professor para esclarecer suas dúvidas (Classroom > Atividades > Dúvidas sobre o conteúdo).

Introdução ao Protocolo HTTP

Prof. MSc. Yuri Maximian Rottner Dirickson

Texto revisto, atualizado e ampliado por:

Prof. Dr. Rafael Will M. de Araujo

Resumo

A Internet é uma rede formada por diversas outras redes de computadores interligadas e é utilizada para o tráfego de todos os tipos de informações. Apesar de não ser o único, o principal protocolo de comunicação utilizado na Internet é o HyperText Transfer Protocol, ou HTTP. Entender a estrutura básica deste protocolo é essencial para todo desenvolvedor que for construir sistemas na web, e para isso é necessário entender o que são requisições e respostas no HTTP, bem como entender o modelo cliente-servidor e saber mais sobre as tecnologias envolvidas de uma maneira mais global.

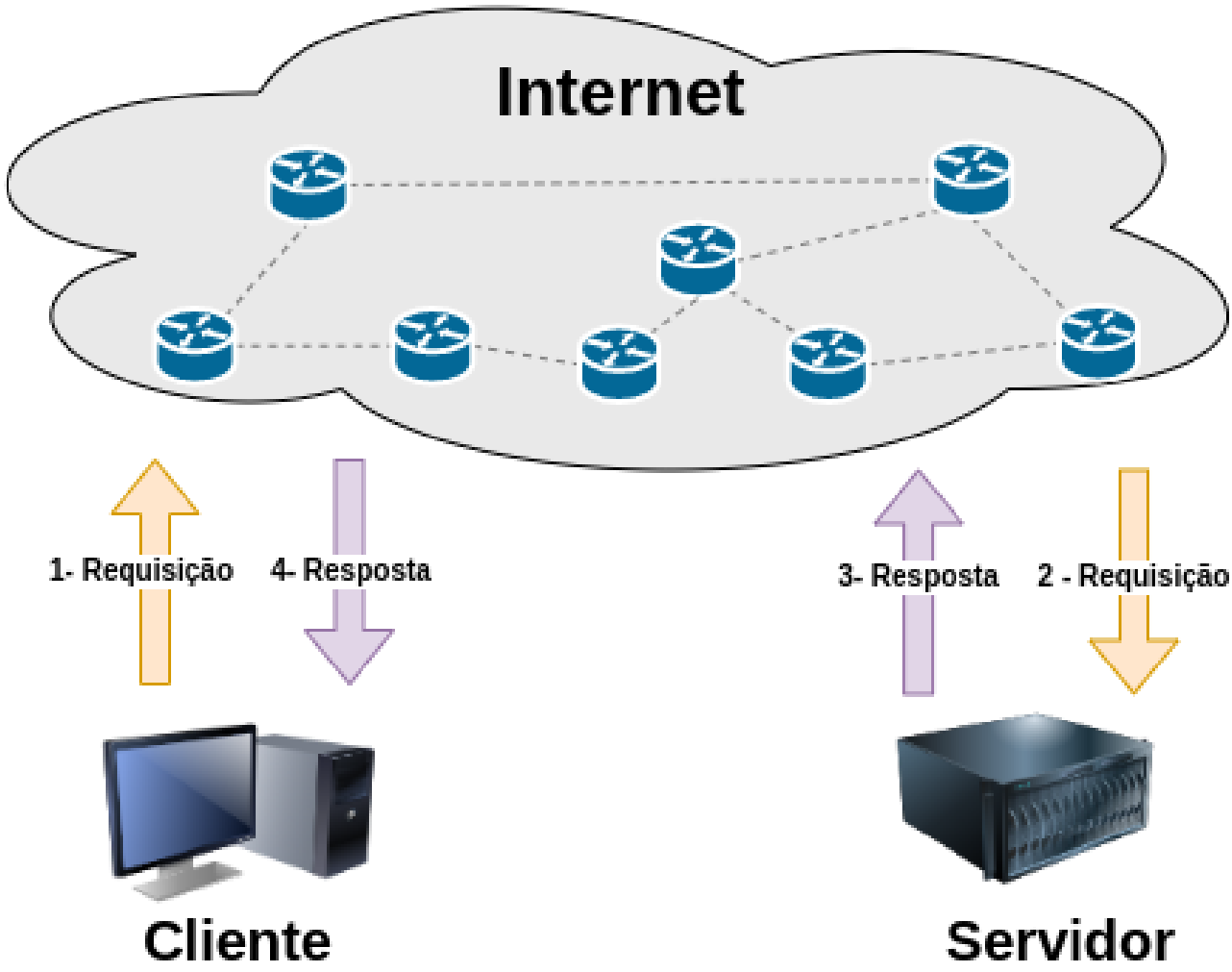
Introdução

A Internet é formada por diversas redes de computadores conectadas entre si, trocando informações de todo tipo a todo instante. De uma simples mensagem de e-mail até o streaming de um vídeo, temos computadores (além de outros dispositivos conectados) trocando mensagens de forma padronizada e organizada.

