

Relatório do EP1

Alexandre Kenji Okamoto	11208371
Fernanda Cavalcante Nascimento	11390827
Gabriel Afonso Carnaiba Silva	11270886
Karina Duran	11295911
Mirela Mei	11208392

1. Descrição

O presente relatório tem como objetivo apresentar e analisar o EP de desenvolvimento que foi entregue em conjunto.

Este projeto tem como tema a camada de aplicação, e seu objetivo é servir como um sistema de pizzaria. Ele é composto por dois programas: um para uso interno da pizzaria e outro para uso externo pelos clientes. Ambos são sistemas desktop (terminal), e sua comunicação acontece via servidor.

O programa de uso da pizzaria é responsável por gerenciar informações como a disponibilidade das mesas (alocar e desalocar clientes), disponibilidade de itens do cardápio, além de enviar todas essas informações para o servidor, a fim de serem recebidas e utilizadas pelo programa dos clientes, assim como gerenciar as informações recebidas deles pelo servidor.

Já o programa utilizado pelos clientes, por sua vez, permite que os mesmos solicitem mesas, realizem pedidos, solicitem o fechamento de suas comandas para realizarem o pagamento e liberem as mesas, além de enviar todas essas informações para o servidor para que sejam administradas pelo programa da pizzaria.

2. Detalhes

Dado que o tema do projeto é a camada de aplicação, o protocolo de comunicação utilizado foi o TCP. No código, cada cliente se comunica com o servidor usando um socket, e existe uma thread para cada cliente.

O protocolo TCP, que é um protocolo confiável da camada de transporte, tem como objetivo garantir a transmissão integral e sequencial dos dados para os hosts de destino.

Esse protocolo segmenta as informações recebidas da camada de aplicação em pequenos blocos (datagramas) e insere um cabeçalho de identificação contendo um conjunto de bits (checksum) que fazem possível uma validação dos dados e do próprio cabeçalho. Além da validação, o checksum também permite a recuperação da informação em casos de erros na transmissão.

O host de destino envia, após receber os pacotes, um ACK para o host de origem, a fim de que este tenha a garantia de que eles chegaram sem erros.

3. Como compilar

O projeto foi desenvolvido utilizando a linguagem de programação Java, no ambiente de desenvolvimento Visual Studio Code. A classe main do servidor está no arquivo "PizzeriaServer.java", enquanto que a classe main do cliente está no arquivo "PizzeriaClient.java". Para compilar o código, basta abrir um terminal na pasta "src" e executar o comando "javac *.java". Entretanto, a pasta "bin" já possui todos os arquivos compilados para agilizar o processo, caso necessário.

4. Como executar

Para executar o programa, é necessário executar o arquivo *pizzariaServer.jar* para iniciar o lado do servidor, onde será mostrada a mensagem "Servidor aberto na porta 4000".

Após isso, para iniciar o lado do cliente é necessário executar o arquivo *pizzariaClient.jar*, fazendo com que o menu de comandos seja exibido.

(Obs.: Para executar no Windows, os arquivos são: *servidor.jar* e *cliente.jar*, dentro da pasta /bin)

Deve-se então, no lado do cliente, digitar o número 1, para realizar a reserva da mesa, indicando a quantidade de pessoas. É possível visualizar os comandos recebidos pelo cliente no terminal do servidor.

Após a reserva da mesa, digitando o número 2 será enviado o cardápio para realizar o pedido. É preciso digitar o número das pizzas desejadas, uma a uma. Depois digitar 0 para finalizar o pedido, voltando para o menu inicial.

Após isso, há a possibilidade de fazer outros pedidos (que serão somados na conta final) ou fechar a conta, para realizar o pagamento e liberar a mesa (opção de número 3).

Então, depois de efetuar o pagamento, deve-se avaliar o atendimento (número 4), e desconectar o cliente (número 5).

