Laboratorio 2
Aplicação cliente/servidor básica

Sistemas Distribuídos (MAB-733)
Profa. Silvana Rossetto

[1]Instituto de Computação/UFRJ

Introdução

O objetivo deste Laboratório é desenvolver uma aplicação distribuída básica para aplicar os conceitos estudados sobre arquitetura de software em camada e arquitetura de sistema centralizada (cliente/servidor); e seguir praticando com a programação distribuída usando sockets.

A aplicação que vamos desenvolver consiste em *contar o número de ocorrências de uma palavra em um arquivo texto*.

- Entrada: O usuário informa o nome do arquivo texto e a palavra de busca.
- Saída (com sucesso): exibe o número de ocorrências (pode ser zero ou mais) da palavra no arquivo.
- Saída (com erro): *informa que o arquivo solicitado não foi encontrado*

Nas atividades 1 e 2, já temos os projetos de arquitetura de software e arquitetura de sistema pré-concebidos, faltando apenas fazer os refinamentos solicitados.

Atividade 1

Objetivo: Refinar a arquitetura de software — usando o estilo arquitetural em camadas — apresentada abaixo.

Camadas:

1. Funcionalidades da camada de interface com o usuário: recebe do usuário o nome do arquivo e a palavra de busca e exibe na tela o resultado do processamento. O resultado do processamento poderá ser: *(i) uma mensagem de erro indicando que o arquivo não foi encontrado*; ou *(ii) o número de ocorrências da palavra no arquivo*. As mensagens já virão prontas do lado servidor. Já sairão prontas desde a camada de processamento (no projeto, foi feita no arquivo “service.py”)
2. Funcionalidades da camada de processamento: solicita o acesso ao arquivo texto. Se o arquivo for válido, realiza a busca pela palavra informada e prepara a resposta para ser devolvida para a camada de interface. Se o arquivo for invalido, responde com a mensagem de erro. O resultado foi entregue conforme o enunciado. A camada de processamento, recebe ou o texto do arquivo, ou uma mensagem

informando que não encontrou o arquivo. Na própria camada de processamento, a palavra pesquisada é contada, e uma mensagem já pronta e formatada (tanto com a informação de sucesso ou de erro) é devolvida para o lado cliente da aplicação.

3. Funcionalidades da camada de acesso aos dados: verifica se o arquivo existe em sua base. Se sim, devolve o seu conteúdo inteiro. Caso contrário, devolve uma mensagem de erro.
   A camada de acesso a dados foi implementada conforme solicitado no item3.

Tarefa: Em um arquivo PDF, repita as funcionalidades descritas para cada camada, substituindo as partes em vermelho pelas decisões tomadas.

Atividade 2

Objetivo: Refinar a proposta de instanciação da arquitetura de software da aplicação definida na Atividade 1 para uma arquitetura de sistema cliente/servidor de dois n´ıveis, com um servidor e um cliente, apresentada abaixo.

Proposta de arquitetura de sistema:

1. Lado cliente: implementa a camada de interface com o usuário (no projeto, feita pelo arquivo search.py). O usuário poderá solicitar o processamento de uma ou mais buscas em uma única execução da aplicação: o programa espera pelo nome do arquivo e da palavra de busca, faz o processamento, retorna o resultado, e então aguarda um novo pedido de arquivo e palavra ou o comando de finalização.
2. Lado servidor: implementa a camada de processamento (no projeto, feita pelo arquivo service.py) e a camada de acesso aos dados (no projeto, feito pelo arquivo repository.py). Projete um servidor iterativo, isto é, que trata as requisições de um cliente de cada vez, em um único fluxo de execução (estudaremos essa classificação depois). Terminada a interação com um cliente, ele poderá voltar a esperar por uma nova conexão. Dessa forma, o programa do servidor fica em loop infinito (depois veremos como lidar com isso).

Refinar:

1. **Especificar os tipos e a sequência de mensagens que serão trocadas entre cliente e servidor, considerando um comportamento requisição/resposta:**

As mensagens trocadas entre o cliente e o servidor são strings do tipo simples, seguiram a seguinte sequência:

O processo é iniciado com a camada do servidor (arquivo server.py) aguarda o envio de uma mensagem;

A camada cliente (arquivo search.py) é iniciado em um loop, que aguarda a string com a informação do arquivo e da palavra que será pesquisada, ou o comando para encerrar o programa; Uma vez informado os dados de entrada, envia os dados para a camada servidora (arquivo server.py);

A camada de entrada do lado server (arquivo server.py), também é realizada em um loop. Repassa os dados para a camada de processamento (arquivo service.py), que por sua vez consome a camada de acesso a dados (arquivo repository.py), que identifica se o arquivo informado existe e retorna o seu conteúdo (ou um mensagem de erro) para a camada de processamento.

A camada de processamento (arquivo service.py) recebe o texto do documento solicitado, conta as ocorrências da palavra pesquisada, formata a massagem e retorna para o arquivo server.py, que por sua vez vai enviar a mensagem formatada via socket para a camada cliente, voltando em seguida a aguardar uma nova mensagem.

A camada cliente apresenta a mensagem com o resultado da consulta (seja um resultado de sucesso ou de erro) e inicia uma nova pedido de entrada de dados, para iniciar uma nova pesquisa

2. Definir as estruturas de dados que serão usadas e o conteúdo das mensagens que serão trocadas entre cliente e servidor;

Não foi definida uma estrutura formal (como um struct ou uma classe) para a troca de mensagens. As mensagens foram basicamente strings simples. O cliente em uma única mensagem informa o nome do arquivo e a palavra que se deseja pesquisar (separadas por vírgula), no formato: NomeDocumento,PalavraPesquisa. Ex: book01.txt,master isso vai informar a quantidade de ocorrências da palavra master no documento book01.txt

3. Detalhar outras decisões de implementação do lado do cliente e do lado do servidor.

Ainda na camada cliente (no arquivo search.py), foi feita uma validação da string de entrada, para garantir que esteja no formato esperado. Caso a string não esteja no formato adequado, a solicitação não é encaminhada para o lado do servidor, e é solicitado ao usuário, que refaça a sua consulta.

Tarefa: No arquivo PDF, repita as funcionalidades descritas para o lado cliente e o lado servidor, complementando as definições solicitadas em vermelho.

Atividade 3

Objetivo: Implementar e avaliar a aplicação distribuída proposta, seguindo as definições da Atividade 2.

Roteiro:

1. Implemente o código do lado cliente e do lado servidor;
2. Documente o codigo de forma concisa e clara;
3. Experimente a aplicação usando diferentes arquivos de entrada e palavras de busca.

Disponibilize seu código e disponibilize o arquivo PDF e o código da sua aplicação em um ambiente de acesso remoto (GitHub ou GitLab), e use o formulário de entrega desse laboratório para passar as informações solicitadas.