

O servidor web gerado para o trabalho de redes tem uma interface de administração que pode ser acessado pela URL /gerenciador.

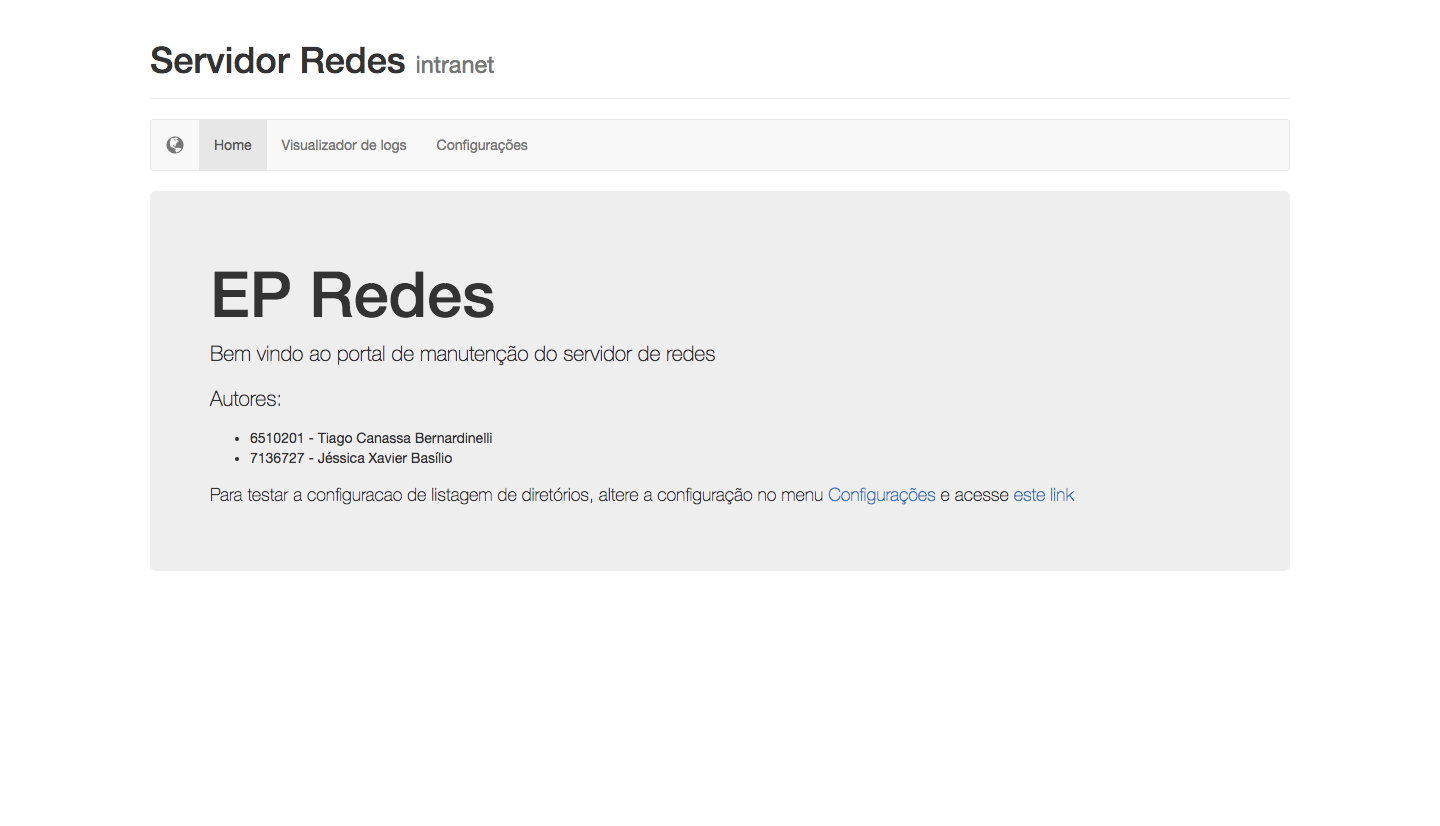


Figure 1 Página principal de administração do servidor (index).

