Transmissão de Dados – Turma A – Trabalho 1
Professor: Marcos F. Caetano <mfcaetano@unb.br>
Monitoras: Camila Camargo <130104868@aluno.unb.br>,
Mariana Makiuchi <marimakiuchi@aluno.unb.br>
Resumo: Desenvolvimento de um sistema Dropbox-*like*

1 Introdução

Nos últimos 10 anos, serviços de armazenamento, compartilhamento e sincronização de arquivos em nuvem se popularizaram e hoje em dia são utilizados frequentemente nos meios pessoal, acadêmico e profissional. Provavelmente, o primeiro grande serviço desse gênero a ser lançado no mercado foi o *Dropbox* [1]. Este serviço de armazenamento remoto de arquivos foi disponibilizado em 2007 e seu funcionamento consiste na criação de uma pasta especial no computador do usuário, cujo conteúdo (arquivos de toda a espécie: fotos, vídeos, texto, etc.) é sincronizado com os servidores *Dropbox*, que de fato armazena os arquivos, e com outros computadores e dispositivos nos quais o mesmo usuário instalou este mesmo serviço, mantendo, assim, todos os arquivos atualizados em todos esses dispositivos.

O serviço *Dropbox* funciona por meio de aplicativos de computador para *Windows*, *macOS* e *Linux*, bem como aplicativos de celular e *tablet* para as plataformas *iOS*, *Android* e *Windows Phone*. A companhia também oferece uma interface web.

Outro serviço disponibilizado pelo *Dropbox* é o de carregamento (*upload*) automático de fotos e vídeos, ou seja, usuários podem carregar suas fotos ou vídeos de câmeras, cartões SD, *tablets* ou celulares na pasta "Camera Uploads"dedicada a este armazenamento [2]. Além desse serviço, o *Dropbox* ainda é projetado com várias camadas de proteção, serviços de criptografia, para que os dados do usuário não sejam alvo de ataques.

A figura 1 mostra um esquema básico das camadas de desenvolvimento envolvidas no projeto de um Dropbox completo.

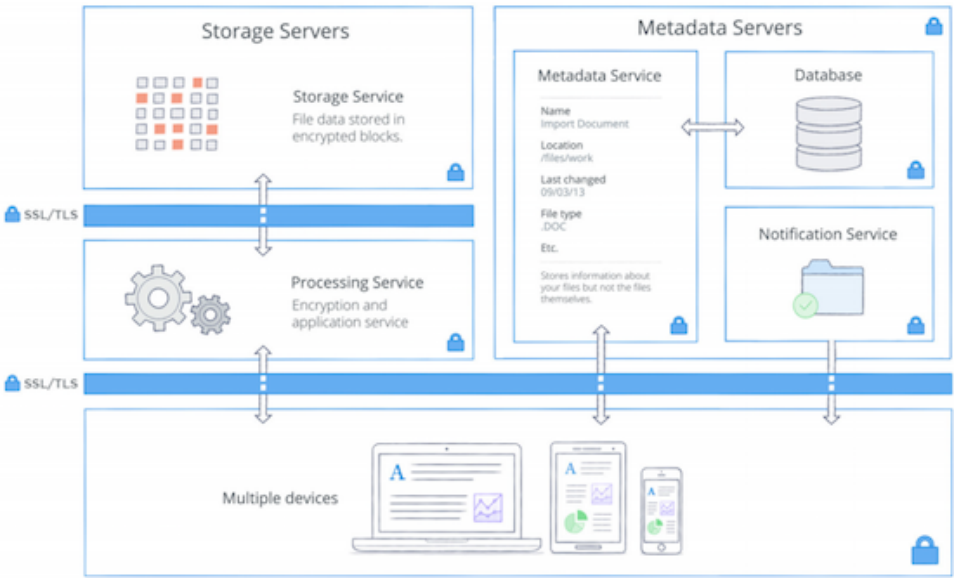

Figura 1: Esquema de camadas de desenvolvimento do Dropbox

Tendo em vista a complexidade da ferramenta completa do *Dropbox*, este trabalho terá como foco o desenvolvimento das funcionalidades de **armazenamento** e **sincronização** de arquivos em nuvem, a serem detalhadas na secção seguinte.

