UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Práctica de simulador a medida - Supermercado con diferentes tipos de colas

SIM - FACULTAT D'INFORMÀTICA DE BARCELONA

Autores: Alex Catalan Muñoz
Aleix Vila Berenguel

Fecha: 25/03/2021

Índice

1. Descripción del sistema

Nuestro sistema se corresponde al de un supermercado, específicamente a la parte del pasar por caja para pagar los productos escogidos. El sistema, está formado por 6 cajeros, 3 de los cuales tienen la misma cola y los otros 3 tienen una cola para cada uno. Las personas, cuando llegan, hacen cola en la cola con menos cantidad de gente y una vez les llegue su turno, procederán a ejecutar el proceso de pago para poder irse del supermercado.

Queremos observar cual de los dos sistemas de colas será el mejor, si el de cola unificada o el de cola por cajero, teniendo en cuenta, tal y como se ha mencionado que los clientes entraran a la cola más vacía en el momento en el que llegan.

2. Entidades del sistema

En cuanto a las entidades que forman parte de nuestro sistema tenemos:

- **Entidades fijas:**
    - Colas
    - Cajeros
    - Entrada
- **Entidades móviles:**
    - Personas

3. Especificación del sistema

Para entender mejor el sistema, hemos realizado una especificación de nuestro modelo a través de un diagrama UML para observar las distintas entidades del modelo y cómo están relacionadas, un diagrama BPMN para observar cual es el flujo de procesos/eventos del sistema y un diagrama de estados para observar el comportamiento de las entidades a lo largo de la ejecución.

