Serviços em Redes de Computadores

Apache, SSH, Cron, Samba

Apache Web Server

O Apache Web Server é um dos servidores web mais populares, gratuito e de código aberto. Ele começou na forma de pequenas correções de software para o **httpd**, daemon web produzido pela **NCSA** (***National Center For Supercomputing Applications***), da Universidade de Illinois. Esse daemon acabou sendo adquirido, na metade da década de 90, por um grupo de desenvolvedores conhecido como Apache Group (que, mais tarde, formaria a empresa **ASF** - ***Apache Software Foundation***). O grupo decidiu reescrever a aplicação, batizando-a de **Apache HTTPD Server**. Atualmente, a ASF tem várias aplicações, mas acabou ficando mais conhecida pelo servidor web.

Para instalar o Apache, digite:

```
apt-get install apache2 -y
```

Logo após a instalação, o servidor já é iniciado e deverá estar funcionando. Para checar isso, basta digitar:

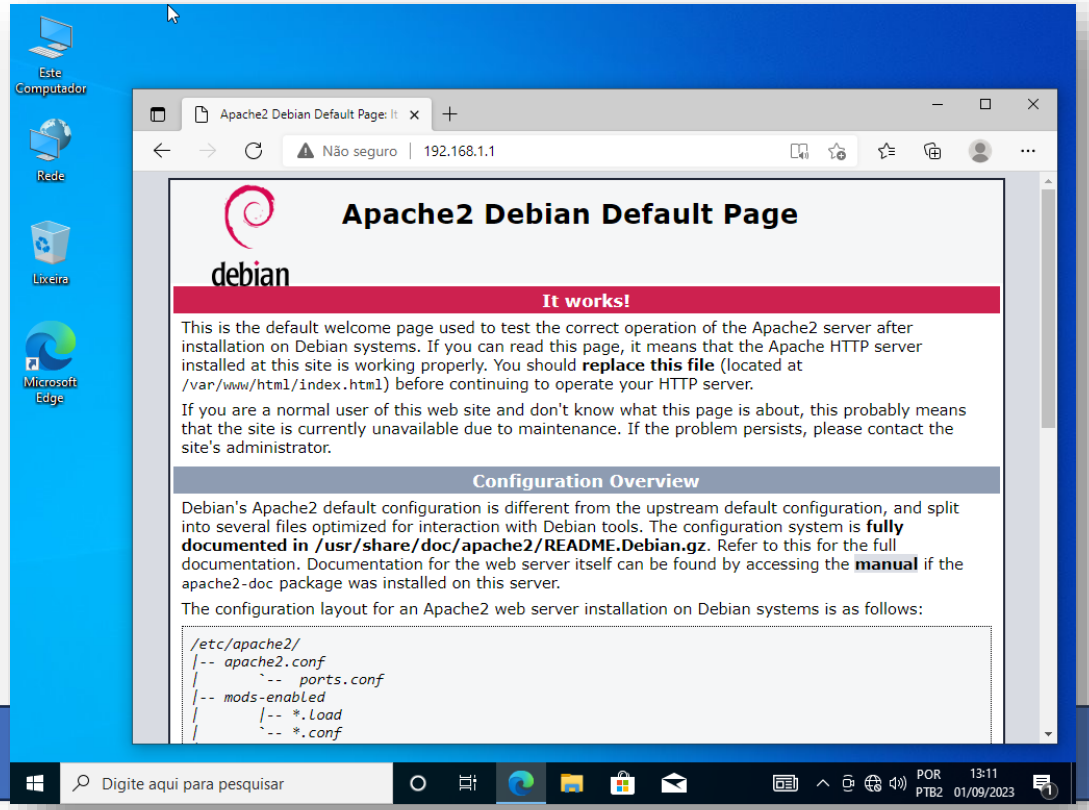
```
service apache2 status
```

```
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-08-31 15:00:54 -03; 2h 5min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1265 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 1269 (apache2)
      Tasks: 55 (limit: 9492)
     Memory: 17.5M
        CPU: 596ms
     CGroup: /system.slice/apache2.service
             ├─1269 /usr/sbin/apache2 -k start
             ├─1270 /usr/sbin/apache2 -k start
             └─1271 /usr/sbin/apache2 -k start
```

Se você vir as palavras “active (running)” em verde, significa que deu tudo certo: o Apache está no ar.

Podemos testar a funcionalidade do servidor usando uma outra máquina – por exemplo, uma estação **Windows 10**.

Se seu firewall estiver com a porta 80 liberada, basta abrir o navegador na sua máquina Windows e digitar `192.168.1.1`. A página do Apache deve ser carregada, conforme mostrado na figura ao lado.



O **Apache** pode funcionar tanto como um servidor para sua intranet, onde apenas o pessoal da empresa tem acesso, como ser acessível via Internet, de fora da empresa.

Dentro da empresa, essa página inicial do Apache pode ser modificada para que ela funcione como quadro de avisos, com notícias sobre a empresa e eventos da organização. Se você souber um pouco de **HTML** e **CSS**, poderá cuidar disso. Para isso, basta modificar a página inicial do Apache. No entanto, antes de fazer isso, é melhor conhecermos um pouco do seu funcionamento e onde estão os principais arquivos desse *web server*.

Arquivos de Configuração

Essa página inicial que você vê ao acessar o Apache está armazenada não na internet, mas localmente, em `/var/www/html`. O arquivo contendo a página é o `index.html`. Ela, geralmente, contém uma mensagem de boas-vindas padrão ou uma página de exemplo, mas que pode variar de acordo com a distribuição Linux. Qualquer modificação no conteúdo desse arquivo irá **refletir** no que você verá ao acessar o endereço 192.168.1.1 pelo navegador. Você pode usar qualquer editor de texto para modificá-la. Por exemplo, você pode substituir o conteúdo padrão por informações sobre o seu site, links para as páginas principais, ou qualquer outra coisa que desejar.

O local onde o Apache procura este arquivo, no entanto, pode ser alterado editando-se os dados contidos no arquivo de configuração localizado em `/etc/apache2/sites-available/000-default.conf`.

Apache Web Server: Arquivos de Configuração

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
```

Ao lado, vemos as primeiras linhas do arquivo `000-default.conf`. A linha contendo `DocumentRoot` indica onde o apache deve buscar pelo arquivo inicial do site. Você pode optar por mover os arquivos do site para, por exemplo, `/var/meu-site`. Fazendo isso, você precisaria substituir `/var/www/html` por `/var/meu-site`.

Também podemos adicionar uma diretiva que instrui o Apache qual deve ser o arquivo inicial do site, a `DirectoryIndex`.

Se ela não estiver presente, o Apache supõe que seja um arquivo chamado `index.html` (na página de teste do Apache, é esse o nome dele). Mas se você preferir especificar um outro padrão para suas páginas iniciais, basta acrescentar o nome na diretiva:

```
DirectoryIndex  default.html
```

No exemplo acima, mudamos a página principal para `default.html`. Ou seja, o Apache, para acessar sua página inicial, deve buscar por “default.html” a partir de agora. Se essa página não existir, ele exibirá um erro ou a listagem de arquivos encontrados na pasta.

Também podemos dar mais de uma opção para essa busca:

```
DirectoryIndex  default.html index.html index.php index.asp
```

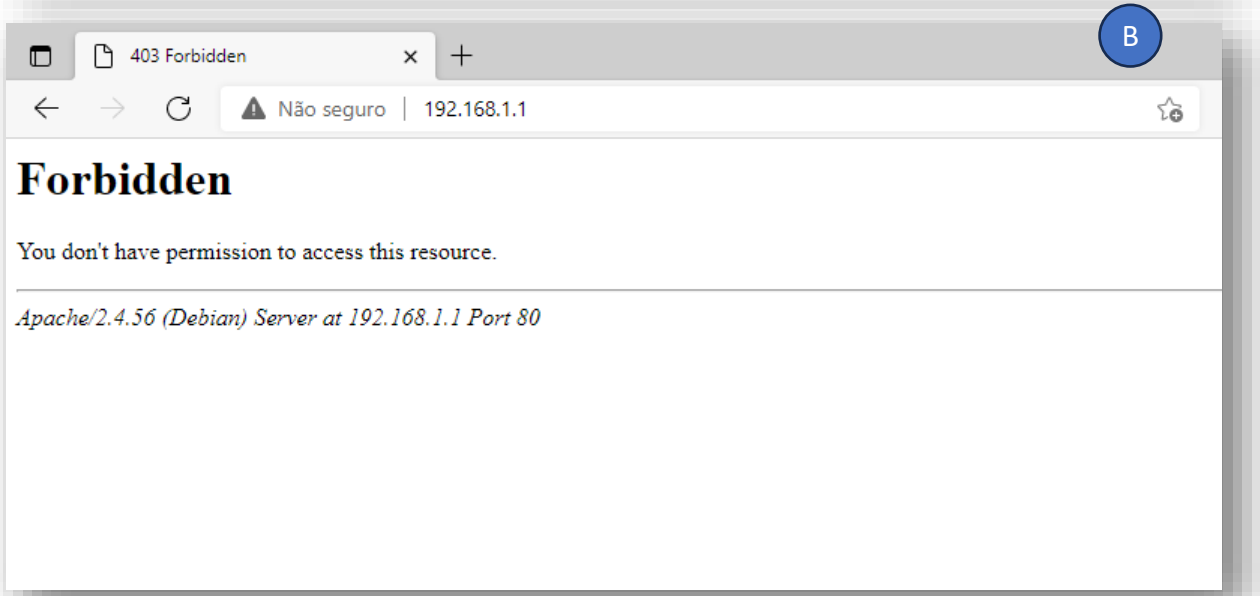
Nesse exemplo, se a página “default.html” não existir, ele procurará por “index.html”; se também não houver, pesquisará por “index.php” e, por fim, não existindo também, tentará carregar a “index.asp”. Se nenhuma das opções tiver sucesso, então ele exibirá uma mensagem de erro ou a listagem de arquivos encontrados na pasta.

Um problema comum (e que pode ser um brecha de segurança) que acontece quando uma página não é encontrada é justamente permitir que o servidor web exiba os diretórios existentes em um determinado local. Por exemplo, se o Apache procura pelo arquivo principal da pasta e não o encontra, provavelmente ele irá mostrar na tela uma listagem de diretórios:

Apache Web Server: Arquivos de Configuração

A

B



A figura ao lado mostra que, na ausência da página principal procurada pelo Apache, ele simplesmente listou o conteúdo do diretório principal, revelando os arquivos ali existentes.

Listar diretórios pode expor informações sensíveis, como estrutura de pastas e nomes de arquivos, a potenciais invasores. Bloquear essa listagem pode ajudar a proteger o sistema contra ataques direcionados e aumentar a privacidade dos recursos no servidor.

Para evitarmos essa listagem, basta incluirmos uma nova diretiva no mesmo arquivo `000-default.conf`:

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
<Directory /var/www/html>
        Options -Indexes
</Directory>
```

O trecho com a diretiva `<Directory>` indica que aquela pasta (`/var/www/html`) não deve ter sua listagem indexada (ou seja, não deve ser exibida) – indicado pelo sinal de "–" antes da palavra "Indexes".

Uma vez adicionada essa diretiva, o resultado disso passaria a ser o mostrado na figura **B** (a mensagem de "Forbidden").

Para que essa modificação funcione, reinicie o Apache digitando `service apache2 restart`.

Apache Web Server: Arquivos de Configuração

O módulo `mod_gnutls` é semelhante ao `mod_ssl`, mas suporta recursos e protocolos que o `mod_ssl` não suporta e não usa a biblioteca OpenSSL

Quando o Apache é iniciado, ele se conecta a uma determinada porta na máquina local e aguarda por pedidos de conexão. As portas estão especificadas em `/etc/apache2/ports.conf`. Ao lado, um exemplo do conteúdo desse arquivo. Nele, a diretiva `Listen` diz ao servidor para escutar a porta 80 e responder nela aos pedidos de conexão. Se for um pedido usando SSL ou TLS (a depender do módulo que o Apache encontrar), então a porta 443 deverá ser usada. Essa porta só funcionará se o modulo `mod_ssl` ou o `mod_gnutls` estiver ativo.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

Por padrão, ele escuta todos os endereços IP da máquina nas mesmas portas. No entanto, pode ser necessário que ele escute em portas específicas, ou apenas em endereços selecionados, ou em uma combinação de ambos.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 192.168.1.1:80
Listen 192.168.0.1:8082
```

Por exemplo, eu posso determinar que o servidor responda apenas na porta 80 quando o IP for 192.168.1.1 e na porta 8082 quando o IP for 192.168.0.1. Nesse caso, o arquivo poderia ser modificado para que se parecesse como a segunda figura, de cima para baixo. Esse tipo de direcionamento específico é conhecido como **Virtual Host**. Podemos verificar quais portas estão sendo usadas pelo web server digitando:

```
ss -l | grep http
```

Você deverá ver algo semelhante ao mostrado abaixo:

```
NetId  State    Recv-Q Send-Q            Local Address:Port
nl      UNCONN   0      0                       rtnl:kernel
tcp     LISTEN   0      128                     0.0.0.0:ssh
tcp     LISTEN   0      128                     [::]:ssh
tcp     LISTEN   0      511                     *:http
```

A linha em destaque mostra que apenas a porta 80 está sendo usada pelo Apache. A porta 443, embora esteja configurada no arquivo `ports.conf`, não teve os módulos SSL ou TLS carregados e, por isso, ela não está ativa.

O asterisco antes de “http” indica que o servidor está atendendo requisições tanto através do protocolo IPv4 quanto IPv6.

Apache Web Server: Arquivos de Configuração

O Apache tem um diretório contendo vários módulos que ampliam sua funcionalidade. Esses módulos estão em `/etc/apache2/mods-available`, que indica que são módulos disponíveis para uso, mas que não estão ainda habilitados. Dentre esses módulos, temos o de SSL (para criptografia), o de Status (que permite obter algumas estatísticas do servidor através da própria URL), etc. Cada módulo contém dois arquivos, um de configuração (terminado em `.conf`) e outro que aponta o local físico onde o módulo de fato está (terminado em `.load`).

Neste diretório, por exemplo, o módulo SSL está definido no arquivo `ssl.load`. Nele, é possível ver que esse módulo, na verdade, está em `/usr/lib/apache2/modules`:

```
# Depends: setenvif mime socache_shmcb
LoadModule ssl_module /usr/lib/apache2/modules/mod_ssl.so
```

Já o arquivo `ssl.conf` contém parâmetros de configuração do protocolo SSL (figura abaixo). As primeiras linhas, por exemplo (que começam com "SSLRandomSeed"), são usadas para configurar a geração de números aleatórios criptograficamente seguros, que são necessários para o funcionamento seguro do protocolo SSL/TLS. Isso é necessário para a criação de chaves de sessão seguras e a geração de números de inicialização para cifras.

```
#
SSLRandomSeed startup builtin
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect builtin
SSLRandomSeed connect file:/dev/urandom 512

##
##  SSL Global Context
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##

#
#   Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl
```

A geração de números aleatórios seguros é uma parte crítica da criptografia, pois ajuda a evitar a previsibilidade e a repetição de valores.

Apache Web Server: Arquivos de Configuração

As diretivas "AddType" são usadas para mapear extensões de arquivo para tipos MIME (Multipurpose Internet Mail Extensions). Os tipos MIME dizem aos navegadores e clientes como interpretar o conteúdo do arquivo que estão recebendo do servidor. Por exemplo, no caso desta linha:

```
        AddType application/x-x509-ca-cert .crt
```

essa diretiva está dizendo ao Apache que os arquivos com a extensão `.crt` devem ser tratados como "`application/x-x509-ca-cert`", tipo comumente usado para certificados de **Autoridade de Certificação** (***CA***) ou certificados raiz. Os navegadores e outros clientes podem usar essa informação para determinar como lidar com o arquivo quando o servidor envia um certificado com essa extensão. Assim, os navegadores ficam sabendo que estão lidando com um certificado e não com outro arquivo qualquer.

Há um diretório que nos mostra quais dos módulos presentes em `/etc/apache2/mods-available` estão, de fato, habilitados. É o `/etc/apache2/mods-enabled`. Os módulos listados ali estão, de fato, funcionais e pode ser usados (embora, às vezes, precisem de certa configuração).

Um dos módulos habilitados é o de **status** (`mod_status`), que permite que obtenhamos estatísticas referentes ao servidor, como a versão do Apache, há quanto tempo o servidor está no ar, quais são os clientes conectados, e várias outras. O arquivo de configuração desse módulo é o `status.conf`. Abaixo, um exemplo de seu conteúdo:

```
<IfModule mod_status.c>
        # Allow server status reports generated by mod_status,
        # with the URL of http://servername/server-status
        # Uncomment and change the "192.0.2.0/24" to allow access from other hosts.

