Laboratório 4
Aplicação de transferência de arquivos

Sistemas Distribuídos (MAB-733)
Profa. Silvana Rossetto

[1]Instituto de Computação/UFRJ

Aluno: Saulo Andrade Almeida

Período: Bloco IV, 2021

Atividade 1

Objetivo: Projetar a interface de usuário da aplicação.

A interface da aplicação continuará simples como as versões anteriores. Continuará tendo uma interface em modo texto.

Existirão apenas dois comandos possíveis.

1. No primeiro (e principal), que é executado assim que o usuário inicia a aplicação cliente. Será solicitado ao usuário informar o nome do arquivo que ele deseja transferir do lado servidor para o lado cliente, ou digitar fim para finalizar a aplicação.
    a. Caso o usuário digite um arquivo que exista no lado do servidor, uma cópia do arquivo de texto será criado no lado do cliente, na pasta de download.
    b. Caso digite um arquivo que não exista no lado do servidor, uma mensagem de erro será informada ao usuário
2. Se durante a solicitação do arquivo, o usuário digitar a palavra "fim" a aplicação cliente será finalizada.

Atividade 2

Objetivo: Projetar a arquitetura de software da aplicação

A arquitetura de software utilizada no laboratório também manteve o padrão de camadas utilizado nos laboratórios anteriores.

Onde cada componente faz uma chamada para uma camada inferior e normalmente recebe uma resposta. Conforme apresentado na Figura 1 apresentada logo abaixo.

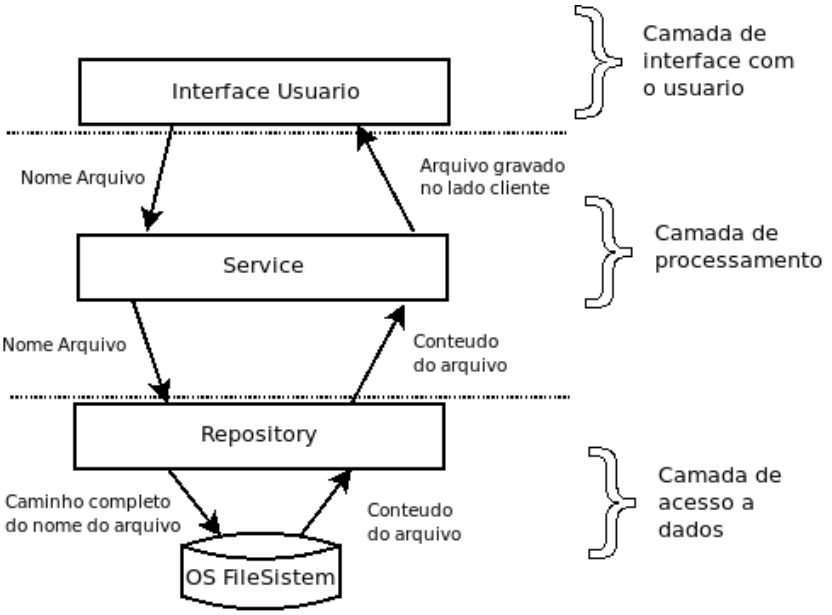

Figura 1

Do lado cliente, também foi criada uma camada para estabelecer o protocolo de comunicação e maiores detalhes dessa camada serão apresentados na arquitetura da aplicação e no protocolo de aplicação, nas atividades 3 e 4.

Atividade 3

Objetivo: Projetar a arquitetura de sistema da aplicação.

A arquitetura da aplicação também continuou evoluindo às versões anteriores da aplicação. Continuou sendo feita de forma centralizada, seguindo o modelo Cliente-Servidor, com uma arquitetura multi-camadas. O modelo adotado na aplicação é representado na Figura 2.