Mas como essa troca se dá? Quais tecnologias estão envolvidas? Sabemos que tudo começa no *navegador de Internet,* o software que todos os computadores e dispositivos móveis possuem para entrar nos sites da Internet. Entender as partes envolvidas nesse processo, mesmo que de forma mais superficial, nos permite ter uma compreensão muito importante para o desenvolvimento de aplicações na web, seja qual for o foco desta aplicação.

Modelo Cliente-Servidor

Em termos gerais, essa troca de mensagens é feita sempre através de um aparelho que **requisita** (solicita) alguma informação e outro que **responde** a essa solicitação. Essa forma de comunicação, onde um ponto *requisita* uma informação e outro *responde* com ela é representada pelo modelo cliente-servidor [Maia, 2013], conforme pode ser visto na Figura 1.1.



Fonte: do autor, 2021. Atualizado em 2022.

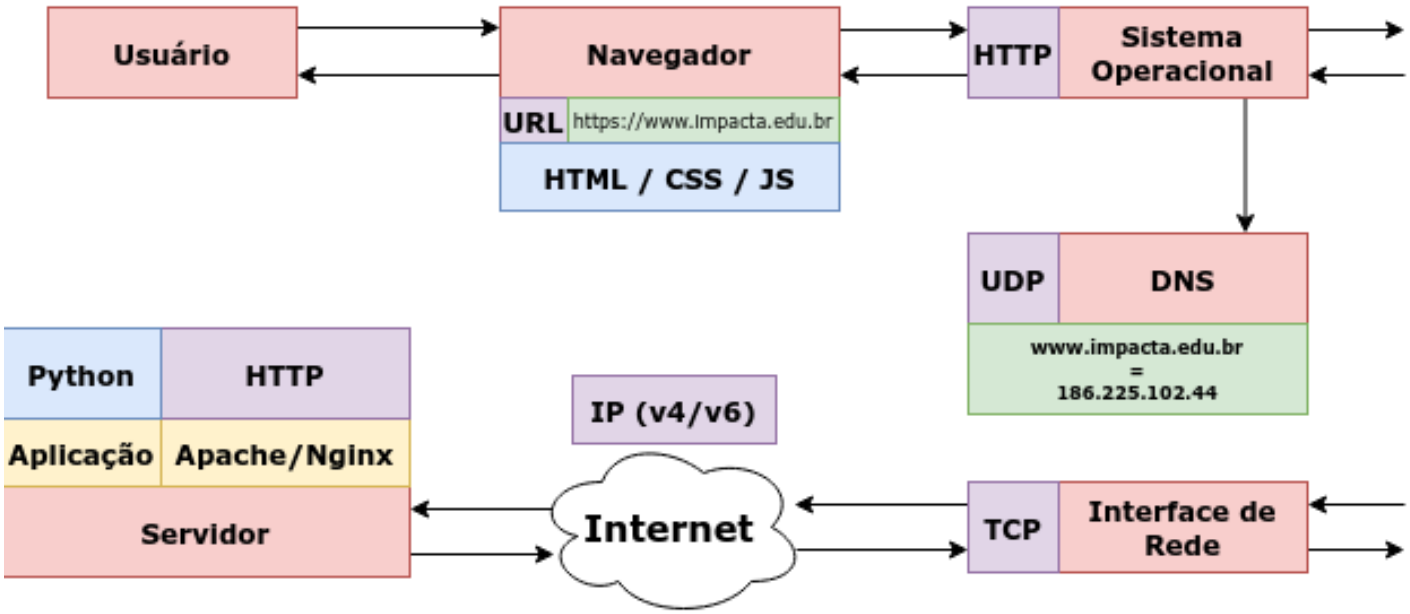
Internet, costumeiramente, o cliente será o **navegador de Internet** e o servidor algum **computador que hospeda páginas** e outros recursos (imagens, vídeos, arquivos Word ou PDF, arquivos com código CSS, etc) que podem ser solicitados pelo navegador (cliente).

