UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e Computação

EA872 - Laboratório de Programação de Software Básico

Projeto Final - Servidor Web em C

Aluno
João Felipe Contreras de Moraes – 174140
jfcm.pipe@gmail.com

(06 de dezembro de 2023)

Conteúdo

1 Descrição do Sistema

O servidor web desenvolvido, em linguagem C,[1] tem por objetivo demonstrar e motivar a aplicação de conceitos de software básico. Sendo assim, o sistema em sua forma final foi responsável por exercitar conceitos nas áreas de

- **compiladores**, com a construção de um parser para interpretação de requisições
- **sistemas operacionais**, com processos, threads, syscalls
- **redes**, com sockets, protocolo HTTP, protocolo TCP e documentos RFC

Na sua atual forma, o sistema é capaz de atender requisições do tipos GET, HEAD, TRACE, OPTIONS e POST (particularmente, nos formulários de troca de senha). Isso indica que o servidor consegue até mesmo interagir com o browser, onde serão realizados os teste.

As capacidades efetivas do servidor consistem em: atender múltiplas requisições com threads, manusear um webspace com arquivos protegidos por `.htaccess`, arquivos sem permissão apropriada (de leitura e varredura), responder requisições com diversos tipos de recursos, como HTML, imagens (jpg, png e gif), pdf, texto. Além disso, o servidor implementa uma forma rudimentar de autenticação fazendo uso de arquivos `.htpasswd` com senhas criptografadas.

Trata-se, portanto, de um servidor simplificado que não apresenta uma interação com banco de dados, não implementa técnicas robustas de segurança e não busca atingir o maior grau de otimização.

Para realizar os testes, será preciso compilar o programa e construir o parser, com ajuda do script `"scripts/compile_server.sh"`, e ajustar as permissões dos webspaces, a partir dos scripts `"configurar_permissoes.sh"`.

```
# Compila o servidor e gera o executável "server" na raiz do projeto
./scripts/compile_server.sh

# Configura a permissão do webspace desejado
cd web/meu-webspace/
./configura_permissoes.sh
cd ../../

# Executa o programa com
# ./server <portNum> <webspace> <logfile> <threads>
./server 8080 meu-webspace log.txt 4

# Testes podem ser realizados acessando http://localhost:8080/
```

A maioria dos testes levou em conta as seguintes versões dos softwares gcc (9.4.0), bison (3.5.1) e flex (2.6.4).

[1] https://github.com/JF235/C-WebServer

2 Fluxograma do Sistema

2.1 Fluxo geral

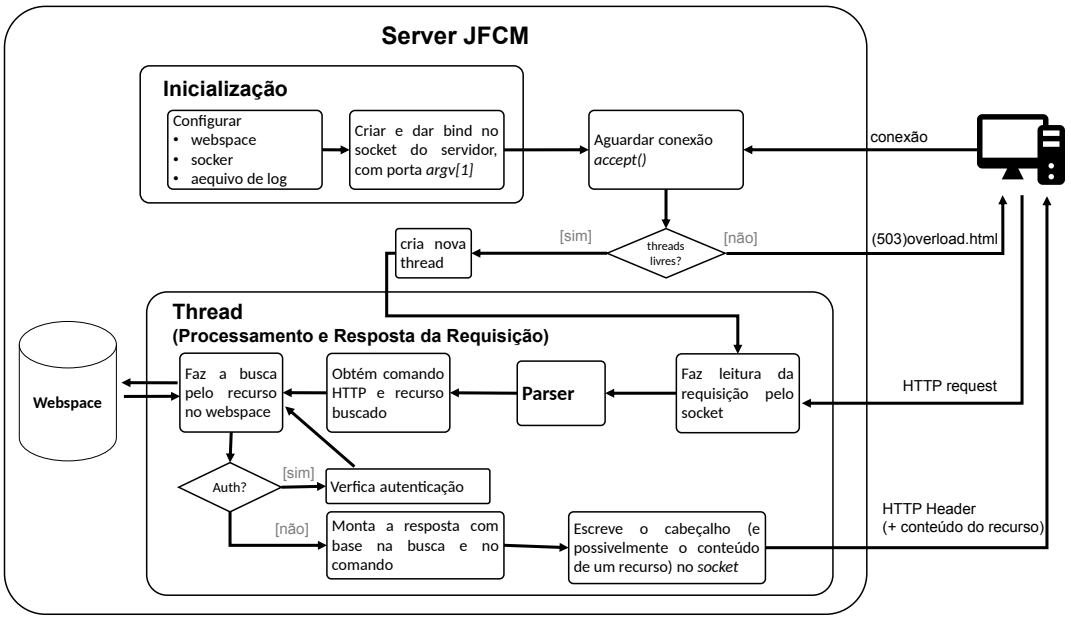


Figure 1: Fluxo geral do programa, desde a inicialização até o envio da resposta pelo socket.