2 Descrição do projeto

O projeto a ser desenvolvido pode ser representado graficamente de forma generalizada pela figura 2, que envolve as funções mais fundamentais do Dropbox: o **armazenamento** e a **sincronização** de arquivos.

Figura 2: Armazenamento e sincronização de arquivos utilizando o Dropbox

A nível de desenvolvimento, o projeto pode ser descrito por um lado cliente e um lado servidor. O lado cliente é representado pelo fluxograma da figura 3.

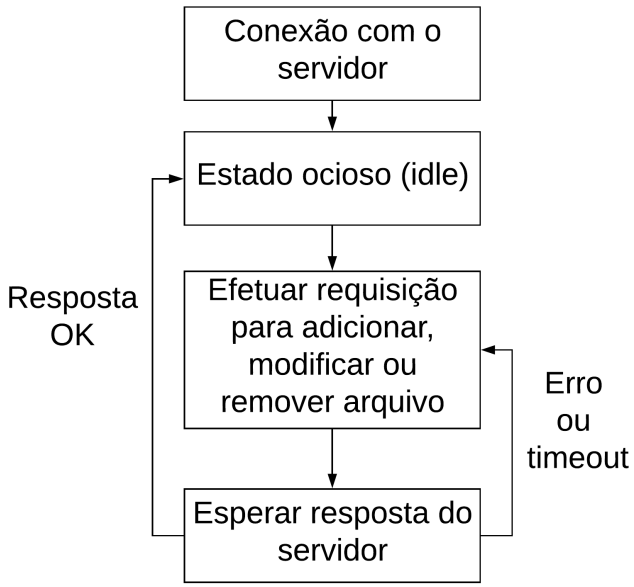

Figura 3: Fluxograma da parte cliente do projeto

Em resumo, temos que as funcionalidades da plataforma a serem exploradas neste trabalho podem ser descritas por um cliente que realiza as seguintes tarefas: ele é capaz de se conectar a um servidor remoto; ele envia e recebe arquivos do servidor; ele é capaz, a partir da utilização de um protocolo, de trocar mensagens com o servidor e requisitar que este realize uma tarefa; ele é capaz de interpretar as mensagens retornadas por um servidor (que também devem ser enviadas seguindo um protocolo), e apresentá-las de forma compreensível para o usuário executando o programa cliente em sua máquina; ele modifica o diretório local de acordo com as instruções enviadas pelo servidor.

Já o lado servidor da aplicação pode ser visualizado pelo fluxograma da figura 4.