Diagrama UML

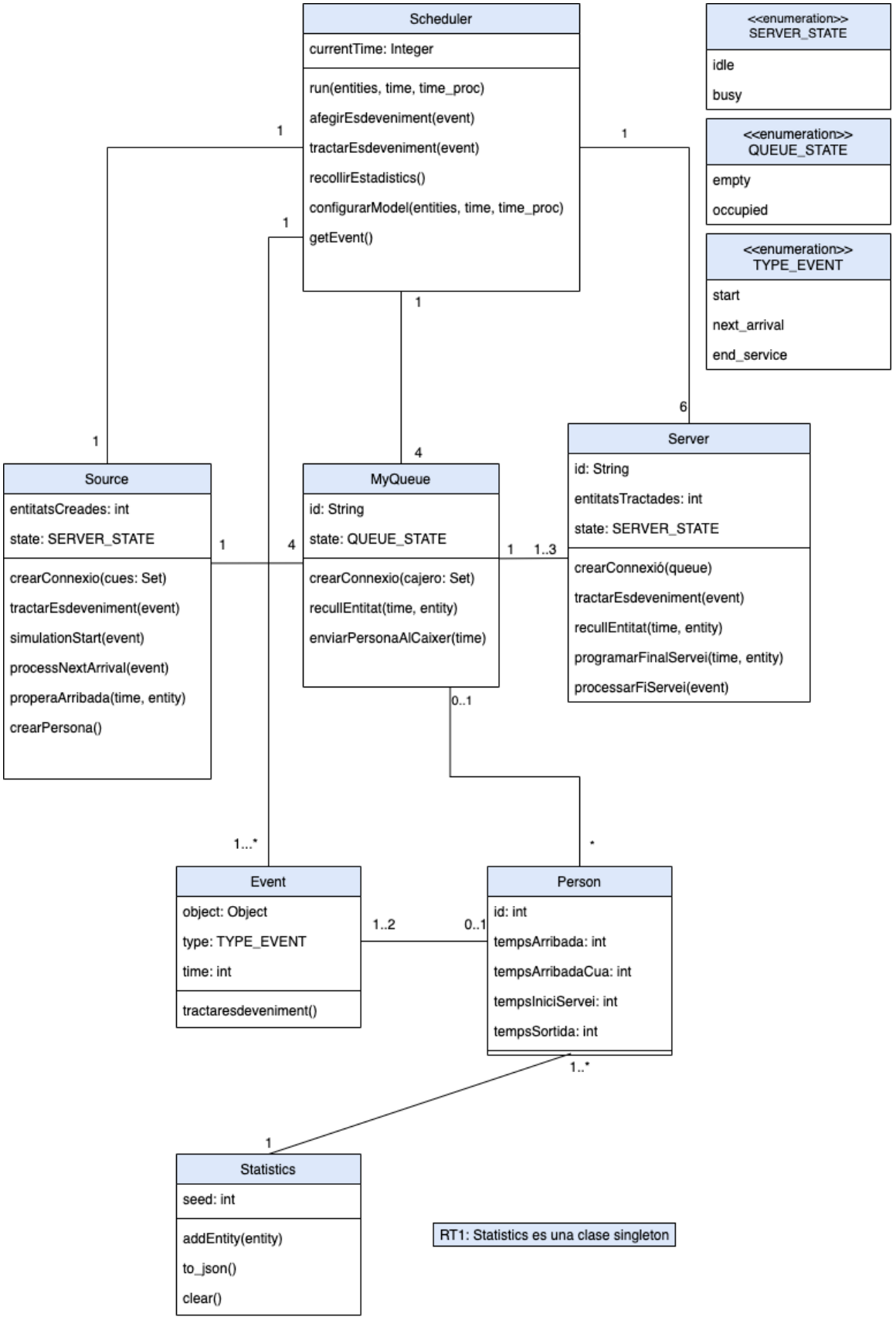

Figura 1: Diagrama UML de nuestro simulador.

En la figura anterior, podemos ver cuál es la estructura general de nuestro simulador a medida a través de un diagrama UML. Inicialmente, tenemos la clase scheduler, elemento base de un event scheduler, y encargado, tal y como indica su nombre, de ir programando los diferentes eventos del sistema a las entidades que lo componen.

Las entidades que se encargarán de ir ejecutando los eventos del sistema son:

- **Source**: Se encarga de hacer llegar a las personas a la zona de pago cuando le llega un evento de tipo “next_arrival” y de asignar la siguiente llegada al scheduler (evento de tipos “next_arrival).
- **MyQueue**: Paso intermedio entre el source y los cajeros. Se encargan de coger a las personas que van llegando al sistema en caso de que estas decidan ir a la cola y de enviar las personas al cajero correspondiente.
- **Server(cajero):** Recogen las personas de su cola cuando se encuentran vacíos y de procesar el pago de esa persona cuando llega un evento de tipo “end_service” de la persona que han recogido de la cola (el evento “end_service” lo ha asignado el propio server al scheduler al recoger a la persona de la cola).

Además, tenemos las clases **Event**, correspondiente a los diferentes eventos que se irán ejecutando en el sistema y que está compuesta por el objeto que lo va a ejecutar, la persona que la va a ejecutar y el tiempo en el que se va a ejecutar, la clase **Person**, que está compuesta de todas las personas que van a formar parte del sistema con toda la información relevante sobre ellas (que se va a ir actualizando a lo largo de la ejecución) y la clase **Statistics**, que está compuesta por las personas que toman parte de cada una de las ejecuciones que se desee realizar. Está última clase, nos ayudará a sacar los estadísticos del sistema una vez se haya acabado toda la ejecución.