        <Location /server-status>
                SetHandler server-status
                Require local
                #Require ip 192.0.2.0/24
        </Location>
```

As estatísticas, do modo como esse arquivo está configurado, somente serão acessíveis através do próprio servidor, através de um navegador web, digitando-se o IP do servidor (ou `localhost`) e acrescentando-se `/server-status` ao final.

Mas é possível liberar para outros IPs, removendo o sinal de comentário (“#”) da linha “Require ip ...” e informando um IP ou uma rede válida:

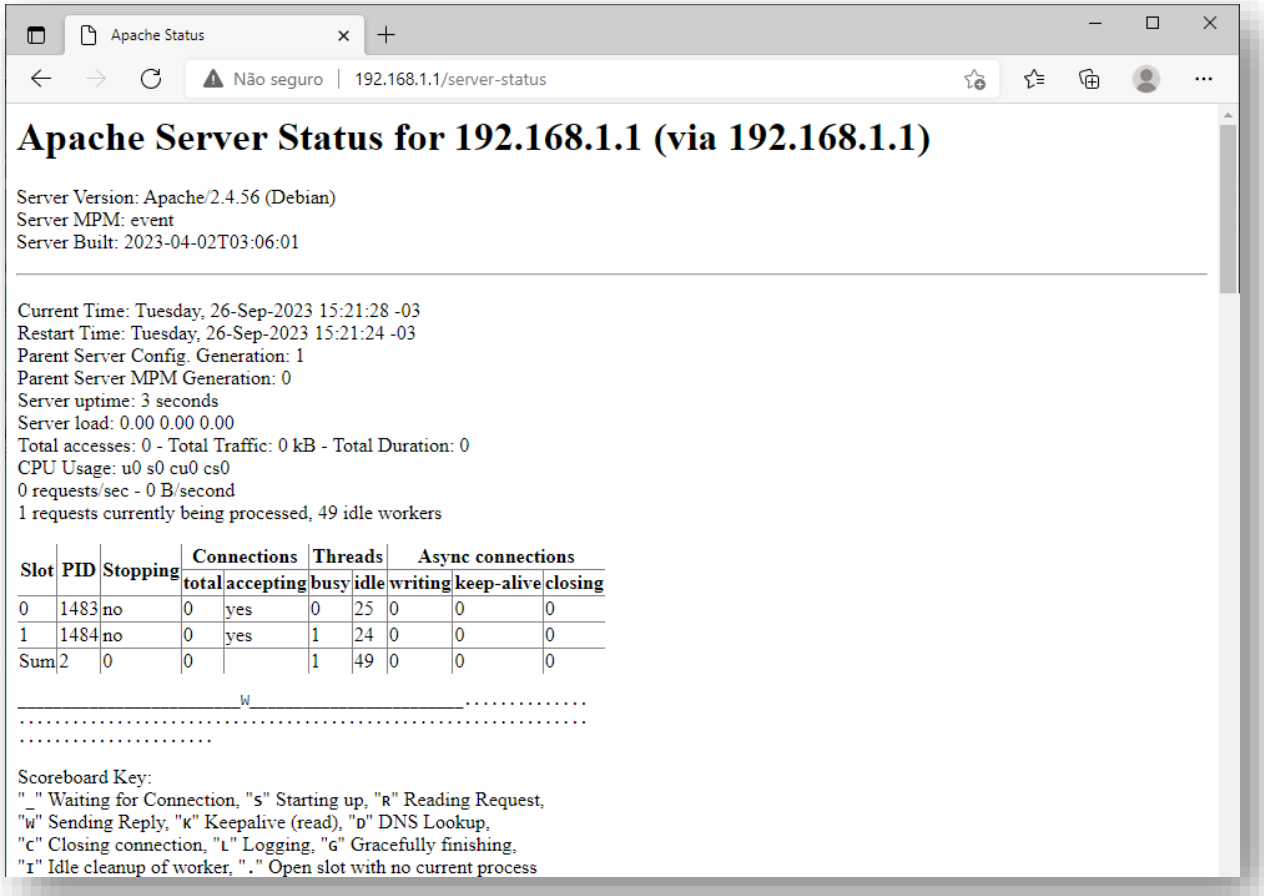
```
  Require ip 192.168.1.0/24
```

Apache Web Server: Arquivos de Configuração

Como não temos uma interface gráfica em nosso servidor, vamos fazer essa alteração para que nossa estação Windows consiga ter acesso a estas estatísticas. Para isso, digite `nano /etc/apache2/mods-enabled/status.conf`. Lá, localize a linha que começa com "Require ip..." e faça com que fique conforme abaixo:

```
Require ip 192.168.1.0/24
```

Pressione CTRL+O e, depois, ENTER para salvar. Em seguida, tecle CTRL+X para sair. Agora, é necessário reiniciar o Apache. Para isso, basta digitar `service apache2 restart`. Usando sua máquina **Windows 10**, navegue até `192.168.1.1/server-status`. Uma página semelhante a esta deverá aparecer:



Note que, em `status.conf`, a linha `<Location /server-status>` é que determinada que, ao ser digitado `/server-status` na URL, as estatísticas do servidor sejam mostradas.

Se você alterar essa linha para `<Location /status>`, por exemplo, então `/server-status` não mais funcionará e, ao invés disso, você deverá digitar `/status` para obter as mesmas estatísticas. Isso nos mostra que a diretiva `Location` permite a personalização da URL.

Após configurar um módulo, podemos habilitar ou desabilitá-lo, conforme a necessidade. Para desabilitar um módulo, usamos o comando `a2dismod` seguido do nome do módulo (aqui, o módulo *status*):

```
a2dismod status
```

Para reativá-lo:

```
a2enmod status
```

Apache Web Server: Arquivos de Configuração

Não deixe o módulo de status habilitado para redes fora da empresa – isso configuraria uma tremenda falha de segurança, já que informações sensíveis são expostas ali.

Um dos principais arquivos de configuração do Apache é o `/etc/apache2/apache2.conf`. Ali estão presentes algumas configurações globais do serviço. Veja um exemplo dos parâmetros encontrados nele (algumas linhas começadas por "#" foram suprimidas para melhorar a legibilidade):

```
# KeepAlive: whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
KeepAlive On
#
MaxKeepAliveRequests 100
#
KeepAliveTimeOut 5

