TC2006 Lenguajes de Programación

Tarea 6: Programación Distribuida en Erlang

(* Equipos de 3 integrantes *)

Esta tarea les dará la oportunidad de practicar y familiarizarse con el lenguaje de programación Erlang. **NO** se aceptarán trabajos que no tengan la cantidad de integrantes especificada. Cualquier situación al respecto debe negociarse a la brevedad con el profesor.

Internamente, mediante comentarios de Erlang deberán incluir sus matrículas y nombres en el archivo, la descripción de las funciones de interfaz para los compradores y la tienda, los protocolos de creación y comunicación de los procesos, y las funciones auxiliares. Se **penalizará** fuertemente el NO documentar adecuadamente su código.

FUNCIONAMIENTO DEL SISTEMA

Deben programar un **sistema distribuido de compras** que involucre los siguientes 3 tipos de entidades:

- **Socios**: compradores que se suscribieron al pagar la membresía de la tienda.
- **Productos**: artículos, registrados en la tienda, que pueden ser comprados por los socios.
- **Tienda**: servidor que administra las suscripciones de socios y la venta de productos.

Los compradores se deben suscribir a la tienda como **socios** pagando una membresía. Para suscribirse deben utilizar el comando **suscribir_socio(Socio)**, donde Socio debe ser un átomo (nombre) que identifique de manera única al socio para que pueda comprar productos. Si ya existe un socio con ese nombre, deberá intentarlo de nuevo con otro nombre. Se puede eliminar una suscripción mediante el comando **elimina_socio(Socio)**, en cuyo caso se tienen que eliminar todos sus pedidos no entregados, ajustando adecuadamente toda la información relacionada.

Los socios pueden realizar pedidos de productos a la tienda mediante el comando **crea_pedido(Socio,ListaDeProductos)**, donde la lista de productos debe ser una lista de tuplas que identifiquen {Producto, CantidadPedida}. La tienda generará un número de pedido y responderá con la lista de pedido ajustada de acuerdo a las existencias de productos, para que el socio acepte o rechace el pedido con los comandos **acepta_pedido(Socio,Pedido)** o **rechaza_pedido(Socio,Pedido)**,

respectivamente, donde Pedido será un número asignado por la tienda al pedido del cliente. El cliente puede solicitar una lista de productos en existencia mediante el comando **lista_existencias()**.

Los **productos** en existencia se deben de registrar en la tienda mediante el comando **registra_producto(Producto,Cantidad)**, donde Producto es un átomo que identifica de manera mnemónica y única al producto, mientras que Cantidad debe ser un entero mayor o igual a 0. El registro creará un proceso que controlará la cantidad de producto en existencia. Se puede eliminar un producto, eliminando su proceso, con el comando **elimina_producto(Producto).** La cantidad de un producto se puede incrementar o reducir mediante los comandos **modifica_producto(Producto,Cantidad)**, donde el aumento o la reducción se determina por el signo de la cantidad. No se debe reducir si la cantidad excede las existencias. Todas estas tareas se realizan a través del proceso Tienda.

La **tienda** se deberá manejar como un proceso que concentre la información de socios, productos y pedidos. De los socios debe guardar una lista con sus claves y nombres. De los productos solo debe guardar la lista de las claves de los productos existentes (con proceso vivo). Y de los pedidos debe guardar dos listas: una para los pedidos atendidos y otra para los pedidos en proceso. Un pedido se encuentra en proceso mientras no haya sido aceptado por el socio. Cada vez que se acepte un pedido, este debe ser desplegado en el nodo de la tienda.

El proceso de la tienda se crea con el comando **abre_tienda()** y se elimina con el comando **cierra_tienda()**. El proceso de la tienda debe registrarse con el nombre **tienda** para que los socios y los que registran productos no requieran conocer su PID. Este proceso debe implementar los protocolos de comunicación con los socios para los pedidos y con los productos para su creación, modificación de existencias y terminación. Además, se debe de poder desplegar, de manera formateada, la información de la lista de socios con el comando **lista_socios()**, la lista de pedidos en proceso con el comando **pedidos_en_proceso()** y la lista de pedidos atendidos con el comando **pedidos_atendidos()**. Cada lista debe contener toda la información relevante. Un pedido en proceso es un pedido que todavía no ha aceptado o rechazado el socio. Los pedidos atendidos solo deben incluir la lista de los nombres de los socios que han aceptado pedidos y la cantidad de pedidos que han aceptado.

No está de más mencionar que si el proceso de la tienda termina, todos los procesos de los productos y cliente haciendo un pedido también lo deben hacer.

Como es un prototipo de sistema distribuido, deben incluir código de despliegue que indiquen clara y detalladamente lo que está pasando en cada nodo del sistema. Particularmente cuando se manden y reciban mensajes. Por ejemplo, cuando un comprador haga una solicitud de suscripción, este debe desplegar algo como “Comprador <fulanito> solicita suscripción”, antes de suspenderse para esperar la respuesta de la tienda. También, el proceso de la tienda al recibir la solicitud del comprador debe desplegar “Solicitud de suscripción de <fulanito>”, antes de atender dicha solicitud. Estos despliegues aparecerán en los distintos nodos donde se encuentren corriendo los procesos.

Finalmente, **DEBERÁN** incluir dentro del mismo programa, por medio de comentarios de Erlang, información que describa los pasos a realizar y la secuencia de comandos a ejecutar para probar todas las características que incluyeron en su sistema distribuido. Esta debe incluir todos los detalles para crear los nodos y procesos, así como utilizar todas sus funciones de interfaz, de modo que se ilustren todas las facilidades programadas. Se **penalizará fuertemente** el NO documentar el uso de su sistema.

¡¡ÉXITO!!