Diagrama BPMN

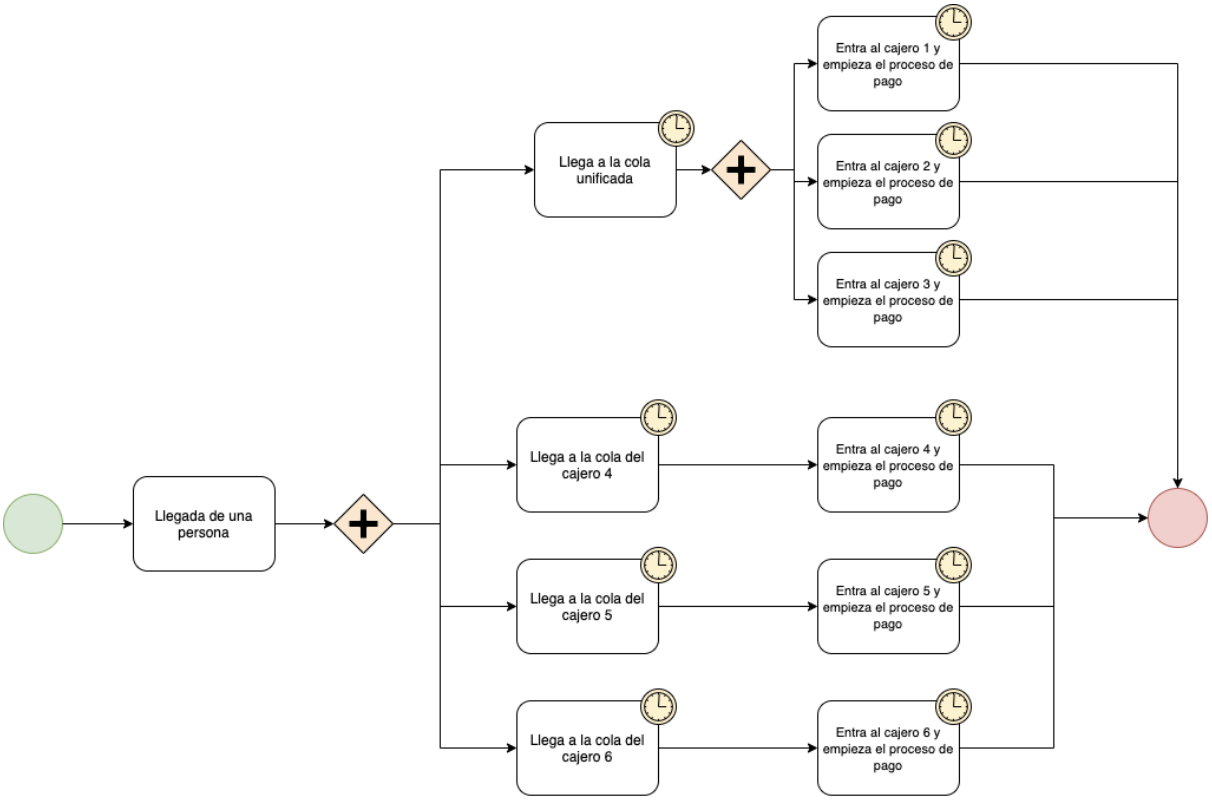

Figura 2: Diagrama BPMN de nuestro simulador.

En la **figura 2**, podemos observar cuál será el flujo de cada una de las personas que formen parte del sistema a través de un diagrama business process modelling and notation (BPMN). Se observa cómo al llegar la persona al sistema, esta entrará a una de las colas dependiendo del número de personas que haya en estas. Una vez esté en una de las colas, la persona pasará un tiempo indefinido en la cola a la espera de que alguno de sus cajeros esté vacío cuando se encuentre el primero de la cola. Una vez suceda lo anterior, la persona entrará al cajero correspondiente y pasará a realizar el proceso de pago de su compra (que durará un tiempo medio que podrá determinar el usuario) con el que acabará su recorrido en el sistema al finalizar el proceso.

