Refinamentos do laboratório 2 de Sistemas Distribuídos:

Matheus Simões - DRE: 117091021

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 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) a lista de palavras com suas ocorrências.
Refinamento:
A lista de palavras virá já ordenada e limitada às 10 mais comuns para a camada de interface. Logo, a camada de interface só precisa receber a lista de palavras e exibir cada uma mantendo a ordem.

2. Funcionalidades da camada de processamento:
Solicita o acesso ao arquivo texto. Se o arquivo for válido, realiza a contagem das palavras e prepara a resposta para ser devolvida para a camada de interface. Se o arquivo for inválido, responde com a mensagem de erro.
Refinamento:
Caso o arquivo seja válido, as palavras serão ordenadas de acordo com suas respectivas contagens e as 10 primeiras serão enviadas para a camada de interface, a partir de um dicionário do python, em formato de json, onde as palavras são as chaves e seus contadores são os valores.

3. Funcionalidades da camada de acesso aos dados:
Verifica se o arquivo existe em sua base. Se sim, devolve seu conteúdo inteiro. Caso contrário, envia uma mensagem de erro.

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 níveis, com um servidor e um cliente, apresentada abaixo.

Proposta de arquitetura de sistema:
1. Lado cliente: implementa a camada de interface com o usuário. O usuário poderá solicitar o processamento de um ou mais arquivos em uma única execução da aplicação: o programa espera pelo nome do arquivo, faz o processamento, retorna o resultado, e então aguarda um novo pedido de arquivo ou o comando de finalização.

2. Lado servidor: implementa a camada de processamento e a cada de acesso aos dados. Implemente um servidor iterativo, isto é, que trata as requisições de uma cliente de cada vez, em um único fluxo de execução (estudaremos essa classificação depois). Terminada a

interação com o 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).

Refinamento:
1.
- Estrutura de dados que serão utilizadas: Dicionário do python para associar as palavras (keys) aos seus respectivos contadores (valor).
- A mensagem de requisição do cliente para o servidor possui apenas a string com o nome do arquivo de busca. O nome do arquivo deve conter o sufixo.
- A mensagem de resposta envia o Dicionário do python com as 10 palavras mais comuns em formato de um json serializado, ou seja, um json como string, no caso em que o arquivo é válido e envia uma string com a mensagem de erro no caso em que o arquivo é inválido. Na mensagem de erro, os primeiros 5 caracteres serão "Erro:" para que seja possível diferenciar uma mensagem de erro e uma mensagem de sucesso no lado do cliente.
2.
O cliente envia uma requisição com uma string com o nome do arquivo de busca ao servidor.
O servidor recebe o nome do arquivo do cliente e, após o processamento, envia uma resposta com o json das 10 palavras mais comuns
ou a mensagem de erro.
3.
Do lado do cliente, ao receber a resposta, verifica se os primeiro caracteres correspondem à "Erro:", se sim, apenas printa a mensagem de erro, caso contrário pega a string que deve conter um json e transforma novamente em Dicionário do python, então itera esse Dicionário para printar cada palavra mantendo a ordem a qual veio.

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 código de forma concisa e clara;
3. Experimente a aplicação usando diferentes arquivos de entrada.

Implementação do código do lado do cliente feito e comentado/documentado no arquivo: cliente.py
Implementação do código do lado do servidor feito e comentado/documentado no arquivo: servidor.py

Arquivos utilizados como teste foram: inputTest1.txt e inputTest2.txt