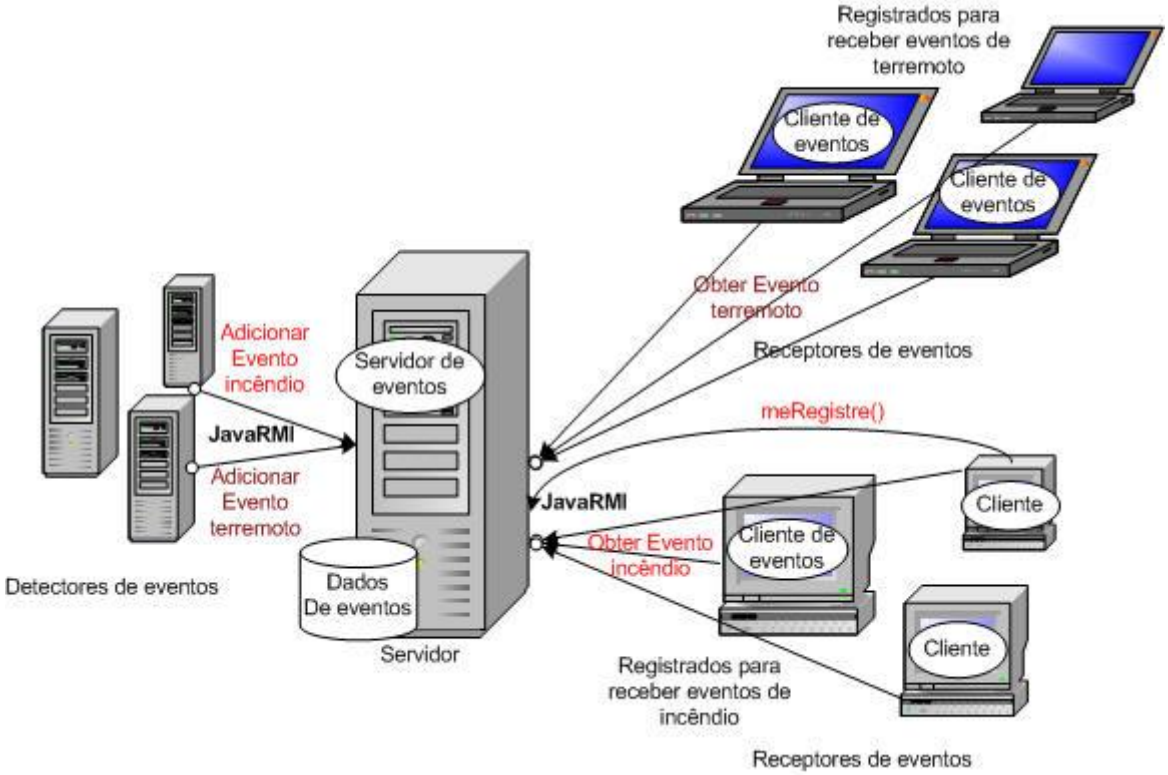
LABORATÓRIO 5 - Eventos e Notificações

Descrição | **Implementação** | **Apresentação** |

Descrição

Nesta atividade de laboratório a equipe deverá implementar um Serviço de Eventos que **oferece** mensagens de eventos para objetos clientes que se cadastraram nesse serviço para obter eventos específicos (ex. alerta de incêndio, assalto, enchente, terremoto, etc.). A aplicação de vocês deve ter a seguinte arquitetura:



O objeto Servidor de eventos deve receber mensagens de eventos dos detectores de eventos. Os clientes de eventos devem **buscar** no Servidor de eventos os eventos que lhes interessam. Ou seja, o sistema funciona similar ao **modelo produtor/consumidor**, o Detector de eventos é o produtor e os clientes são os consumidores de eventos. Por exemplo, a ocorrência de um evento de *incêndio* vai fazer com que um Detector de evento envie uma mensagem de alerta para o Servidor de eventos, o qual guardará esse evento para que todos os Clientes que tenham interesse nesse evento possam obtê-lo através de uma invocação de método do JavaRMI (ver figura acima).