Diagrama de estados

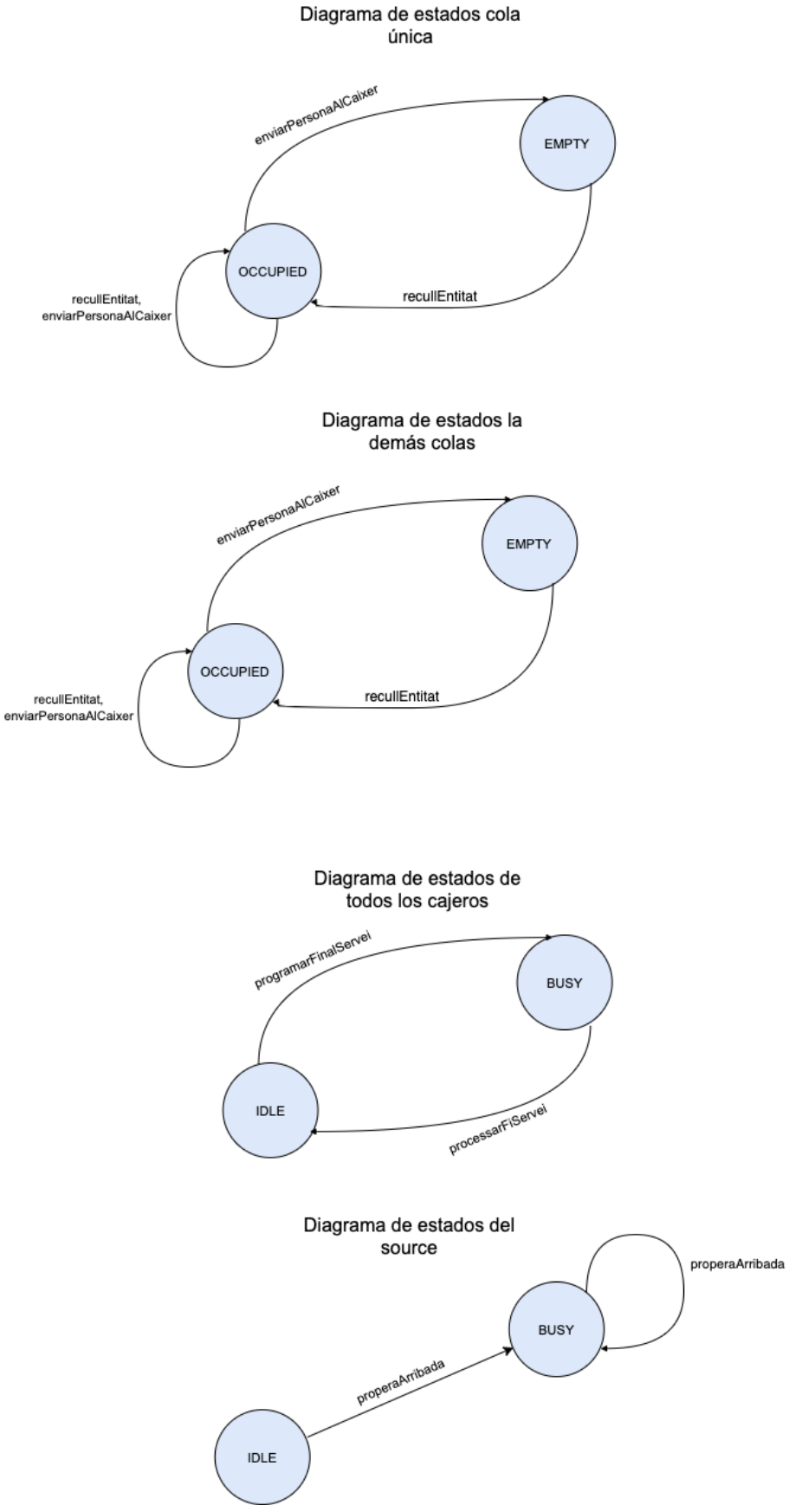

Figura 3: Diagrama d'estats de les entitats del nostre simulador.

Finalmente, en la **figura 3**, podemos observar el diagrama de estados de las diferentes entidades que conforman el sistema.

Primeramente, para las **colas**, encontramos que tienen dos estados: *"OCCUPIED"* y *"EMPTY"*. Inicialmente, estas se encontrarán en estado *"EMPTY"* hasta que llegue una persona (recullEntitat), donde pasarán a estado *"OCCUPIED"*. Una vez en estado *"OCCUPIED"* podrán:

- Seguir en el mismo estado si les llegan más personas (recullEntitat) o si van enviando personas al cajero pero hay más personas en ellas (enviarPersonaAlCaixer).
- Volver a estado *"EMPTY"* si envían una persona al cajero y solo tenían a esa persona en la cola.

Para los **cajeros** (server), también encontramos dos estados: *"BUSY"* y *"IDLE"*, correspondientes a sí están ejecutando un proceso de pago o no. Inicialmente, se encontrarán en estado *"IDLE"* y cambiarán a *"BUSY"* cuando empiece el proceso de pago (programarFinalServei). Una vez se encuentren *"BUSY"*, volverán a estado *"IDLE"* cuando acabe el proceso (processarFiServei). Entonces, los cajeros irán cambiando de estado cuando empiece o acabe un proceso de pago.

Finalmente, para el **source** también tenemos los estados *"BUSY"* y *"IDLE"*. En este caso, el source iniciará en estado *"IDLE"* y pasará a *"BUSY"* con la primera llegada de una persona (properaArribada). A partir de entonces, este se mantendrá en estado *"BUSY"* toda la ejecución e irá procesando las llegadas de personas al sistema.

4. Datos de entrada y distribuciones usadas

En cuanto a los datos de entrada del simulador, encontramos dos opciones:

1. **Opción por defecto:**
   Esta opción se corresponde a la que se ejecutará si decidimos dejar la configuración por defecto implementada en el simulador (más adelante explicaremos cómo ejecutar el programa). Los datos de entrada de esta opción son los siguientes:
   - *Cantidad de personas del sistema:* 25.
   - *Tiempo medio entre llegadas:* 2.
   - *Tiempo medio de proceso:* 20.
   - *Semilla para las distribuciones:* 0.
   - *Cantidad de experimentos:* 2.