2.2 Fluxo da busca pelo Webspace

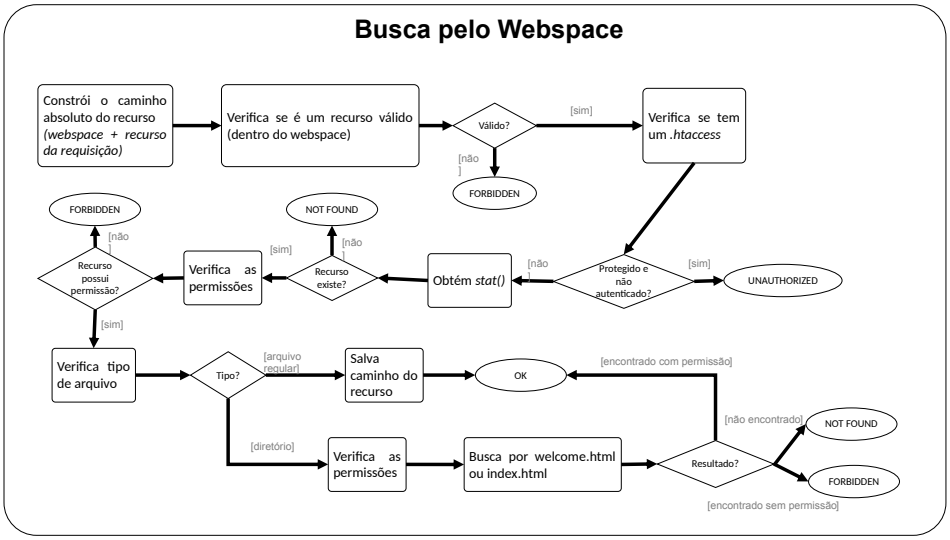


Figure 2: Uma das etapas mais complexas do fluxo geral foi detalhada nesse fluxograma.

2.3 Fluxo da montagem de resposta

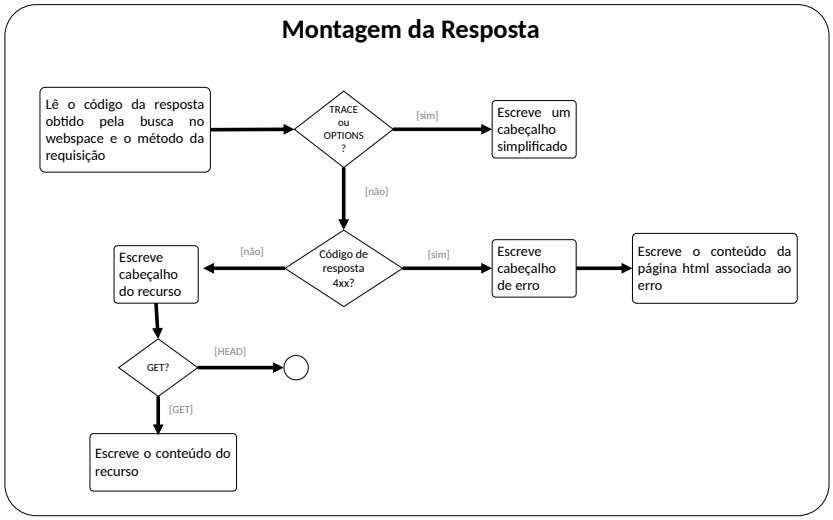


Figure 3: Fluxo da montagem da resposta, desde a interpretação da requisição e do código de erro, até a eventual escrita do conteúdo do recurso.

3 Conteúdo do Webspace

3.1 Visão Geral

A árvore geral de diretórios é exibida abaixo, na Figura 4, em forma vetorizada com a finalidade de preservar a qualidade diante de zoom.

