**ATIVIDADE 2 – MÓDULO WEB SERVICES**

Arthur Silva

Softex Recife

Faça um projeto explicando como funcionam os métodos HTTP com exemplos, sua utilização e em qual momento cada um dos métodos deve ser utilizado. Além disso, descreva o que é WSDL, seu significado e sua função.

*“O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso. Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como HTTP Verbs (Verbos HTTP)”.* Descrição retirada do site: developer.mozilla.org.

Existem 9 métodos HTTP, são eles: GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, CONNECT, TRACE

Para esta atividade irei propor um cenário de uma biblioteca de livros simplificada, apenas para objetivo de exemplificação dos métodos.

**Cenário**: em uma biblioteca um funcionário é capaz de consultar livros, através do nome do livro, do autor, editora, editar os dados de livros, saber em que prateleira esse livro se encontra, etc. Através de um sistema online. Sem ter conhecimento, este funcionário já faz uso de diversos métodos.

Método **GET**: Método que serve para consultar dados. Quando o funcionário faz uma pesquisa sobre um livro X, o sistema utiliza o método GET (Pegue), para retornar a informação solicitada. Este método também é utilizado ao acessar qualquer página web, o navegador faz um GET no servidor da aplicação, e retorna a página solicitada.

Exemplo:

GET | https://biblioteca.com/livros/x

{

Id: 1657

Nome: “X”,

Ano: 2001,

Autor: xxxxx

}

Método **HEAD**: solicita uma resposta de forma idêntica ao método GET, a diferença é que apenas o cabeçalho da requisição é retornado.

Exemplo:

HEAD | https://biblioteca.com/livros/x

HTTP/2 200

Content-type: application/json

Cache-control: no-cache

Método **POST**: Método que serve para enviar dados para a aplicação, como realizar um login ou cadastro de novas informações no servidor, como um novo livro.

Exemplo:

Enviando para o servidor um novo livro para ser cadastrado, utilizando o método POST na url.

POST | https://biblioteca.com/livros

{

Nome: “Y”,

Ano: 2002,

Autor: zzzz

}

Resposta:

Como o método POST é um método para enviar dados, esperasse que retorne uma resposta, como no exemplo é um cadastro de um novo livro, a resposta foi o livro cadastrado, percebesse que a resposta retornou o livro com um “ID”, que é o identificador do livro na base de dados.

{

Id: 2498,

Nome:” Y”,

Ano:202,

Autor: zzzz

}

O programador pode escolher retornar apenas o status de sucesso que é o código 201 HTTP, mas não irei abordar códigos nessa atividade.

Método **PUT**: Método que serve para atualizar algum livro na biblioteca.

Exemplo: suponhamos que o indivíduo responsável por cadastrar o livro Y, digitou o autor com algum erro. Mas o programador disponibilizou apenas o método PUT para atualizações.

PUT | https://biblioteca.com/livros/2498

*Note que ao final da url, está se referindo ao ID do livro Y. Por conversão, o método PUT exige que todos os dados sejam reenviados.*

Envio:

{

Nome: “Y”,

Ano: 2002,

Autor: yyyyy *<- único dado que necessitava de atualização*

}

Resposta:

{

Id: 2498,

Nome: “Y”,

Ano: 2002,

Autor: yyyyy

}

Método **PATCH**: Outro método que serve para atualização, a diferença é que este é utilizado para fazer uma atualização parcial nos dados, diferente do método PUT.

Exemplo: o exemplo abaixo é de como seria enviado a atualização anterior, se fosse utilizado o método PATCH

PATCH | https://biblioteca.com/livros/2498

*Note que ao final da url, está se referindo ao ID do livro Y.*

Envio:

{

Autor: yyyyy *<- único dado que necessitava de atualização*

}

Resposta:

{

Id: 2498,

Nome: “Y”,

Ano: 2002,

Autor: yyyyy

}

Método **DELETE**: como o próprio nome sugere, este método é responsável por deletar livros de nossa biblioteca.

Exemplo: por um motivo adverso, a biblioteca não terá mais o livro Y na sua estante e o livro precisa ser deletado do sistema.

DELETE | https://biblioteca.com/livros/2498

*Note que ao final da url, está se referindo ao ID do livro Y.*

Neste método o envio de informação é opcional, da mesma forma que a resposta pode conter apenas o código de status.

Método **OPTION**: este método é utilizado para consultar as opções permitidas de execução, como quais métodos podem ser executados.

Exemplo:

OPTION | https://biblioteca.com/

A resposta ao executar este método é um cabeçalho, semelhante ao HEAD, porém com informações apenas de métodos que podem ser utilizados.

Resposta:

HTTP/2 200

Allow: GET, POST, PUT, PATCH, DELETE, OPTIONS

Access-Control-Allow-Origin: \*

Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS

Access-Control-Allow-headers: Content-type

Método **CONNECT:** este método é responsável por estabelecer uma conexão ponta a ponta entre dois computadores, normalmente cliente - servidor, através de proxy/firewall. Funciona como uma espécie de túnel por onde as comunicações serão feitas.

Exemplo:

CONNECT | www.biblioteca.com

Resposta:

HTTP/2 200 CONNECTION ESTABLISHED

DATE: DD/MM/AAAA HH-MM-SS GMT

SERVER: APACHE/3.1.4(WIN32)

Método **TRACE:** este método é usado para a realização de um teste de loopback enviando uma mensagem até o destino, e retornando um cabeçalho que será útil para debugs.

Exemplo:

TRACE | www.biblioteca.com

Resposta:

HTTP/2 200 CONNECTION ESTABLISHED

DATE: DD/MM/AAAA HH-MM-SS GMT

SERVER: APACHE/3.1.4(WIN32)

CONNECTION: CLOSE