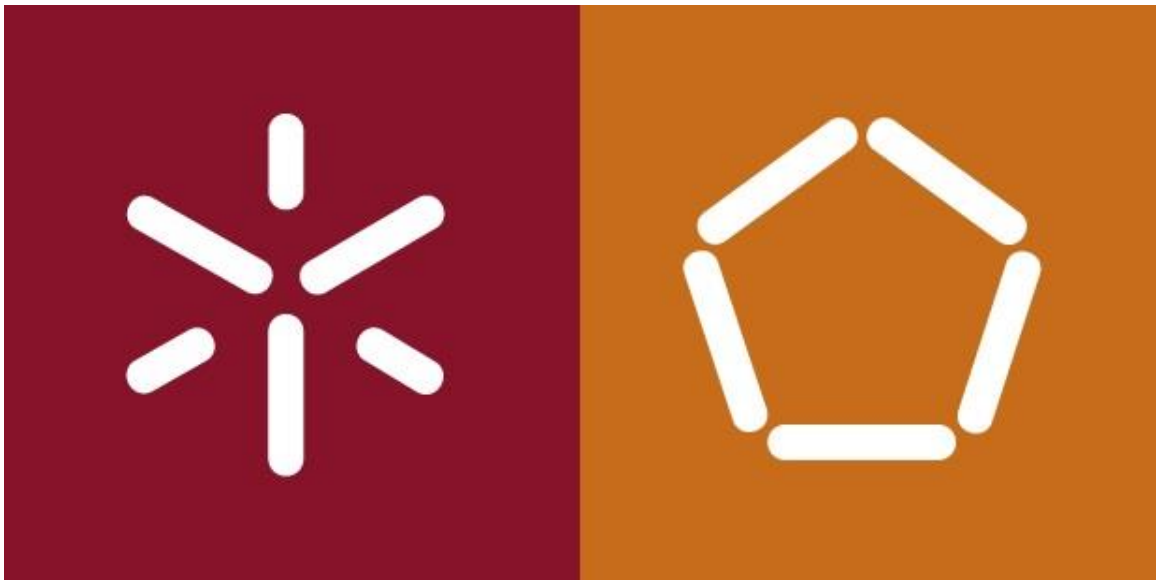
UGOTSERVED

SISTEMAS DISTRIBUÍDOS



Alunos:
Duarte Freitas: A63129
Ana Guimarães: A79987
Hugo Moreira : A43148
2018-2019

INTRODUÇÃO

O projecto a desenvolver consiste numa empresa, que através de uma plataforma fornece o seu serviço aos vários utilizadores.

O serviço que esta propociona ao seus utilizadores é a reserva e utilização dos seus servidores virtuais, quer seja para processamento ou armazenamento. Estes servidores são de diferentes tipos e cada um tem diferentes caracteristicas de forma a satisfazer todas as necessidades dos seu utilizadores. Cada tipo de servidore é indentificado por um nome, este irá corresponder às suas características de configuração, de hardware e a um preço que será fixo.

Existem duas possibilidades dos utilizadores adquirirem estes servidores. Por reserva em que caso o servidor desse tipo esteja disponível, este é lhe atríbuido automáticamente. Ou por leilão em que a plataforma indica o preço mínimo desejado pelo aluguer do servidor e os seus utilizadores vão licitando até que o leilão termine. Este é alugado ao utilizador com a licitação mais alta.

Caso um utilizador esteja disposto a pagar o preço da reserva de um servidor que se encontre em leilão, esse leilão é automáticamente terminado, o servidor é atribuído ao utilizador que fez essa mesma reserva. O utilizador com a licitação mais alta é informado de que este leilão foi terminado.

Irá também proporcionar um outra opção aos seus utilizadores, caso este pretendão reservar um servidor. Esta opção consiste na possibilidade de o utilizador, numa situação em que o tipo de servidor desejado não se encontre disponível, ele poderá ficar a aguardar numa lista de espera para a reserva do mesmo. Ela será satisfeita com prioridade de ordem de chegada.

Assim que não pretenda mais usufruir de um dos servidores que tem alugado, o utilizador poder proceder ao cancelamento da reserva. Após este cancelamento é verificado se existe algum utilizador em fila de espera, para que o seu pedido possa ser staisfeito.

O utilizador pode visualizar os seus servidores assim como o custo incorrido pela utilização dos seus serviços.