Implementação:

*“O servidor Web deve implementar a versão 1.0 do HTTP, definido na RFC-1945 [1], onde requisições HTTP separadas são enviadas para cada componente da página Web. Este servidor deverá manipular múltiplas requisições simultâneas de serviços em paralelo. Isso significa que o servidor Web é multithreaded. No thread principal, o servidor escuta uma porta fixa. Quando recebe uma requisição de conexão TCP, ele ajusta a conexão TCP através de outra porta e atende essa requisição em um thread separado”.*

Para a implementação do servidor WEB *Multithread* criamos a classe *HttpRequest* que implementa a interface *Runnable*, o método *ProcessMessage* trata todas as requisições do servidor.



Figure Classe java multthread.

A Implementação da classe *Runnable* faz com que o servidor seja capaz de manipular varias requisições simultaneamente, uma vez que foi aberto uma thread para cada Accept da classe Socket.



Figure Entry Point da aplicação servidora.

Implementação:

*“Quando o servidor Web encontrar um erro, ele deve enviar uma mensagem de resposta com a fonte HTML apropriada, de forma que a informação do erro seja exibida na janela do browser”.*

Ao acessar uma página não existente, o usuário receberá a resposta com código HTTP 404 – NOT FOUND.

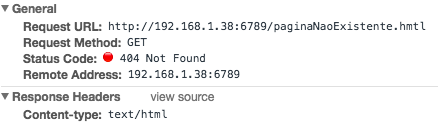


Figure Resultado de requisição feita pelo google Chrome

Dentro do método HandleGet da classe HttpRequest temos uma variável booleano preenchida com o resultado da busca do arquivo dentro do diretório “contente” da aplicação, seu resultado serve de insumo para a varíavel statusRequest e contentLineType.



Figure Trecho de código de retorno sobre existencia de arquivos

A página web renderizada no browser é produto da variável inputStream, quando não é encontrada a página e o servidor não permite a listagem de diretórios preenchemos a variável com um FileInputStream apontando para o arquivo ./content/gerenciamento/404.html



Figure Código de retorno da página não encontrada.

O Browser apresenta a seguinte tela:

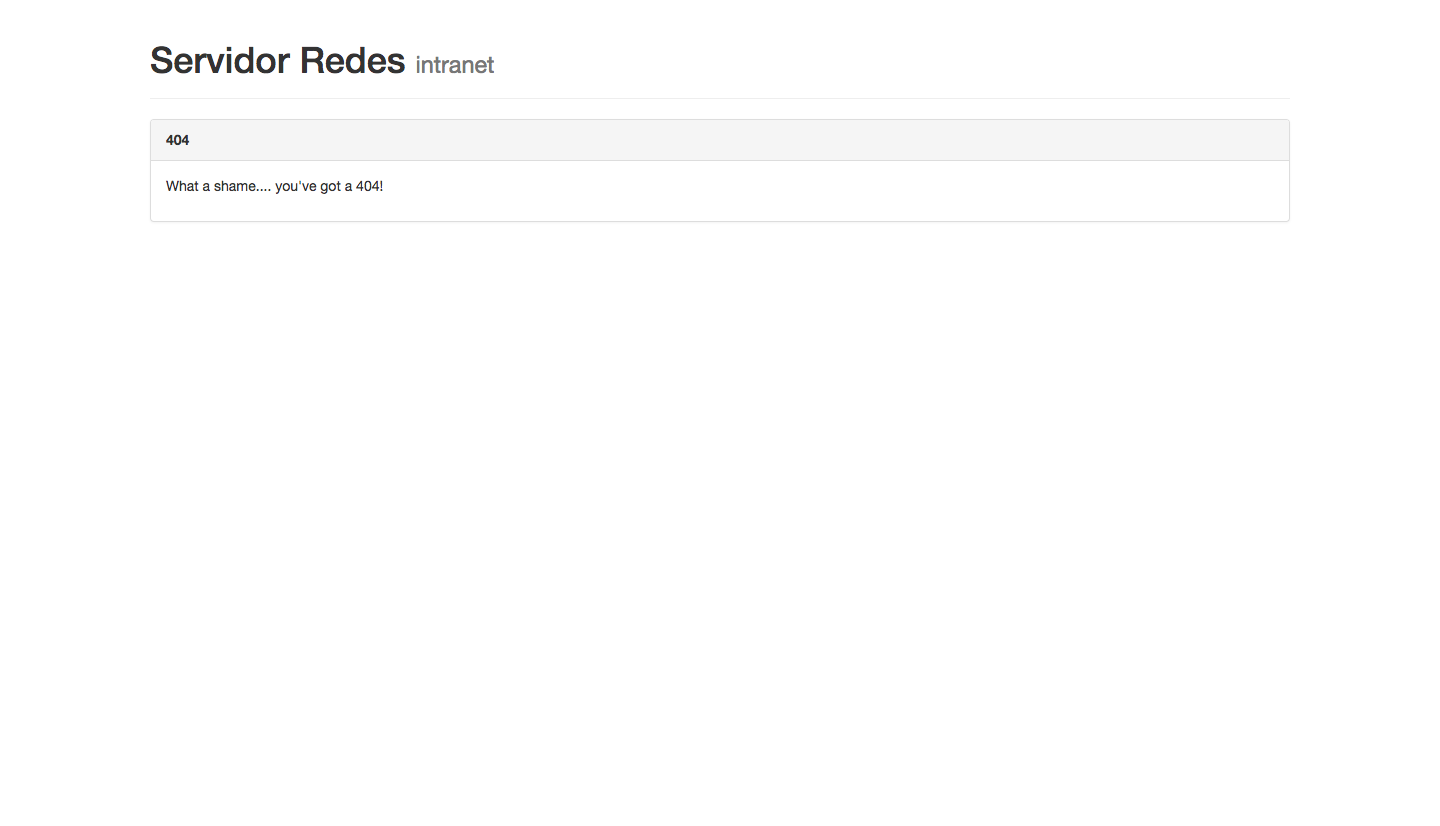


Figure Página não encontrada

Quando o arquivo existe e o mime-type esta dentro das especificações do servidor retornamos o arquivo existente, para testes poderão ser acessadas a raiz do website, retornando a página index.html e o arquivo /giphy.gif, além deles, todos os arquivos que fazem parte do “gerenciamento intranet” bem como arquivos css e js que compõe a intranet também são acessíveis via requisição HTTP.

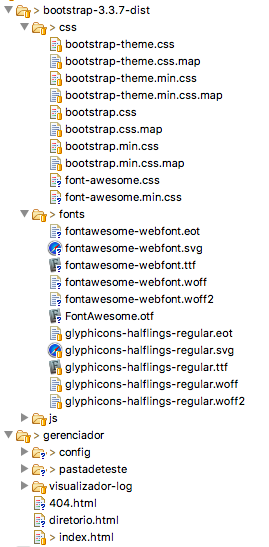
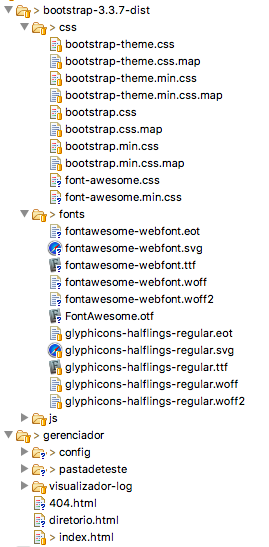


Figure Arquivos acessíveis

Implementação:

*“O arquivo de log deve ser escrito em formato texto, e ser acessível através do servidor web. Ele deve conter as seguintes informações (mínimas): endereço origem, porta origem, horário da requisição, conteúdo requisitado, quantidade de bytes transmitidas em resposta a requisição”.*

Os logs do sistema são gerados dentro do arquivo /requests.log, o arquivo fica na raiz do projeto portanto não é acessível diretamente pelo portal via requisição GET, construímos uma interface de administração onde é possível visualizar os logs, esta é acessível pelo menu de gerenciamento e pela URL /gerenciador/visualizador-log.

