75.59 - Técnicas de Programación Concurrente

Primer Proyecto

Docente titular de cátedra:	Lic. Echeverria, Adriana Adela
Jefe de trabajos prácticos:	Ing. Garibaldi, Julia
Ayudante:	Ing. Garbarino, Jimena Daniela
Cuatrimestre:	1er cuatrimestre de 2015

Integrantes:

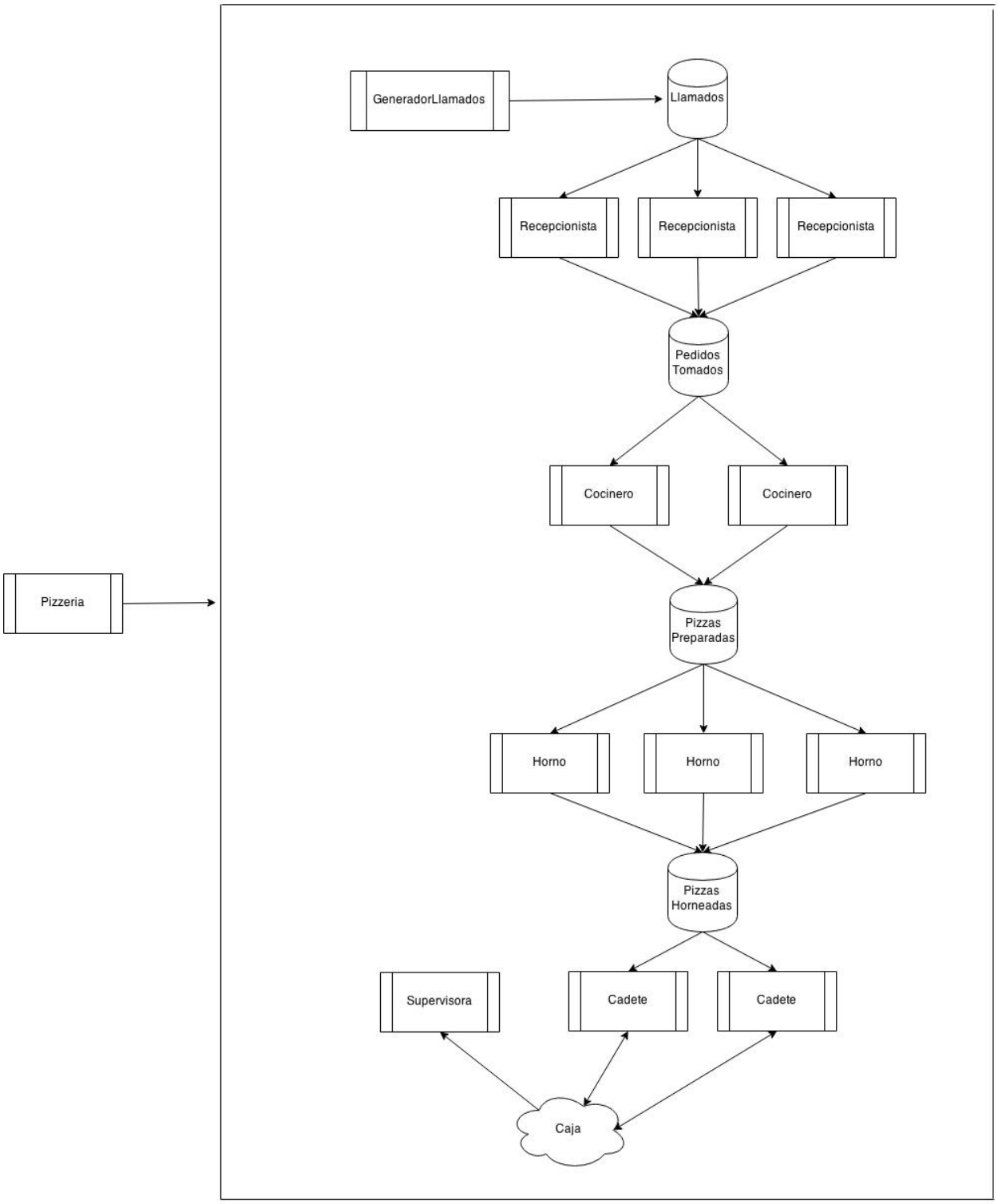
Mateo Bosco	93488	mateo.bosco@gmail.com
Juan Manuel Hidalgo	93383	juanma06@gmail.com

Introducción:

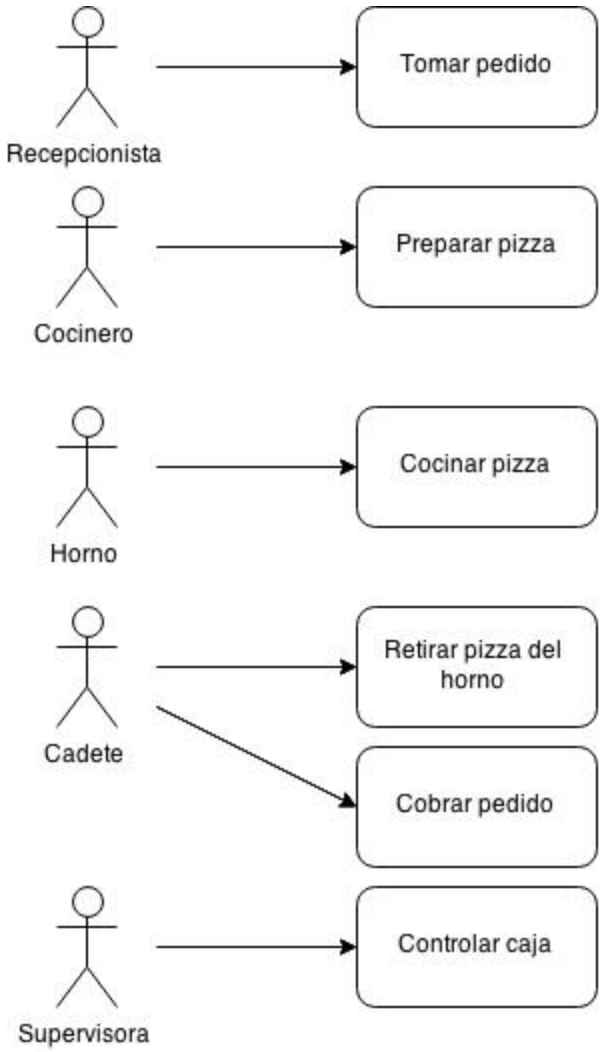
El siguiente informe corresponde a la documentación del análisis, especificación, diseño e implementación del primer proyecto en donde debíamos realizar una simulación de una pizzería concurrentemente.

Análisis del problema:

Al leer varias veces el enunciado llegamos a identificar varias entidades. Algunas otras las tuvimos que inventar para facilitar la simulación. Primero vimos que debía haber un generador de llamados que sea el que genere pedidos periódicamente para que luego las recepcionistas vayan atendiendo. Una vez que las recepcionistas toman el pedido, lo ponen en una cola para que lo tomen los cocineros. Luego ellos amasan y preparan la pizza que luego va a ir al horno. En el horno se cocina durante un tiempo hasta que un cadete la retira, la entrega, luego cobra y deposita esa plata en la caja. Además debía haber una supervisora que revise esa caja periódicamente. Las entidades que identificamos del enunciado fueron Recepcionista, Cocinero, Hornos, Cadete y Supervisora; y las que creamos nosotros fueron Zappi que representa una pizza, Pizzería que es el encargado del control de todas las demás entidades y GeneradorLlamados que va a simular los llamados entrantes a las Recepcionistas.
Este es el primer diagrama que hicimos:


GeneradorLlamados
Llamados
Recepcionista
Recepcionista
Recepcionista
Pedidos Tomados
Cocinero
Cocinero
Pizzeria
Pizzas Preparadas
Horno
Horno
Horno
Pizzas Horneadas
Supervisora
Cadete
Cadete
Caja

Casos de Uso

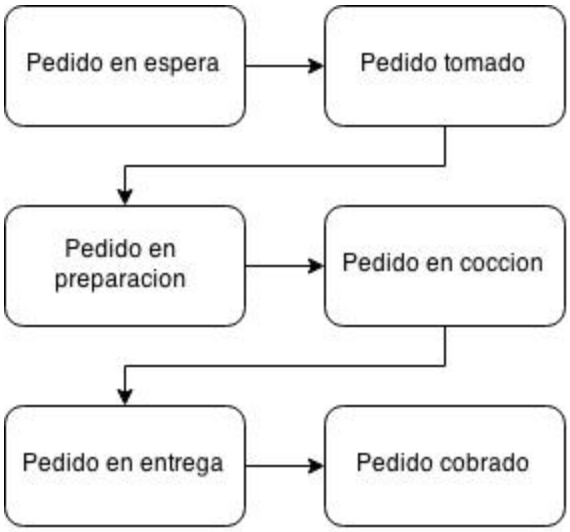


Descripción de los Casos de Uso:

- Tomar pedido: Una recepcionista atiende el telefono, toma el pedido que va a consistir de una pizza que va a tener un nombre, un precio y un tiempo de demora para su preparación.
- Preparar pizza: Una cocinera recibe un pedido creado por la recepcionista, amasa y prepara la pizza. Luego espera a que se libere un horno y la coloca.

