HTTP: Entendendo a web por baixo dos panos

- Esse curso capacita a:
  - Tornar-se um programador web completo;
  - Entender os detalhes do protocolo HTTP;
  - Saber quando usar GET e POST;
  - Estudar sobre segurança na web e o HTTPS;
  - A web stateful e a web stateless;
  - Conhecer as melhorias do HTTP/2;

Aulas:

1. **O que é o HTTP?**
   - A arquitetura Cliente-Servidor;
   - HTTP é um protocolo que define as regras de comunicação entre cliente e servidor na internet, é o protocolo mais importante da Internet;

2. **A web segura – HTTPs:**
   - Um certificado digital prova uma identidade para um site, onde temos informações sobre o seu domínio e a data de expiração desse certificado. Além disso, o certificado ainda guarda a chave pública que é utilizada para criptografar (cifrar) os dados que são trafegados entre cliente e servidor;
     - Por padrão, os dados são trafegados como texto puro na web;
     - Apenas com HTTPS a Web é segura;
     - O protocolo HTTPS nada mais é do que o protocolo HTTP mais uma camada adicional de segurança, a TLS/SSL;
     - O tipo de criptografia de chave pública/chave privada;
     - O que são os certificados digitais;
     - Certificados possuem identidade e validade;
     - As chaves públicas estão no certificado, a chave privada fica apenas no servidor;
     - O que é uma autoridade certificadora;
     - O navegador utiliza a chave pública para criptografar os dados;

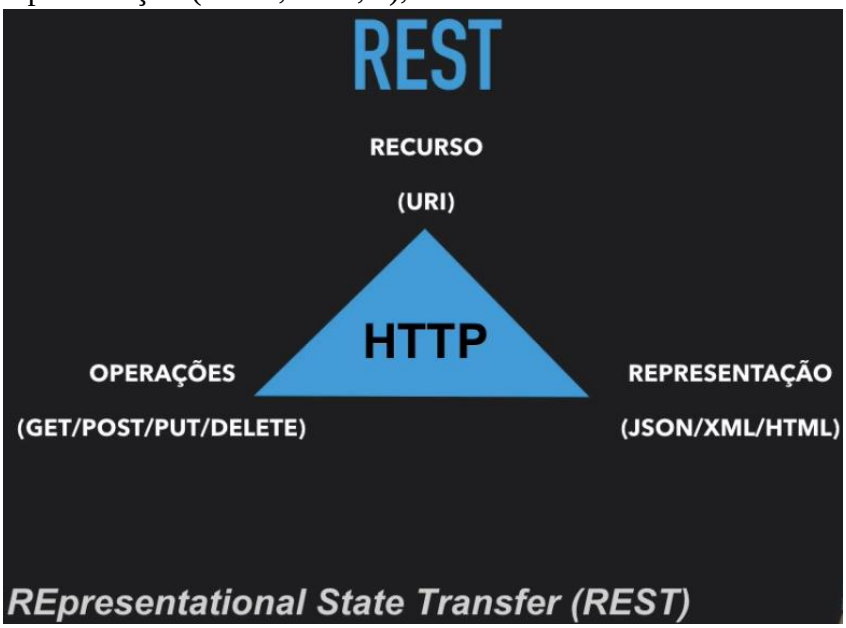
3. **Endereços sob seu domínio:**
   - URL são os endereços da Web;
   - Uma URL começa com o protocolo (por exemplo https://) seguido pelo domínio (www.alura.com.br);
   - Depois do domínio pode vir a porta, se não for definida é utilizada a porta padrão desse protocolo;
   - Após o domínio:porta, é especificado o caminho para um recurso (/course/introducao-html-css);
   - Um recurso é algo concreto na aplicação que queremos acessar;

4. **O cliente pede e o servidor responde:**
    - O protocolo HTTP segue o modelo Requisição-Resposta;
    - Sempre o cliente inicia a comunicação;
    - Uma requisição precisa ter todas as informações para o servidor gerar a resposta;
    - HTTP é **stateless**, não mantém informações entre requisições;
    - As plataformas de desenvolvimento usam sessões para guardar informações entre requisições;
    - Uma sessão HTTP nada mais é que um tempo que o cliente permanece ativo no sistema!

5. **Depurando a requisição HTTP:**
    - Console de depuração;
    - Método HTTP – GET (“Quero receber algo”);
    - Cabeçalhos de resposta;
    - Códigos de resposta (Status Code):
        - 2xx – Successful responses;
        - 3xx – Redirection messages;
        - 4xx – Client error responses;
        - 5xx – Server error responses;

6. **Parâmetros da requisição:**
    - Methods (Verbos):
        - GET - Receber dados (params na URL);
        - POST - Submeter dados (params no corpo da requisição);
        - DELETE - Remover um recurso;
        - PUT/PATCH - Atualizar um recurso;
    - Quando enviamos parâmetros na URL, devemos iniciar pelo ‘?’, o nome do parâmetro e um ‘=’, para separar o nome do parâmetro do seu valor:
      ?nome_do_parametro=seu_valor&nome_do_outro_param=valor

7. **Serviços web REST:**
    - REST é um padrão arquitetural para comunicações entre aplicações que se aproveita da estrutura que o HTTP proporciona;
    - Recursos são definidos via URI em conjunto com as operações com os métodos do HTTP (GET/POST/PUT/DELETE);
    - Cabeçalhos(Accept/Content-Type) são usados para especificar as representações(JSON,XML,...);



8. **HTTP2 - Por uma web mais eficiente:**
    - Apresenta Headers binários e comprimidos (**HPACK**);
    - **GZIP** padrão na resposta;
    - Multiplexing (Requisição e respostas são paralelas);
    - **Headers Stateful**: permitem que apenas os cabeçalhos que mudem sejam enviados a cada requisição, economizando muita banda que seria gasta em cabeçalhos repetidos;
    - **Server Push**: O servidor pode empurrar para o clientes certos recursos antes mesmo de serem solicitados, pois ele consegue analisar o HTML e ver o que mais é preciso para carregar a página fazendo com que não seja necessário gastar tempo pedindo todos os outros recursos;