# These need to be set in /etc/apache2/envvars
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}
```

A diretiva `KeepAlive` é usada no servidor para habilitar ou desabilitar a funcionalidade de conexão persistente (também conhecida como "Keep-Alive"). Quando definida como "On" (que é o padrão), permite-se que várias solicitações e respostas HTTP sejam transmitidas pela mesma conexão TCP, em vez de abrir uma nova conexão para cada solicitação. Essa funcionalidade pode melhorar o desempenho do servidor web e a eficiência da comunicação entre o cliente e o servidor, especialmente para sites que servem várias solicitações a partir de uma única página (por exemplo, carregamento de recursos como imagens, folhas de estilo e scripts). Em alguns casos, no entanto, desativar o "Keep-Alive" pode ser uma boa ideia, como em servidores que atendem a muitos clientes, cada um com uma única solicitação – é o caso de servidores de streaming, por exemplo. Em ambientes de alto tráfego, isso pode reduzir o consumo de recursos do servidor, pois as conexões são encerradas rapidamente após cada solicitação.

O parâmetro `MaxKeepAliveRequests` é uma diretiva usada para controlar o número máximo de solicitações que podem ser feitas em uma única conexão persistente entre um cliente e o servidor. Como a conexão de "Keep-Alive" permite que vários pedidos e respostas HTTP sejam transmitidos pela mesma conexão TCP, é uma boa prática limitar o número máximo de solicitações para ele. Isso ajuda a evitar que uma única conexão seja mantida indefinidamente, o que poderia consumir demais os recursos do servidor.

Por outro lado, o `KeepAliveTimeout` é uma diretiva de configuração que serve para controlar o tempo máximo que uma conexão de "Keep-Alive" deve ser mantida aberta quando não há nenhuma atividade entre o cliente e o servidor. Assim, o parâmetro determina quanto tempo o servidor web deve esperar por uma nova solicitação do cliente antes de encerrar a conexão, evitando que ela seja mantida em espera por um período excessivo de tempo.

Apache Web Server: Arquivos de Configuração

Neste arquivo também está definido qual usuário o Apache deverá usar no sistema. Um usuário e um grupo são criados para ele assim que o Apache é instalado, e esses nomes são atribuídos às variáveis `APACHE_RUN_USER` e `APACHE_RUN_GROUP` e estão definidas no arquivo `/etc/apache2/envvars`.

No Debian, normalmente usuário e grupo são `www-data` mas, em outras distribuições, como a Red Hat, podem estar definidos como `apache`.

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory /var/www>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Outras diretivas dizem respeito à estrutura de diretórios. As diretivas agrupadas em `<Directory>` informam ao Apache o que pode e o que não pode ser visualizado pela própria aplicação em termos de pastas do sistema.

A linha `Options FollowSymLinks` define as opções para este diretório. Neste caso, permite que links simbólicos sejam seguidos. Um link simbólico é um tipo especial de arquivo que atua como um atalho para outro arquivo ou diretório. Quando você acessa um link simbólico, na verdade está acessando o arquivo ou diretório para o qual ele aponta.

A opção "Seguir Links Simbólicos" em um servidor web, como o Apache, controla como o servidor lida com esses links ao acessar recursos em seu sistema de arquivos. Ele pode tanto ser encaminhado para o arquivo ou diretório para o qual o link aponta (e daí ele estaria “seguindo” o link), como pode segui-lo apenas se o arquivo ou diretório-alvo for da mesma pessoa que está acessando o atalho.

A linha `AllowOverride None` especifica se as diretivas de configuração podem ser modificadas através da presença de arquivos `.htaccess` dentro deste diretório. Arquivos `.htaccess` também são arquivos de diretivas e eles conferir permissão de fazer justamente o oposto do que diz a diretiva principal. Mas, neste caso, como a “AllowOverride” foi definida como "None", então não serão permitidas alterações por meio desses arquivos (mesmo que eles estejam presentes).

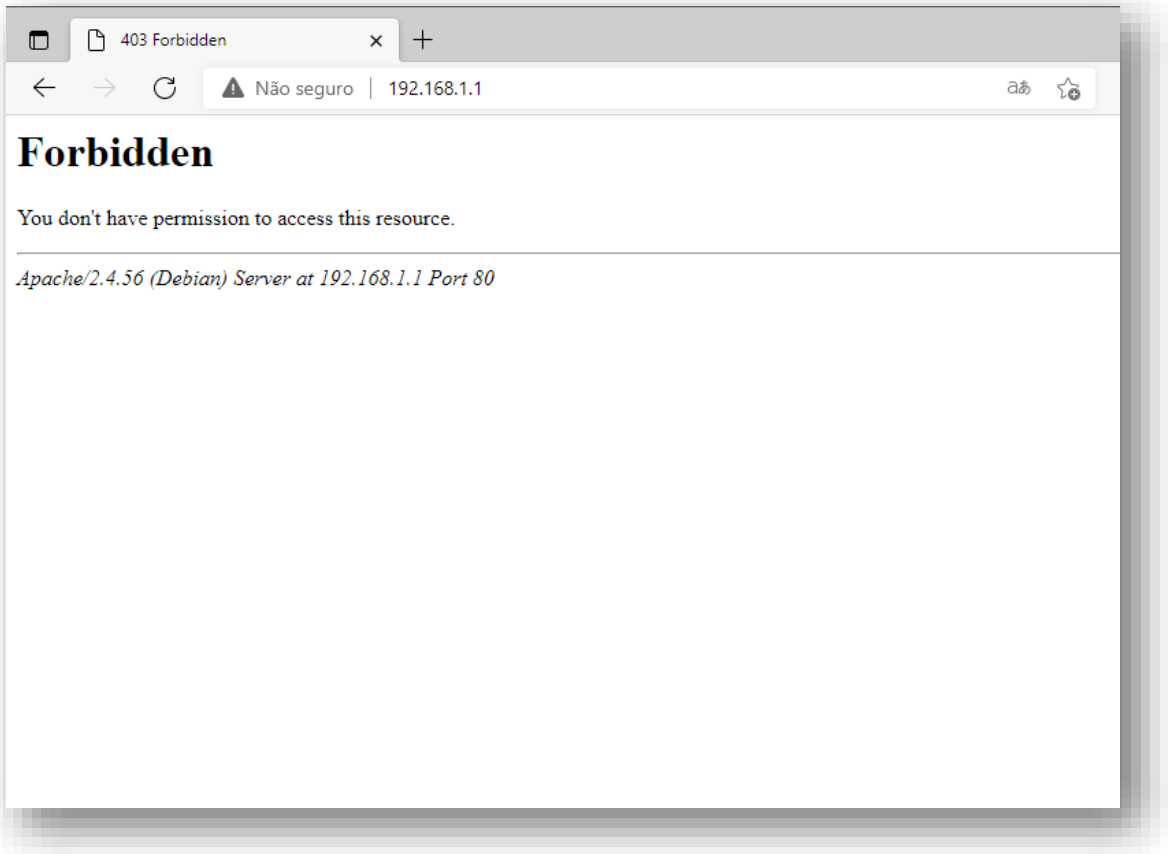
E a última linha diz que tudo o que estiver localizado a partir da raiz do sistema (“/”) deve ser negado para todos os usuários (`Require all denied`).

A segunda diretiva trata do diretório `/var/www` (`<Directory /var/www/>`). Neste, além de seguir links simbólicos, está definido que o diretório pode ser indexado (`Options Indexes` ...). Indexar um diretório significa que, se a página padrão do diretório não for localizada, o Apache poderá listar o conteúdo

encontrado no diretório. Foi justamente isso que tentamos impedir quando editamos o arquivo `000-default.conf` anteriormente. A listagem acontecia justamente porque o padrão é listar. Mas, como foi possível perceber, acrescentando outra diretiva que negue isso (com `Options -Indexes`) conseguimos barrar a listagem.

Como nós já havíamos negado o acesso à listagem, esse `Options Indexes` do arquivo `etc/apache2/apache2.conf` não surtirá efeito (a configuração do arquivo `000-default.conf` tem preferência ao `apache2.conf`). E a opção `Require all granted` indica que todos devem ter acesso ao diretório `/var/www` e, consequentemente, aos seus subdiretórios.

Uma curiosidade aqui: se você alterar `Require all granted` para `Require all denied`, nós simplesmente perderemos o acesso às páginas do servidor e veremos uma mensagem de "proibido" (figura ao lado) (para que a alteração funcione, claro, você deve reiniciar o Apache, digitando `service apache restart`).



Faz sentido que percamos o acesso a todo o site se negarmos permissão de acesso à pasta `/var/www`: já que o diretório que contém a página principal está logo abaixo, em `html`, negando o acesso a `/var/www`, o diretório subsequente também "herda" essa negativa – e, consequentemente, o arquivo `index.html` que está nele.

Falando em acessar arquivos, é possível usar uma diretiva específica caso se queira permitir ou negar acesso a arquivos de um determinado nome. Tomemos como base o mesmo arquivo `apache2.conf`, onde temos a diretiva `<Directory /var/www>`: vamos negar, a partir desse diretório, o acesso a qualquer arquivo chamado `teste.html`.

Para isso, vamos editar novamente as configurações, digitando `nano /etc/apache2/apache2.conf`.

Apache Web Server: Arquivos de Configuração

Procure a linha que `<Directory /var/www>` e, antes da *tag* de encerramento `</Directory>`, adicione estas linhas:

```
<Files "teste.html">
   Require all denied
</Files>
```

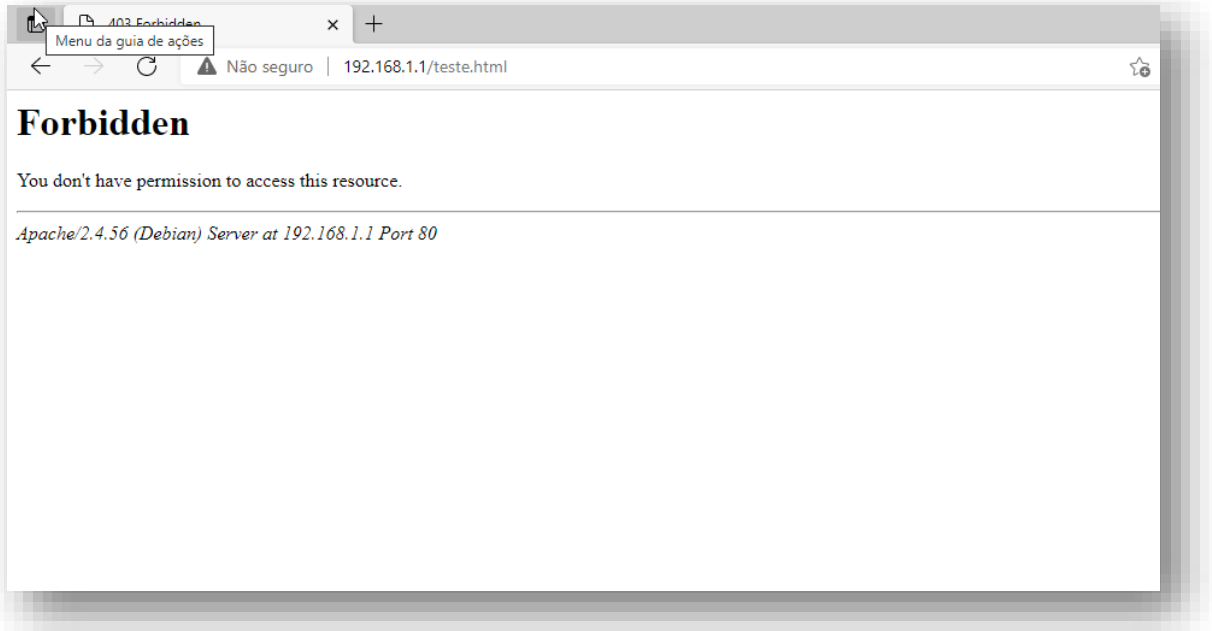
```
<Directory /var/www>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
    <Files "teste.html">
        Require all denied
    </Files>
</Directory>
```

O resultado final deve se parecer com o mostrado na segunda figura, logo abaixo.

Salve o arquivo, teclando CTRL+O e pressionando ENTER. Depois, pressione CTRL+X para sair. Agora, vamos criar um arquivo chamado `teste.html`. Para facilitar, podemos fazer uma cópia do arquivo `index.html` e salvá-lo como `teste.html`. Para isso, digite:

```
cp /var/www/html/index.html /var/www/html/teste.html
```

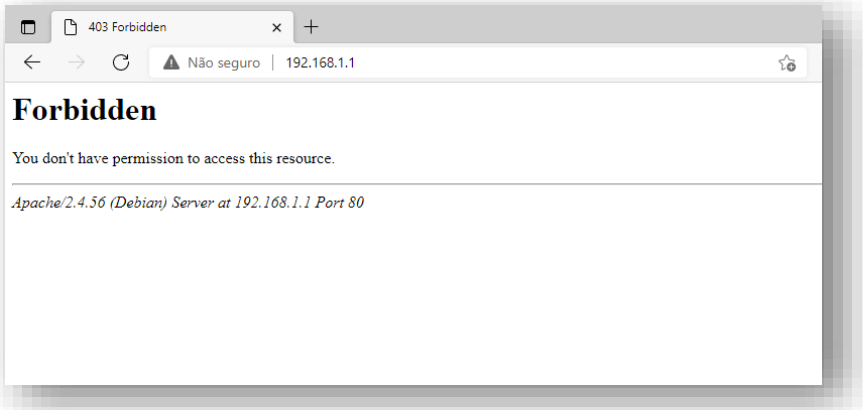
Agora, na máquina Windows 10, use o navegador web e acesse a URL `192.168.1.1/teste.html`. Você deverá ver a mensagem mostrada na terceira figura ao lado, indicando que a página está bloqueada. Na prática, a partir do diretório `www` (o que inclui todos os seus subdiretórios), qualquer página chamada `teste.html` estará bloqueada.



Isso ocorre porque, vale lembrar, as permissões são herdadas da pasta de nível mais alto e, se não houver nada que diga o contrário em um diretório específico, a herança continua, mesmo que seja para um nome de arquivo.

Isso significa que, se você também tiver um arquivo `teste.html` em `/var/www/html/novodir`, por exemplo, ele também não poderá ser acessado por causa da diretiva `Require all denied`, que se aplica a tudo que está em `/var/www` e abaixo.

Personalizando os Erros do Apache




```
Content-language: en
Content-type: text/html; charset=UTF-8
Body:----------en--
<!--#set var="TITLE" value="Access forbidden!"
--><!--#include virtual="include/top.html" -->

  <!--#if expr="v('REDIRECT_URL') =~ m:/$:" -->

    You don't have permission to access the requested directory.
    There is either no index document or the directory is read-protected.

  <!--#else -->

    You don't have permission to access the requested object.
    It is either read-protected or not readable by the server.

  <!--#endif -->

<!--#include virtual="include/bottom.html" -->
----------en--

Content-language: es
Content-type: text/html
Body:----------es--
<!--#set var="TITLE" value="&iexcl;Acceso prohibido!" -->
<!--#include virtual="include/top.html" -->

   <!--#if expr="v('REDIRECT_URL') =~ m:/$:" -->

      Usted no tiene permiso para acceder al directorio solicitado.
      No existe un documento &iacute;ndice, o el directorio est&aacute;
```

Mensagens como as da página de erro ao lado podem ser personalizadas, usando HTML e CSS com um editor de texto simples.

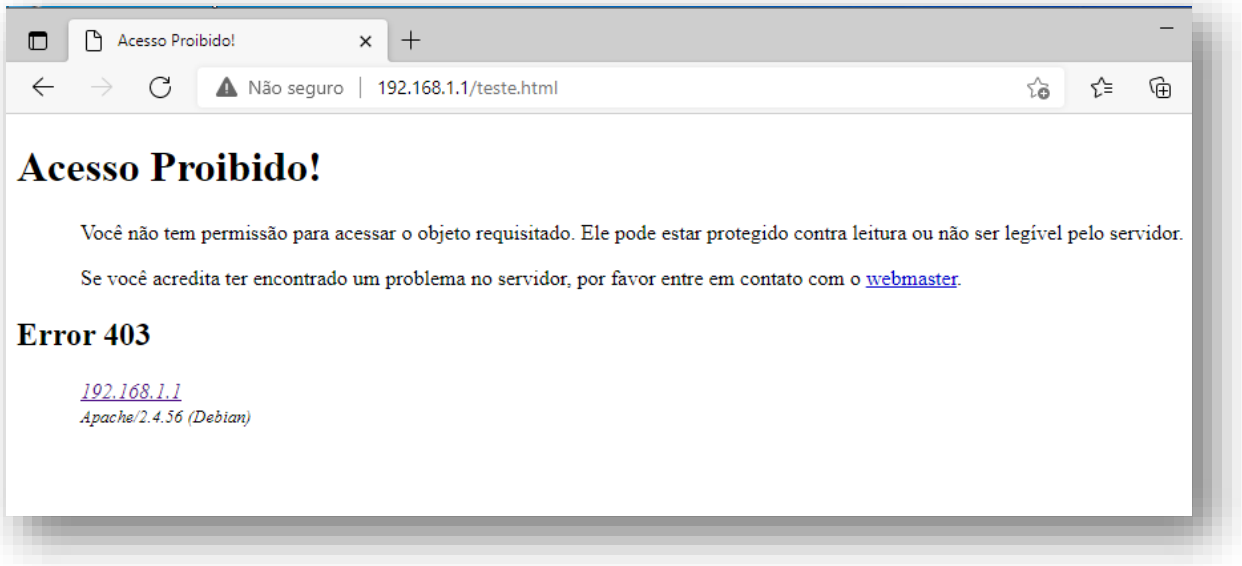
Por padrão, o próprio daemon dispara as mensagens apropriadas para cada situação, mas você pode modificar isso. Alguns arquivos pré-configurados já estão presentes no Apache, como os encontrados em `/usr/share/apache2/error`. Por exemplo, o arquivo `HTTP_FORBIDDEN.html.var` pode ser usado para indicar que o acesso a um determinado recurso está bloqueado, ao invés de se usar a mensagem padrão da figura ao lado. Arquivos como este contém, em diversos idiomas, a mensagem que informa o usuário de que um determinado arquivo ou diretório teve seu acesso proibido. Um exemplo de parte desse arquivo pode ser visto na segunda figura à esquerda. Ali, há texto em inglês (a partir da linha “Content-language: en”) e o mesmo texto em espanhol (a partir de “Content-language: es”) .

Normalmente, arquivos como este trazem uma mensagem um pouco mais completa do que as mensagens padrão. Para permitir que o Apache use estes arquivos, precisamos habilitar o módulo `mod_include`, digitando:

```
a2enmod include
```

Uma vez feito isso, tente acessar de novo o seu arquivo “teste.html” com o navegador da estação Windows. Você verá que a mensagem de “forbidden” mudou.

Apache Web Server: Personalizando os Erros do Apache



Agora, a mensagem está em português e tem um pouco mais de detalhes sobre o que pode ter acontecido para que este acesso tenha sido bloqueado.

Outra maneira de modificar as mensagens é editar o arquivo `/etc/apache2/apache.conf` e escrever, nele, a mensagem você gostaria de ver para cada erro, ou apontar um novo local para um arquivo html que contém a mensagem.

Por exemplo, para mostrar uma nova mensagem de erro de "Acesso Proibido", basta adicionar a linha abaixo ao final do arquivo:

```
ErrorDocument 403 "Desculpe, este documento não pode mais ser exibido"
```

Usando o parâmetro `ErrorDocument` podemos indicar qual o conteúdo da mensagem desejado para cada situação. Neste exemplo, para erros 403, o texto exibido será "Desculpe, este documento não pode mais ser exibido". Com mensagens de erro como esta especificadas diretamente no arquivo `/etc/apache2/apache.conf`, os arquivos de mensagens tradicionais serão ignorados.

Mas ao invés de indicar o texto, posso preferir indicar o local onde o texto está. Por exemplo, posso criar um arquivo html em `/usr/share/apache2/error` e, nele, colocar minha mensagem. Vamos fazer isso. Para criarmos um arquivo html com uma mensagem personalizada, digite:

```
nano /usr/share/apache2/error/MeuErro403.html
```

Na janela do editor *nano*, digite o texto a seguir:

Apache Web Server: Personalizando os Erros do Apache

