Programação de Sistemas para Internet

Prof. Diego Cirilo

Aula 02: Conceitos Básicos

Internet

- Programação de Sistemas para **INTERNET**;
- *Rede mundial de computadores*.

Histórico

- A internet começou a ser criada com o projeto do governo americano chamado **ARPANET**;
- Tinha o objetivo de interligar Universidades e Instituições de pesquisa e militares.

Histórico

- Em 1980 várias outras universidades foram incorporadas;
- Em 1985 a NSF (*National Science Foundation*) interligou seus computadores em rede;
- Em 1986 a NSF se junta à ARPANET e essa rede passa a ser chamada de *Internet*;
- Em 1989 a FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) e a LNCC (Laboratório Nacional de Computação Científica) se ligam a Internet;
- No mesmo ano foi criada a RNP (Rede Nacional de Pesquisa).

Histórico

- Em 1991 o cientista Tim Berners-Lee do CERN cria o WWW (*World Wide Web*);
  - Antes só existia email, FTP e Telnet;
  - Serviços;
- Depois do WWW foi criado o Mosaic, o primeiro navegador para *web* que renderizava imagens;
- Em 1993 a internet é aberta para exploração comercial nos EUA, e o mesmo ocorre um ano depois no Brasil.

WWW

- Sistema de informação que utiliza a internet como meio de transmissão;
- Páginas de multimídia interligadas através de *hyperlinks*;
- Arquitetura Cliente/Servidor.

www

- Conceitos:
  - HTTP - Protocolo de transmissão de hypertexto (*Hypertext Transmission Protocol*):
    - Realiza a comunicação com um servidor Web através de requisições e respostas bem definidas.
  - URL - Sistema de endereços:
    - `http://www.ifrn.edu.br/`;
  - HTML - Linguagem de marcação de hypertexto (*Hypertext Markup Language*):
    - *Hyper*?
    - Formatação para texto, inserção de imagens e *links*.

URL

- *Uniform Resource Locator*
- Protocolo/Domínio/Recurso
  - Ex. *www.ifrn.edu.br*
  - Primeiro nível:
    - .br indica o país. Ex. *.it*, *.pt*, *.mx*
    - Quando não possui o endereço é americano ou internacional.
  - Segundo nível:
    - Tipo de instiuição. Ex. *.edu*, *.com*, *.gov*
  - Terceiro:
    - Nome do site
  - Quarto:
    - Tipo de serviço. Ex. *www*, *mail*, *suap*, etc.

URL

- https://github.com/dvcirilo/psi-ifrn
- https://www.youtube.com/watch?v=GggUi3KQpLc
- https://suap.ifrn.edu.br
- https://g1.globo.com/rn

Protocolo TCP/IP

- *Transmission Control Protocol/Internet Protocol*;
- Conjunto de regras e padrões que permitem a comunicação entre computadores na rede;
- Pode ser organizado em quatro camadas:
  - Enlace: controla o hardware e o meio de comunicação;
  - Internet: encontra o melhor caminho através da rede;
  - Transporte: mantém uma comunicação confiável entre os dispositivos e executa a correção de erros;
  - Aplicação: protocolos de comunicação entre serviços e cliente-serviço.
- O escopo dessa disciplina é a camada de aplicação.

IP

- O IP (*Internet Protocol*) é responsável pelo encaminhamento de dados entre as máquinas na rede;
- As máquinas são identificadas por um endereço IP, ex: 122.220.98.4;
- Além do IP também existem *portas*, que podem ser acessadas por diferentes serviços;
- As portas são representadas com `:` após o IP, ex. `127.0.0.1:2046`.

DNS

- DNS (*Domain Name Server*) é responsável por traduzir as URLs para os IPs dos servidores associados.
- Conveniência.
- Para obter uma URL é necessário pagar para um *registrar*. Ex. GoDaddy, Registro.br

Navegador

- Também chamado de *browser*:
  - Aplicativo responsável pelo acesso às páginas web;
  - Transforma/*Renderiza* códigos HTML/JS/CSS em páginas interativas;
  - Ex. Google Chrome, Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Brave, Chromium...
- Faz requisições HTTP/HTTPS e renderiza os dados recebidos;
- É o cliente.

Resumo (simplificado)

- O navegador (cliente) faz uma requisição HTTP direcionada a uma URL:PORTA;
- O servidor DNS traduz a URL para um IP e direciona a requisição para o IP do servidor;
- O servidor recebe a requisição e responde de volta para o cliente;
- Caso a resposta seja positiva, o navegador renderiza o HTML recebido.

Protocolo HTTP

- *Hyper text transfer protocol*
- Camada de aplicação
- Baseado no modelo cliente-servidor
- Padrão de mensagens de requisição e respostas
- Porta 80 (ou 443 para HTTPS)
- Cada mensagem é composta por Método/Cabeçalho/Corpo
- Referência

Métodos HTTP

Método	Descrição
GET	Recebe um recurso existente
POST	Cria um novo recurso
PUT	Atualiza um recurso existente
PATCH	Atualiza parcialmente um recurso existente
DELETE	Remove um recurso

Códigos de status

Faixa	Categoria
2xx	Sucesso
3xx	Redirecionamento
4xx	Erro de cliente
5xx	Erro de servidor

Códigos de status

Código	Significado	Descrição
200	OK	The requested action was successful.
201	Created	A new resource was created.
202	Accepted	The request was received, but no modification has been made yet.
204	No Content	The request was successful, but the response has no content.
400	Bad Request	The request was malformed.
401	Unauthorized	The client is not authorized to perform the requested action.
404	Not Found	The requested resource was not found.
415	Unsupported Media Type	The request data format is not supported by the server.
422	Unprocessable Entity	The request data was properly formatted but contained invalid or missing data.
500	Internal Server Error	The server threw an error when processing the request.

Cabeçalho HTTP

Dado	Descrição
Accept	O tipo de conteúdo que o cliente aceita
Content-Type	O tipo de conteúdo que o servidor retorna
User-Agent	Que software o cliente está usando para comunicar com o servidor
Server	Que software o servidor usa para comunicar com o cliente
Authentication	Quem chama a API que quais suas credenciais

Programação de Sistemas para Internet

Sites Estáticos

- Primeiro tipo de site existente;
- O que foi feito nas disciplinas anteriores;
- O conteúdo do site é definido no momento da escrita do HTML;
- O usuário não é capaz de armazenar/modificar dados no sistema;
- É o suficiente?

Sites Dinâmicos

- A página é construída em `tempo de execução`;
- O conteúdo pode estar armazenado em um banco de dados;
- Permitem a apresentação de dados e funcionalidades mais complexas;
- O processamento é realizado no servidor;
- "Web 2.0";
- Exemplos?

Sites Dinâmicos

- CGI (*Common Gateway Interface*):
  - Processamento de *forms* e acesso a banco de dados);
  - Executa um programa em uma linguagem qualquer que processa a requisição e retorna o HTML formado.
- PHP (*Personal Home Page*, depois *PHP: Hypertext Preprocessor*):
  - Permite descrever a lógica dentro de um arquivo HTML que é pré-processado pelo servidor antes de ser disponibilizado para o cliente;
  - Linguagem muito popular até hoje e inicialmente desenvolvida para *web*.
- ASP, Java, Ruby, Python, JS, etc;
- Frameworks.

Sistema Web

- Aplicação acessada pela Web;
- Aplicação Desktop;
- Aplicação Mobile;
- Vantagens/Desvantagens?

Front-end e Back-end e Full-Stack

- *Front-end*: parte visual, interface com o usuário;
  - Tecnologias: HTML, CSS, JavaScript.
- *Back-end*: processamento, armazenamento de dados, segurança, páginas dinâmicas;
  - Tecnologias: PHP, Java, Python, Ruby, JavaScript, SQL, etc.
- *Full-stack*: tudo;
- E o Designer?

Referências

- Dye, M. A., McDonald, R., Rufi, A. W. (2007). Network Fundamentals, CCNA Exploration Companion Guide (2nd Edition) (Companion Guide). Reino Unido: Cisco Press.
- https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

Dúvidas?