Sistemas Distribuídos - GSI529

Sistema de gerenciamento de um restaurante

Nicolly Ribeiro Luz
Matrícula: 32221BSI022

1 Estado atual do projeto

Atualmente meu projeto se encontra em seu estado inicial, contando apenas com a execução por linha de comando (terminal), iniciei o desenvolvimento do sistema na linguagem Java e mais para frente pretendo fazer a integração com outras tecnologias, como por exemplo, SpringBoot. Por enquanto, o meu projeto está simples, a comunicação entre as entidades Cliente e Servidor do Restaurante já está funcionando e foi implementada com a utilização de um Socket TCP Multithreading, pois nessa primeira etapa eu já queria ver múltiplos clientes interagindo com o servidor.

Meu sistema está seguindo a seguinte interação:

1. O Cliente se conecta com o Servidor do Restaurante
2. O Servidor envia o Menu do restaurante para o cliente
3. O cliente faz um pedido por meio do ID do item do Menu e as quantidades desejadas
4. O Servidor verifica se as quantidades estão disponíveis em estoque
   - Se as quantidades forem suficientes, o Servidor confirma o pedido.
   - Caso contrário, o Servidor recusa o pedido e solicita que o cliente faça outro pedido.

2 Implementação de processos, Threads e comunicação básica

Para a implementação de uma comunicação básica no meu sistema, utilizei um Socket que opera sobre o protocolo TCP e é Multithreading, com o intuito de que os clientes pudessem se conectar com o Servidor, terem acesso ao cardápio do restaurante e que conseguissem fazer seus pedidos, de forma direta e didática.

A implementação das Threads ficou envolvida na criação do Socket, uma vez que optei para que, cada vez que um novo cliente se conectasse com o servidor, uma nova thread seja aberta e que elas possam ser processadas concorrentemente. Além disso, também me preocupei com tratamento de exceções dentro de cada thread do cliente para garantir que uma falha em uma thread não derrube o servidor inteiro.

No entanto, com múltiplos clientes fazendo pedidos ao mesmo tempo, percebi que poderia surgir um desafio de concorrência, especialmente quando se trata do controle de estoque. Para evitar que dois clientes tentassem pegar o último item disponível simultaneamente

e causassem inconsistências (como estoque negativo ou pedidos duplicados), implementei um mecanismo de sincronização de threads, que atua como um cadeado, enquanto uma thread está com o cadeado, as outras esperam na fila. Assim que a thread termina sua operação no estoque (libera o cadeado), a próxima thread na fila pode pegá-lo.

Dessa forma, garanto que a verificação completa do estoque para um pedido (mesmo que ele contenha vários itens) e a próxima atualização aconteçam de forma atômica e consistente.

Os clientes, por padrão, são single-thread em sua implementação atual. Cada Cliente do Restaurante estabelece uma única conexão com o servidor e processa suas próprias entradas e saídas.

E no que diz respeito aos processos, é possível ver que tanto os clientes quanto o servidor funcionam como programas independentes, cada um rodando por conta própria e usando seus próprios espaços de memória e capacidade de processamento.

2.1 Justificativa da utilização do TCP:

O primeiro aspecto que levei em conta foi a confiabilidade, pois o TCP garante que cada pedido (e a resposta do servidor) chegue ao destino sem perdas e na ordem correta, uma vez que um cliente fazendo um pedido e ele não chegar ao servidor, ou a confirmação do servidor não chegar ao cliente, seria um problema grave para um sistema de pedidos. Outro ponto que considerei foi a necessidade de uma conexão estabelecida para cada cliente, de forma persistente e dedicada, o que me ajuda a vizualizar que os dados que vêm de um socket particular pertencem a um cliente particular. E por fim, o controle de Fluxo e Congestionamento, já que o TCP lida automaticamente com o controle de fluxo (para não sobrecarregar o receptor) e o controle de congestionamento (para não sobrecarregar a rede), o que é vital para garantir uma comunicação estável, especialmente quando muitos clientes estão ativos.

3 Guia de Execução do Projeto

Pré requisitos essenciais

1. **Java Development Kit (JDK) e Java Virtual Machine (JVM) configurados:** Certifique-se de ter o JDK instalado e o ambiente configurado corretamente em seu sistema.
2. **Obtenção dos projetos:** Baixe os projetos do Cliente e do Servidor da pasta do GitHub onde os códigos estão publicados. Acesse o repositório através do link: Link para o Repositório do Projeto aqui.

   Agora que os projetos já foram baixados:

3. Compile primeiro o projeto do Servidor do Restaurante
4. Compile o projeto do Cliente, uma ou mais vezes, para ver a comunicação multithread.
5. Teste a comunicação entre essas duas entidades e a lógica do sistema de pedidos.