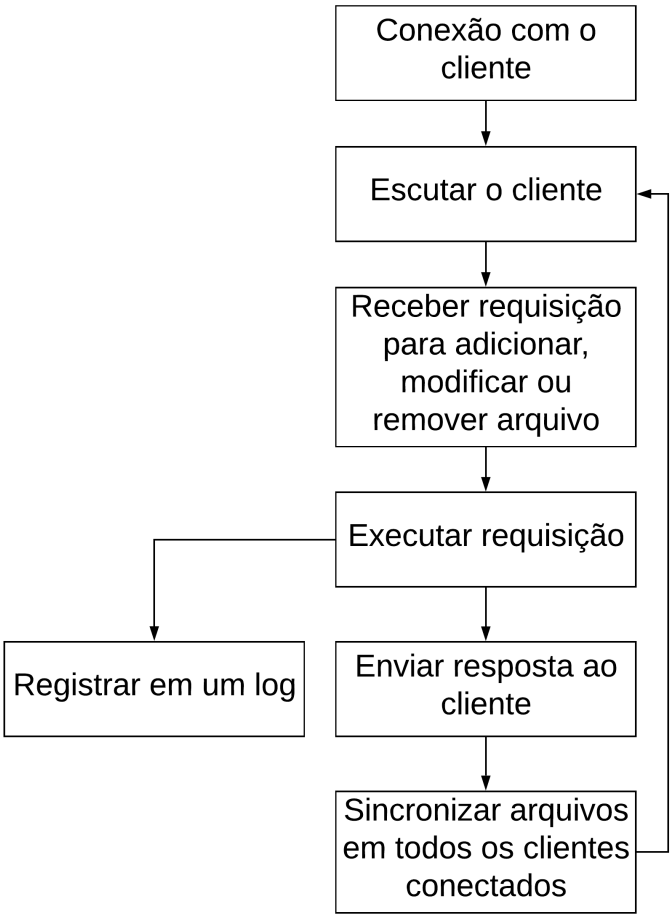

Figura 4: Fluxograma da parte servidor do projeto

Dessa forma, temos que as funcionalidades da plataforma a serem exploradas neste trabalho podem ser resumidas em um servidor que realiza as seguintes tarefas: ele apresenta uma tela de boas vindas para o usuário após requisitar deste usuário suas credenciais já autenticadas ou após requisitar a realização de uma autenticação; ele é capaz de cadastrar um novo usuário e, na máquina onde ele executa, criar um diretório onde os arquivos de tal usuário serão armazenados; ele é capaz de enviar e receber arquivos; ele permite que um usuário adicione, remova ou modifique a localização de outros arquivos e diretórios presentes dentro do diretório raiz; e quando modificações como as previamente mencionadas são feitas, ele é capaz de sincronizar com todas as máquinas que estão conectadas com este mesmo usuário, para que estas mantenham o conteúdo de seus diretórios locais coerentemente atualizados quando comparados com o diretório que se encontra no servidor; finalmente, temos que o servidor precisa ser **capaz de atender múltiplas requisições** (vindas de máquinas diferentes) ao mesmo tempo (a utilização de *threads*, portanto, torna-se indispensável para a realização deste trabalho).

A ideia de implementar um serviço deste tipo é que o cliente seja capaz de possuir um diretório que não exista de fato em sua máquina, apenas no servidor remoto. Portanto, o diretório presente no servidor não deve ser recriado na máquina cliente, caso contrário, perde-se o propósito da criação

remota dele. Além disso, cabe ao aluno decidir como que o diretório de um cliente irá ser reproduzido no próprio servidor. É possível que, para o cliente, quando este acessa o servidor, o diretório raiz apresentado como sendo seu seja o diretório `/home/cliente`, enquanto no servidor este diretório na verdade é o diretório `/usr/local/servidor/cliente`. Os arquivos dos clientes, portanto, podem estar organizados no servidor na forma que se desejar, porém, o servidor deve ser capaz de reproduzir o formato de organização destes arquivos de acordo com as operações realizadas pelo cliente, quando este realizar algum comando. Em outras palavras, um cliente irá criar, remover e alterar diretórios e, por mais que o servidor não de fato crie, remova e altere diretórios na máquina em que ele próprio executa, ele precisa ser capaz de, quando o cliente requisitar, apresentar a organização de diretórios por ele criados. Um exemplo: suponha que o cliente cria no seu diretório raiz, `/home/cliente`, o diretório `trabalhos`, e, neste diretório ele inclui o arquivo `trabalho_td2018.txt`, quando é feito o upload deste para o servidor. O servidor, neste caso, precisaria de fato armazenar o arquivo `trabalho_td2018.txt`, mas ele não necessariamente precisa, por exemplo, criar o diretório `/usr/local/servidor/cliente/trabalhos` para armazenar tal arquivo. Ele pode simplesmente armazenar o arquivo em `/usr/local/servidor/cliente`, e, neste mesmo diretório criar um arquivo que mantenha registro dos diretórios criados por este cliente e dos arquivos que lá estão contidos. Quando o cliente navegar por seus diretórios, porém, o servidor precisa indicar para ele que existe `/home/cliente/trabalhos` e que o arquivo que lá se encontra é `trabalho_td2018.txt`, e não apresentar para ele o caminho `/usr/local/servidor/cliente/trabalhos/trabalho_td2018.txt`. É possível também estabelecer um servidor que recrie exatamente aquilo que está sendo apresentado para o cliente. Em nosso exemplo, teríamos o diretório `/home/cliente` sendo o mesmo que um diretório `/usr/local/servidor/home/cliente` criado no servidor (todos os diretório raiz ficariam em `/usr/local/servidor/home`). Neste caso, todas as operações feitas pelo cliente poderiam ser reproduzidas em sua totalidade no servidor, não sendo necessário nenhum arquivo auxiliar para armazenar metadados que o servidor precisaria consultar antes de repassar para o cliente dados do seu diretório. Ambos os modos de implementação serão aceitos, contato que sejam implementados de modo coerente e organizado.

