Aluno Victor Augusto Souza de Oliveira

Atividade 1

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*.
A lista de palavras virá pronta para exibição.

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.
Entrega a lista de palavras já pronta, sem necessidade de modificações para a camada de interface.

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, envia uma mensagem de erro.

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.

A camada de interface estará toda contida no lado do cliente.
Cliente envia mensagem contendo o nome do arquivo e recebe do servidor uma mensagem contendo a lista de palavras com suas ocorrências, já ordenada, ou uma mensagem de erro caso o arquivo não exista. Para desconectar o cliente digite *stop.*

2. Lado servidor: implementa a camada de processamento e a camada de acesso aos dados. Implemente um servidor concorrente, ou seja, que trate cada nova conexão de cliente como um novo fluxo de execução e atenda as requisições desse cliente dentro do novo fluxo de execução. O servidor é portanto capaz de lidar com múltiplos clientes ao mesmo tempo bem como de receber comandos básicos da entrada padrão. Caso um cliente se desconecte, o servidor continua disponível para novas conexões. As camadas de dados e processamento estão totalmente contidas no lado do servidor.

Para implementar a concorrência no servidor foi escolhida a criação de threads sempre que há uma nova conexão. A thread será responsável por atender às requisições de um cliente e termina quando o cliente desconecta.

Será utilizado uma estrutura chamada Counter, da classe collections. Seu funcionamento é semelhante a um dicionário em Python porém possui funções que facilitam muito a contagem do número de palavras.

Funcionamento resumido do atendimento de um cliente no servidor:

O servidor recebe uma mensagem do cliente contendo o nome do arquivo.
A camada de processamento recebe essa mensagem e transmite para a camada de dados que tentará ler o arquivo, caso exista retornará uma *string* contendo o texto no arquivo, caso não exista retorna um booleano *False*.
A camada de processamento checa se o retorno é *False* ou texto. Se é *False* então a camada de processamento envia para o cliente a mensagem de erro *"Arquivo não encontrado"* .
Se é texto então a camada de processamento executa uma série de passos para limpar o texto de símbolos, acentos,letras maiúsculas e etc e por fim executa o procedimento para descobrir as 10 palavras com mais ocorrências. Em seguida, envia uma string contendo essa informação, já ordenada, para o cliente.

Comandos da Entrada Padrão:

fim: termina o processo servidor caso não haja conexões ativas.
listar: lista conexões ativas.