Implementação

O cliente ao invocar o método `obterEvento()` envia como parâmetro uma string com o nome do **tipo de evento** que deseja obter. Se o Servidor de eventos tiver este tipo de evento

disponível, o retorna ao cliente. A string de retorno deve ter o seguinte formato: nome_do_tipo_de_evento: local (p.ex. "terremoto: Tóquio"). O cliente deve criar uma nova thread para cada evento de seu interesse. Esta thread irá invocar o método `obterEvento`() solicitando eventos do tipo especificado. Por exemplo, se um cliente deseja obter eventos de *incêndio*, *enchente* e *terremoto*, deve criar três *threads*, uma para cada um dos eventos. As threads criadas devem, então, invocar concorrentemente o método `obterEvento` solicitando o evento desejado. Caso o evento exista, a thread obtém imediatamente e, caso contrário, a thread entra em estado de espera (invocando `wait()`).

Um cliente não pode obter o mesmo evento mais de uma vez, portanto, o Servidor de eventos deve fazer esse controle registrando os eventos que já foram obtidos por cada cliente (cada cliente tem um ID único). Quando um cliente invoca o método `meRegistre()` envia o seu ID e o evento de seu interesse. O Servidor então irá cadastra este cliente para contabilizar os eventos em que já obteve.

O tamanho do Buffer que guarda eventos é 4. Desta forma, quando o detector de eventos tentar inserir um evento (método `inserirEvento`()) no buffer do Servidor de eventos e este estiver cheio, o detector entra em estado de espera. Quando um evento é lido duas vezes, por dois clientes diferentes, pode ser removido do buffer e o método `notifyAll()` deve ser invocado. Isto acordará a thread do Detector para que possa inserir novos eventos. Portanto, os métodos `inserirEvento()` e `obterEvento`() devem fazer uso de monitor do Java.

Os detectores de eventos podem ser implementados como um único objeto Detector em uma máquina remota. O objeto Detector deve ter uma interface (pode ser um janela do DOS) para que possamos digitar um evento (uma string) para ser enviado para o Serviço de Eventos.

A classe do Servidor de eventos deve ter os seguintes métodos.

```
public class ServidorEventosImpl extends UnicastRemoteObject
implements ServidorEventosInt {
  public boolean meRegistre(int clientID, String evento) throws
RemoteException {...}
  public synchronized boolean inserirEvento(String evento) throws
RemoteException {...}
  public synchronized String obterEvento(int clientID,
                                         String TipodeEvento) throws
RemoteException {...}
  public String[] obterListaEventos() throws RemoteException {...}
};

public interface ServidorEventosInt extends Remote {
  public boolean meRegistre(int clientID, String evento) throws
RemoteException;
  public boolean inserirEvento(String evento) throws RemoteException;
  public String obterEvento(int clientID, String TipodeEvento) throws
RemoteException;
  public String[] obterListaEventos() throws RemoteException;
```

Apresentação

A atividade deve ser desenvolvida **em duplas**. O programa deve ser apresentado ao professor no laboratório **até o dia 08/07 (por ser projeto final, não haverá prorrogação de prazo)**. Os dois componentes do grupo devem estar presentes. Será verificado o funcionamento do

programa e em seguida os alunos devem responder a questões sobre a forma como foram utilizados os mecanismos do JavaRMI e monitores no programa.

Podem ser atribuídas notas diferentes aos alunos de um grupo, dependendo das respostas às perguntas sobre o código do programa efetuadas pelo professor. Caso um dos alunos não esteja presente ou demonstre não conhecer o código do programa, será atribuída nota zero à atividade. Em caso de **cópia** do código de outro grupo, ambos terão nota igual a **zero**.

Dúvidas

Atendimento aos Alunos

- Horário: Segundas ou Terças das18:30 às 20:00.
- Local: Prédio do INE - Sala 515.