Outro ponto importante que deve ser enfatizado é o de *upload* de diretórios e de arquivos da máquina cliente para a máquina servidora. Nunca teremos que diretórios serão de fato transferidos para um servidor, serão transferidos apenas arquivos neles presentes. O programa cliente, quando o usuário desejar fazer o *upload* de apenas um arquivo, deve transferir apenas este arquivo para o servidor, que, por sua vez, deve armazená-lo no diretório corrente em que se encontra o usuário. Quando, porém, o usuário desejar fazer o *upload* de um diretório completo, temos que o servidor precisa reproduzir o diretório para ele transferido no diretório corrente da máquina servidora, que é o diretório onde o usuário executa o comando requisitando a transferência. O usuário especifica o caminho até o diretório, em sua máquina, que ele deseja carregar no servidor, e o lado cliente do programa se encarrega de encontrar o diretório, definir a sua configuração (quais outros diretórios se encontram dentro deste, como os arquivos estão distribuídos nele, etc.) e enviar para o servidor não apenas os arquivos que se encontram no diretório, como a organização deste diretório e de seus conteúdos, justamente para que o servidor seja capaz de reproduzí-lo de forma exata no diretório de seu cliente na máquina de sua própria execução. Um exemplo: suponha que o cliente opta por transferir para o seu diretório `/home/cliente/trabalhos`, no servidor, o diretório local de sua máquina `/C:/Usuários/cliente/Documentos`, onde, dentro deste diretório encontram-se outros diretórios, `UnB` e `Estágio`, e o arquivo `README.txt`. O usuário irá executar um comando no programa cliente que irá indicar para este, o diretório que ele deseja transferir para o servidor, neste caso, `/C:/Usuários/cliente/Documentos`. O programa cliente, por sua vez, encontra este diretório e o mapeia, recuperando a forma com que arquivos e outros diretórios são organizados dentro de `/Documentos`. Feito isso, ele transfere para o servidor, na ordem que julgar adequada, todos os arquivos - folhas na árvore que tem `/Documentos` como raiz - pertencentes à `/Documentos`, bem como um arquivo contendo metadados, que explicitam para o servidor como estes arquivos devem ser nomeados e dispostos no diretório do cliente. Como produto final, teríamos no servidor o diretório `/home/cliente/trabalhos/Documentos`, que, por sua vez, iria conter os diretórios `UnB` e `Estágio`,

bem como os arquivos `README.txt` e `trabalho_td2018.txt` (com base no exemplo anterior).

Note a importância de se desenvolver um protocolo próprio para este programa: todas as operações a serem realizadas pelo cliente irão depender da informação para ele enviada pelo servidor e vice-versa. Um protocolo bem estruturado irá permitir que a informação trocada seja interpretada de forma eficiente, permitindo assim que o cliente e servidor funcionem de forma mais rápida e correta.

Vale ressaltar que, a operação de cadastro, a ser realizada por um usuário para que o servidor saiba qual cliente está se conectando a ele, pode ser implementada com o auxílio de um Sistema Gerenciador de Banco de Dados ou não. O aluno pode optar por armazenar dados persistentes em um arquivo texto, ou armazenar para cada sessão dados de cadastro (informações de um usuário persistindo apenas durante uma sessão do servidor). Independente da escolha feita pelo aluno, é necessário que haja algum tipo de **registro dos usuários** acessando o servidor enquanto este estiver ativo.

Além disso, deixa-se claro que **não é necessário** que o trabalho possua interface gráfica. Se for de desejo do aluno, a implementação de tal interface pode ser feita, porém, **nenhuma pontuação extra será associada a tal interface** dado que o objetivo do trabalho não é a implementação desta. É necessário, porém, que haja uma interface mínima de texto no lado do cliente, para que um usuário possa comunicar ao servidor o que ele deseja fazer.