5. Como ler o código

- **PizzeriaServer:** onde está o método *main* do Server, que irá instanciar o PizzeriaServer e abrirPizzeria. A porta utilizada será de número 4000 e há um LinkedHashSet<ClientSocket> para guardar os clientes que irão se conectar. O método *start* irá instanciar o ServerSocket e chamará o método *clientConnectionLoop* para gerar um loop infinito, instanciar o ClientSocket e criar threads conforme os clientes iniciam a conexão. O método *clientMessageLoop* recebe um ClientSocket como parâmetro e verifica a ocorrência de novos comandos do lado do cliente, para realizar a lógica necessária conforme o número enviado. O método *enviarMsgCliente* irá enviar os comandos recebidos pelo cliente, e o *cardapio* irá imprimir as pizzas.

- **PizzariaClient:** onde está o método *main* do Cliente, que irá instanciar o *pizzariaClient* e chamará o método *start*. O IP será 127.0.0.1.
O método *start* irá instanciar o *ClientSocket* com o IP mencionado e a porta declarada no *Server*.
O método *run* aguarda os comandos do cliente e imprime a mensagem.
O método *messageLoop* irá processar os comandos recebidos e enviar a mensagem para o servidor realizar a lógica necessária. Os métodos *menu*, *ajuda* e *solicitarCardapio* realizam o básico para a impressão do menu inicial.
- **Restaurante:** classe para instanciar o restaurante, possui uma variável *List* para armazenar mesas, cardapio, pedidos, clientesConectados e *avaliacoesPizzaria*. O método *GetRestauranteSingleton* irá garantir apenas uma instância de restaurante sendo utilizada.
O método *abrirPizzaria* limpará as mesas, cardapio, clientes e pedidos, para adicioná-los novamente diante da nova conexão.
O método *reservarMesa* irá realizar a lógica de reserva, garantindo que ainda existam mesas disponíveis e que o cliente esteja conectado.
O método *adicionarAoPedido* irá instanciar as pizzas pedidas pelo cliente e adicionar a lista. O método *clienteEntrou* adiciona um novo cliente ao restaurante.
O método *adicionarPedidoCliente* irá adicionar o pedido ao ID do cliente, para somar ao final e o método *fecharPedido* irá fechar a conta e o cliente, liberando a mesa. O método *adicionarAvaliacao* salva a avaliação do cliente, e o método *acharCliente* busca o cliente pelo ID.
- **Cliente:** classe para instanciar o cliente. Possui um ID único, um *Pedido*, *mesaReservada* e uma tag *possuiReserva* para indicar se há uma mesa instanciada em seu ID.
- **ClientSocket:** classe do *Socket* do cliente. Realiza as operações de *getMessage*, *sendMessage*, *getRemoteSocketAddress*, *getLocalPort* e *close*, para finalizar o cliente.
- **MesasPizzaria:** classe para instanciar uma mesa. Possui um *int* para *quantidadeLugares*, um *String* com o cliente que reservou, e uma tag para indicar se está disponível.
- **Pedido:** classe para instanciar um pedido novo. Possui um identificador do pedido, um ID do cliente que o fez, e um *ArrayList* para guardar os itens. O método *totalPedido* irá somar os itens e seus respectivos valores.
- **ItemCardapio:** classe para um item do cardápio, métodos *getSabor* e *getValor*.
- **Avaliacao:** classe para avaliações recebidas. Possui um *int* para nota e uma *String* para o comentário.

- **Pizza*:** classes com as pizzas disponíveis e seus respectivos valores.