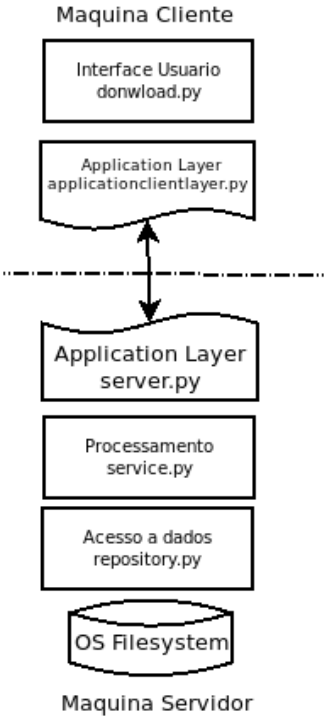

Figura 2

Respondendo às questões solicitadas para o relatório e conforme apresentado na Figura 2:

1. **Como os componentes de software serão agrupados ou particionados:** Serão particionado nas camadas Interface do usuário (download.py), Camada de aplicação cliente (applicationclientlayer.py), Camada de aplicacao server (server.py), Camada de processamento (service.py) e camada de acesso a dados (repository.py).
2. **Onde os componentes de software serão implantados:** No lado cliente ficarão: camadas Interface do usuário (download.py), Camada de aplicação cliente (applicationclientlayer.py). Já no lado servidor ficarão: Camada de aplicacao server (server.py), Camada de processamento (service.py) e camada de acesso a dados (repository.py)
3. **Como os componentes de software deverão interagir:** Uma explicação mais detalhada sobre como as camadas da arquitetura de software e de aplicação interagem é apresentada na Atividade 4, durante a explicação da camada de aplicação.

Atividade 4

Objetivo: Projetar o protocolo de camada de aplicação.

A camada de aplicação foi uma camada nova dentro do projeto. Ela foi pensada apenas para o lado cliente, visando criar uma abstração sobre a remotabilidade que existe na aplicação.

Devido a complexidade em criar uma abstração para o lado do servidor, o lado server continuou a enxergar e tratar os detalhes do socket, relativos à comunicação com o lado cliente.

1. **Protocolo de camada de transporte:** O protocolo da camada de transporte ficou definido apenas no lado cliente da aplicação. Ela teve por objetivo, do ponto de vista do cliente, ocultar os detalhes da remotabilidade da aplicação. O cliente informa e recebe apenas Strings. Para ele não é necessário interagir com bytes, nem conhecer sobre a API de socket do Python.
2. **Tipos de mensagens trocadas:** O formato das mensagens trocadas é do tipo requisição/resposta. O lado cliente, por intermédio da camada de aplicação, só precisa acessar um método do lado servidor para obter o conteúdo do arquivo que ele solicitou, e em seguida, salvar esse conteúdo, em um novo arquivo com o mesmo nome, no diretório de transferência (na pasta download pré-determinada). Caso o arquivo não exista, no mesmo método, o lado servidor, devolve no conteúdo, uma constante contendo a string "FILE_NOT_FOUND", que indica que o arquivo não existe no lado server. A camada de aplicação ao identificar esse conteúdo, levanta uma exceção para o lado cliente, que ira tratar o erro e apresentar a mensagem de forma amigável para o usuário.
3. **Sintaxe/formato de cada tipo de mensagem:** Na única mensagem enviada entre os lados da aplicação, o cliente envia uma string com o nome do arquivo que deseja obter, e recebe como retorno, também uma string com todo o conteúdo do arquivo solicitado. O tratamento de erro, quando o arquivo não existir, foi feito utilizando exceção. A exceção nasce no lado

servidor, e é empacotada e desempacotada pela camada de aplicação.

4. **Regras que determinam quando e como um processo envia/responde mensagens:** Como regras as mensagens trocadas foram apenas strings, a única regra que precisou ser definida, foi a estratégia para tratar o retorno do conteúdo do arquivo. Como o arquivo pode ter um tamanho grande, muito provavelmente demandaria executar o método sock.recv() em um loop, e definir qual a hora de encerrar essa agregação. Na aplicação foi implementada a estratégia vista em aula de fechamento do socket a cada operação. Dessa forma a agregação do recebimento do conteúdo do arquivo é interrompida quando a conexão é fechada pelo lado server. Com isso é criado uma thread com um socket para atender cada requisição, e o mesmo é fechado imediatamente após o envio dos dados solicitados.

O diagrama de sequência que está na última página (Figura 3) apresenta em maiores detalhes a interação entre os componentes da aplicação.

Atividade 5

Objetivo: Implementar a aplicação de acordo com o projeto elaborado nas atividades anteriores.