Caminho de uma requisição

Ao digitarmos o endereço do site que queremos acessar no navegador, a requisição feita por ele passa por diversos componentes, tecnologias e protocolos antes de podermos ver o resultado. Entender o papel de alguns destes pedaços ajuda a perceber a complexidade envolvida na construção da web e a construir aplicações com foco no funcionamento do sistema completo.

Vamos simplificar o caminho com os conceitos mais importantes para o desenvolvimento web, mostrados no diagrama da Figura 1.2.

Figura 1.2. Caminho da requisição



Fonte: do autor, 2021.

procurando na barra de endereço do navegador, usando o formato da **URL**. Com isso, o navegador empacota o pedido para o **sistema operacional**, usando o protocolo **HTTP** como o pacote da mensagem.

Caso o **navegador** não saiba o **IP** (*Internet Protocol*) do domínio digitado (se for a primeira vez que visita o site, por exemplo), o **sistema operacional** pergunta ao servidor de **DNS** (*Domain Name System*), que é o serviço responsável por traduzir domínios para IPs públicos na Internet. Essa conexão tradicionalmente é feita usando o protocolo **UDP** (*User Datagram Protocol*) que é um protocolo de transporte de mensagens característico por ser mais rápido por não checar se a informação chegou ou não ao destinatário. O **IP** é um protocolo de endereçamento na Internet, formado por números de 32 bits (IPv4) ou 128 bits (IPv6).

Após descobrir o **IP** público do servidor que possui o site pedido, o **sistema operacional** passa a mensagem para a **interface de rede** que irá estabelecer uma conexão com o servidor que possui o **IP** solicitado. A partir daqui, esses dados trafegam pela Internet usando protocolo **TCP** (*Transfer Control Protocol*) que, ao contrário do **UDP**, sempre verifica se a mensagem de fato chegou ao destinatário, fazendo com que seja um protocolo mais robusto (mas com uma perda no desempenho quando comparado com o UDP).

Com o **IP** e a conexão **TCP** estabelecida, a requisição **HTTP** chega ao servidor. Lá, ela precisa ser desempacotada por algum programa que entenda o protocolo **HTTP**, nesse caso chamado de *servidor de aplicação HTTP*, onde os exemplos mais famosos são o Apache, Nginx (pronuncia-se "*engine-X*") e o Microsoft IIS. Na maioria dos casos esses softwares apenas traduzem a mensagem e repassam para alguma outra **aplicação**, que irá executar as ações necessárias para retornar a resposta para o cliente.

Com a resposta já formada, o *servidor de aplicação HTTP* a empacota para o retorno ao cliente (passando por todos os pontos novamente). Chegando no **navegador** (cliente), ele usa seus recursos para interpretar a resposta e mostrar o resultado ao **usuário**. Em geral, esses recursos são escritos usando as linguagens que o **navegador** consegue interpretar, como por exemplo o **HTML, CSS e JavaScript**.

Você sabia?

A Internet é o local com a maior quantidade de conhecimento aberto do mundo, mas nem sempre foi assim. O conceito foi criado pelo Departamento de Defesa dos EUA, no contexto da Guerra Fria ainda (década de 1960). A ideia era transferir documentos secretos entre pontos estratégicos, descentralizando as informações entre vários locais. A subdivisão deste departamento que criou o conceito foi a ARPA (*Advanced Research Projects Agency*), por isso a primeira versão se chamava ARPANET (ESCOLA, s.d.).

A requisição no HTTP é formada basicamente por três grandes partes: a **URL**, o **método HTTP** e os **cabeçalhos de requisição**. Algumas requisições podem possuir um corpo (*payload*), que abriga dados adicionais enviados ao servidor.

Estrutura da URL

A **URL** (*Uniform Resource Locator*) é um sistema de endereçamento de recursos para web. Ela traz informações de localização do servidor na web, do recurso dentro do servidor, e qual protocolo deve ser utilizado para trazer o recurso. A sua estrutura básica e simplificada é:

esquema://domínio:porta/caminho/recurso?query_string#fragmento

O **esquema** mostra qual protocolo será utilizado na transmissão da mensagem. Dentre os mais comuns, temos como exemplo: http, ftp, smtp, etc.

O **domínio** é o conjunto de nomes e identificadores mais amigáveis aos humanos para computadores pela Internet. Um **domínio** em geral corresponde apenas a um endereço **IP**, embora existam técnicas para permitir múltiplos endereços. Tradicionalmente chamamos de **domínio** a parte do endereço que compõe o nome do negócio representado (ex: *impacta*) e os identificadores de negócio e local (*.edu* para educação e *.br* para sites no Brasil). Qualquer prefixo na frente do domínio principal (*impacta.edu.br*) é chamado de ***subdomínio***. Por exemplo: *www.impacta.edu.br* e *account.impacta.edu.br* são subdomínios de *impacta.edu.br*.

A **porta** é um número inteiro que identifica o caminho lógico por onde uma comunicação de rede está passando. Toda comunicação de rede (e processos nos computadores) passam por uma porta lógica. Por padrão, o HTTP sempre utiliza a porta 80 (ou 443 para o HTTPS, uma versão do HTTP que utiliza criptografia). No navegador, sempre que se utiliza o HTTP e o HTTPS não é necessário escrever suas portas padrão, pois nesses casos ele já utilizará as portas 80 ou 443, implicitamente.

O **caminho** (também conhecido como ***path***) identifica onde o recurso está dentro do servidor. Este caminho pode ser tanto físico (indicando pastas no servidor) quanto lógico (caminhos configurados dentro da aplicação para encontrar um recurso).

O **recurso** é o arquivo que se deseja acessar. Nem sempre essa parte da URL vai existir, pois alguns recursos são dinâmicos, ou seja, são gerados em tempo de execução da requisição (ex: lista de chamada de uma turma no dia). Quando o recurso for estático (ex: imagens) essa parte existirá.

A **query string** é uma forma de passar dados a mais para o servidor, como forma de ajudar na busca de recursos mais específicos. É bastante usada nos buscadores e possui o formato **atributo=valor**. Essa parte da URL é análoga ao envio de valores através de parâmetros para uma função em uma linguagem de programação, isto é, podemos enviar valores quaisquer para o servidor através da URL

geral um *id* (identificador único) de algum elemento HTML para que o navegador já entre visualizando o elemento específico.

Métodos HTTP

Os métodos HTTP (ou verbos HTTP) são informações que servem para indicar uma intenção do que pode ocorrer no servidor ao enviar a requisição. Alguns métodos são usados apenas para obtenção de recursos (ex: GET), outros são específicos para alteração do estado da aplicação (ex: PUT, DELETE, POST) (HTTP request methods, s.d.).

Dentre os métodos HTTP mais comuns, destacamos:

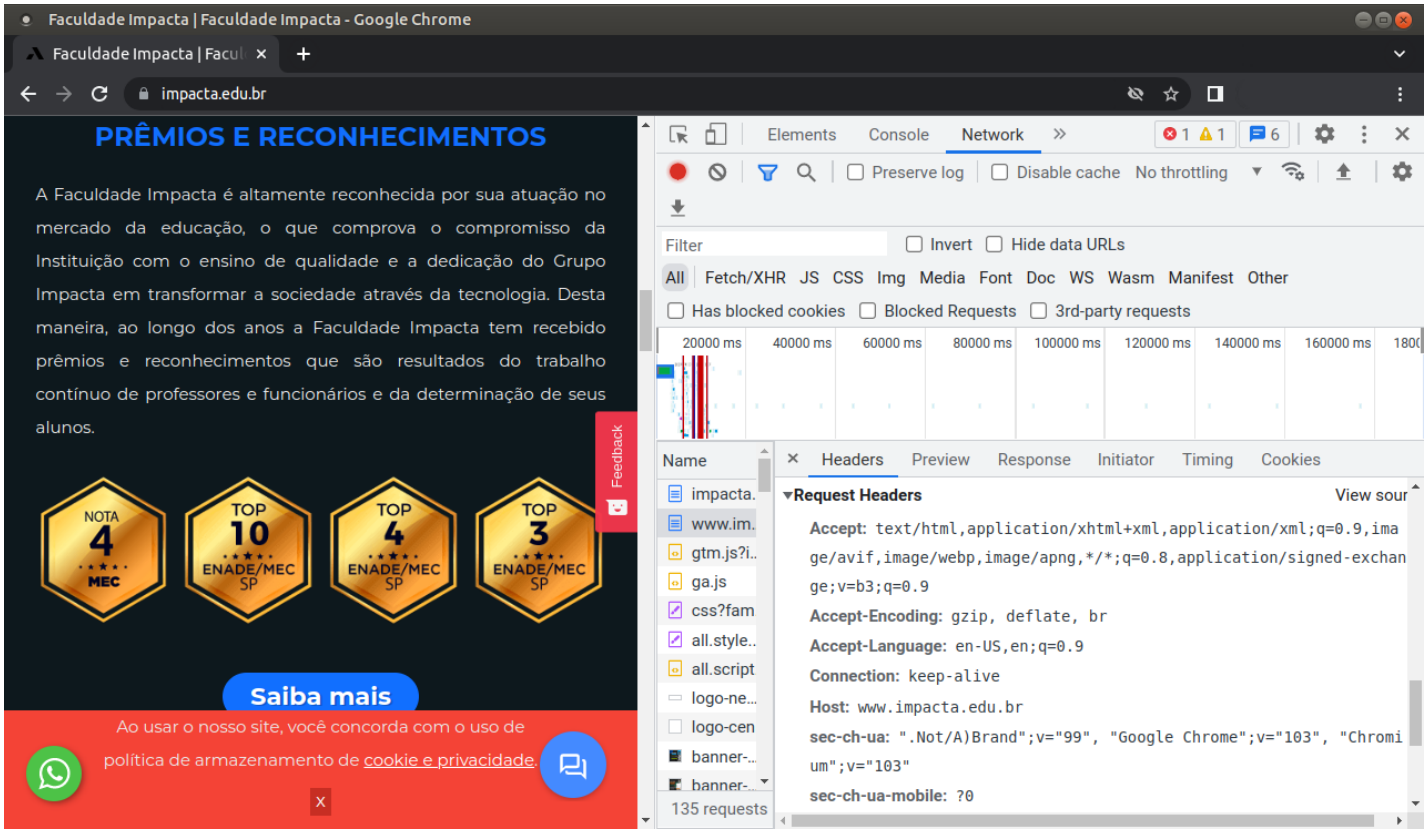
- **GET**: usado para obter um recurso qualquer do servidor;
- **POST**: envia dados para serem processados pelo servidor, em geral para criar ou alterar um recurso;
- **DELETE**: remove um determinado recurso do servidor;
- **PUT**: atualiza todas as informações de um recurso no servidor;
- **PATCH**: atualiza parte das informações de um recurso no servidor;

Em nosso curso focaremos apenas nos métodos **GET** e **POST**. Entendê-los será especialmente útil para enviar informações de formulários (entrada de dados do usuário) para o servidor. Detalharemos esses métodos no conteúdo de formulários em HTML.

Cabeçalhos

Os cabeçalhos (*headers*) de requisição são uma série de informações necessárias para a comunicação entre o cliente e o servidor. Essas informações trafegam sempre no formato **chave: valor**, onde os valores são sempre tratados como dados textuais.

Algumas dessas informações são restritas ao navegador, como por exemplo o *user-agent* e os *cookies*. Também é possível criar nossos próprios cabeçalhos com bibliotecas JavaScript para passar informações específicas de nossa aplicação. A Figura 1.3 mostra alguns exemplos de informações contidas no cabeçalho de uma requisição. Essa mesma tela pode ser acessada no menu "*Ferramentas do Desenvolvedor*", ou usando o atalho *F12* (ou também *Control+Shift+I*) no navegador Google Chrome.



Fonte: do autor, 2022.

domínio onde eles são aplicados. Toda requisição feita no mesmo domínio carrega todos os cookies registrados nele. O uso dos cookies é uma das maneiras mais clássicas de criar uma sessão de usuário na web. Estudaremos a relação entre sessões de usuário e cookies mais adiante em nosso curso.

Estrutura de uma resposta

As respostas HTTP possuem sempre três partes: código de status, cabeçalhos de resposta e corpo da resposta.

Códigos de status

Os códigos de status indicam se a requisição foi sucedida ou não, e o que aconteceu com ela em ambos os casos. Eles são divididos em números que podem ir de 100 a 599, mas divididos em faixas de 100 de acordo com o seu objetivo. Alguns dos códigos de status mais famosos são o 200 (***OK***), 404 (***Not Found***), 400 (***Bad Request***) e 500 (***Internal Server Error***).

As faixas dos códigos são dividas em (HTTP response status codes, s.d.):

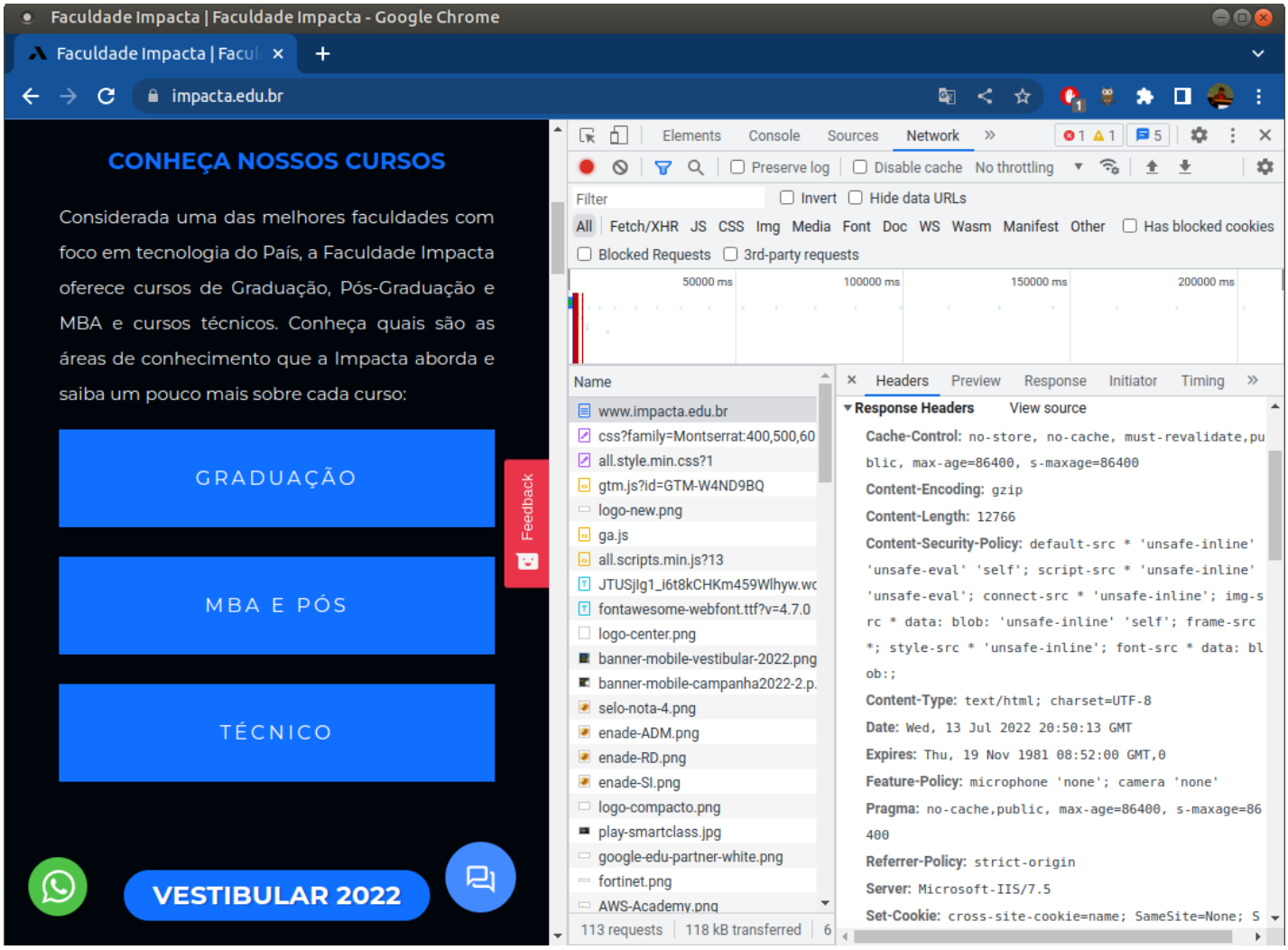
- 100 até 199: códigos informativos. São usados na comunicação do cliente com o servidor com informações intermediárias sobre a comunicação de rede. Requisições com números nessa faixa não são muito comuns no dia a dia;
- 200 até 299: códigos de sucesso. Indicam que a requisição foi bem sucedida no servidor. O sucesso de cada requisição depende muito do método sendo usado;
- 300 até 399: códigos de redirecionamento. Indicam que o navegador precisou tomar uma ação a mais para completar a requisição, como por exemplo, o redirecionamento de páginas, uso do cache, entre outros.
- 400 até 499: códigos de erro do cliente. Mostram que alguma coisa deu errada no cliente (navegador). Situações comuns são: algum problema na comunicação estabelecida pelo navegador, ou falta alguma informação ou condição para a requisição ser concluída (ex: não está logado).
- 500 até 599: códigos de erro do servidor. Indicam que o erro aconteceu no lado do servidor, podendo ser problemas na rede interna ou uma inconsistência na aplicação rodando no servidor.

Cabeçalhos de resposta

Da mesma forma que na requisição, a resposta também possui seus próprios cabeçalhos, no mesmo formato.

Muitos destes cabeçalhos são criados pelo servidor, baseado na resposta. Por exemplo, o cabeçalho **Date** mostra a data em que a resposta foi gerada e o **Server** indica o software rodando no servidor. Outros cabeçalhos podem ser criados pela aplicação de acordo com a necessidade dela. Já os cookies são definidos no navegador através do cabeçalho de resposta **Set-Cookie**. A Figura 1.4 mostra algumas informações disponíveis no cabeçalho

Figura 1.4. Visualização do cabeçalho de uma requisição usando o navegador Google Chrome 2



Fonte: do autor, 2022.

O corpo da resposta contempla o recurso que foi requisitado. Esse recurso pode ser qualquer coisa, por exemplo: um documento HTML, uma imagem, um vídeo, um arquivo xml, um arquivo PDF, etc.

Referências

ESCOLA, Equipe Brasil. **Como surgiu a internet?** Brasil Escola, s.d. Disponível em: <https://brasilescola.uol.com.br/curiosidades/como-surgiu-a-internet.htm>. Acesso em: 22 jun. de 2022.

HTTP request methods. MDN Web Docs, s.d. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>. Acesso em: 28 jun. 2022.

HTTP response status codes. MDN Web Docs, s.d. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>. Acesso em: 17 jun. 2022.

MAIA, Luiz Paulo. **Arquitetura de redes de computadores**. 2. ed. São Paulo: LTC; Grupo Gen, 2013.