A seguir, as figuras de 5 a 11 mostram um exemplo de interface texto que poderia ser disponibilizada ao usuário por um programa cliente, após recebidas e interpretadas as mensagens de resposta enviadas pelo servidor (lembrando que as requisições de um cliente dependem da resposta enviada pelo usuário executando o programa). A ideia por elas sendo apresentada é de que a plataforma criada deve se assemelhar a um *shell* executando em um sistema operacional *Linux*, onde um usuário é capaz de realizar operações com diretórios e arquivos após inserir simples comandos de operação, seguidos do alvo (ou alvos) de tal operação. Para este trabalho, propõe-se que o aluno desenvolva algo de natureza semelhante.

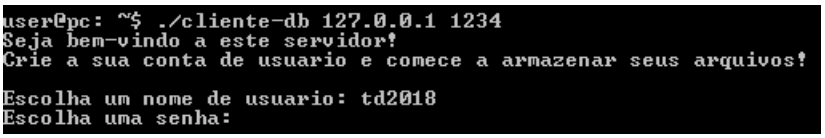

Figura 5: Tela de cadastro (cliente executa o programa sem argumentos)

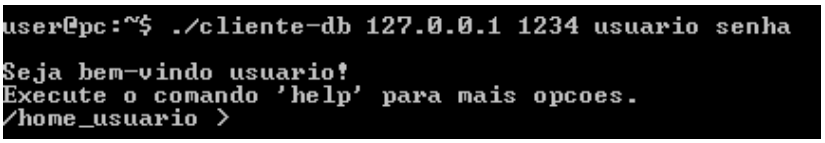

Figura 6: Tela inicial do programa cliente (cliente executa o programa usando dados de autenticação como argumentos)

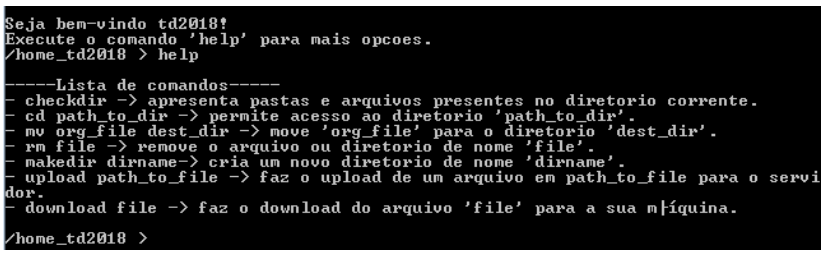

Figura 7: Utilização de um comando de ajuda que apresenta para o usuário uma lista dos comandos aceitos pelo programa

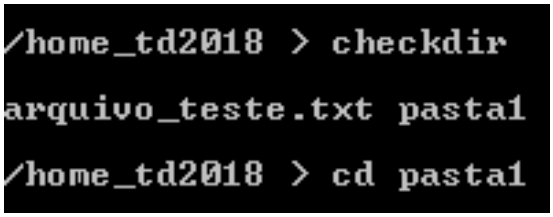

Figura 8: Usuário verificando os arquivos que se encontram em um diretório

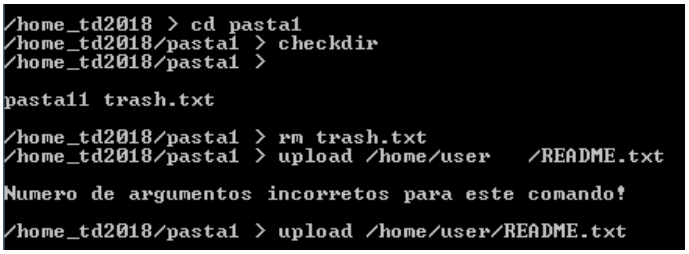

Figura 9: Usuário executando uma sequência de comandos e recebendo uma mensagem de erro após utilizar um dos comandos incorretamente