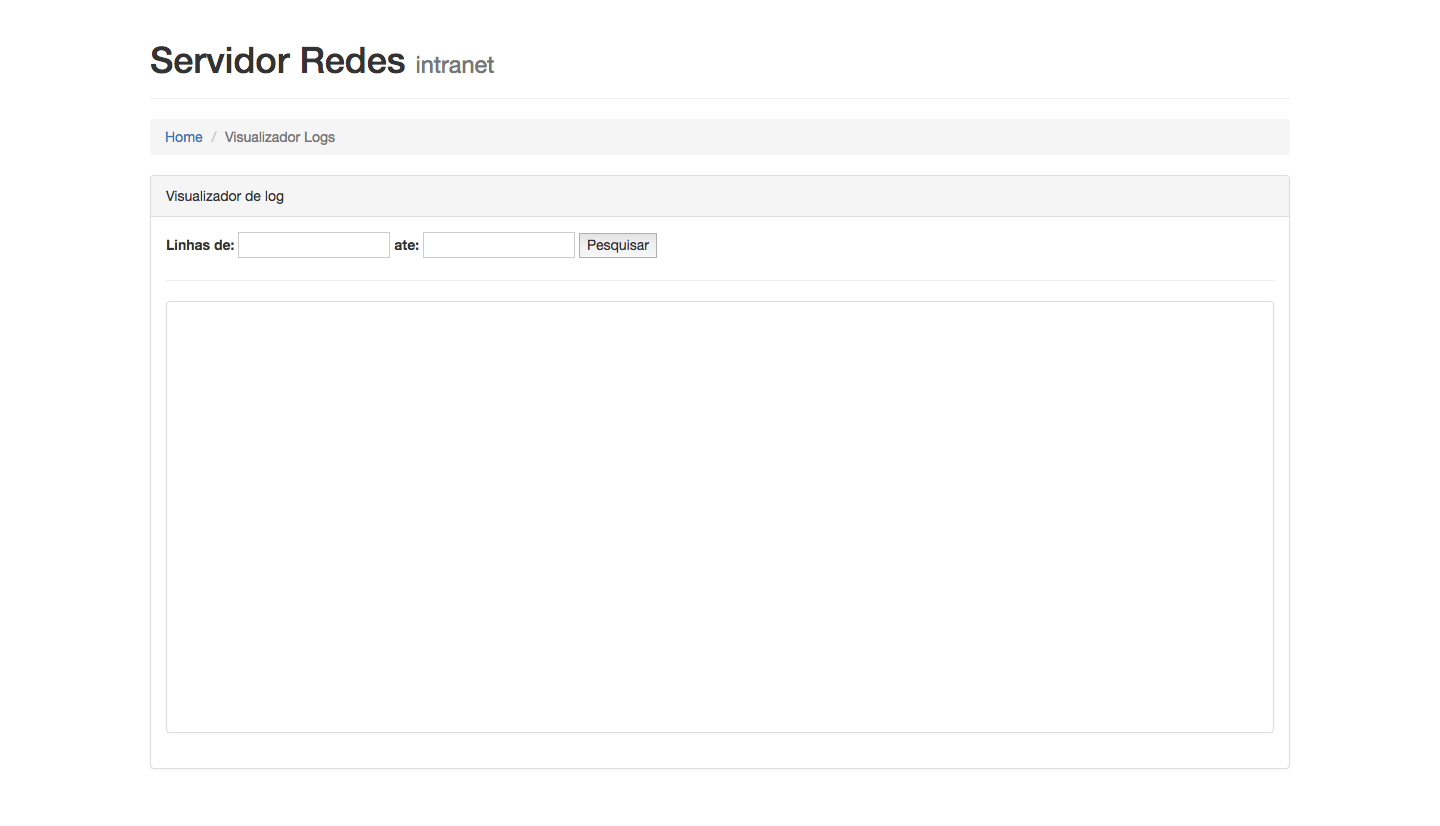


Figure Página de logs

A interface permite que se escolha um range de linhas que o usuário queira ver, o botão pesquisar contém validações no range, apenas após o range ser validado o botão pesquisar efetua a requisição para o Browser.

É feita então uma chamada ajax via POST para o servidor de aplicação que manipula ela pelo método HandlePost da classe HttpRequest.



Figure 10 Código JavaScript para AJAX POST



Figure 11 Código JAVA de tratamento ao método HTTP Post

Existem diversas chamadas AJAX POST pelo sistema, o código de cada uma delas implementa a interface ProgramaWEB, podem ser vistos no pacote epredes.programas.

O log do sistema contém as informações mínimas pedias pelo enunciado, estão destacados em amarelo na figura 11.

