Nome: Douglas Batista de Souza Junior / Matricula: 1201900167

Em relação ao HTTP descreva e entenda os seguintes conceitos:

1. Qual a diferença entre HTTP e HTTPS?

   O Protocolo **HTTP** é um protocolo de transferência de hipertexto padrão é a forma que os dados são transferidos do seu computador ou outros aparelhos com os servidores, é uma forma de comunicação, o protocolo **HTTP** se baseia em textos, esse protocolo não tem um método de segurança e por esse motivo esses “textos” podem sofrer interceptações ou interrupção da transferência de dados do Banco de dados até o cliente. Já o Protocolo **HTTPS** é um protocolo que funciona igual ao **HTTP** através de textos, porém camadas de segurança como **SSL** (Secure Sockets Layer) ou **TLS** (Transfer Layer Security) foram adicionadas ao **HTTPS** essas camadas que nada mais são que autenticação, integridade isso garante que a “mensagem” ou requisição vai chegar ao seu destino sem alterações, e com garantia de um servidor seguro.

2. Qual o formato de uma requisição HTTP?

   O formato de requisição do **HTTP** funciona da seguinte maneira, o cliente abre um “**socket**” para conversar com o servidor e após isso realiza os **Requests** de acordo com o **Requests** (Requisições) o servidor envia as **Responses** (Respostas). Essa requisição funciona em três etapas: Request line, Headers e Body.

   - Request line:
     Métodos das requisições **GET** baixa informações, **POST** envia informações dentre outros comandos como **DELETE**, **PUT** e etc.

   - Headers:
     Utilizado para informações adicionais, alguns exemplos de **headers** são **Accept-Encoding** e **User-Agent** e seu funcionamento é baseado em **chave:valor**.

   - Body:
     Onde contém os dados associados a requisição, são mais usados nos métodos **PUT** E **POST**, mas também pode ser usado no **GET**, pode se encontrar nos formatos **XML**, **JSON** por exemplo.

3. Qual o formato de uma resposta HTTP?

   O formato de resposta **HTTP** segue a estrutura dos seguintes campos:
   - (**Status-Line**) linha inicial;
   - (**Response header**) linhas de cabeçalhos;
   - Linha em branco obrigatória;
   - Corpo da mensagem.

   O retorno da linha inicial, possui três etapas:
   - (**HTTP-Version**) Mostra a versão do protocolo **HTTP**;
   - (**Status-Code**) um código de status (resultado da requisição);
   - (**Reason-Phrase**) justificativa descrevendo o que diz o código do status.

4. Quando um servidor não encontra um recurso, quais os principais códigos de status

que existem? Por exemplo 404? o que significa?

Existem diversos códigos de erro ou sucesso exemplos são: 1XX Informacional, 2XX Sucesso, 3XX Redirecionamento, 4XX Erro do Cliente e 5XX Erro do Servidor, mas em questão de erros do servidor esses são os mais falados:

4XX Erro do Cliente

401 – Não autorizado, possivelmente Falta de autorização (Login);

403 – Proibido;

404 – Não encontrado página ou arquivo que não existe no servidor, arquivo apagado.

5XX Erro do Servidor.

500 – Erro interno do servidor.

503 – Serviço indisponível, pode ser um erro temporário, uma manutenção ou quantidade de requisições/acessos que pode derrubar o servidor.

ERRO 404: Não encontrado página ou arquivo que não existe no servidor, arquivo apagado.

5. Quais as principais diferenças do HTTP v1 para o HTTP v2?

As principais diferenças é que **HTTP 1** usa protocolo texto, já o **HTTP 2** utiliza protocolo Binário, algumas diferenças também são:

- **Priorização de requests:**

  No **HTTP 2** podemos designar nos requests quais são mais importantes, o navegador consegue dar prioridade a um arquivo **CSS** e uma prioridade menor ao **JS**.

- **Headers que mudam são reenviados:**

  No **HTTP 1** os headers são enviados em plain text, No **HTTP 2** os headers são binários e comprimidos, reduzindo o volume de dados.

- **Paralelização de requests:**

  HTTP 1 é um protocolo sequencial, 1 request por vez, No **HTTP 2** as requisições e respostas são paralelas em uma única conexão, o chamado **multiplexing**.

- **Compressão automática:**

  No **HTTP 1** usamos o **GZIP** para comprimir as informações que enviamos nas respostas, No **HTTP 2 GZIP** é algo padrão e obrigatório.

- **Server Push:**

  HTTP 1 faz inline de recursos, visando a renderização inicial mais rápida, o problema é que descartamos o **cache** do navegador, no **HTTP 2** o servidor envia recursos para o navegador sem ele ter requisitado, assim quando for preciso vai ter em cache os recursos.

- **Segurança:**

  HTTP 2 somente aceita conexões com **HTTPS**, ou seja, com **SSL** ou **TSL**.