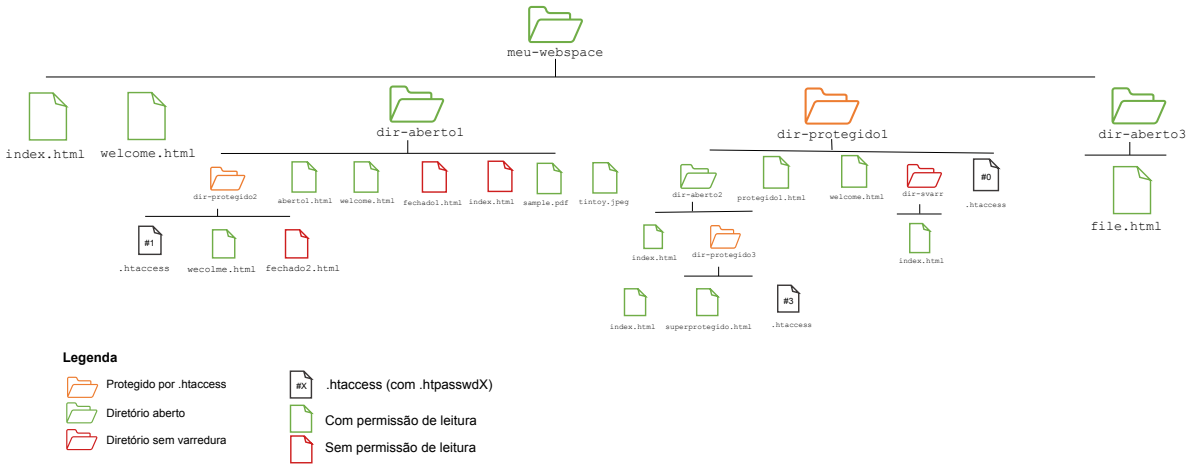


Figure 4: Árvore de diretórios do webspace.

3.2 Detalhamento

Com a finalidade de detalhar o conteúdo do webspace, na seção atual o conteúdo de cada arquivo será mostrado.

- **HTML**, os arquivos html contém um texto que indica qual é o nome do arquivo sendo acessado, além do diretório em que está presente. Arquivos *index* e *welcome* também possuem um link para cada um dos outros arquivos na mesma altura do diretório.

  Além disso, arquivos *index* protegidos possuem um link para um formulário de troca de senha.

  Exemplo:

```html
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
        <title>welcome.html</title>
    </head>
    <body>
        <h1>Conteudo do welcome.html</h1>
        <h2>Dentro de dir-aberto1</h2>
        <br>
        <a href="aberto1.html">aberto1</a>
        <br>
        <a href="fechado1.html">fechado1</a>
        <br>
        <a href="dir-protegido2/">dir-protegido2</a>
        <br>
        <a href="sample.pdf">sample.pdf</a>
        <br>
        <a href="tintoy.jpeg">tintoy.jpeg</a>
    </body>
</html>
```

- **JPEG** é uma imagem da animação Tin Toy da Pixar (1988).
- **PDF** é um arquivo pdf de exemplo, com texto simples.
- **.htaccess e .htpasswd** Os arquivos *.htaccess* apresentam somente uma linha, indicando o caminho do arquivo *.htpasswd*, que contém as senhas.

  Existem 3 arquivos de senha ativos nesse webspace:

```
# .htpasswd0 (dir-protegido1)
# Senhas: user0:0user0, user1:0user1, user2:0user2, admin:admin, 123:123
user0:EAawXWXTygJ6s
user1:EAL5t1QMVsnVU
user2:EAKJYUMMQjOMg
admin:EAj1T7YvoWRIw
123:EAuO9sgNsgg1g
```

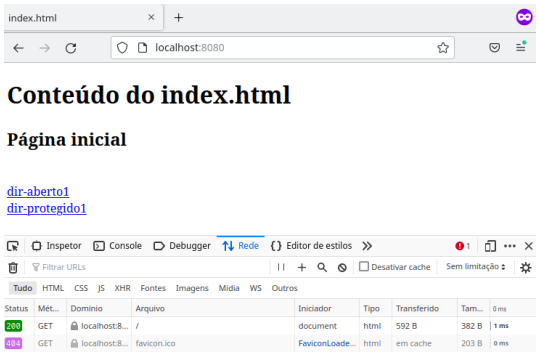
```
# .htpasswd1 (dir-protegido2)
# Senhas: user0:1user0, user1:1user1, user2:1user2, admin:admin
user0:EAdxmqOHal48Y
user1:EAtZyy5cuPt02
user2:EAJRDRbG3Y5W.
admin:EAj1T7YvoWRIw
```

```
# .htpasswd3 (dir-protegido3)
# Senhas: admin:admin, 123:123
admin:EAj1T7YvoWRIw
123:EAuO9sgNsgg1g
```

