MC833 - Relatório do Projeto 1

Nome: Felipe Escórcio de Sousa - **RA:** 171043
Nome: Ricardo Ribeiro Cordeiro - **RA:** 186633

Abril de 2021

1 Introdução

O intuito deste trabalho é prover uma primeira experiência com a programação de um servidor TCP concorrente. Para isso foi proposto este projeto que implementa um servidor TCP básico que realiza o cadastro, listagem, edição e exclusão de dados de alunos formados em uma universidade fictícia. Também é proposto a programação de uma plataforma também simples a ser usada pelo usuário, que realiza as requisições ao servidor, localmente ou remotamente.

2 Descrição geral e casos de uso

Primeiramente, para rodar são necessários os seguintes requisitos:

- LibMongoC
- Docker-Compose ou MongoDB rodando localmente

E para um guia simples para instalação das mesmas, consulte o apêndice A deste relatório.

Já para compilar basta utilizar o comando Make na pasta do projeto ou nas pastas específicas do cliente ou servidor para compilação de apenas alguma parte.

O servidor é configurado para aceitar conexões TCP e efetivamente se comunicar com até 5 usuários ao mesmo tempo, podendo facilmente ser configurado para que sejam aceitos até mais conexões, dependendo do sistema operacional utilizado, mas que pode ser suficiente para efeitos de exemplificação. A partir do estabelecimento da conexão pelo socket, o mesmo espera pelas operações a serem enviadas pelo client. Por padrão a porta configurada é a `3490`, mas para que se configure a porta a ser utilizada basta passar

a porta como parametro na hora da execução do programa e a url do banco de dados, por exemplo:

```
$ ./server 4242 mongodb://localhost:27017
```

O cliente, por sua vez, pode abrir uma conexão cada com o servidor, podendo, assim que a mesma for estabelecida, enviar comandos e dados para o lado do servidor. Há também a possibilidade deste ver um menu com ajuda e também requisitar por terminar a conexão com o servidor. Para se configurar um IP de uma máquina remota (podendo ser IPv4 ou IPv6) e a porta, basta passar como parâmetro o endereço seguido da porta na hora da execução, por exemplo:

```
$ ./client 123.0.1.23 1234
```

As operações realizadas pelo servidor são:

1. Registrar perfil
2. Adicionar novas experiências à um perfil existente
3. Listagem por curso
4. Listagem por habilidade
5. Listagem por ano de graduação
6. Listagem de todos os perfis
7. Encontrar por email
8. Deletar perfil

Sendo as operações de criação, edição e remoção de registros restrita ao administrador do sistema. O username que precisa ser enviado para realizar estas operaçãoes é apenas `admin`, por simplicidade.

3 Armazenamento e estrutura de dados

Primeiramente, a razão da escolha de MongoDB para gerenciamento de dados sobre outras implementaçãoes se dá ao fato de simplificar razoavelmente o tratamento dos dados, dadas as naturezas dos dados e operações requeridas pelo enunciado do trabalho. Porém, apesar de deixar o tratamento dos dados mais simples, a instalação do driver necessário requereu algumas adaptações no comando de compilação, necessitando incluir as libs manualmente e linkar os executáveis.

A principal simplicação que essa escolha trouxe foi no envio de mensagens, essas podendo ser convertidas automaticamente em strings e enviadas,

além de facilitar a edição e deleção. As informações serão salvas formatadas da forma de um BSON, dessa maneira:

```
{
    email: <string>,
    name: <string>,
    surname: <string>,
    residence: <string>,
    graduation: <string>,
    graduationYear: <string>
    skills: <string>,
    experiences: <array<string>>
}
```

Sendo essas limitadas apenas pelos tamanhos máximos de vetor estabelecidos no programa cliente.

4 Divisão do código

A codificação é baseada principalemente em 4 arquivos `".c"` em específco, são esses:

4.1 server.c

Esse arquivo é responsavel por implementar as interações do servidor, é neste arquivos que recebemos as requisições do client, e coordenamos o fluxo do programa, seja resolvendo no próprio server.c ou enviando requisições para o banco de dados.

4.2 db.c

Esse arquivo é responsável por implementar as funções de comunicação com o banco de dados (MongoDB). Utilizando a bíblioteca libmongoc podemos implementar as operações básicas de um CRUD como: Registrar um Perfil, Listar por determinado parâmetro, adicionar uma nova experiência à um perfil existente e deletar um perfil.

4.3 client.c

Esse arquivo é responsavel por ser a interface com o usuário, além de se conectar com o servidor, também envia as requisições para o server,

de modo que ao receber o dado solicitado, devolve o feedback com o dado requisitado pelo o usuário.

4.4 shared.c

Esse arquivo é responsável por manter as funções que são compartilhaveis entre os demais arquivos, evitando replicação de código.

5 Implementação

A implementação dos sockets e da conexão TCP se deu por funções já implementadas por bibliotecas nativas do C. A configuração, tanto para o cliente quanto para o servidor, acontece de maneira muito semelhante, primeiro, é criado um socket e então este é ligado à uma porta. No caso do servidor, ele passa a escutar este socket, esperando por alguma conexão, e se alguma chegar, ele aceita então a conexão. Para o cliente, este já tenta iniciar logo uma conexão com o servidor e se conseguir estabelecer, já pode começar a operar as funções estabelecidas.

Operação	**Descrição**	**Formato dos dados**
0	Inserir novo perfil	[OpCode][Id][Dados]
1	Inserir novas experiências em um perfil existente	[OpCode][Id][Email] [Dados]
2	Listagem por curso	[OpCode][Curso]
3	Listagem por habilidade	[OpCode][Habilidade]
4	Listagem por ano de graduação	[OpCode][Graduação]
5	Procurar por email	[OpCode][Email]
6	Listar todos os perfis	[OpCode]
7	Deletar um perfil	[OpCode][Id][Email]
8	Desconectar	[OpCode]

Após a execução da operação é enviado um feedback textual ao cliente, informando-o se a operação foi concluída com sucesso ou não ou há o retorno dos dados consultados por este, para manter um padrão na comunicação e melhorar a experiência do usuário.

6 Conclusão

Percebe-se então que a maior função dentre o servidor é a troca de mensagens, sendo um requerente, e o outro o servente, que opera as ações

nos dados. Podendo, os dois, estarem em máquinas diferentes ou não.

Para uma implementação mais robusta da troca de mensagens, poderia-se pensar em formar de envio de pacotes maiores ou menores, mensagens maiores e também números maiores de conexões simultanêas.

Podemos concluir então com o trabalho, que conexões TCP são bastante confiáveis, todas mensagens chegam intactas, havendo uma certa latencia maior para mensagens maiores, mas sem qualquer problema.

A Referências

1. http://beej.us/guide/bgnet/translations/bgnet_ptbr.pdf
2. http://mongoc.org/libmongoc/current/tutorial.html

B Instalação de dependências

Link para instalação do `docker-compose`:

`https://docs.docker.com/compose/install/`

Link para instalação da lib do banco de dados:

`http://mongoc.org/libmongoc/current/installing.html`

É possível que seja necessário também realizar a instalação dos pacotes `*-dev`(Ubuntu) ou `*-devel`(Fedora).

C Possíveis problemas com a instalação do driver do banco de dados

É possível que as os headers da lib do banco de dados tenha sido instalado em algum diretório diferente dos usados na construção do programa, para isso verifique em que diretório dentro de `/usr` estão contidos os headers das libs e altere nos Makefiles necessários.