6. Testes e resultados obtidos

O teste inicial: Abertura do socket.

```

Selecionar Prompt de Comando - java PizzeriaServer
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\kah_d\src_rc_pizzaria>src

C:\Users\kah_d\src_rc_pizzaria>java PizzeriaServer
Servidor aberto na porta 4000

Prompt de Comando - java PizzeriaClient
C:\Users\kah_d\src_rc_pizzaria>java PizzeriaClient
Cliente 53554 conectado ao servidor 127.0.0.1:4000
===== INICIO MENU =====
- Insira o código do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliação para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
  
```

Teste para reserva de mesa para 5 pessoas

```

Prompt de Comando - java PizzeriaServer
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\kah_d\src_rc_pizzaria>src

C:\Users\kah_d\src_rc_pizzaria>java PizzeriaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906

Prompt de Comando - java PizzeriaClient
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\kah_d\src_rc_pizzaria>src

C:\Users\kah_d\src_rc_pizzaria>java PizzeriaClient
Cliente 61906 conectado ao servidor 127.0.0.1:4000
===== INICIO MENU =====
- Insira o código do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliação para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
1
Insira o número de pessoas:
5
===== INICIO MENU =====
- Insira o código do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliação para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
Seu número de cliente é: 61906
Mesa 4 reservada para o cliente 61906
  
```

Teste com número excedente de pessoas(9, sendo o máximo 6):

```
Prompt de Comando - java PizzariaServer
C:\Users\ksh_d\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:51028: 1 6
Mesa 4 reservada para o cliente 51028
Comando recebido do cliente /127.0.0.1:51030: 1 3
Mesa 2 reservada para o cliente 51030
Comando recebido do cliente /127.0.0.1:51024: 1 9
Comando recebido do cliente /127.0.0.1:51024: 1 6
Mesa 5 reservada para o cliente 51024

Prompt de Comando - java PizzariaClient
Error: Could not find or load main class PizzariaCliente
Caused by: java.lang.ClassNotFoundException: PizzariaCliente
C:\Users\ksh_d\src>java PizzariaClient
Cliente 51024 conectado ao servidor 127.0.0.1:4000
===== INICIO MENU =====
- Insira o cdigo do comando desejado:

(0) Exibir menu.

(1) Reservar mesa: Solicita a reserva de uma mesa.

(2) Fazer pedido: Adiciona o item desejado ao pedido.

(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.

(4) Avaliar: Deixe uma avaliacao para nosso atendimento.

(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
1
Insira o nmero de pessoas:
9
===== INICIO MENU =====
- Insira o cdigo do comando desejado:

(0) Exibir menu.

(1) Reservar mesa: Solicita a reserva de uma mesa.

(2) Fazer pedido: Adiciona o item desejado ao pedido.

(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.

(4) Avaliar: Deixe uma avaliacao para nosso atendimento.

(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
Seu nmero de cliente : 51024
Nao ha mesas disponiveis.
4
Insira o nmero de pessoas:
6
===== INICIO MENU =====
- Insira o cdigo do comando desejado:

(0) Exibir menu.
```

Teste: pedindo as pizzas

```
Prompt de Comando - java PizzariaServer
C:\Users\ksh_d\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio

Prompt de Comando - java PizzariaClient
===== FIM MENU =====
Seu nmero de cliente : 61906
Mesa 4 reservada para o cliente 61906
2
===== CARDAPIO =====
Insira o nmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.

2
===== CARDAPIO =====
Insira o nmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.

3
===== CARDAPIO =====
Insira o nmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.

5
===== CARDAPIO =====
Insira o nmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.
```

Teste: adicionando novos sabores a um pedido j existente:

```
Prompt de Comando - java PizzariaServer
C:\Users\ksh_dxcd_rc_pizzaria\src
C:\Users\ksh_dxcd_rc_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 2 3 5 0
Pizza de Calabresa adicionada ao pedido
Pizza de Frango adicionada ao pedido
Pizza de chocolate adicionada ao pedido
Pedido realizado pelo cliente 61906 no valor de 78 reais.
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 6 0
Pizza de Romeu e Julieta adicionado ao pedido
Pedido realizado pelo cliente 61906 no valor de 26 reais.

Prompt de Comando - java PizzariaClient
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Pedido realizado pelo cliente 61906 no valor de 78 reais.
Caso ja tenha realizado um pedido anterior, os itens serao adicionados a ele.
2
===== CARDAPIO =====
Insira o nÂmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.
6
===== CARDAPIO =====
Insira o nÂmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.
0
===== INICIO MENU =====
- Insira o cÂdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Caso ja tenha realizado um pedido anterior, os itens serao adicionados a ele.
```