Figura 10: Usuário realizando operações sobre um diretório

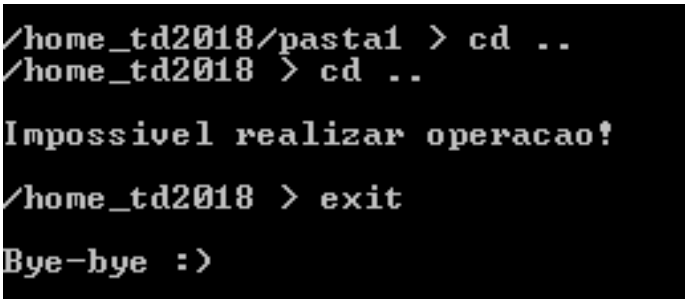

Figura 11: Usuário se desconectando do servidor após receber mensagem de erro por tentar acessar um diretório "superior"ao seu diretório raiz (não deve existir)

3 Avaliação

A avaliação consiste em 2 etapas:

- Relatório de desenvolvimento do projeto.
  - Apresentação detalhada sobre o projeto, que deve ser desenvolvido a partir das especificações deste roteiro;
  - Apresentação de problemas encontrados, bem como explicações sobre o porquê dos problemas e sobre suas possíveis soluções;
  - O comportamento do protocolo da camada de aplicação desenvolvido deve estar explicado na forma de uma máquina de estados finitos;

  - Deverão ser adicionados *Screenshots* e explicações a respeito do funcionamento das funções (ou métodos) implementadas;
  - Podem ser adicionados *links* de vídeos hospedados na internet demonstrando o funcionamento do projeto;
  - Não serão aceitos trabalhos enviados fora do prazo;

- Apresentação em sala de aula.
  - Trabalhos não apresentados não serão considerados;
  - As apresentações devem ter duração de no máximo **10 minutos**;
  - As datas de realização das apresentações são as que seguem: **26/6**, **28/6**, **3/7** e **5/7**.

3.1 Pontuação extra

O aluno que desenvolver a função do Dropbox de compartilhamento de arquivos com outros usuários ("*Shared folder*") acrescentará um ponto à nota do trabalho. Dessa forma, o aluno poderá contabilizar até 11 pontos em sua nota final do trabalho.

Para tanto, o cadastro de usuários deve persistir por entre sessões do servidor (utilizando um banco de dados ou um arquivo para armazenar usuários cadastrados).

4 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

- O trabalho deve ser **OBRIGATORIAMENTE** executável na plataforma GNU/Linux;
- A linguagem de programação na qual o projeto será desenvolvido fica a critério do aluno;
- É responsabilidade do aluno verificar se o código desenvolvido compila e executa corretamente na plataforma GNU/Linux;
- código que não compila é ZERO!
- É responsabilidade do aluno se certificar que o arquivo, no formato ZIP, contendo todo projeto (código e relatório) foi corretamente armazenado no Moodle. Ou seja, após a submissão, faça o download do seu projeto e descompacte o arquivo para ter certeza que ele está correto;
- Códigos copiados serão considerados "cola"e todos os alunos envolvidos ganharão nota zero;
- É necessário incluir referências no trabalho escrito. Cite a origem de seus dados!
- O relatório deverá ser entregue **OBRIGATORIAMENTE** no formato de arquivo PDF;
- Dúvidas sobre o trabalho deverão ser tiradas **utilizando a plataforma virtual moodle**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.
- Será considerada como parte da avaliação a organização do código. Comentários coerentes e não óbvios devem ser incluídos.
- O prazo definido para submissão do trabalho é FIRME e já contempla possíveis indisponibilidades do Moodle. Não deixem para submeter o seu projeto na última hora. O sistema está configurado para vocês poderem fazer múltiplos uploads.
  - E-mails contendo o projeto enviados para o professor ou os monitores serão automaticamente descartados.

Referências

[1] Dropbox. "**Dropbox**". [Online] Disponível em: "`https://www.dropbox.com/?landing=dbv2`". Acessado em: 31 de Março de 2018.

[2] Wikipedia. "**Dropbox (service)**". [Online] Disponível em: "`https://en.wikipedia.org/wiki/Dropbox_(service)`". Acessado em: 31 de Março de 2018.