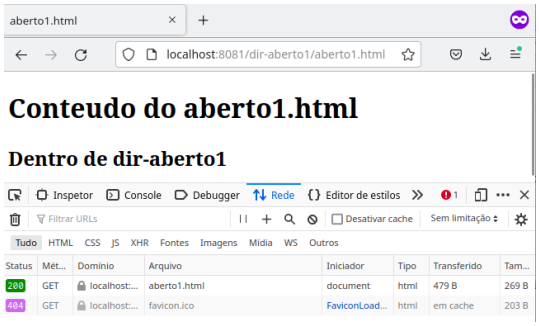
4 Casos de Teste

4.1 Obtendo recursos (sem erros)

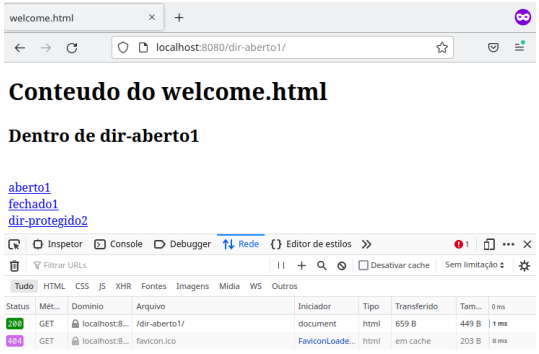
- Obtendo `index.html` presente diretamente no webspace. Observe que `index.html` tem maior precedência sobre o arquivo `welcome.html`.



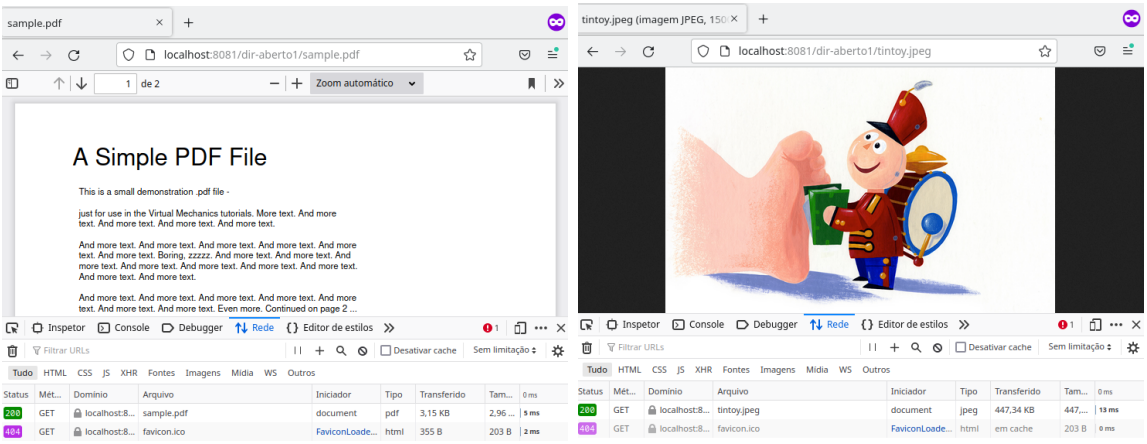
- Obtendo um arquivo com permissão de leitura, dentro de um diretório não protegido por `.htaccess`.



- Obtendo um arquivo `welcome.html`, quando o arquivo `index.html` está inacessível.



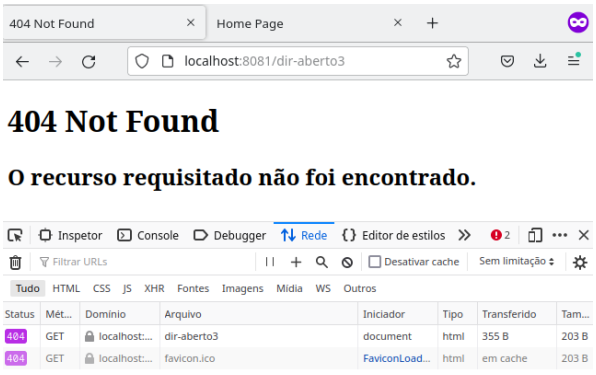
- Obtendo recursos não textuais (`.pdf` e `.jpeg`)



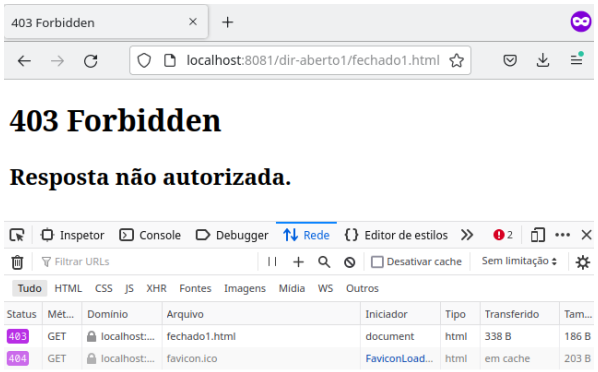
4.2 Erros

- `404 Not Found`, Recurso inexistente.