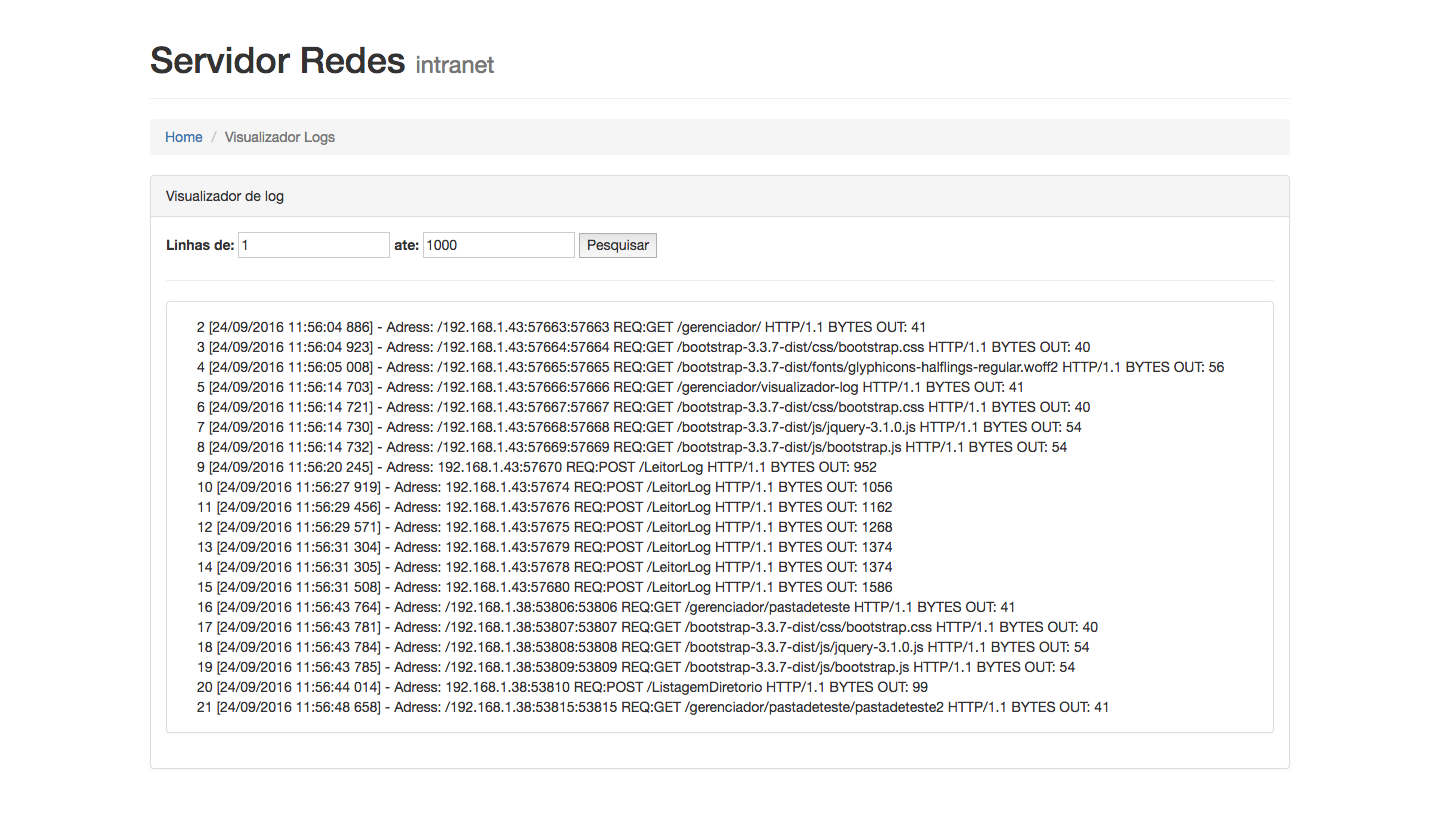


Figure Página web com logs.

A interface web permite o scroll para verificar todas as linhas de log requisitadas.

