



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

Servidor de Comunicação Baseada em Eventos

José André Pereira da Silva Neto

Marlus Marcos Rodrigues Costa da Silva

Natal - RN

Maio de 2022

Sumário

1 Introdução	14
2 Arquitetura, Implementação e Execução	14
2.1 Arquitetura	14
2.2 Implementação	14
2.3 Execução	15
3 Conclusão	16
Referências	18

1 Introdução

O presente relatório tem como objetivo auxiliar no entendimento do desenvolvimento de um sistema de comunicação baseado em eventos. O sistema tem como objetivo possibilitar que um usuário gere um evento sobre um determinado tópico/assunto e publicá-lo no servidor de eventos, usuários do tipo cliente podem cadastrar no servidor de eventos o interesse em determinado(s) tópico(s), e toda vez que um evento for publicado no servidor de eventos, todos os clientes interessados irão receber uma notificação.

No desenvolvimento foi utilizado o conceito de programação distribuída para gerar uma comunicação em tempo real interligada por uma rede, foi usada a linguagem Java com RMI para fazer a construção do servidor e cliente, também foi feito o uso do padrão de projeto Observer para gerar as notificações de eventos que o usuário demonstrou interesse.

Mais abaixo será detalhado de forma mais específica os conceitos de arquitetura, implementação e execução do sistema.

2 Arquitetura, Implementação e Execução

Na figura 1, mostraremos a arquitetura do nosso projeto. Onde temos um servidor, que fornece três interfaces, IEvento, Iobservers e IServerEvents. Essas interfaces são responsáveis pelo Evento, que terá um nome de tópico e uma lista de interessados nele. Observers, que são os elementos interessados em um determinado tópico do evento. e Server Events, que é o

servidor de Eventos responsável por gerenciar os Eventos.

No cliente, temos as mesmas interfaces para usarmos e a implementação dos Observers (que são os interessados nos eventos).

2.1 Arquitetura

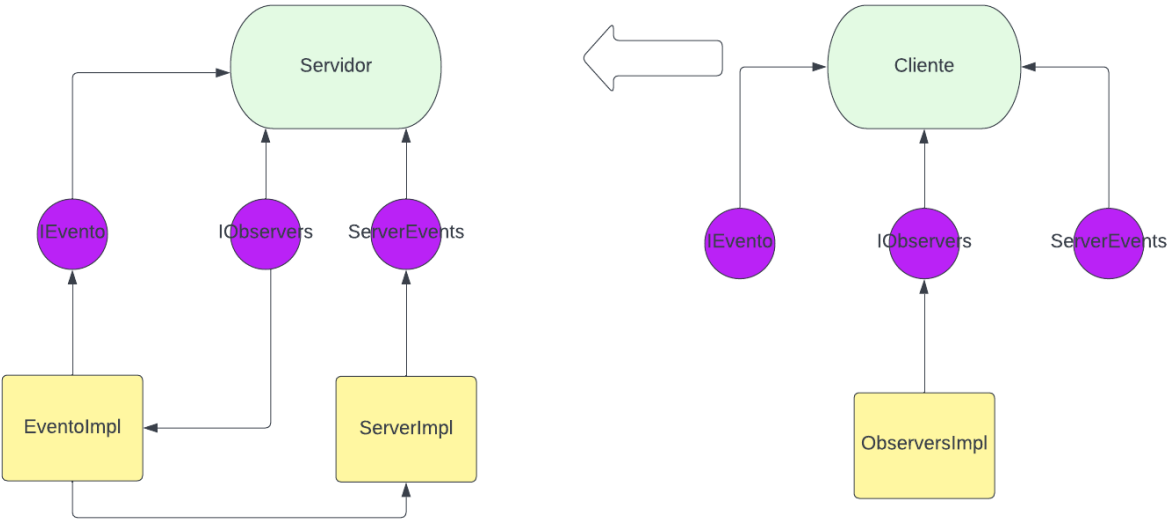


figura 1

2.2 Implementação

No servidor de Eventos temos a sua interface ServerEvents que é responsável por criar um evento de um tópico x e retorna um Evento.

```
public interface ServerEvents extends Remote{  
  
    public Evento criarTopico (String titulo)throws RemoteException;  
  
    public Evento getEvento (int i)throws RemoteException;  
}
```

No Observer, vamos ter a função de receber a notificação.