2. **Opción con valores personalizados:**
   Esta opción se corresponde a la que se ejecutará si decidimos cambiar los valores por defecto del simulador (más adelante explicaremos cómo ejecutar el programa). En este caso, los mismos datos de entrada que para la opción por defecto (*cantidad de personas del sistema*, *tiempo medio entre llegadas, tiempo medio de proceso*, *semilla para las distribuciones, cantidad de experimentos* 2) vendrán dados por el usuario antes de empezar con la simulación.

Además, para ambas opciones, a través de la librería **numpy**, se han elegido dos distribuciones con tal de poder acercar el sistema a la realidad de la mejor forma posible. Las distribuciones escogidas han sido:

1. **Distribución exponencial para la llegada de personas a la zona de pago:**

```
def exponential_distribution(time):
    return np.random.exponential(scale=time)
```

Figura 4: Distribución usada para la llegada de personas.

Como se observa en la **figura 4**, la distribución exponencial usada se basa en la variable time, que corresponde al tiempo medio de llegadas que se haya asignado a la ejecución (ya sea el valor por defecto o se haya decidido cambiar).

2. **Distribución normal para los proceso de pago en los cajeros del supermercado:**

```
def normal(time_processing):
    return np.random.normal(time_processing,time_processing/1.5,1)[0]
```

Figura 5: Distribución usada para el proceso de pago.

Como se observa en la **figura 5**, y del mismo modo que para la anterior distribución, la distribución normal usada se basa en una variable del sistema, en este caso, al tiempo medio de proceso, el cual también se puede decidir si cambiar o no.

5. Funcionamiento

Para facilitar el uso del simulador, y que sean personalizables los experimentos, el programa nos pregunta si queremos modificar una serie de parámetros.
Los parámetros que se pueden modificar son los siguientes:

- **Cantidad de entidades:** Indica el número de personas que se van a utilizar para realizar los diferentes experimentos.

- **Tiempo entre llegadas:** Es el tiempo para la distribución entre llegadas, que está distribuido con una exponencial.

- **Tiempo de proceso:** Tiempo medio de proceso, distribuido con una normal.

- **Semilla:** Este parámetro es la semilla utilizada para las distribuciones y los experimentos.

- **Cantidad de experimentos:** Es la cantidad de experimentos que se van a ejecutar, simulando lo que sería el experimenter de flexsim.

También, para facilitar la ejecución y no tener que introducir los parámetros en cada ejecución, tenemos unos parámetros por defecto que nos ha facilitado hacer las pruebas y el desarrollo del simulador.

Los parámetros por defecto son estos:

```
### DEFAULT SETTINGS ####
ENTITIES_QUANTITY = 25
TIME_BETWEEN_ARRIVALS = 2
TIME_PROCESSING = 20
START_SEED = 0
QUANTITY_OF_EXPERIMENTS = 2
```

Figura 6: Parámetros por defecto.

Este fichero es el de settings.py, por lo tanto también es posible cambiar estos parámetros para realizar distintas pruebas sin tener que introducirlos por terminal.

6. Resultados obtenidos

A continuación, mostramos y explicamos un poco los resultados que hemos obtenido. En el siguiente apartado mostraremos las conclusiones que hemos sacado a partir de estas gráficas.

Primero de todo, hemos ejecutado los experimentos con la siguiente configuración.

- ***ENTITIES_QUANTITY = 200***
- ***TIME_BETWEEN_ARRIVALS = 3***
- ***TIME_PROCESSING = 25***
- ***START_SEED = 0***
- ***QUANTITY_OF_EXPERIMENTS = 10***

Las siguientes dos gráficas tienen el número de personas en cada una de las colas por cada experimento. En el eje de las X tenemos el tiempo transcurrido en el experimento, y en el eje de las Y se ve la cantidad de personas para ese tiempo concreto.