As classes HandlePost e HandleGet logam as informações de entrada por meio da classe LogRequests, ela acessa o FileSystem por um AsynchronousFileChannel aberto durante a existência da instância, a escrita acontece Byte a Byte, toda vêz que o servidor inicia a execução o arquivo é limpo e inicado novamente.



Figure Classe Java de log

Implementação:

*“Configuração da listagem ou não de conteúdo de diretórios:*

*Esta característica deve ser configurável (opção de execução do programa ou arquivo de configuração). Quando usuário acessa uma url do tipo* [*http://host.usp.br/diretorio/*](http://host.usp.br/diretorio/)*, as seguintes respostas são possíveis (ou seja, opções de configuração):*

1. *o conteúdo do diretório é listado em uma página html (opção que permite a listagem do diretório).*
2. *uma página com uma mensagem do tipo “O conteúdo do diretório não pode ser listado” é retornada ao usuário (opção que não permite a listagem do diretório).*
3. *uma página padrão é exibida: nesse caso, o servidor web exibe a página chamada index.html, e se esta não existir, exibe a mensagem de erro da opção 2”.*

A configuração de listagem ou não de diretórios é feita on-line e tem reflexo em real-time para a aplicação, a página de configuração pode ser acessada pela HOME de administração ou pelo endereço /gerenciamento/config/.