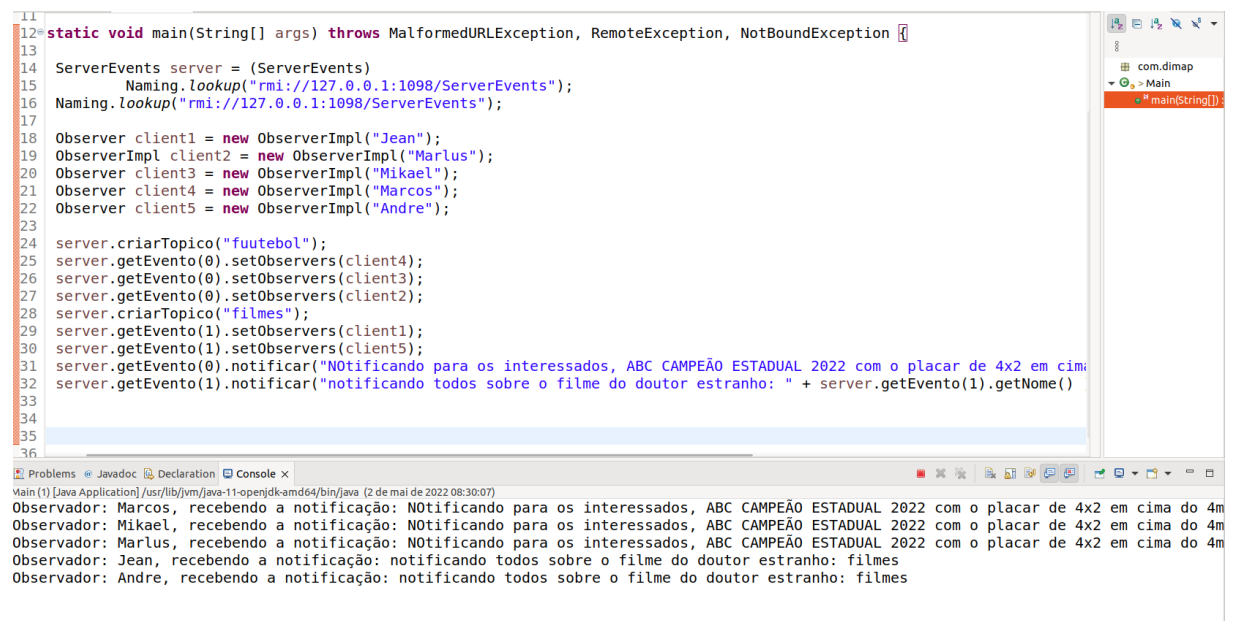
```
public interface Observer extends Remote {  
  
    public void updateEvents(String m) throws RemoteException;  
}
```

E no Evento, teremos as funções de criar um tópico, adicionar elementos interessados e notificar os elementos.

```
public interface Evento extends Remote {  
  
    public String getNome()throws RemoteException;  
  
    public void setNome(String nome)throws RemoteException;  
  
    public List<Observer> getObservers()throws RemoteException;  
  
    public void setObservers(Observer ob)throws RemoteException;  
  
    void notificar (String m) throws RemoteException;  
}
```

2.3 Execução

Na figura 2, mostraremos a Execução do nosso projeto. Onde temos a criação de de 5 clientes, (Observers interessados). Nisso, foi se criado também um Evento com o tópico futebol, e após isso foi adicionado 3 elementos interessados nesse tópico. Similar ao futebol, foi criado um tópico sobre filme e adicionado interessados nele. Nisso, foi se emitido uma notificação para interessados em ambos e recebidos por todos.



```
11
12 static void main(String[] args) throws MalformedURLException, RemoteException, NotBoundException {
13
14     ServerEvents server = (ServerEvents)
15         Naming.lookup("rmi://127.0.0.1:1098/ServerEvents");
16     Naming.lookup("rmi://127.0.0.1:1098/ServerEvents");
17
18     Observer client1 = new ObserverImpl("Jean");
19     ObserverImpl client2 = new ObserverImpl("Marlus");
20     Observer client3 = new ObserverImpl("Mikael");
21     Observer client4 = new ObserverImpl("Marcos");
22     Observer client5 = new ObserverImpl("Andre");
23
24     server.criarTopico("futebol");
25     server.getEvento(0).setObservers(client4);
26     server.getEvento(0).setObservers(client3);
27     server.getEvento(0).setObservers(client2);
28     server.criarTopico("filmes");
29     server.getEvento(1).setObservers(client1);
30     server.getEvento(1).setObservers(client5);
31     server.getEvento(0).notificar("Notificando para os interessados, ABC CAMPEÃO ESTADUAL 2022 com o placar de 4x2 em cima do 4m");
32     server.getEvento(1).notificar("notificando todos sobre o filme do doutor estranho: " + server.getEvento(1).getNome());
33
34
35
36
```

Problems Javadoc Declaration Console x

4main (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (2 de mai de 2022 08:30:07)

Observador: Marcos, recebendo a notificação: Notificando para os interessados, ABC CAMPEÃO ESTADUAL 2022 com o placar de 4x2 em cima do 4m

Observador: Mikael, recebendo a notificação: Notificando para os interessados, ABC CAMPEÃO ESTADUAL 2022 com o placar de 4x2 em cima do 4m

Observador: Marlus, recebendo a notificação: Notificando para os interessados, ABC CAMPEÃO ESTADUAL 2022 com o placar de 4x2 em cima do 4m

Observador: Jean, recebendo a notificação: notificando todos sobre o filme do doutor estranho: filmes

Observador: Andre, recebendo a notificação: notificando todos sobre o filme do doutor estranho: filmes

figura 2

3 Conclusão

Nesse trabalho foi possível implementar o servidor de eventos como foi proposto na atividade usando o JavaRMI, projeto bastante interessante para mostrar a possibilidade de uso de sistemas distribuídos.

Referências

<https://github.com/marlusmarcos/-distributed-programming>

