optimizando los tiempos de respuesta.