Teste: fechando o pedido:

```
Prompt de Comando - java PizzariaServer
C:\Users\ksh_dxcd_rc_pizzaria\src
C:\Users\ksh_dxcd_rc_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 2 3 5 0
Pizza de Calabresa adicionada ao pedido
Pizza de Frango adicionada ao pedido
Pizza de chocolate adicionada ao pedido
Pedido realizado pelo cliente 61906 no valor de 78 reais.
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 6 0
Pizza de Romeu e Julieta adicionado ao pedido
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Comando recebido do cliente /127.0.0.1:61906: 3

Prompt de Comando - java PizzariaClient
^ Insira o nÂmero do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.
0
===== INICIO MENU =====
- Insira o cÂdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Caso ja tenha realizado um pedido anterior, os itens serao adicionados a ele.
3
Estamos processando sua solicitatÂo.
===== INICIO MENU =====
- Insira o cÂdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu pedido no valor de: 104 reais sera finalizado e sua mesa numero: 4 liberada.
Seu pedido foi finalizado e sua mesa liberada. Favor aguardar garÂom para realizar pagamento.
```

Teste com mÚltiplos clientes(3):

```
Prompt de Comando - java PizzariaClient
Insira o n mero de pessoas:
===== INICIO MENU =====
- Insira o c digo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu n mero de cliente  : 51028
Mesa 4 reservada para o cliente 51028

Prompt de Comando - java PizzariaClient
===== INICIO MENU =====
- Insira o c digo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu n mero de cliente  : 51030
Mesa 2 reservada para o cliente 51030

Prompt de Comando - java PizzariaServer
C:\Users\ksh_d\src_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:51028: 1 6
Mesa 4 reservada para o cliente 51028
Comando recebido do cliente /127.0.0.1:51030: 1 3
Mesa 2 reservada para o cliente 51030
Comando recebido do cliente /127.0.0.1:51024: 1 9
Comando recebido do cliente /127.0.0.1:51024: 1 6
Mesa 5 reservada para o cliente 51024

Prompt de Comando - java PizzariaClient
===== INICIO MENU =====
- Insira o c digo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu n mero de cliente  : 51024
Mesa 5 reservada para o cliente 51024
```

Teste: todas as mesas ocupadas.

```
Prompt de Comando - java PizzariaServer
Microsoft Windows [vers o 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\ksh_d>cd src_pizzaria\src
C:\Users\ksh_d\src_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:53764: 1 6
Mesa 4 reservada para o cliente 53764
Comando recebido do cliente /127.0.0.1:53773: 1 5
Mesa 5 reservada para o cliente 53773
Comando recebido do cliente /127.0.0.1:53775: 1 4
Mesa 2 reservada para o cliente 53775
Comando recebido do cliente /127.0.0.1:53777: 1 3
Mesa 3 reservada para o cliente 53777
Comando recebido do cliente /127.0.0.1:53779: 1 5
Comando recebido do cliente /127.0.0.1:53779: 1 2
Mesa 0 reservada para o cliente 53779
Comando recebido do cliente /127.0.0.1:53768: 1 4
Comando recebido do cliente /127.0.0.1:53774: 1 2
Mesa 1 reservada para o cliente 53774
Comando recebido do cliente /127.0.0.1:53768: 1 2

Prompt de Comando - java PizzariaClient
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu n mero de cliente  : 53768
Nao ha mesas disponiveis.
1
Insira o n mero de pessoas:
2
===== INICIO MENU =====
- Insira o c digo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.
===== FIM MENU =====
Seu n mero de cliente  : 53768
Nao ha mesas disponiveis.
```