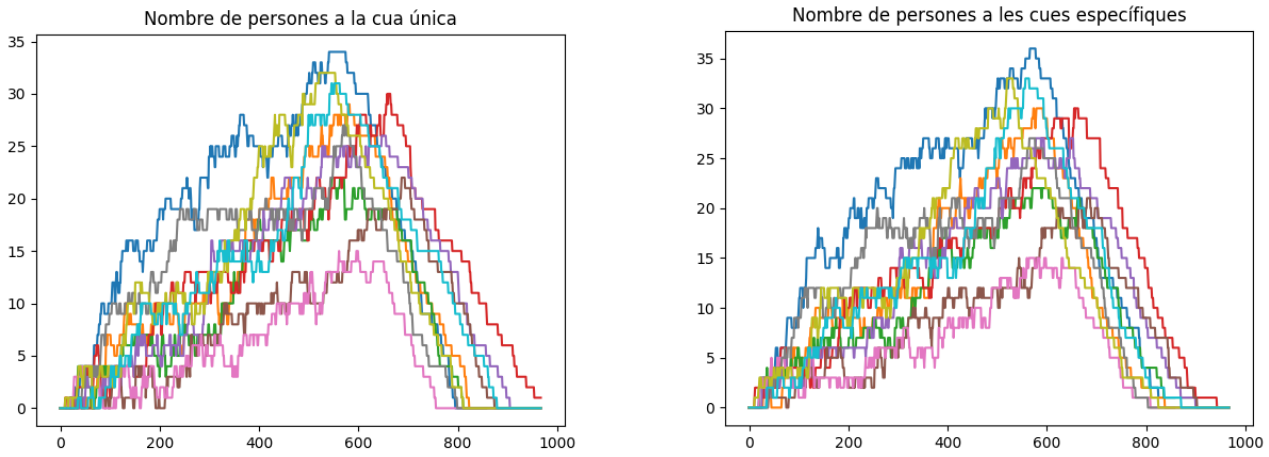

Figura 7: Gráficas correspondientes al número de personas en las colas.

Se puede observar en la siguiente gráfica que cuando hacemos la media de la cantidad de personas/tiempo de todos los experimentos para los dos tipos de cola, es muy similar.

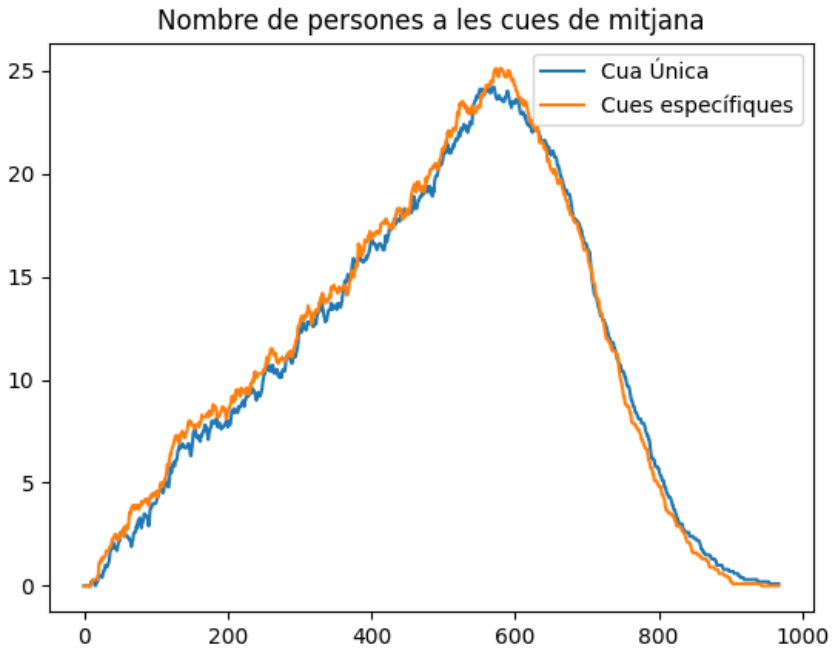

Figura 8: Gráfica correspondiente al número de personas en las colas de media.

Las siguiente dos gráficas muestran el histograma del tiempo para cola unificada, en una sobre la media de los tiempos de todos los experimentos, y en la otra desglosado por cada experimento.