IMPLEMENTAÇÃO

De forma a satisfazer os serviços que a plataforma deve fornecer aos seus utilizadores, foram considerados os seguintes aspectos no desenvolvimento do projecto.
Uma vez que vários utilizadores concorrentemente podem fazer a utilização de todos as funcionalidades que a plataforma proporciona, foi considerado para implementação a utilização de ReentrantLock para a utilização de bloqueios explícitos sobre as seguintes "Entidades/Objectos": Servidor,Cliente,UserQueue,Server,Client entre outras mas consideramos estas as mais importantes.
O Server garante toda a gestão sobre a plataforma e comunicação com os seus clientes. Sendo que ao ser efectuada o bloqueio deste quando está a ser efectuada alguma consulta ou registo que seja considerado crítico sobre esse Objecto. Como nos casos de efectuar uma reserva de um servidor, efectuar um registo de utilizador, licitar e outras que foram consideradas como "críticas", para manter a prefeita funcionalidade da plataforma.
Libertando-o assim que termine a manipulação sobre esse mesmo Objecto para que outros utilizadores possam ter acesso a este.
Os utilizadores devem, quando estão dispostos a ficar à espera de um servidor, aquardar enquanto o seu pedido é satisfeito. E após a libertacão de um servidor do tipo pretendido dessa fila de espera, ela deve ser satisfeita por ordem de chegada. Para obtermos essa propriedade da fila de espera, consideramos a utilização de uma UserQueue em que ,conforme um utilizador pretenda aguardar ele é inserido ou removido nessa estrutura de dados.
Para garatirmos que o utilizador fica a aguardar recorremos à utilização de uma Condição, em que o utilizador ao ser colocado em fila de espera ele fica bloqueado a agurdar sinal de que o seu pedido pode ser satisfeito.

Tanto a UserQueue como o utilzador também implementam uma variável de ReentrantLock() que premitem o seu repectivo bloqueio como foi referido anteriormente. Sempre que é feita a inserção de um dos utilizadores nesse Objecto(UserQueue), e desbloqueando a UserQueue após essas operações terem terminado. Ficando assim garantido a situações de concorrência, starvation e deadlock aquando estas referidas operações são efectuadas.

Recorremos aos bloqueios explícitos também no Objecto servidor. Caso seja efectuada uma reserva, ela é efectuada peor ordem de chegada, garantindo assim a concorrência sobre a reserva de um tipo de servidor. Como exemplo o caso em que dois utilizadores estejam a visualizar um servidor de um tipo e em que apenas um se encontra diponível, o que efectuou primeiro a reserva será o que vai ser satisfeito e o que fez a reserva posteriormente, irá ser colocado em fila de reserva a aguradar que um servidor desse tipo lhe seja disponibilizado.

Os utilizadores também podem ter de ser notificados não só em lista de espera, para avisar que o seu pedido pode ser satisfeito, como quando fizeram uma licitação e esse servidor foi reservado. Para garantir que os utilizadores correctos eram notificados recorremos à utilização de uma segunda classe no server que regista os outputs dos repectivos utilizadores à medida que estes se conectam ao server.

Por fim verificamos que precisamos de ter o client em modo espera para receber notificaçoes para isso criamos uma segunda thread que fica em "espera ativa" essa thread depois verifica o que é notificaçao e o que precisa de resposta.

E utilizamos um classe que chamamos de Clog que comunica entre estas duas threads.

Conclusão

Acabado o trabalho, cabe-nos fazer uma retrospetiva de todo o trabalho feito. Com a realização deste projeto usando JAVA, sem descuraros principios de encapsulamento de dados, que garantem a segurança e o bom funcionamento do programa .Em suma,todo este projeto foi realizado de modo a responder a todos os problemas propostos no enunciado. Concluímos também que para a perfeita utilização de um sistema com concorrência paralela é necessário verificar todas regiões críticas e resolver essas situções garantindo que não há starvation, ou seja que todoas as operações pretendidas são efectuadas correctamente e com a garantia de que são realmente efectuadas. Que são necessários bloqueios para que apenas o Objecto seja manipulado correctamente. E também é necessário garantir que durante todas estas operações nenhum dos Objectos efectue um bloqueio sobre alguma classe sem que efectue o desbloqueio de forma a garantir que o sitema nunca terá uma situação em que ocoora um deadlock.