- Cocinar pizza: Un horno recibe una pizza, la cocina y se apaga. Se mantiene ocupado hasta que un cadete la retira.
- Retirar pizza del horno: un cadete espera que se termine de cocinar una pizza en un horno y la retira para que este quede libre.
- Cobrar pedido: Un cadete entrega el pedido, lo cobra y tiene que depositar la plata en la caja, para eso debe esperar a que esta se libere si alguien la esta usando.
- Controlar caja: La supervisora revisa periódicamente la caja, si esta está ocupada debe esperar a que la liberen para poder consultar el saldo.

Diagrama de estados:



Diseño:

En este punto describiremos cómo fue la solución concurrente planteada detallando cómo fue la representación de las entidades, los estados y la comunicación entre procesos.
Nosotros diseñamos la pizzería como un conjunto de procesos. El primero de los procesos lo denominaremos Pizzeria, este es el encargado de crear los demás procesos, de iniciarlos conjuntamente y también de terminarlos. Los procesos a crear son:

- GeneradorLlamados: es el encargado de periódicamente generar pedidos para que atiendan las Recepcionistas

- Recepcionista: proceso encargado de atender los llamados generados y de crear las pedidos en la pizzería para que sean preparados por los Cocineros.

- Cocinero: es el proceso encargado de preparar las pizzas de los pedidos tomados por las recepcionistas y para luego meterlas en el un horno.

- Horno: proceso encargado de cocinar las pizzas y de entregarsela al cadete cuando este se libere que representa la acción de un cadete retirando la pizza del horno.

- Cadete: proceso encargado de retirar la pizza del horno, entregarla, cobrar, depositar el dinero en la caja.
- Supervisora: proceso encargado de controlar la caja.

Comunicación entre procesos:

GeneradorLlamados - Recepcionista: para la comunicación entre estos dos procesos utilizamos un FIFO de escritura en el GeneradorLlamados y de lectura en la recepcionista. Cuando una recepcionista no tenga pedidos para leer se va a bloquear hasta que se genere algún pedido y cuando el FIFO se llene el generador se va a bloquear. Como en el enunciado no hay ninguna restricción adicional para esta comunicación, con un FIFO es suficiente.

Recepcionista - Cocinero: entre estos dos procesos también se utiliza un FIFO, de escritura en el caso de Recepcionista y de lectura en el caso de Cocinero. Además utilizó un Semáforo, que creó e inicializo Pizzeria en un valor que es el doble de cocineros existentes. Cuando un cocinero termina un pedido, incrementa el semáforo y cuando una Recepcionista quiere entregar un pedido decrementa el semáforo, si este se encuentra en 0, la Recepcionista se bloqueará hasta que se termine de preparar algún pedido.

Cocinero - Horno: para que estos dos procesos se sincronicen, Pizzeria crea e inicializa un Semáforo en 0, cuando un Horno se crea o se libera se incrementa en uno y cuando un cocinero quiere poner una pizza en el horno lo decrementa. Si el semáforo esta en

0, este se bloqueará hasta que se libere un horno. Y para que se pasen el pedido, se utilizó un FIFO, de escritura en el caso de Cocinero y de lectura en el caso de Horno.

Horno - Cadete: este caso es similar al de Horno-Cocinero. Un FIFO de escritura y lectura para horno y cadete respectivamente y un Semáforo que horno decrementa y Cadete incrementa.

Cadete - Supervisora: estos dos procesos deben acceder al mismo recurso que es la caja. La caja está guardada en una memoria compartida pero como Cadete hace lectura y escritura, se utilizó un LockFile para que la lectura de Supervisora o la lectura de otro Cadete de la caja no sea de un valor que se va a pisar.

Pizzeria - GeneradorLlamados: Estos dos procesos se comunican para finalizar la simulación, para eso se utilizó un Semáforo inicializado en 0 que Pizzería va a intentar decrementar y mientras sea 0 se va a bloquear. Cuando GeneradorLlamados recibe la señal 2 (SIGINT) desde el exterior, por ejemplo una terminal, el handler de esta señal hará que se incremente el semáforo para que se desbloquee el finalizador de Pizzeria.

Pizzeria - Procesos: Pizzeria crea a todos los procesos (GeneradorLlamados, Recepcionista, Cocinero, Horno, Cadete, Supervisora) y guarda todos sus pids menos al de GeneradorLlamados. Cuando logra decrementar el semáforo va a mandar la señal 2 (SIGINT) a todos estos procesos que almaceno para que cada uno haga una salida elegante liberando todos los recursos.

Logger: El logger fue implementado como una clase estática que tiene un método que se llama log. Este método lo que hace es pedir un LockFile para el archivo log.txt, bloquearse si algún otro proceso esta escribiendo, luego logear y liberar el LockFile.