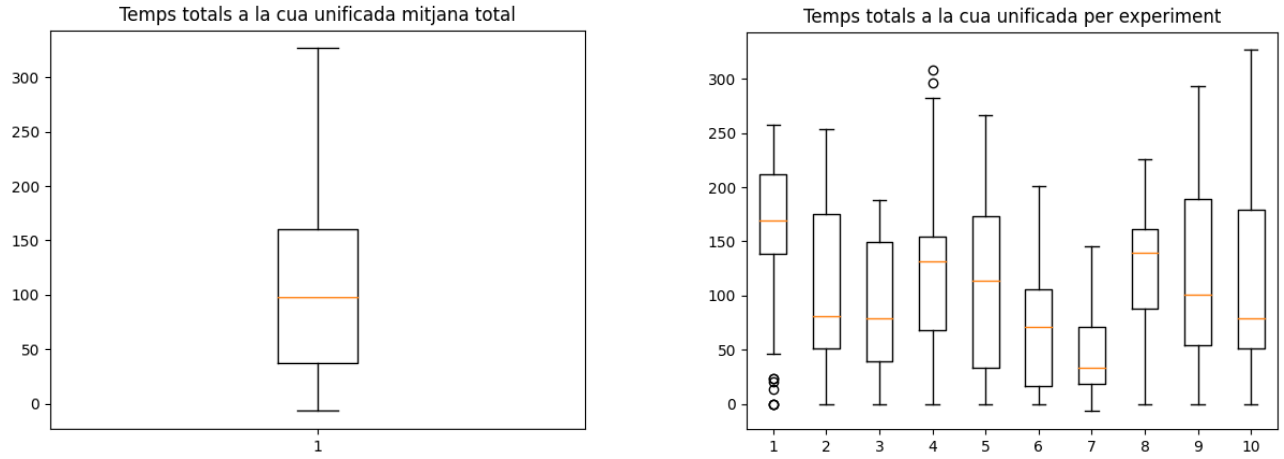

Figura 9: Gráficas correspondientes a los tiempos de los clientes en la cola unificada.

Las siguientes dos gráficas muestran el tiempo medio en las colas específicas, una de ellas es el tiempo medio de todos los experimentos, y en la otra gráfica se presenta el tiempo de espera en las colas específicas por cada experimento por separado.

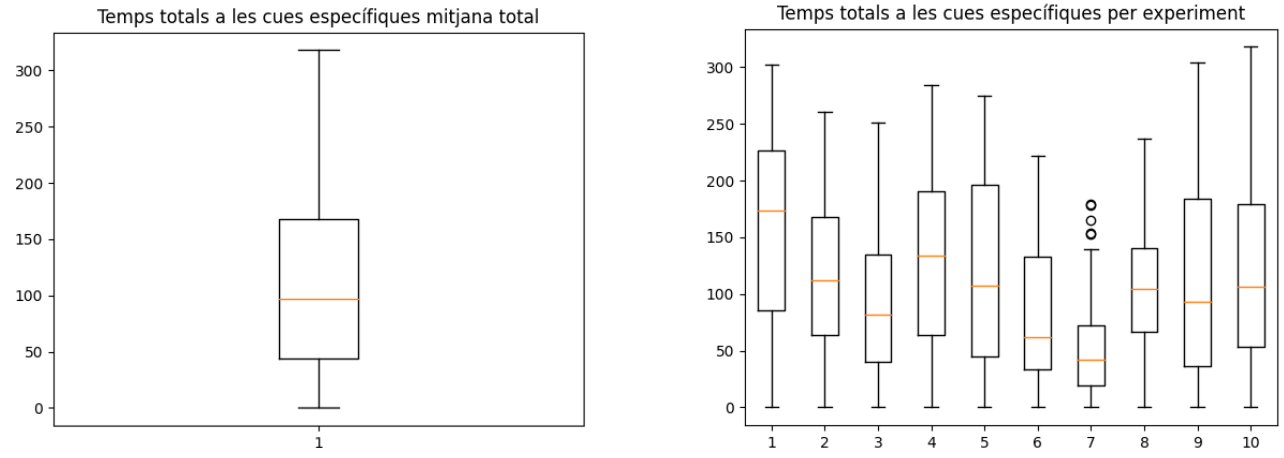

Figura 10: Gráficas correspondientes a los tiempos de los clientes en las colas específicas.

Estas dos gráficas muestran el histograma del tiempo en las colas, tanto en la unificada como en las específicas, esto nos permitirá estudiar si hay una diferencia significativa entre implementar la cola unificada, la cola específica, o dejar el mixto como tenemos ahora mismo en el sistema.

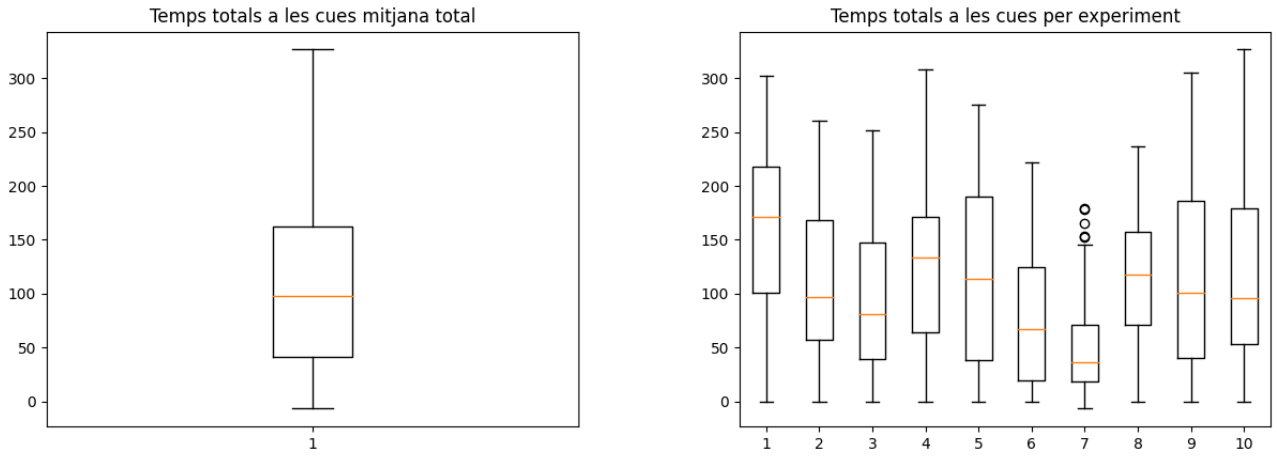

Figura 11: Gráficas correspondientes a los tiempos de los clientes en todas las colas.

En estas dos últimas gráficas mostramos el tiempo total por experimento y el tiempo total medio que se han pasado las personas en el supermercado entre el tiempo de espera en la cola y el tiempo de procesar el pago. Estas dos gráficas también nos sirven para comparar los dos tipos de cola y determinar que combinación es mejor.

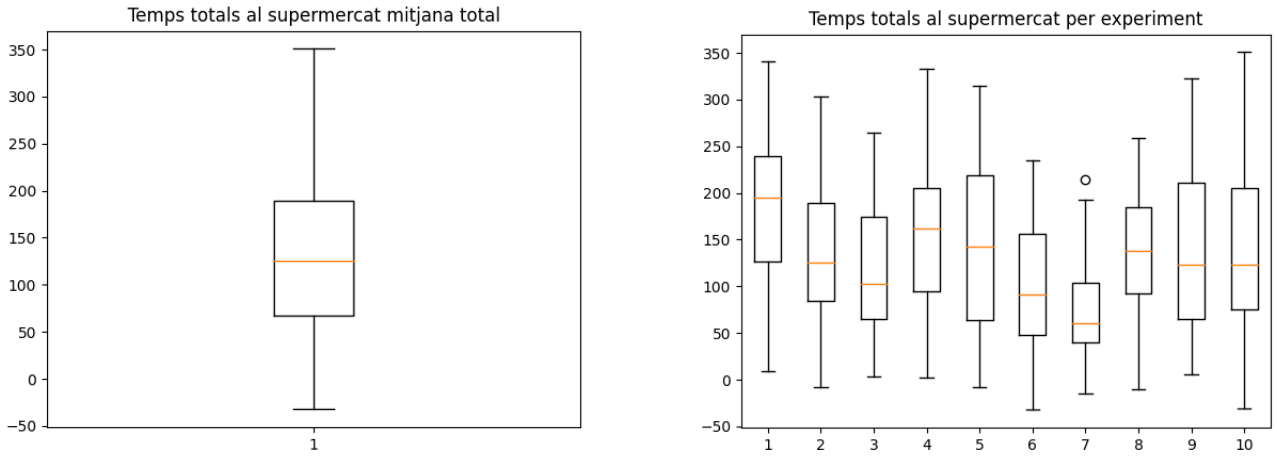

Figura 12: Gráficas correspondientes a los tiempos de los clientes en el supermercado.

7. Conclusiones

Después de observar los resultados obtenidos una vez finalizada la ejecución de nuestro sistema, podemos observar que no hay una diferencia significativa entre los tiempos para una cola unificada con 3 cajeros y una cola independiente por cajero.

Al iniciar el experimento, teníamos la idea de que la cola unificada sería mejor que las colas específicas, y que encontraríamos *outliers* para las colas que dependen de un único cajero, pero los experimentos nos demuestran lo contrario.

Por este motivo creemos que la decisión de poner de poner un tipo u otro de cola no debe estar supeditado al tiempo que los usuarios pasan en ellas en este supermercado, si no en otros factores como puede ser el espacio utilizado para las colas, distancia entre cajeros o su distribución en la sala.