Teste: Cliente desconectado:


```

Prompt de Comando - java PizzariaServer
C:\Users\kah_d\src
C:\Users\kah_d\src>cd rc_pizzaria\src
C:\Users\kah_d\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 2 3 5 0
Pizza de Calabresa adicionada ao pedido
Pizza de Frango adicionada ao pedido
Pizza de chocolate adicionada ao pedido
Pedido realizado pelo cliente 61906 no valor de 78 reais.
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 6 0
Pizza de Romeu e Julieta adicionado ao pedido
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Comando recebido do cliente /127.0.0.1:61906: 3
Comando recebido do cliente /127.0.0.1:61906: 4 6 demorou muito
Avaliacao adicionada.
Comando recebido do cliente /127.0.0.1:61906: 5
Cliente 61906 se desconnectou.
java.io.IOException: Stream closed
    at java.base/java.io.BufferedReader.ensureOpen(BufferedReader.java:122)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:319)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:392)
    at ClientSocket.getMessage(ClientSocket.java:21)
    at PizzariaServer.clientMessageLoop(PizzariaServer.java:41)
    at PizzariaServer.lambda$ClientConnectionLoop$0(PizzariaServer.java:32)
    at java.base/java.lang.Thread.run(Thread.java:834)

Prompt de Comando
- Insira o cÃdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
Seu pedido no valor de: 104 reais sera finalizado e sua mesa numero: 4 liberada.
Seu pedido foi finalizado e sua mesa liberada. Favor aguardar garÃom para realizar pagamento.
Por favor, informe uma nota de 0 a 10 para nosso atendimento:
6
Gostaria de deixar um comentario?
1 - Sim
2- Nao
1
Insira seu comentario:
demorou muito
===== INICIO MENU =====
- Insira o cÃdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
Desconectado.
Obrigado e volte sempre!
C:\Users\kah_d\src>

```

Teste: mesa liberada

```

Prompt de Comando - java PizzariaServer
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\kah_d\src
C:\Users\kah_d\src>cd rc_pizzaria\src
C:\Users\kah_d\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:53764: 1 6
Mesa 4 reservada para o cliente 53764
Comando recebido do cliente /127.0.0.1:53773: 1 5
Mesa 5 reservada para o cliente 53773
Comando recebido do cliente /127.0.0.1:53775: 1 4
Mesa 2 reservada para o cliente 53775
Comando recebido do cliente /127.0.0.1:53777: 1 3
Mesa 3 reservada para o cliente 53777
Comando recebido do cliente /127.0.0.1:53779: 1 5
Comando recebido do cliente /127.0.0.1:53779: 1 2
Mesa 0 reservada para o cliente 53779
Comando recebido do cliente /127.0.0.1:53768: 1 4
Comando recebido do cliente /127.0.0.1:53774: 1 2
Mesa 1 reservada para o cliente 53774
Comando recebido do cliente /127.0.0.1:53768: 1 2
Comando recebido do cliente /127.0.0.1:53773: cardapio
Comando recebido do cliente /127.0.0.1:53773: cardapio
Comando recebido do cliente /127.0.0.1:53773: cardapio
Comando recebido do cliente /127.0.0.1:53773: 2 6 4 0
Pizza de Romeu e Julieta adicionado ao pedido
Pizza de palmito adicionada ao pedido.
Pedido realizado pelo cliente 53773 no valor de 49 reais.
Comando recebido do cliente /127.0.0.1:53773: 3
Comando recebido do cliente /127.0.0.1:53773: 4 10 Sem comentario
Avaliacao adicionada.
Comando recebido do cliente /127.0.0.1:53773: 5
Cliente 53773 se desconnectou.
java.util.ConcurrentModificationException
    at java.base/java.util.LinkedHashMap$LinkedHashMapIterator.nextNode(LinkedHashMap.java:719)
    at java.base/java.util.LinkedHashMap$LinkedKeyIterator.next(LinkedHashMap.java:741)
    at PizzariaServer.clientMessageLoop(PizzariaServer.java:120)
    at PizzariaServer.lambda$ClientConnectionLoop$0(PizzariaServer.java:32)
    at java.base/java.lang.Thread.run(Thread.java:834)
Comando recebido do cliente /127.0.0.1:53768: 1 2
Mesa 5 reservada para o cliente 53768

Selecionar Prompt de Comando - java PizzariaClient
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
Seu nÃmero de cliente Ã0: 53768
Nao ha mesas disponiveis.
1
Insira o nÃmero de pessoas:
2
===== INICIO MENU =====
- Insira o cÃdigo do comando desejado:
(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzaria.
===== FIM MENU =====
Seu nÃmero de cliente Ã0: 53768
Mesa 5 reservada para o cliente 53768