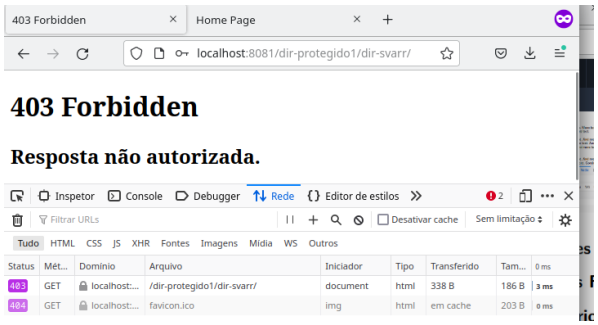
- `404 Not Found`, Diretório sem `index.html` ou `welcome.html`.



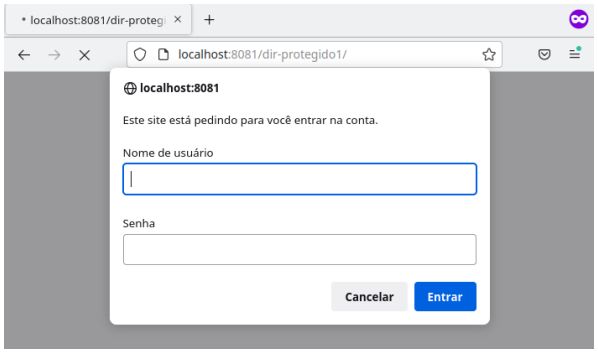
- `403 Forbidden`, Arquivo sem permissão de leitura.



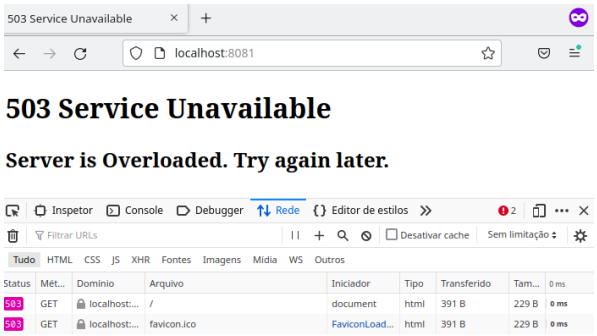
- 403 `Forbidden`, Diretório sem permissão de varredura.



- 401 `Unauthorized`, tentando acessar diretório sobre a proteção de um `.htaccess` ou quando inserimos credenciais erradas.



- 503 `Service Unavailable`, todas as threads estão ocupadas.



4.3 OPTIONS, HEAD e TRACE

Nessa seção, serão exemplificadas as respostas obtidas por meio da interação direta com o socket usando o simulador de cliente desenvolvido.

- `OPTIONS`

```
HTTP/1.1 200 OK
Allow: OPTIONS, GET, HEAD, TRACE, POST
Date: Tue Dec 05 16:37:36 2023 BRT
Server: JFCM Server 0.1
Connection: keep-alive
Content-Length: 0
Content-Type: text/plain
```

- `HEAD`

  Para o recurso `"/"`

```
HTTP/1.1 200 OK
Date: Tue Dec 05 16:40:19 2023 BRT
Server: JFCM Server 0.1
Connection: keep-alive
Last-Modified: Tue Dec 05 11:33:56 2023 BRT
Content-Length: 382
Content-Type: text/html; charset=utf-8
```

  Para o recurso `"/dir-aberto1/tintoy.jpeg"`

```
HTTP/1.1 200 OK
Date: Tue Dec 05 16:42:08 2023 BRT
Server: JFCM Server 0.1
Connection: keep-alive
Last-Modified: Tue Dec 05 15:27:42 2023 BRT
Content-Length: 457878
Content-Type: image/jpeg
```

  Para o recurso `"inexistente"`

```
HTTP/1.1 404 Not Found
Date: Tue Dec 05 16:45:23 2023 BRT
Server: JFCM Server 0.1
Content-Type: text/html
Content-Length: 203
Connection: close
```

