IFMG *CAMPUS* FORMIGA

CURSO: **CIÊNCIAS DA COMPUTAÇÃO**
DISCIPLINA: **SISTEMAS DISTRIBUÍDOS I**
PROF.º EVERTHON VALADÃO everthon.valadao@ifmg.edu.br

TRABALHO EM GRUPO VALOR: 20 PONTOS

Deverá ser projetado um **sistema distribuído** (SD) que proveja um *Serviço de e-Marketplace*. A infraestrutura do SD deverá ser provida pelo middleware JGroups, com a comunicação entre os componentes — membros do(s) *cluster*(s) — realizada através de conectores JChannel, do componente MessageDispatcher (*multicast*, *anycast*, *unicast*), demais *building blocks* e protocolos que se fizerem necessários.

REQUISITOS DO TRABALHO:

Leia atentamente as instruções a seguir:

- Trabalho prático em grupo, a ser realizado em **trios** (três alunos).
  - Exceções devem ser previamente comunicadas e aprovadas pelo professor.
- Para a implementação do trabalho, deve-se utilizar o **middleware** JGroups;
  - Sugere-se a utilização da IDE Netbeans, no sistema operacional Ubuntu (GNU/Linux).
- Deverá ser submetido no portal ***meuIFMG*** (https://meu.ifmg.edu.br) um arquivo compactado (.zip) contendo:
  - a implementação do trabalho: **códigos-fonte** + programa compilado (.java e .class);
  - o arquivo **XML** contendo a configuração **personalizada** dos protocolos do JGroups a serem utilizados pelo programa;
  - um breve **relatório**:
    - apresentando a arquitetura do sistema, explicando e justificando as principais decisões de projeto do grupo, apontando os pontos fortes e fracos da solução desenvolvida;
    - justificando a escolha de cada protocolo do JGroups utilizado e fornecendo uma análise de desempenho da pilha de protocolos configurada (latência, Bytes/seg, mensagens/seg, etc);
  - um arquivo **executável** (ex.: shellscript.sh ou programa.jar) para facilitar a implantação e execução do trabalho.
  - um arquivo texto **README.txt** contendo informações de (i) como compilar e de (ii) como executar (via terminal CLI) o programa no sistema operacional Linux Ubuntu;

DATAS DAS ENTREGAS:

Etapa 1: funcionalidades **básicas**
até **30/05** via portal *meuIFMG* (https://meu.ifmg.edu.br)
Deverá ser agendada a apresentação parcial do trabalho ao professor.

Etapa 2: funcionalidades básicas e **intermediárias**
até **25/06** via portal *meuIFMG* (https://meu.ifmg.edu.br)
Deverá ser agendada a apresentação do trabalho ao professor.

REQUISITOS DA APLICAÇÃO:

O serviço de e-marketplace deverá permitir o **anúncio** de produtos (de vendedores) para serem comercializados (aos consumidores) em uma grande loja virtual. Como exemplo, inspire-se em plataformas como MercadoLivre, eBay e Amazon. Mas ATENÇÃO: o foco do trabalho não é criar uma super-hiper loja virtual mas sim criar uma loja virtual básica que funcione de maneira distribuída. Portanto, a interface com o usuário poderá ser bem básica, CLI ou GUI como preferirem. O professor irá avaliar quais técnicas, algoritmos distribuídos e protocolos de middleware foram utilizados no trabalho.

Ao se cadastrar, cada novo **consumidor** iniciará com um valor padrão de **créditos** na loja virtual (ex.: $1000). Ao se cadastrar, um **vendedor** iniciará com zero créditos na loja virtual. Logo após, um vendedor poderá cadastrar no sistema quantos anúncios de produtos deseje. Um determinado **produto** pode ter sido anunciado por diferentes vendedores, que fornecerão uma determinada quantidade de itens daquele mesmo produto. Assim, cada produto anunciado terá um **estoque de itens** finito, de acordo com a quantidade agregada de itens daquele produto anunciada pelos vendedores. ATENÇÃO: **um item** (unidade) de determinado produto pode ser **vendido** para apenas **um consumidor** (e fornecido por um determinado vendedor)**.** Ou seja, não venda o mesmo item duas vezes!

Um consumidor deverá ser capaz de **pesquisar** pelos produtos anunciados, ordenando os resultados (ex.: menor preço, anúncio mais recente, dentre outras possibilidades que julgue necessárias). O consumidor deverá ser capaz de efetuar a **compra** do produto desejado, de maneira que os créditos serão **transferidos** do cliente para o vendedor que fornecerá o item do produto. Um consumidor deverá ser capaz de listar o **histórico das compras** efetuadas e consultar seu saldo de créditos. De maneira equivalente, um vendedor deverá ser capaz de listar o **histórico das vendas** realizadas, bem como consultar seu saldo de créditos delas proveniente.

Como mecanismo de auditoria financeira do sistema distribuído, deverá ser possível calcular o **montante de créditos** em circulação no sistema de e-marketplace, somando-se os saldos de cada consumidor e os saldos de cada vendedor em um dado momento. Naturalmente, o montante de créditos do sistema deverá ser previsível, ou seja, o total de dinheiro virtual no sistema será proporcional à quantidade de usuários cadastrados no sistema (montante = quantidade usuários * crédito inicial de cada usuário). ATENÇÃO: dinheiro virtual não deverá ser perdido nem criado em uma transação de compra-venda. Igualmente, um item não deverá sair do estoque sem que haja o registro da sua respectiva transação financeira.