```

Teste: avaliação feita:


```
Prompt de Comando - java PizzariaServer
C:\Users\kqh_d\src\pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:61906: 1 5
Mesa 4 reservada para o cliente 61906
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 2 3 5 0
Pizza de Calabresa adicionada ao pedido
Pizza de Frango adicionada ao pedido
Pizza de chocolate adicionada ao pedido
Pedido realizado pelo cliente 61906 no valor de 78 reais.
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: cardapio
Comando recebido do cliente /127.0.0.1:61906: 2 6 0
Pizza de Romeu e Julieta adicionada ao pedido
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Comando recebido do cliente /127.0.0.1:61906: 3
Comando recebido do cliente /127.0.0.1:61906: 4 6 demorou muito
Avaliacao adicionada.

Prompt de Comando - java PizzariaClient
===== FIM MENU =====
Pedido realizado pelo cliente 61906 no valor de 26 reais.
Caso ja tenha realizado um pedido anterior, os itens serao adicionados a ele.
3
Estamos processando sua solicitacao.
===== INICIO MENU =====
- Insira o cAldigo do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.

4
===== FIM MENU =====
Seu pedido no valor de: 104 reais sera finalizado e sua mesa numero: 4 liberada.
Seu pedido foi finalizado e sua mesa liberada. Favor aguardar garAaom para realizar pagamento.
Por favor, informe uma nota de 0 a 10 para nosso atendimento:
6
Gostaria de deixar um comentario?
1 - Sim
2- Nao
1
Insira seu comentario:
demorou muito
===== INICIO MENU =====
- Insira o cAldigo do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
```

Teste: Comando inválido(6);

```
Prompt de Comando - java PizzariaServer
C:\Users\kqh_d\src\pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:53066: 6

Prompt de Comando - java PizzariaClient
C:\Users\kqh_d\src\pizzaria\src>java PizzariaClient
Cliente 53066 conectado ao servidor 127.0.0.1:4000
===== INICIO MENU =====
- Insira o cAldigo do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.

6
===== FIM MENU =====
Comando invalido
Digite "0" para ver o menu do cliente.
===== INICIO MENU =====
- Insira o cAldigo do comando desejado:

(0) Exibir menu.
(1) Reservar mesa: Solicita a reserva de uma mesa.
(2) Fazer pedido: Adiciona o item desejado ao pedido.
(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.
(4) Avaliar: Deixe uma avaliacao para nosso atendimento.
(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
```