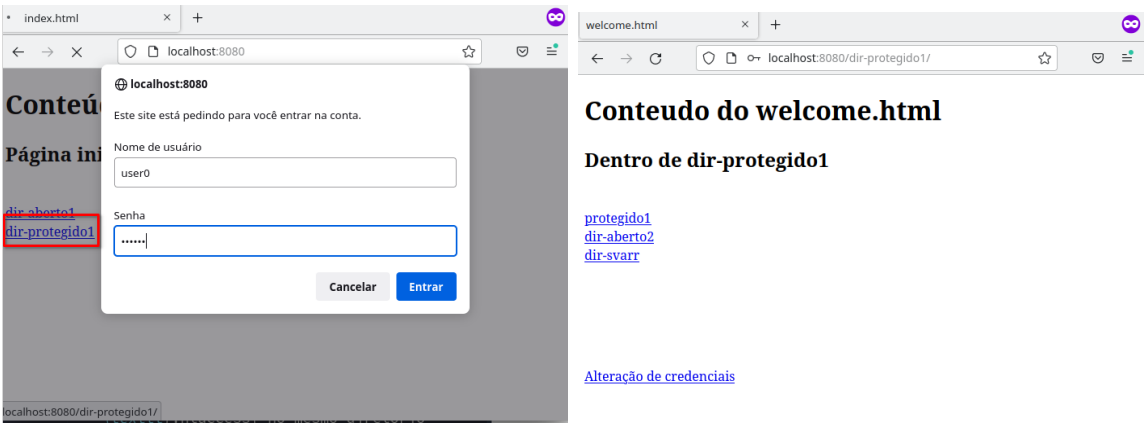
- TRACE

```
HTTP/1.1 200 OK
Date: Tue Dec 05 16:55:18 2023 BRT
Server: JFCM Server 0.1
Connection: keep-alive
Content-Type: text/plain
```

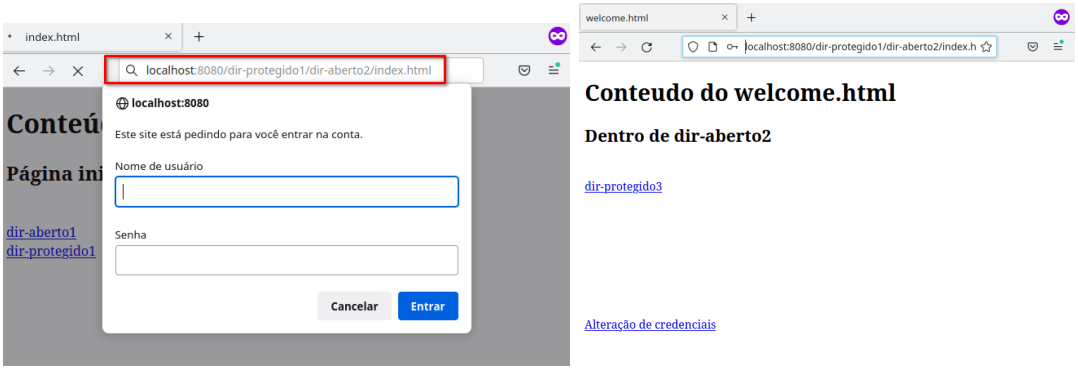
4.4 Autenticação

Nessa seção, vamos testar o recurso de autenticação e proteção do webspace.

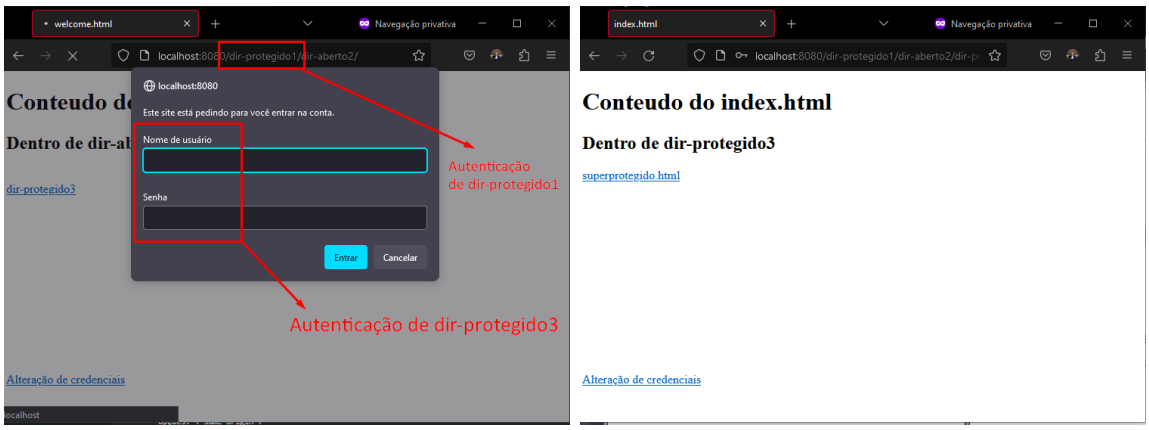
- Acesso a um recurso protegido pelo `.htaccess` no mesmo diretório