```
<html>
 <meta charset="utf-8">
 <head><title> Erro! </title></head>
<body bgcolor="blue" text="white">
<h2>Erro ao acessar esta página:</h2>
<h3><font color="red">Página bloqueada!</font></h3>
</body>
</html>
```

Salve as modificações no arquivo `/etc/apache2/apache2.conf`, pressionando CTRL+O e ENTER. Para sair do nano, pressione CTRL+X.

Reinicie o Apache, digitando `service apache2 restart`.

Agora, use novamente sua máquina **Windows 10** para acessar a página "teste.html", que nós bloqueamos. Você deverá ver o erro da página html que acabamos de criar, conforme figura ao lado.

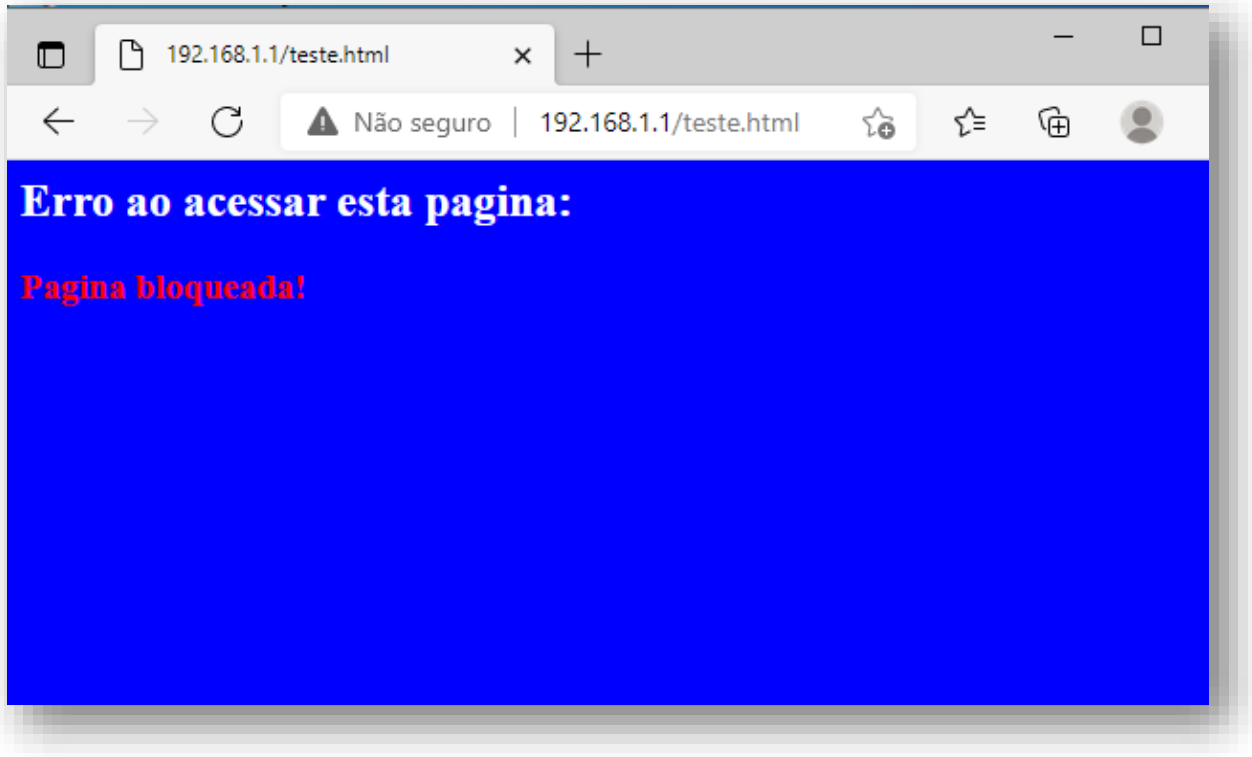
No código html, note a presença da linha `<meta charset="utf-8">`. Ela é necessária para que o navegador interprete corretamente a acentuação das palavras em português.

Note também que, no arquivo `/etc/apache2/apache2.conf`, não precisamos indicar o caminho completo da mensagem de erro (que seria `/usr/share/apache2/error`); basta indicar o diretório `/error` e o Apache já saberá que este diretório está em `/usr/share/apache2`.

Salve o seu texto, pressionando CTRL+O e, em seguida, ENTER. Pressione CTRL+X para sair do nano.

Agora, voltemos ao arquivo de configuração do Apache, digitando `nano /etc/apache2/apache2.conf`. Modifique a linha "ErrorDocument 403" para que fique conforme mostrado a seguir:

```
ErrorDocument 403 /error/MeuErro403.html
```



PHP Como Módulo do Apache

Outros módulos pode ser instalados no Apache para adicionar funcionalidades ao servidor. Um muito comum é o PHP, uma linguagem de script usada para implementar páginas dinâmicas (para melhorar a estática linguagem HTML). Para instalá-lo, basta digitar:

```
apt-get install php -y
```

Em poucos instantes, a instalação estará concluída e um novo módulo chamado `libphp8.2.so` estará disponível em `/usr/lib/apache2/modules`. Uma referência a ele também será adicionada em `/etc/apache2/mods-available/php8.2.load`, além de seu arquivo de configuração, `php8.2.conf`.

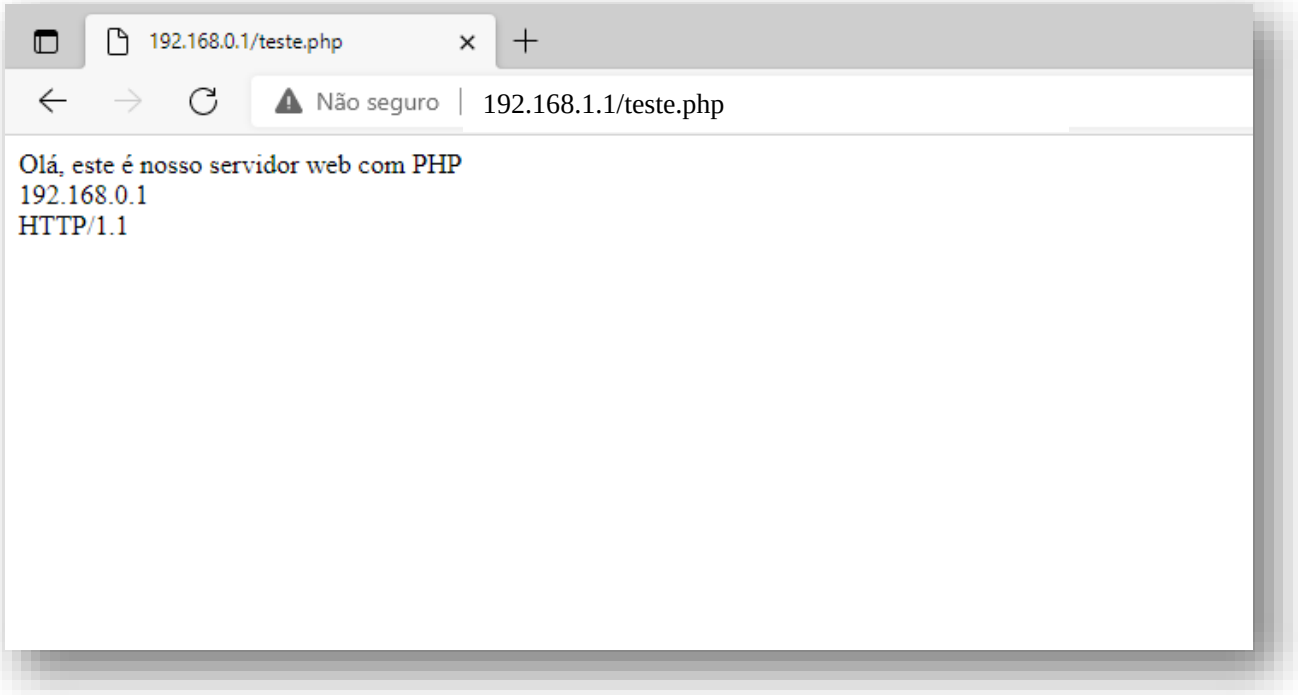
Após a instalação, o PHP já estará funcionando. Vamos criar uma página simples para comprovar isso. Digite nano `/var/www/html/teste.php` e escreva o seguinte:

```
<?php
  echo "Olá, este é nosso servidor web com PHP <br>";
  echo $_SERVER['SERVER_NAME'] . "<br>";
  echo $_SERVER['SERVER_PROTOCOL'];
 ?>
```

Esta página mostra na tela uma mensagem e o nome (ou o IP) do nosso servidor e o protocolo usado para acessá-lo (que é o HTTP).

Na nossa máquina **Windows 10**, use o navegador web para acessar nosso servidor, digitando `192.168.1.1/teste.php`.

Você deverá ver algo como mostrado na figura à direita.



PHP e MySQL

Se instalarmos um software Gerenciador de Banco de Dados (SGBD), como o MySQL, as possibilidades com o PHP aumentam muito. É possível realizar consultas com o PHP e trazer dados do Banco de Dados (BD) na tela.

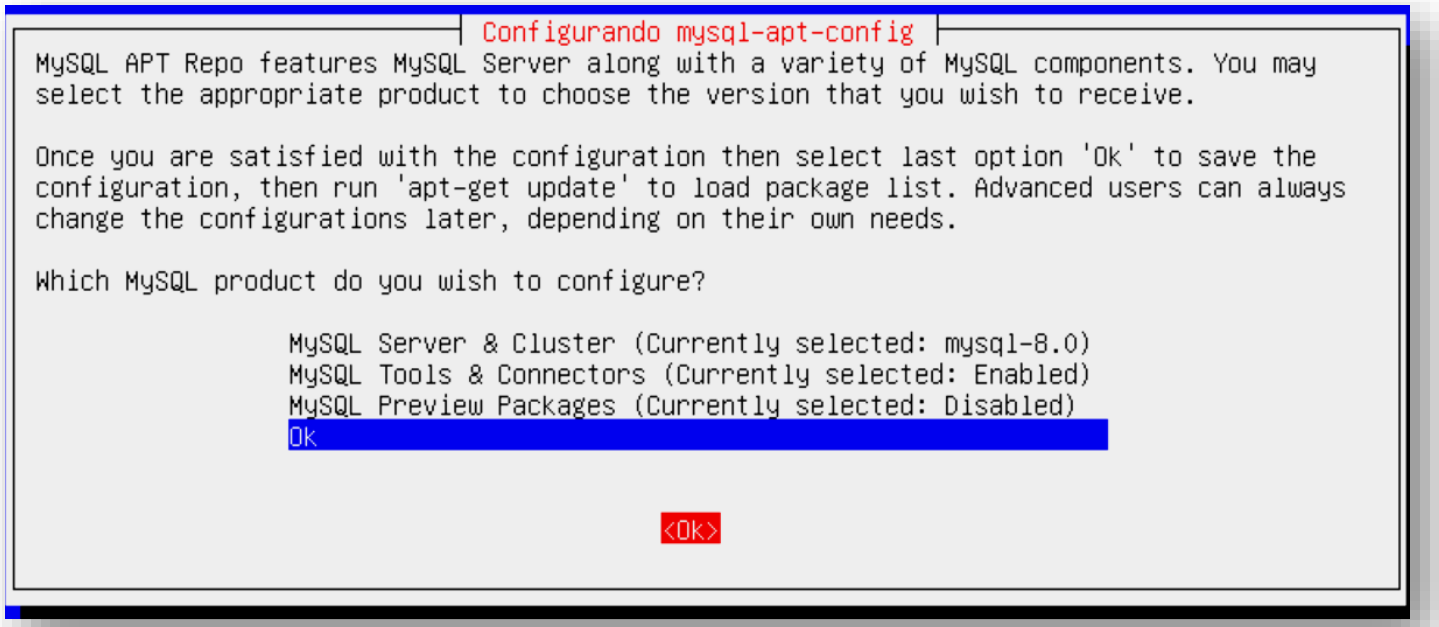
Vamos fazer uma experiência, instalando o MySQL. Não adianta usarmos o APT para baixá-lo, pois o MySQL não está no repositório. Temos que baixá-lo diretamente do site do MySQL com o comando abaixo:

```
wget -c https://repo.mysql.com/mysql-apt-config_0.8.22-1_all.deb
```

Com isso, o pacote `mysql-apt-config_0.8.22-1_all.deb` está disponível em nossa máquina, mas nada foi instalado. Para instalarmos, precisamos usar o DPKG, digitando o comando a seguir:

```
dpkg -i mysql-apt-config_0.8.22-1_all.deb
```

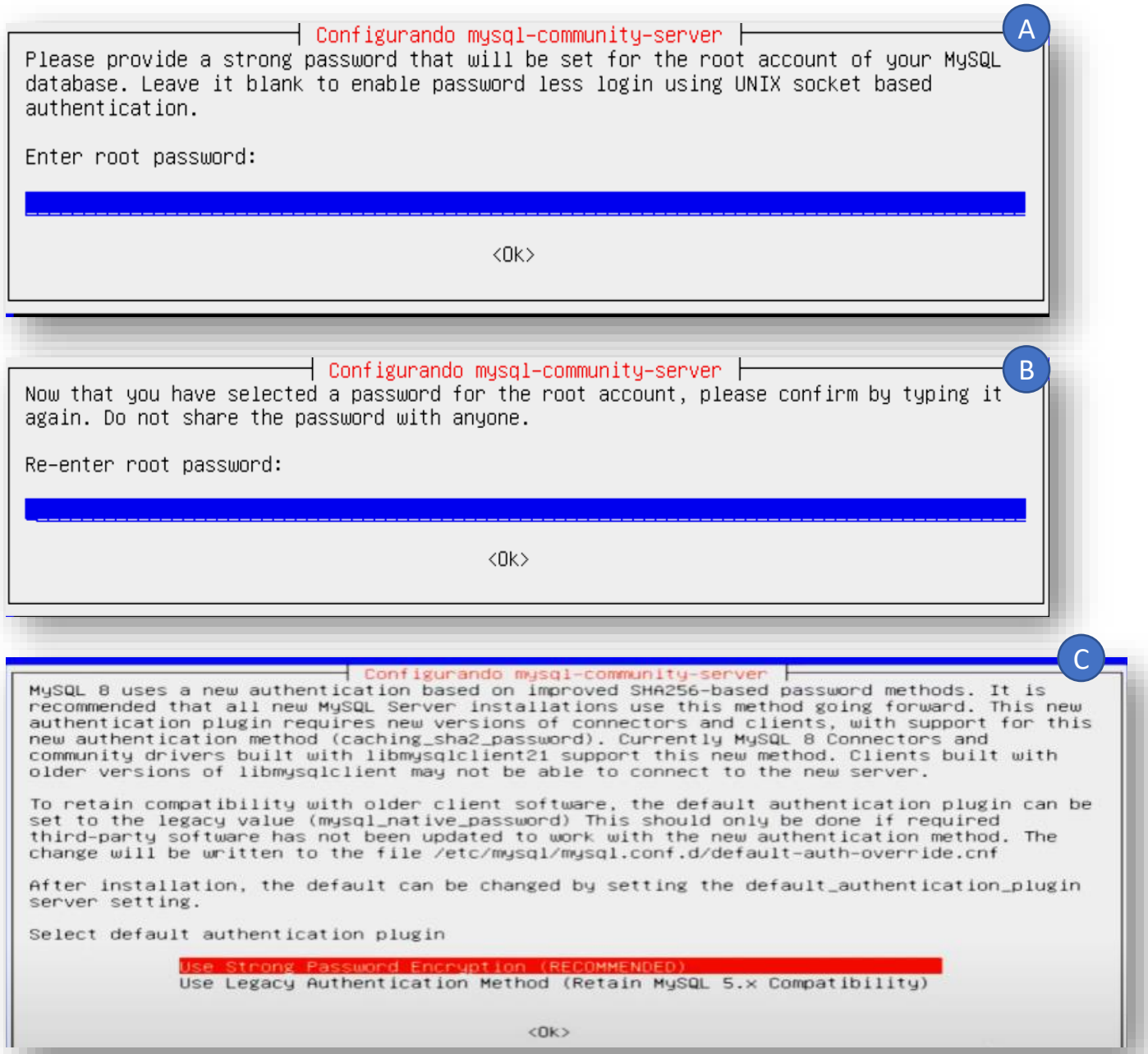
Isso fará com que a tela ao lado apareça. Basta andar com as setas do teclado até a opção “OK” e pressionar ENTER. Isso fará com que os repositórios do MySQL sejam adicionados à lista do Debian, permitindo a instalação. Precisamos, em seguida, fazer o Linux reler as URLs do repositório:




```
apt-get update
```

Agora, sim, é possível instalá-lo com o comando abaixo:

```
apt-get install mysql-server -y
```



Na sequência, uma tela pedindo para definir uma senha surgirá (figura **A**). Esta será a senha do usuário root do banco, que não tem nada a ver com o usuário root do Linux. Apesar de utilizar o mesmo nome, este é apenas o nome padrão para definir o usuário que é o administrador do MySQL.

Digite uma senha e pressione ENTER. Uma outra tela pedirá para você confirmar a senha, redigitando-a (figura **B**). Pressione ENTER novamente.

É importante que você se lembre dessa senha ou não será possível acessar o banco de dados! Se preferir, use a mesma senha que você definiu para o root do Linux.

Na tela seguinte, devemos escolher qual método de autenticação o MySQL irá utilizar (figura **C**). Podemos selecionar a segunda opção, “Use Legacy Authentication Method”, com as setas do teclado e pressionar ENTER.

Pronto! A instalação está finalizada e o MySQL já deve estar funcionando. Mas para podermos fazer algo útil com o BD, precisamos que ele contenha alguns dados. Vamos baixar uma tabela de cadastro de clientes bem simples que já está pronta em **http://www.alexandresmcampos.adm.br/fa/tabela.sql,** usando o comando a seguir:

```
 wget -O /root/tabela.sql -c
http://www.alexandresmcampos.adm.br/fa/tabela.sql
```

Depois, vamos importar essa tabela para o MySQL. Para isso, vamos iniciar o cliente MySQL via linha de comando:

```
mysql -u root -pabc123 -h localhost
```

Pressionando ENTER após digitar a linha acima, você deverá ver o "prompt" do MySQL:

```
mysql>
```

Isso significa que, tudo o que for digitado a partir de agora, diz respeito exclusivamente ao MySQL, e não ao shell do Linux. Portanto, comandos do shell não vão funcionar mas, sim, comandos que dizem respeito ao MySQL. Vamos importar os dados da tabela que estão no arquivo `tabela.sql`, que acabamos de baixar:

```
mysql> source /root/tabela.sql
```

Você deverá ver na tela mensagens semelhantes a esta:

```
Query OK, 1 row affected (0,06 sec)

Query OK, 0 rows affected (0,04 sec)

Query OK, 4 rows affected (0,04 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Isso significa que a importação funcionou. Para dar uma olhada nos registros existentes no BD, digite o comando abaixo. Você verá alguns nomes de clientes cadastrados:

```
mysql> select * from Alex.Cliente ;
```

```
+----+-----------+------------+--------+
| id | nome      | cidade     | estado |
+----+-----------+------------+--------+
|  1 | Alexandre | Indaiatuba | SP     |
|  2 | Felipe    | Campinas   | SP     |
|  3 | Carlos    | Indaiatuba | SP     |
|  4 | Silma     | New York   | NY     |
+----+-----------+------------+--------+
4 rows in set (0,02 sec)

mysql> _
```

Os dados ao lado serão listados na tela. São quatro clientes cadastrados, contendo o número de identificação (“id”), o nome, a cidade e o estado. Podemos, então, criar um pequeno site para nossa intranet que nos permita consultar esses clientes, usando o PHP. Vamos sair do MySQL digitando `exit` e pressionando ENTER.