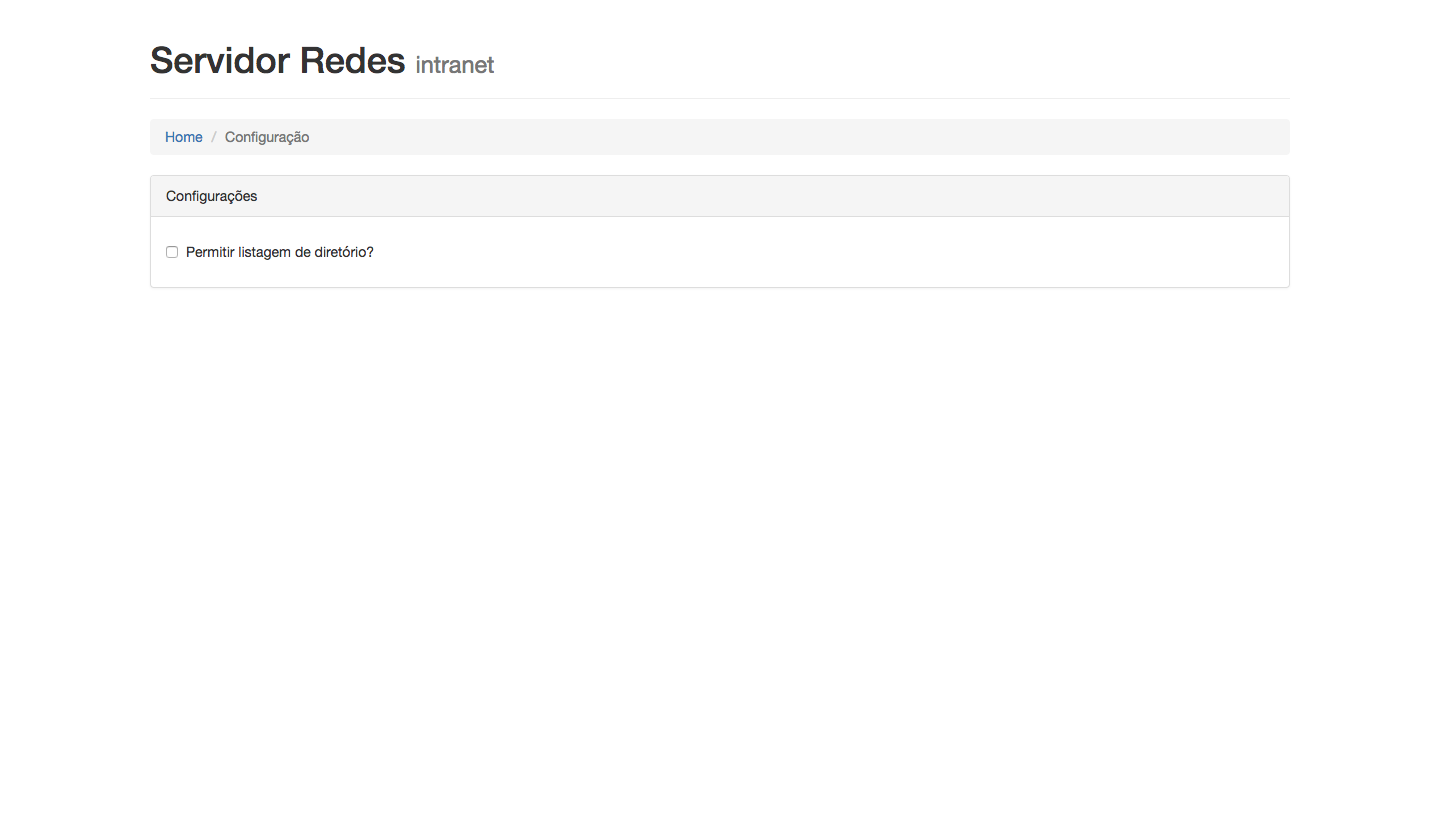


Figure 14 Página de configuração sem acesso a listagem de diretórios

Com a configuração não permitindo acesso a listagem de diretórios recebemos um erro 404 not found no acesso a um diretório existente, conforme ilustrado na figura 7.

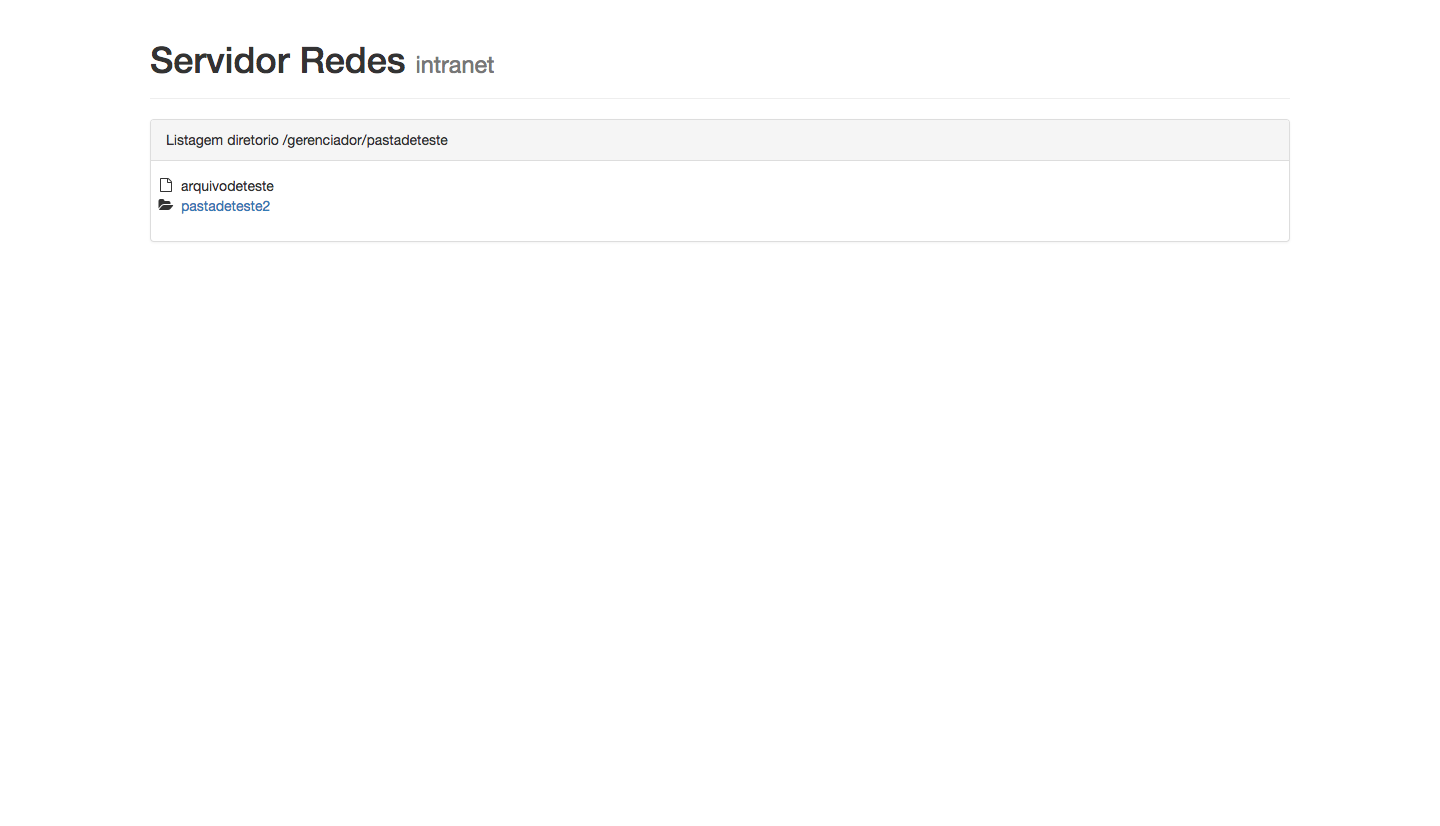
Para alterar a configuração do servidor é feito um POST para o servidor para o endereço /Salvaconfig passando como parâmetro o valor do checkbox.