- Acesso a um recurso protegido pelo `.htaccess` em um diretório superior



- Acessando um arquivo que tem dois `.htaccess` na árvore. Dessa maneira, demonstra-se que o arquivo mais próximo é aquele de predomina.



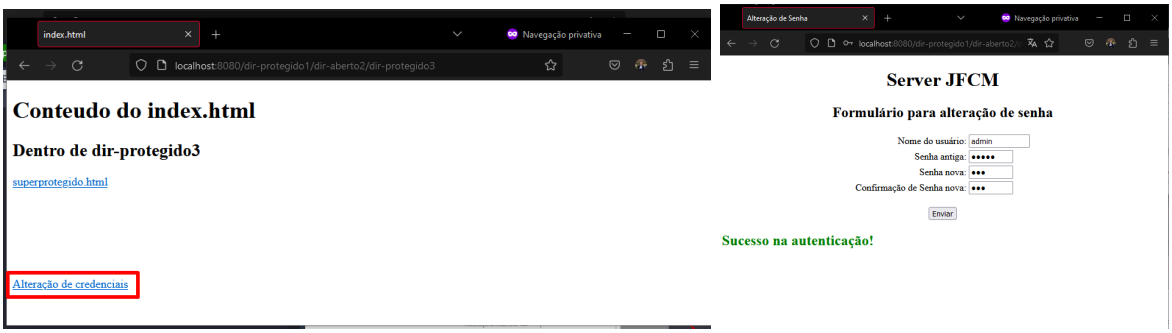
- Alterando a senha em dois diretórios protegidos por arquivos de senhas distintos.

  Os arquivos de senha, antes das alterações, são

```
# .htpasswd0 associado à dir-protegido1
# Senhas: user0:0user0, user1:0user1, user2:0user2, admin:admin, 123:123
user0:EAawXWXTygJ6s
user1:EAL5t1QMVsnVU
user2:EAKJYUMMQjOMg
admin:EAj1T7YvoWRIw
123:EAuO9sgNsgg1g
```

```
# .htpasswd3 associado à dir-protegido3
# Senhas: admin:admin, 123:123
admin:EAj1T7YvoWRIw
123:EAuO9sgNsgg1g
```

  Acessando o `dir-protegido3` com autenticação `admin:admin` e acessando o formulário para alteração de senha, vamos alterar a credencial para `admin:new`.

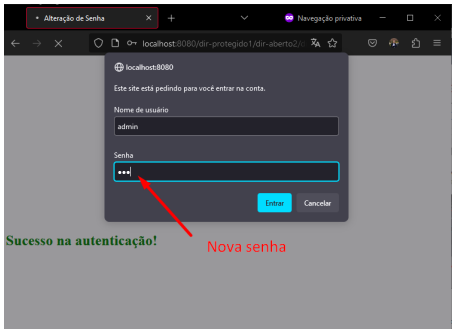


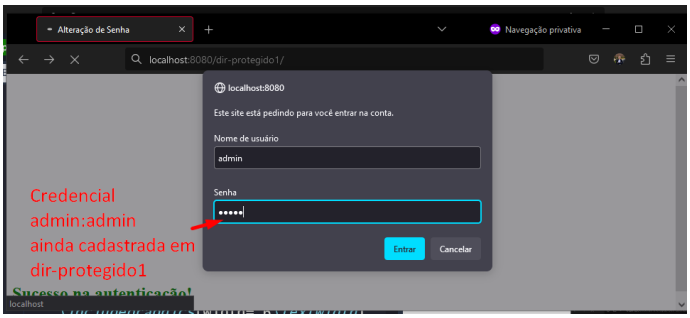
  O novo conteúdo de `.htpasswd3` é dado por

```
# .htpasswd3 associado à dir-protegido3
# Senhas: admin:new, 123:123
admin:EAoI3rLqgpsj2
123:EAuO9sgNsgg1g
```

Quando voltamos para o caminho `.../dir-protegido3`, recebemos a aba de cadastramento de um `401 Unauthorized`, pois será preciso atualizar a credencial.