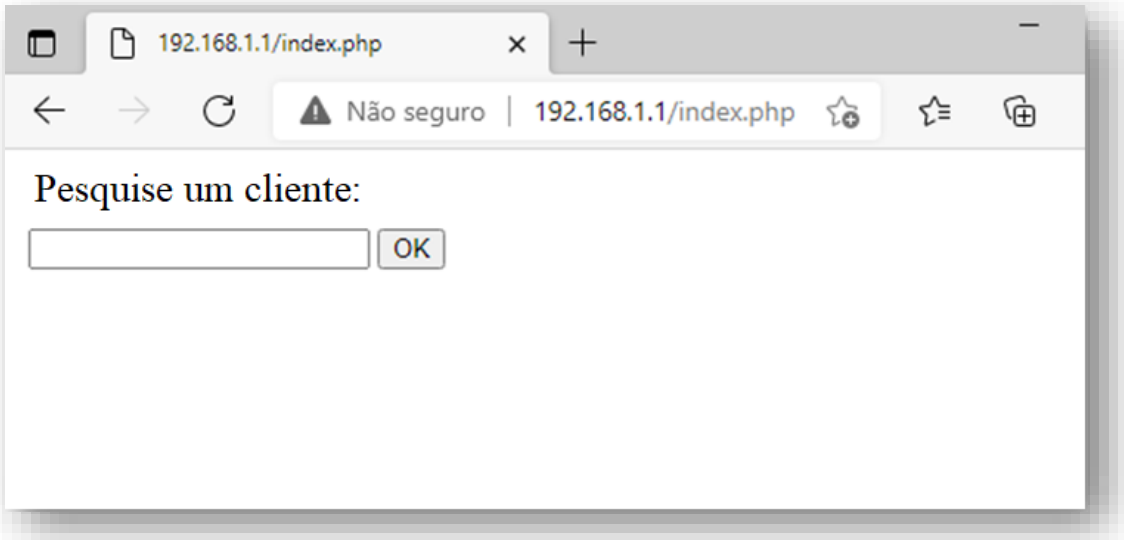
Agora, de volta ao shell do Linux, digite `nano /var/www/html/index.php` e, em seguida, o código a seguir:

```
<html><head> <title>Cadastro</title></head>
<body>
<form action=“POST” action=“consulta.php”>
<h3>Pesquise um cliente: </h3>
<input type=“text” name=“cliente”><input type=“submit” value=“OK”>
</form>
</body>
</html>
```

Para salvá-lo, pressione CTRL+O e, depois, ENTER. Para sair, tecle CTRL+X.

Aqui, apesar da extensão de arquivo `.php`, temos uma página html simples cujo objetivo é disponibilizar uma caixa de texto para que o usuário informe o nome de um cliente e clique no botão de “OK” para realizar a pesquisa.

Esse código, quando visto no navegador, deve se parecer com o mostrado na figura ao lado. Quem faz a “mágica” aqui são os campos definidos como “input”, que permitem ao HTML receber entrada do usuário. A tag `<input type="text" name="cliente">` cria um campo de entrada de texto para que se possa digitar qualquer coisa; o atributo `type` (aqui, definido como "text“) indica que este é um campo de texto; `name` é definido como "cliente” e se tornará uma variável quando o formulário for enviado.



Já a tag `<input type="submit" value="OK">` cria um botão na tela para envio dos dados. O atributo `type` é definido como "submit", o que indica que este botão será usado para enviar o formulário, e `value` é definido como "OK", que é o texto exibido no botão.

Apache Web Server: PHP e MySQL

Agora, vamos criar a página que irá receber o que foi digitado no campo "input" e realizará a pesquisa. Para isso, digite `nano /var/www/html/consulta.php` e, na sequência, o código ao lado.

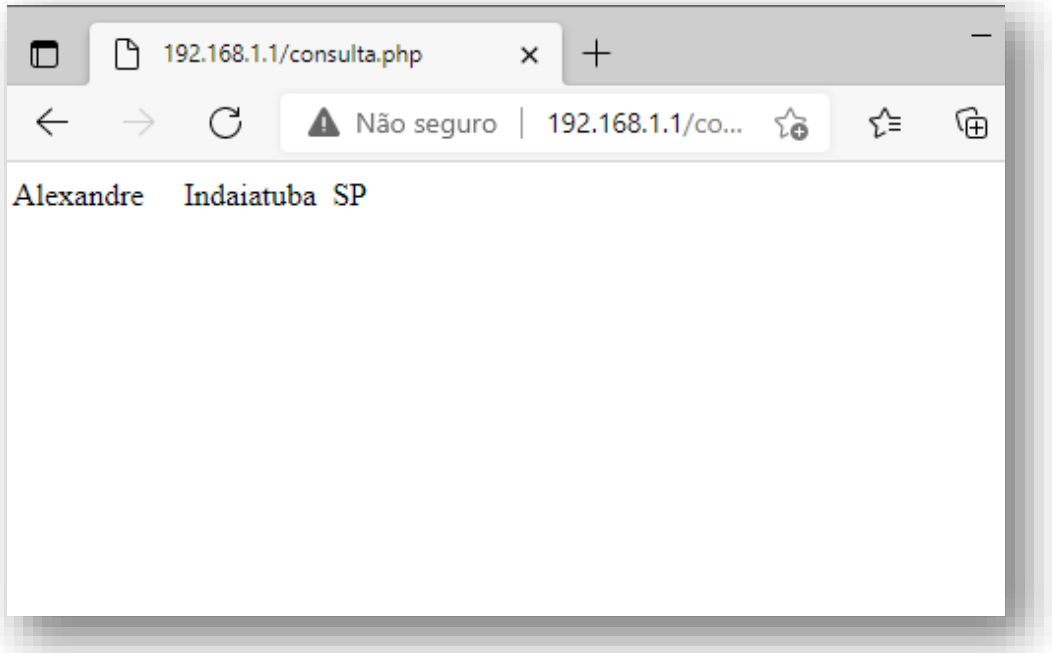
```
<html><head> <title>Consulta</title></head>
<body>
<?php
 $cliente = $_POST['cliente'];
 $c = mysqli_connect("localhost","root","abc123","Alex");
 $r = mysqli_query($c,"select * from Alex.Cliente where nome like '$cliente%' order by nome");
if (mysqli_num_rows($r) >= 1) {
   echo "<h3>Clientes encontrados: </h3>" ;
   while ($linha = mysqli_fetch_array($r)){
     echo $linha['nome'] . "   " . $linha['cidade']." " .$linha['estado'];
   }
 }

?>
</body>
</html>
```

Este código PHP cria recebe na variável `$cliente` o conteúdo que veio da tag "input" da página anterior. Em seguida, faz uma conexão com o MySQL através da função `mysqli_connect()` e pesquisa pelo termo contido na variável $cliente usando a função `mysqli_query()`.

Para testarmos nossa página, na máquina Windows 10, navegue até `192.168.1.1/index.php` e digite, na caixa de texto html, um nome para pesquisa. Por exemplo, digite "Alexandre" (sem as aspas) e pressione "OK".

Se a consulta retornar ao menos um cliente, então seu nome, sua cidade e estado também serão listados na tela. Um exemplo de um possível resultado para essa consulta é mostrado na figura ao lado.



Podemos modificar a sexta linha do nosso código PHP para que seja possível pesquisar por cidades.
Por exemplo, podemos deixa-la assim:

```
$r = mysqli_query($c,"select * from Alex.Cliente where cidade like '$cliente%' order by nome");
```

Bibliografia

- ✓ NEGUS, Christopher. “**Linux Bible** – The Comprehensive Tutorial Resource”. Wiley, 10th Edition. Indianapolis-IN. USA. 2020

- ✓ NEMETH, Evi; SNYDER, Garth; HEIN, Trent. “**Manual Completo do Linux** – Guia do Administrador”. Pearson Education do Brasil, 2ª Edição. São Paulo. SP. 2007

- ✓ RIBEIRO, Uirá. "**Certificação Linux Essentials**: Guia Para o Exame 010-160 – Versão Revisada e Atualizada". eBook Kindle. 2020