O servidor por sua vez guarda as configurações no arquivo config.properties, acessado pela classe ObtemConfig, descendente também da classe Programaweb, o que a torna automaticamente uma servlet que responde ao método POST com a execução do método Executa.



Ao acessar um diretório com a permissão de visualização de diretório ativa, o browser renderiza a página da figura a seguir.

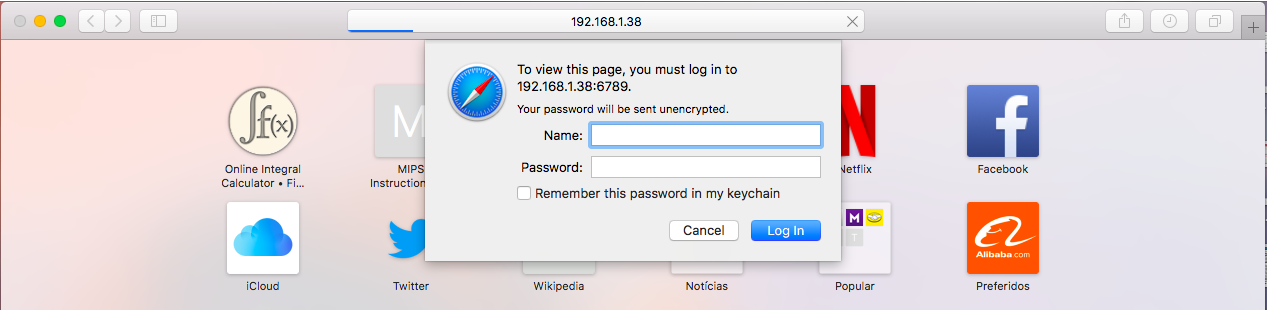


Na página temos o diretório acessado e a listagem de arquivos e diretórios dentro da pasta, os diretórios são links para navegação.

Implementação:

*“O servidor web deve as funções relativas a “Autenticação de Acesso” descritas no item 11 da RFC1945 [1]. Assim, quando o servidor receber uma requisição de acesso ao diretório* ***restrito*** *(sub-diretório do diretório principal do servidor web), ele deverá enviar mensagem ao cliente solicitando as informações de autenticação (usuário e senha)”.*

Ao acessar qualquer URL da interface de administração, o servidor retorna para o browser o código 401 NOT AUTHORIZED e o tipo de autenticação exigida, no nosso caso foi escolhida a autenticação básica, o que faz com que o browser renderize um diálogo de inserção de senha.



O sistema preenche as variáveis que compõe a resposta do GET com o parâmetro WWW-Authenticate: Basic, foi utilizada a resposta contentLineType pois a mesma compõe os HEADERS da response.



Após o usuário digitar seu login e senha o browser envia para o servidor o parâmetro Authorization seguido por um hash 64 com usuário e senha, para o trabalho, não validamos qual usuário foi inserido, apenas que existe um usuário.

A request completa enviada pelo browser pode ser vista no SYSOUT padrão da aplicação.



Figure Request com autorização de execução



Figure Request sem autorização de execução.

Todos os arquivos dentro da pasta gerenciador são de visualização restrita, o código que define os HTTPs Response foi atualizado para responder pelo 401.