Disponibilize o codigo da sua aplicação em um ambiente de acesso remoto (GitHub ou GitLab):

https://github.com/sauloaalmeida/SistDistr/tree/main/Lab04/fileTransfer

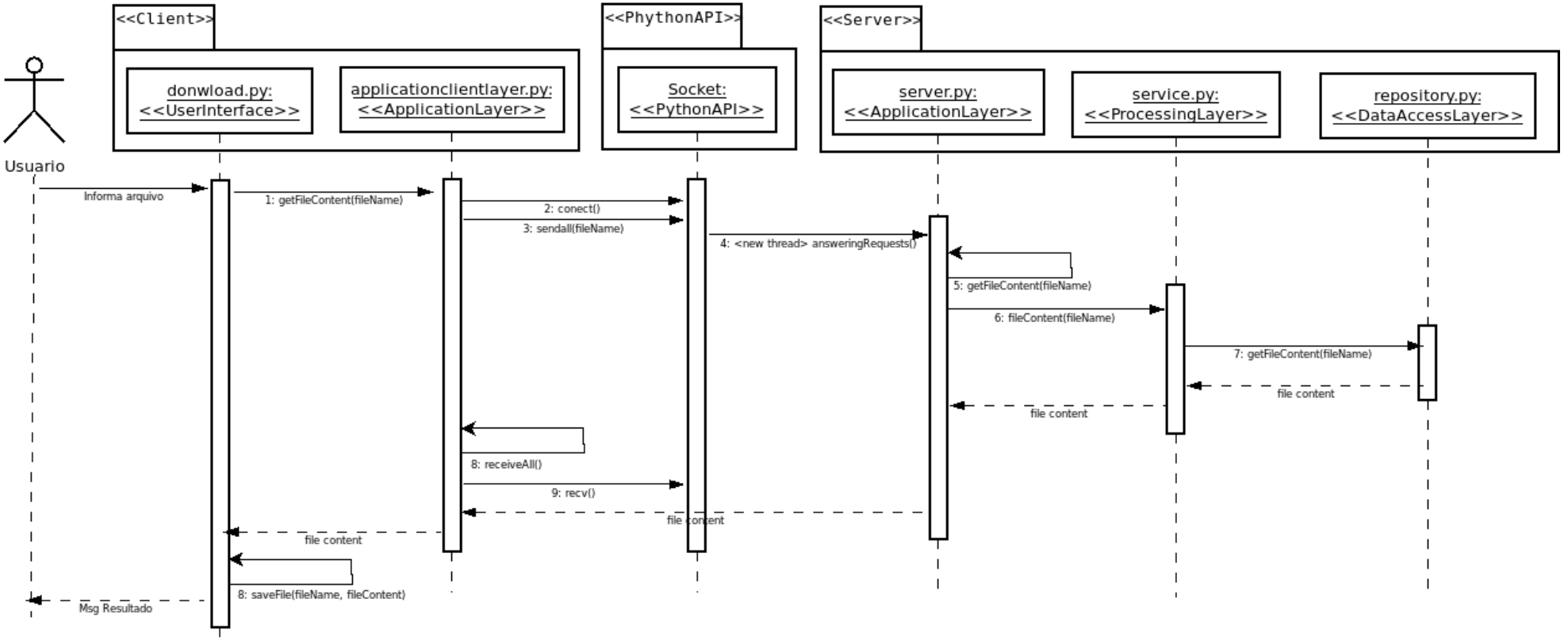
<<Client>>
<<PhythonAPI>>
<<Server>>
donwload.py:
<<UserInterface>>
applicationclientlayer.py:
<<ApplicationLayer>>
Socket:
<<PythonAPI>>
server.py:
<<ApplicationLayer>>
service.py:
<<ProcessingLayer>>
repository.py:
<<DataAccessLayer>>
Usuario
Informa arquivo
1: getFileContent(fileName)
2: conect()
3: sendall(fileName)
4: <new thread> answeringRequests()
5: getFileContent(fileName)
6: fileContent(fileName)
7: getFileContent(fileName)
file content
file content
8: receiveAll()
9: recv()
file content
file content
8: saveFile(fileName, fileContent)
Msg Resultado

Figura 3