Por fim, em cada **produto** anunciado os clientes poderão enviar perguntas aos vendedores que anunciaram itens daquele produto e receber as respostas deles, bem como visualizar as **perguntas/ respostas** de outros consumidores interessados no mesmo produto.

REQUISITOS DO SISTEMA DISTRIBUÍDO:

O sistema deverá ter **distribuição** vertical (sugestão: divisão em camadas MVC) e horizontal (replicação de componentes em cada camada), de maneira que deverá ser **tolerante a falhas**, permanecendo operacional mesmo que alguns de seus componentes (membros do *cluster* daquela camada) sofram falha por colapso.

Quando um novo componente for adicionado ao sistema (ingresso de novo membro no *cluster*) ele deverá receber a **transferência do estado** atualizado do sistema. O estado do sistema consiste em: produtos anunciados e seu estoque atual, usuários cadastrados e seu saldo atual, registro das transações de compra-venda realizadas (comprador, vendedor, produto, quantidade, preço), registro de perguntas&respostas para determinado produto e demais informações que se fizerem necessárias conforme a função daquele componente no sistema.

O sistema deverá **armazenar** periodicamente seu estado de maneira **persistente** (sugestão: serialização de objetos para arquivo binário em disco). Quando todos os processos forem terminados, ao reiniciar o sistema deverá ser **carregado** para a memória principal o último estado (o mais atual possível) armazenado na memória persistente.

O Sistema Distribuído (SD) deverá possuir as seguintes características do **middleware** (JGroups):

- Serviço de composição do cluster, com rápida descoberta de novos membros e detecção de falhas nos membros de um cluster JGroups;
- Transmissão de mensagens entre os membros do cluster JGroups, através de multicast confiável e com ordenação das mensagens.
- Podem ser utilizadas flags de mensagem (sincronização), opções de requisição (votação), bloqueios / travas ou contadores atômicos, de acordo com o serviço necessário;
- OBS.: a pilha de protocolos XML projetada deverá apresentar um bom desempenho, devendo-se portanto ler a documentação do *middleware* JGroups para conhecer melhor os protocolos por ele providos e para configurar os parâmetros do protocolo de acordo com as necessidades da aplicação. Evite incluir protocolos desnecessários ao funcionamento do serviço.

ETAPA 1) o SD deverá prover as seguintes **funcionalidades BÁSICAS**:

- Definição de um identificador único por usuário do sistema, de maneira que um mesmo usuário seja corretamente identificado ao sair e retornar;
- Cadastro de novo produto a ser anunciado, sem duplicidade, em acordo dos membros (todos ou maioria) daquele *cluster*. Ou seja, caso determinado produto já exista, deverá ser incrementada a quantidade de seu estoque global e a qual vendedor aqueles novos itens pertencem;
- Envio da ordem de compra de um consumidor somente para os vendedores que tenham anunciado aquele produto, de maneira que o sistema deverá determinar qual vendedor com estoque daquele produto irá efetivar a venda;
- Criação de nova "sala de perguntas&respostas" sobre determinado produto, identificada pelo produto anunciado e coordenada por algum vendedor, denominado "moderador". Inclusão de nova pergunta ou resposta de usuário para determinado produto, realizada por intermédio do moderador da sala. As perguntas e respostas de determinado produto devem ser visíveis para todos os demais usuários do sistema. No caso de produtos sem itens em estoque, a sala de perguntas&respostas deverá ser trancada, até que novos itens sejam adicionados ao estoque.

ETAPA 2) o SD deverá prover as seguintes **funcionalidades INTERMEDIÁRIAS**:

- Armazenamento persistente do estado do sistema (cadastros, transações, interações);
- No reingresso de um membro, deverá ser obtido o estado do sistema, ou seja, atualizações que ele possa não ter recebido enquanto estava desconectado;
- O sistema deverá utilizar mais de um *cluster* (JChannel), de acordo com a divisão de camadas adotada. Dica: pode-se utilizar (sub)canais de comunicação mais leves e baratos (ForkChannel), para implementar algumas funcionalidades e/ou camadas;
- O sistema deverá ser capaz de continuar funcionando caso haja particionamento na rede, devendo consolidar os estados das partições quando se reunirem;
- O sistema deverá prover mecanismos de segurança, como criptografia das mensagens trocadas, autenticação dos usuários, autenticidade das transações.

EXTRA) OPCIONALMENTE, o SD poderá prover funcionalidades AVANÇADAS:

- O sistema pode ser capaz de permitir que usuários que estejam em LANs diferentes possam se conectar ao mesmo *cluster*, lidando com peculiaridades da infraestrutura de rede;
- O sistema pode implementar um limitador de taxa de dados (quantidade de dados enviada por unidade de tempo), evitando o congestionamento do serviço ou ataques de negação de serviço (DoS);
- O sistema pode ser capaz de comprimir mensagens grandes para agilizar a sua transferência;
- O sistema pode ser capaz de realizar controle de fluxo, ajustando a taxa dos transmissores à taxa do receptor mais lento;
- O sistema poder ser capaz de fragmentar mensagens de aplicação muito grandes em mensagens menores, remontando-as no receptor, agilizando assim sua transferência e evitando fragmentação a nível da camada de rede.