Diagrama de procesos:

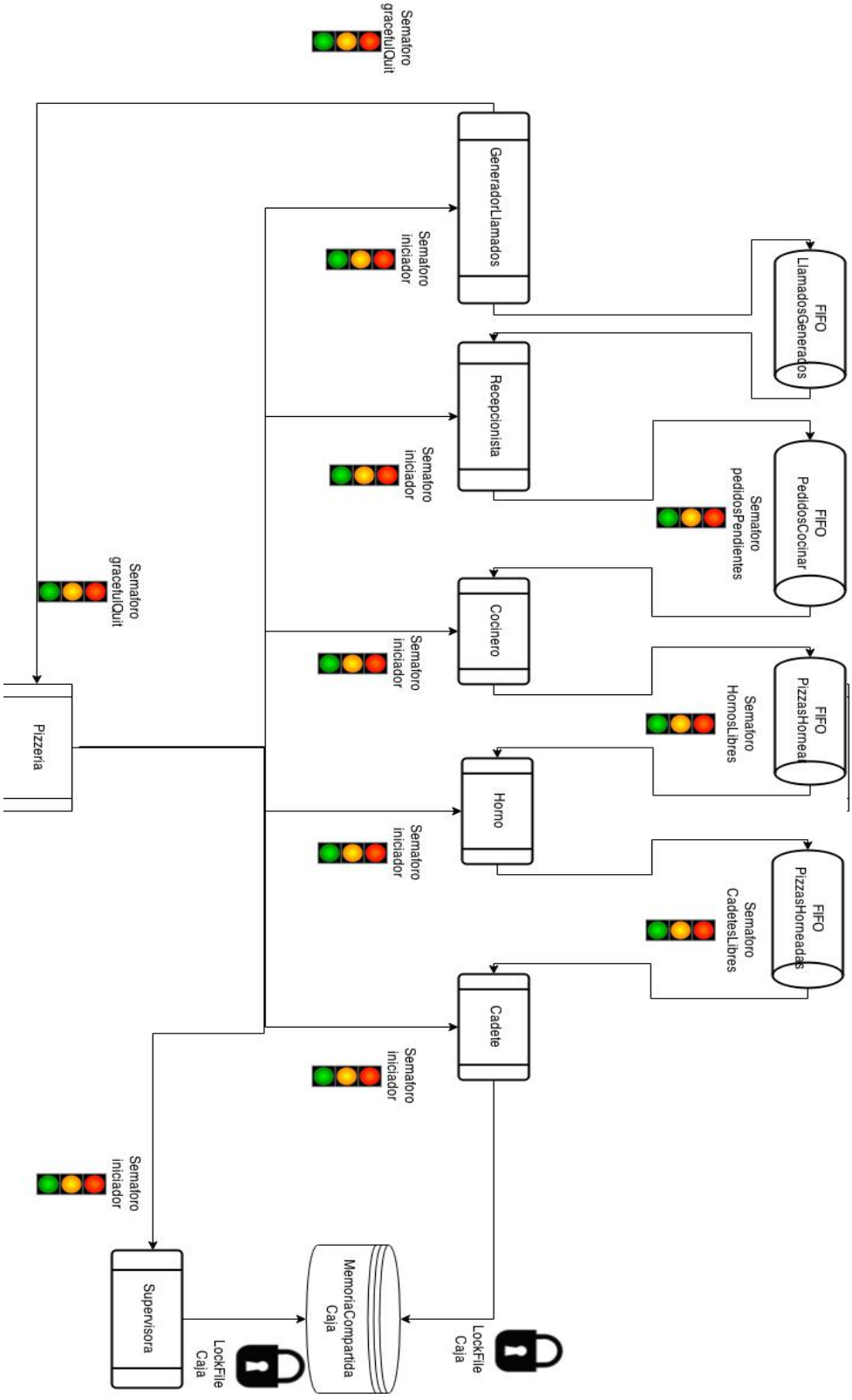

Semaforo gracefulQuit
GeneradorLlamados
Semaforo iniciador
FIFO LlamadosGenerados
Recepcionista
Semaforo iniciador
FIFO PedidosCocinar
Semaforo pedidosPendientes
Semaforo gracefulQuit
Cocinero
Semaforo iniciador
FIFO PizzasHorneadas
Semaforo HornosLibres
Pizzeria
Horno
Semaforo iniciador
FIFO PizzasHorneadas
Semaforo CadetesLibres
Cadete
Semaforo iniciador
Semaforo iniciador
Supervisora
MemoriaCompartida Caja
LockFile Caja
LockFile Caja

Diagrama de Clase: