|  |  |
| --- | --- |
| **Instituto Tecnológico de Costa Rica**  **Ingeniería en Computación**  **Sede Regional San Carlos** | Primer Tarea Programada  Lenguajes de Programación |
| Prof. Oscar Víquez Acuña. | Sistema para compra de entradas a eventos |

**Descripción:**

La compra de entradas a eventos masivos en Costa Rica es un proceso que puede ser agobiante en el tanto la cantidad de solicitudes alcance picos altos en periodos relativamente cortos de tiempo, tanto debido a temas de conectividad y saturación, como a eficiencia en la respuesta de las soluciones hardware-software del sistema de venta de entradas.

Se pretende que el programador diseñe e implemente una aplicación que permita la búsqueda, selección y compra de entradas a eventos. El núcleo de la aplicación se basa en el “motor de búsqueda y compra de entradas” que es un módulo que como su nombre lo dice, debe buscar las mejores entradas disponibles según ciertos parámetros que son solicitados al usuario y la posterior reserva y compra o cancelación de estas. Para la implementación de este motor de búsqueda, se requiere una solución rápida y eficiente (desde el punto de vista de ejecución del sistema) por lo que la implementación de dicho módulo se realizará en el lenguaje Go, que permite diversas técnicas de programación imperativa eficiente, pero sobre todo porque genera un programa ejecutable en máquina real.

Dicho módulo se implementará en una aplicación en Go que permita el acceso a solicitudes en paralelo por parte de diversos clientes. Esta solución, para efectos de simplicidad y no propiamente porque sea la mejor solución posible, se hará utilizando sockets implementando una arquitectura simple de clientes y servidor a través del TCP-IP. En este esquema, la aplicación de búsqueda y eventual reserva o cancelación de boletos funcionará como un servidor que atenderá una serie de solicitudes de clientes que no serán otros que instancias simuladas de compras de clientes reales a partir de un sistema implementado en otro lenguaje de programación siguiendo prácticas y estrategias de programación funcional.

**El Servidor:**

De esta forma se quiere que se diseñe e implemente una solución de este módulo, que permita contemplar las siguientes características:

* Mapear en memoria las zonas del inmueble en donde se realizará el evento mediante el uso de categorías, zonas y por organización de sillas (número de fila y número de silla). Cada categoría podría tener varias zonas, Ej: Categoría Platea Este, zonas A,B,C, etc y cada zona, una serie finita de asientos organizados por filas, Ej: (4 asientos consecutivos) 🡪 Platea Este, Zona A, fila 5, asientos 6,7,8,9. Se debe entonces generar una estructura en memoria que permita mapear las posibilidades totales de asientos para los eventos y que será el espacio de búsquedas en donde se podrá buscar según el criterio de los clientes.
* Contemplar un estado para cada silla del inmueble bajo las características de libre, reservada, comprada. Para la búsqueda de opciones, se tomarán como candidatos solo aquellas que estén libres. Las reservadas se marcan así porque se está en espera de que el cliente decida aceptarlas (mediante un pago hipotético) o cancelarlas. Una vez aceptado lo anterior, el estado de la silla pasaría a comprada.
* Resolver solicitud de compra donde el cliente solicita boletos para una categoría específica y una cantidad de boletos. El algoritmo debe escogen la mejor opción posible de asignación de boletos tratando de cumplir con los siguientes criterios:
  + Mejor vista del evento (criterio que define cada quien)
  + Asientos juntos primero, siempre que sea posible y si no, lo más cerca posible según filas (que, por ejemplo, si no hay 6 boletos juntos en ningún lado, pero si 5 juntos, el sexto esté lo más cerca posible de los otros 5.

Contemplar que existen varias zonas para la misma categoría y por tanto los mismos boletos juntos que se encuentran en una zona, podrían estar más cerca en otra zona de la misma categoría.

* Que no se puedan asignar más de N cantidad de boletos definidos por el programador. Quiere decir que se limite la búsqueda de boletos a un máximo definido previamente y que pueda ser cambiado en el momento que se requiera. Si un cliente quiere comprar más del máximo, se rechaza la compra y se le indica el por qué.

Al estar frente a un sistema que potencialmente puede recibir muchas solicitudes de clientes al mismo tiempo, se debe trabajar mediante el uso de Go Routines que le permitan al servidor, recibir en paralelo una cantidad significativa de solicitudes de entradas de clientes. Para efectos de ilustrar este comportamiento, se estará adjuntando un ejemplo de creación de un servidor mediante sockets usando el protocolo TCP que atienda varios clientes en paralelo con varias peticiones

Se solicitará además que se implemente una simulación mediante timers aleatorios que hagan que la ejecución de cada solicitud de clientes considere un tiempo prudencial similar al tiempo que podría tardarse un usuario real en tomar la decisión de comprar o no.

La interacción entre clientes y servidor en términos generales es la siguiente:

1. Se realiza una conexión al servidor y el cliente de turno solicita las entradas que quiera en la zona que quiera o que le sugieran al servidor.
2. El servidor recibe la solicitud y hace la búsqueda de las mejores opciones según los criterios mencionados.
3. El servidor le comunica al cliente el resultado de la búsqueda. Las entradas quedan en estado reservado
4. El cliente decide si las quiere o no (se puede tardar un tiempo hipotético en este proceso) y le comunica al servidor su decisión
5. El servidor cambia el estado de las entradas a comprado, comunica al cliente la compra exitosa y cierra la conexión con el cliente

*Como el proyecto contempla el uso de lenguajes de programación que utilicen técnica de dos paradigmas y en el curso a la fecha solo hemos visto lo necesario para implementar para uno de ellos, se insta al programador(a) a que, para hacer sus pruebas, cree una aplicación cliente temporal en el mismo lenguaje Go o en otro lenguaje de su preferencia, para simular las interacciones que se harán entre el cliente y el servidor para la compra de entradas*

**El Cliente:**

El cliente será una aplicación distinta a la realizada en el lenguaje Go y deberá implementarse utilizando el lenguaje Haskell de manera que se pueda crear un programa ejecutable para máquina real.

Como en realidad en este sistema podría haber potencialmente N instancias de la aplicación cliente corriendo, lo que se quiere es que se implemente la funcionalidad de una instancia de cliente que pueda ejecutarse para solicitar compra de entradas 100% funcional y que haya además una opción que simule la compra en simultáneo de muchas entradas por diferentes clientes hipotéticos. Esta segunda opción, lo que hará es llamar a las funciones principales con parámetros ficticios y tiempos de espera hipotéticos entre las interacciones con el servidor.

La aplicación del cliente 100% funcional debe cumplir con los siguientes requerimientos:

* Se debe contar con una interfaz gráfica de usuario que mapee el escenario en el cual se quieren comprar entradas con las divisiones planteadas en el lado del servidor (Categorías, Zonas y asientos). La distribución debe ser amplia y queda a criterio del programador(a) la estructuración de esta. Se puede crear la interfaz en Haskell o se puede “interfazar” con otro lenguaje de programación. La decisión queda a criterio del programador(a)
* El usuario podrá escoger la Categoría que quiera y la cantidad de entradas que quiere y con esto se deberá hacer la consulta al servidor. Nótese que no se indica la zona, esto es porque se quiere que se hagan hasta 3 consultas al servidor para aquellas zonas donde haya menos personas. En este sentido, el servidor hará tres búsquedas y dará 3 respuestas
* Cuando el cliente reciba las respuestas que se le mostrará gráficamente en la interfaz del escenario, deberá decidir si compra o no y cual opción compra y se procederá a seguir el procedimiento descrito del lado del servidor para finiquitar dicha compra o cancelación. Si la compra fue exitosa, en la interfaz las sillas se mostrarán como compradas.

**Documentación:**

Para documentar el trabajo se requiere que se haga uso del formato establecido por la Carrera de Computación para este fin. Se solicita especial énfasis en la documentación de aspectos como:

* Algoritmos de resolución de solicitudes y búsqueda de boletos.
* Mapeo del escenario a nivel de memoria
* Hallazgos en torno a la implementación de sockets y procesos en paralelo. Cómo funciona el programa sin sincronización y cómo sería el escenario implementando cierta sincronización.
* Qué elementos consideran, deben tomarse en cuenta mejorar/cambiar, con miras en una implementación robusta de un sistema similar en producción a gran escala?

**Notas Finales para la Tarea:**

* Tarea grupal para máximo dos personas
* Fecha de entrega para las primeras dos tareas: Lunes 19 de setiembre de 2022 antes de las 10:00 pm.
* Se recomienda que se empiece a trabajar desde hoy.
* Cualquier tipo de fraude será severamente castigado.
* La entrega del trabajo se hará a través del Tec-Digital.

Ideas de croquis



