75.59-ConcuBread

Nicolás Longo - Padrón N 98271

27 de mayo 2020

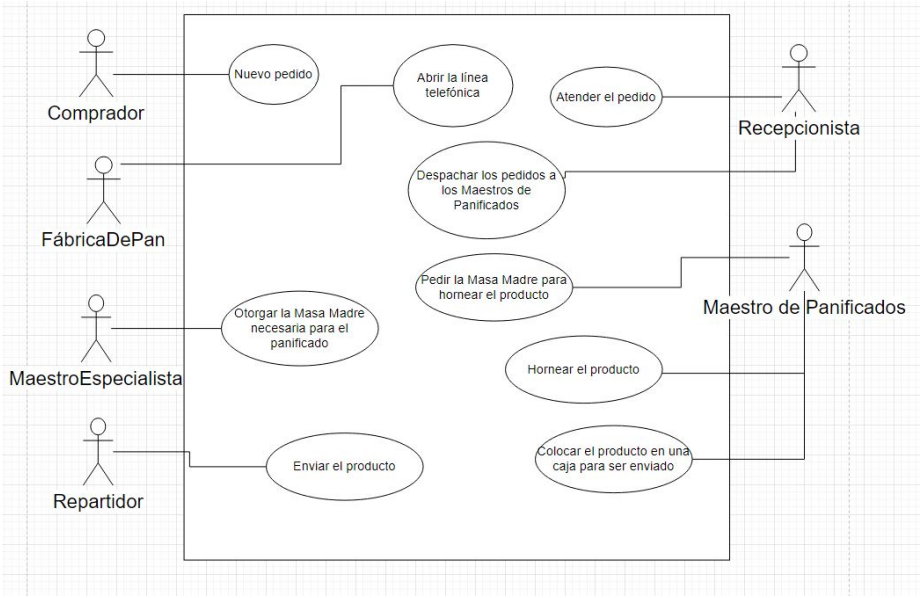
1 Introducción: análisis del problema

Debido a las circunstancias actuales del mundo, en un contexto de aislamiento social preventivo aumentaron tanto el consumo de pan como el formato de venta por delivery. Es por esto que se decidió abordar el problema de modelar el proceso de elaboración de pan haciendo uso de la

`Masa Madre`

Para el modelado, partimos de la idea base de separar la ejecución en procesos que pudieran ir realizando sus tareas de manera más o menos independiente, y luego pudieran, a través de los métodos de IPC (Inter Process Comunication), comunicarse y sincronizarse de la manera que necesiten.

A continuación se adjunta una especificación del caso de uso que describe todo el proceso modelado.



Pedir un producto panificado
1) Al abrir la Fábrica de Pan, se abren las líneas telefónicas para realizar pedidos. 2) Se toma el pedido y se despacha al Maestro del panificado correspondiente. 3) Se utiliza la Masa Madre, provista por el Maestro especialista en Masa Madre, para preparar el producto pedido. 4) Se hornea y una vez listo se pone en una caja. 5) El delivery se encarga de enviarlo.

2 Resolución

2.1 División en procesos

La primera y más natural división del proyecto es la que se puede hacer debido al modelo de negocio. El diseño es el de una fábrica de panificados, que incluye la elaboración del pan, la preparación de la masa madre (a partir de la cual se elabora el pan), la recepción de los pedidos y su posterior envío. Siguiendo ésta misma lógica, se propuso que cada una de estas entidades se ejecuten en un proceso propio. Así, la FabricaDePan se encarga de crear las entidades e inicializar la jornada de trabajo. Para este punto se crea un proceso para todas las entidades: Recepcionistas, MaestrosPanaderos, MaestrosPizzeros, Repartidores y se habilitan las líneas telefónicas desde donde se tomarán los pedidos. Recepcionistas, MaestrosPanaderos, MaestrosPizzeros y Repartidores se corren en un nuevo proceso mientras que el proceso principal se encarga de darle vida al MaestroEspecialista.

2.2 Esquema de comunicación

Para poder trabajar estos procesos deben comunicarse. En principio, para poder realizarse pedidos y avisos: que los recepcionistas puedan extender los pedidos que reciben telefónicamente a los Maestros correspondientes. Que estos puedan pedirle al MaestroEspecialista la Masa Madre necesaria para hornear el producto. Una vez terminado, necesitan enviarle el aviso al Repartidor para que lo envie. Supongamos una serie de situaciones que pueden resultar problemáticas:

- Como ésta es una Fábrica de Pan muy grande situada en el primer cordón del Conurbano, los pedidos son muy frecuentes. Para hacerle frente a la cantidad de pedidos y cuidar a nuestros clientes, tenemos una cantidad de Recepcionistas variada. Los días de semana, por momentos tenemos uno solo. Mientras que los fines de semana podemos llegar a tener varios. Qué pasaría si un Recepcionista atendiera el teléfono, y se dispusiera a tomar un pedido. Empieza a leerlo entonces y cuando está por la mitad es *preempted* por el procesador. Y en su lugar, un nuevo Recepcionista toma el control de la ejecución, atiende el mismo pedido y lo encuentra por la mitad. Ya no entenderá cuál es el pedido, y este no podrá ser reconstruido. Este problema de la concurrencia es la **Exclusión Mutua**. Tiene que ver

con las instrucciones de la Sección Crítica, y hay que ser especialmente cuidadoso para que estas **no se intercalen arbitrariamente**.

- Otro problema a tener en cuenta entonces, surge a partir de este concepto de Sección Crítica. Qué pasa si se da un escenario en el cual los procesos que quieren ingresar en la Sección Crítica no pueden, esperando a otro que no puede salir de ella, o bien porque ninguno pudo entrar aún. O qué pasaría si, por ejemplo, los Maestros de los panificados pidieran la Masa Madre antes de tener un pedido, y el Maestro de la Masa Madre aún no estuviera preparado para recibir pedidos, porque está esperando alguna otra señal. Este problema es conocido como ***deadlock***, y genera un estado de parálisis en los procesos del cual el programa no puede salir.

- Es importante también pensar cómo se le hará saber a todos nuestros procesos que la jornada laboral ha terminado. Para esto, podemos pedir que chequeen constantemente si ya es la hora de irse. Cuando sea la hora de irse, y todos los pedidos hayan sido despachados, entonces ésta variable debe modificarse, para que todos los procesos puedan saberlo la próxima vez que la chequeen. El problema sería si todos los procesos desean modificarla al mismo tiempo. Podría en este contexto suceder una ***race condition*** y el estado del programa pasaría a resultar indefinido.

- Finalmente, es necesario también aclarar que cuando un proceso quiere entrar a su Sección Crítica, debe poder hacerlo. Por esto, se decidió programar ConcuBread usando el lenguaje C++, en ambiente Unix (Linux). Así, la ausencia de ***starvation*** nos la garantizará el Sistema Operativo.

Para abordar la comunicación correctamente y evitar la aparición estos problemas se decidió:

- Los recepcionistas atenderán los pedidos desde un Pipe. Este Pipe implementará Locks para garantizar la lectura atómica.

- Los recepcionistas escribirán los pedidos a un Pipe correspondiente a cada tipo de panificado. Ésta escritura será de menos de $PIPE_BUFbytseparaasegurarsuatomicidad.LosMaest$

- El Maestro Especialista es uno solo. No hay problemas de comunicación en este caso. Escribirá en un Pipe de entregas de Masa Madre

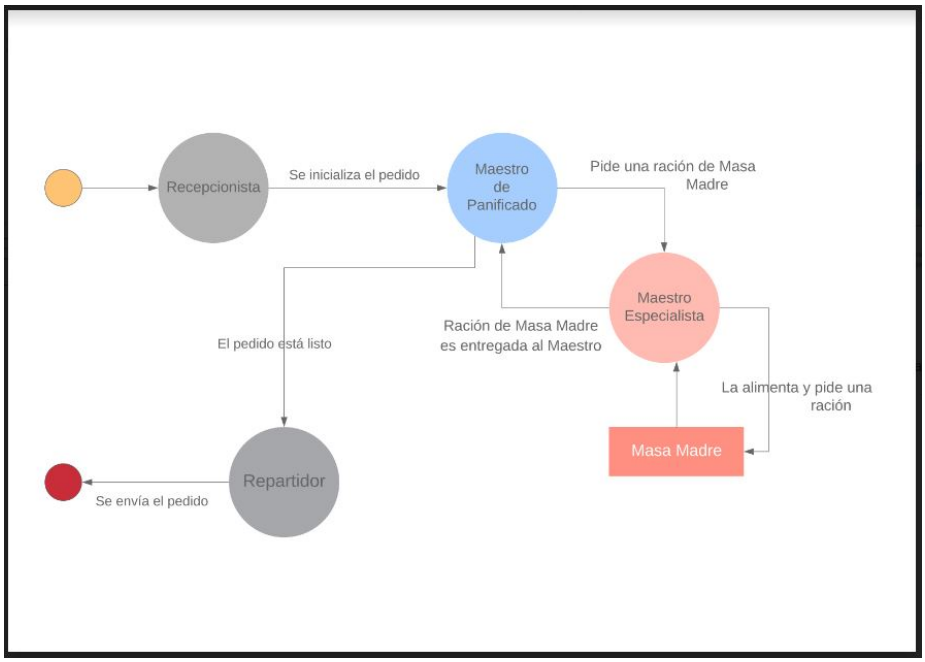
- Para recibir la Masa Madre, los Maestros Panificados leerán del Pipe utilizando locks. De esa manera nos aseguramos de que cada proceso tenga una ración de Masa Madre para cada pedido.

- Una vez que esté horneado, se envía el aviso de que el pedido ha finalizado a los Repartidores. Estos leen utilizando Locks y finalmente realizan el envío correspondiente.

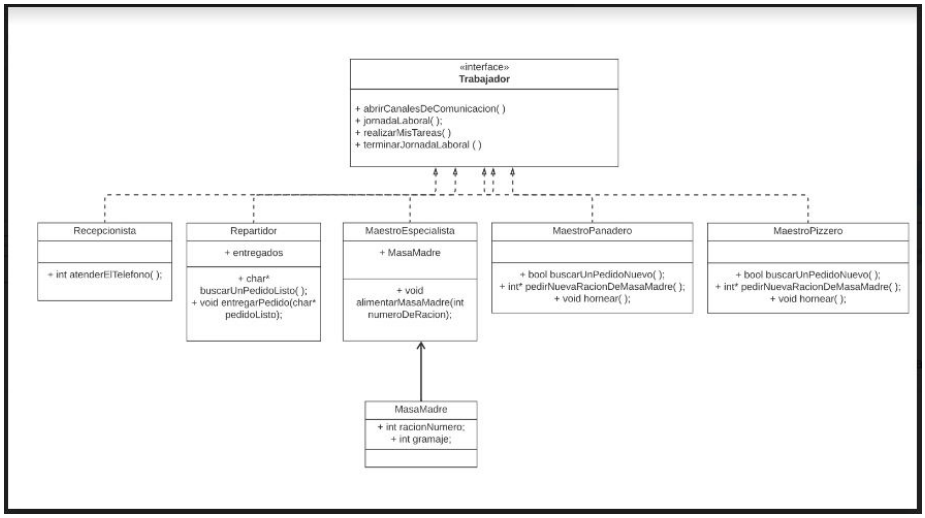
- Cuando todos los procesos hayan terminado, enviaremos una señal SIGINT para avisarle a los procesos que ya es hora de irse. Ellos liberarán los recursos pedidos y terminarán la ejecución del día.

3 Diagramas

3.1 Diagrama de procesos



3.2 Diagrama de clases



3.3 Diagrama de transición de estados