Além disso, para acessar `dir-protegido1` é preciso usar a credencial antiga, pois nesse espaço a senha não foi alterada.





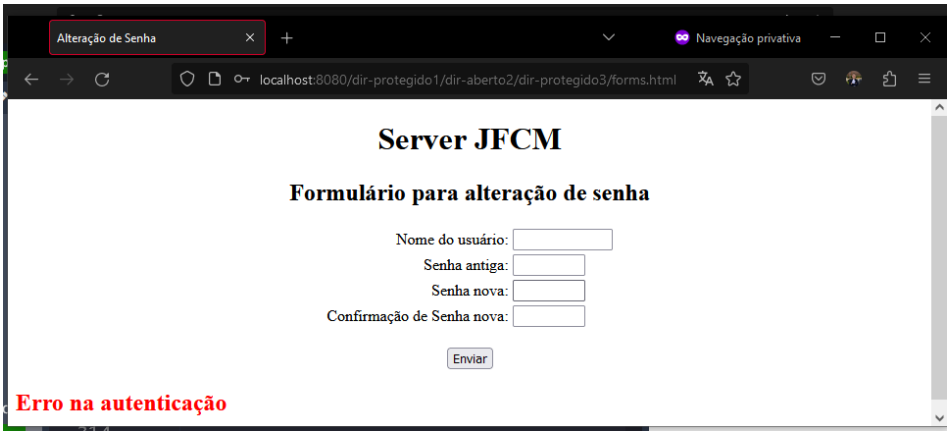
Realizando uma segunda alteração, agora os dois diretórios protegidos podem ser acessados com as credenciais `admin:new`

- Confirmação de senha nova não confere.

  Quando duas senhas diferentes são colocadas nos campos de *Senha nova* e *Confirmação de Senha nova*, o resultado é mostrado abaixo.

- Credenciais erradas

  Quando a autenticação nos campos *Nome do usuário* e *Senha antiga* não são válidas, o resultado é mostrado abaixo.



5 Limitações

A implementação apresenta certas limitações quando comparada com sistemas mais maduros, incluindo limitações de otimização, de segurança, extensibilidade e confiabilidade.

Com relação a otimização, o sistema não implementa um parser reentrante (capaz de funcionar em multithreaded, sem uso de variáveis globais no parser).

A segurança é feita com *salt* fixo e hardcoded no código. O ideal seria variar o *salt* e alterar o método de encriptação, para modelos mais modernos e confiáveis.

O servidor não consegue implementar técnicas de roteamento e controle, como o *express.js*, pois os endereços web estão ligados diretamente com a organização dos arquivos no servidor.

Na parte de confiabilidade, não há padronização de testes, como teste unitários.

6 Trabalhos Futuros

Os próximos passos no desenvolvimento do servidor devem ser guiados pelas limitações presentes atualmente e na construção de uma base mais sólida para adições de novas funcionalidades.

O código do sistema só foi refatorado uma vez durante seu desenvolvimento. Sendo assim, uma segunda refatoração nesse estágio mais maduro poderia agregar muita qualidade à arquitetura do software.

Em segundo lugar, o parser reentrante seria fundamental no aprimoramento de performance do programa, uma vez que o travamento dos *mutexes* na função que realiza o parse, pode (e vai) gerar uma fila. Além disso, a diminuição no número de variáveis globais é desejada para sistemas que buscam crescer ainda mais.

Por fim, deve-se implementar uma solução que melhore a segurança do sistema e a interação com as informações de autenticação (principalmente as senhas). Por exemplo, um banco de dados profissional, que pode estar alocado na nuvem.

7 Comentários Finais

- Sugestão de prazos

  Alguns prazos ficaram curtos durante o andamento do semestre. Uma sugestão para resolver esse problema seria um prazo de duas semanas. Não há necessidade, à princípio, de diminuir o número de atividades, uma vez que elas possam continuar ocorrendo semanalmente.

  A diferença está no prazo final de cada uma das atividades, que é estendido de uma semana, de forma que o aluno sempre tenha duas atividades para entrega após uma aula (a entrega atual com 14 dias de prazo e a entrega da semana anterior com 7 dias de prazo).

  Isso aliviaria a carga em situações específicas e semanas mais cheias.

- Sugestão de conteúdos

  Senti que o tempo para apresentação das ferramentas flex e bison foram muito curtos, ainda mais considerando a falta de documentação extensa desses softwares.

  Portanto, uma aula para discutir somente essas ferramentas, além das atividades, seria de muita valia.