Teste: pedido sem mesa:

```
Prompt de Comando - java PizzariaServer
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\kqh_d>cd rc_pizzaria
C:\Users\kqh_d\rc_pizzaria>cd src
C:\Users\kqh_d\rc_pizzaria\src>javac *.java
C:\Users\kqh_d\rc_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
Comando recebido do cliente /127.0.0.1:64558: cardapio
Comando recebido do cliente /127.0.0.1:64558: cardapio
Comando recebido do cliente /127.0.0.1:64558: cardapio
Comando recebido do cliente /127.0.0.1:64558: cardapio
Comando recebido do cliente /127.0.0.1:64558: cardapio
Comando recebido do cliente /127.0.0.1:64558: 2 5 8 5 4 0
Solicitacao de pedido do cliente negada. Cliente nao possui mesa reservada.

Prompt de Comando - java PizzariaClient
4
===== CARDAPIO =====
Insira o número do item desejado
1 - Pizza Mussarela (R$ 20,00)
2 - Pizza Calabresa (R$ 30,00)
3 - Pizza Frango (R$ 22,00)
4 - Pizza Palmito (R$ 23,00)
5 - Pizza Chocolate (R$ 26,00)
6 - Pizza Romeu e Julieta (R$ 26,00)
0 - Parar.

0
===== INICIO MENU =====
- Insira o código do comando desejado:

(0) Exibir menu.

(1) Reservar mesa: Solicita a reserva de uma mesa.

(2) Fazer pedido: Adiciona o item desejado ao pedido.

(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.

(4) Avaliar: Deixe uma avaliacao para nosso atendimento.

(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
Nao eh possivel realizar um pedido sem reservar uma mesa.
```

Teste: Servidor Fechou inesperadamente

```
Prompt de Comando - java PizzariaClient
===== INICIO MENU =====
- Insira o código do comando desejado:

(0) Exibir menu.

(1) Reservar mesa: Solicita a reserva de uma mesa.

(2) Fazer pedido: Adiciona o item desejado ao pedido.

(3) Fechar pedido: Encerra o pedido para liberar mesa e realizar pagamento.

(4) Avaliar: Deixe uma avaliacao para nosso atendimento.

(5) Desconectar: desconecta o cliente da pizzeria.

===== FIM MENU =====
java.net.SocketException: Connection reset
at java.base/java.net.SocketInputStream.read(SocketInputStream.java:186)
at java.base/java.net.SocketInputStream.read(SocketInputStream.java:140)
at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:284)
at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:326)
at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:178)
at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:181)
at java.base/java.io.InputStreamReader.read(InputStreamReader.java:161)
at java.base/java.io.BufferedReader.readLine(BufferedReader.java:326)
at java.base/java.io.BufferedReader.readLine(BufferedReader.java:392)
at ClientSocket.getMessage(ClientSocket.java:21)
at PizzariaClient.run(PizzariaClient.java:38)
at java.base/java.lang.Thread.run(Thread.java:834)
```

Teste: Se o Client fecha inesperadamente.

```
Prompt de Comando - java PizzariaServer
Microsoft Windows [versão 10.0.19042.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\kah_d>cd rc_pizzaria
C:\Users\kah_d\rc_pizzaria>cd src
C:\Users\kah_d\rc_pizzaria\src>java PizzariaServer
Servidor aberto na porta 4000
java.net.SocketException: Connection reset
    at java.base/java.net.SocketInputStream.read(SocketInputStream.java:186)
    at java.base/java.net.SocketInputStream.read(SocketInputStream.java:140)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:284)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:326)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:178)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:181)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:326)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:392)
    at ClientSocket.getMessage(ClientSocket.java:21)
    at PizzariaServer.clientMessageLoop(PizzariaServer.java:41)
    at PizzariaServer.lambda$clientConnectionLoop$0(PizzariaServer.java:32)
    at java.base/java.lang.Thread.run(Thread.java:834)
```

7. Fontes

Para uso dos padrões Abstract Factory e Singleton:

<https://refactoring.guru/pt-br/design-patterns/java>

Para entender o uso das bibliotecas de Sockets e Threads em java:

▶ #1 Criando aplicação CLIENTE/SERVIDOR de CHAT em JAVA usando SOCK...

▶ #2 Criando aplicação CLIENTE/SERVIDOR de CHAT em JAVA usando SOCK...