



Universidade Federal de Uberlândia
Faculdade de Computação – Prof. Daniel A. Furtado
11º Trabalho de Programação para Internet
Técnica Ajax – Requisições HTTP Assíncronas com o XMLHttpRequest

Instruções Gerais

- Esta atividade deve ser realizada **individualmente**;
- Tecnologias permitidas: HTML5, CSS, JavaScript, Bootstrap, PHP, MySQL, XMLHttpRequest;
- Sintaxe da XHTML como `` ou `
` não é permitida (anulará o trabalho);
- O website deve ser hospedado e disponibilizado online, conforme orientações disponíveis no final deste documento;
- Ao construir o website, utilize dados fictícios. **Jamais utilize** dados pessoais como seu nome, CPF, endereço, e-mail etc.;
- Esteja atento às **observações sobre plágio** apresentadas no final deste documento;
- Trabalhos com implementações utilizando trechos de códigos retirados de sites da Internet ou de trabalhos de semestres anteriores serão anulados;
- As páginas web não devem conter qualquer conteúdo de caráter imoral, desrespeitoso, pornográfico, discurso de ódio, desacato etc.;
- O website deve ser validado utilizando as ferramentas disponíveis nos endereços **validator.w3.org** e **jigsaw.w3.org/css-validator** (não deve conter nenhum erro ou *warning*);
- O trabalho deve ser entregue até a data/hora definida pelo professor. Não deixe para enviar o trabalho nos últimos instantes, pois eventuais problemas relacionados à eventos adversos como instabilidade de conexão, congestionamento de rede etc., não serão aceitos como motivos para entrega da atividade por outras formas ou em outras datas;
- Este trabalho deve ser feito **mantendo os trabalhos anteriores intactos**, ou seja, os trabalhos anteriores devem permanecer online conforme foram entregues;
- Trabalhos enviados por e-mail ou pelo MS Teams **não serão considerados**.

Material de Apoio

<https://furtado.prof.ufu.br/site/teaching/PPI/PPI-Modulo8-Ajax.pdf> (slides 1-54)

Exercício 1

Descompacte o arquivo furtado.prof.ufu.br/site/teaching/PPI/Exemplos-Ajax-XHR.zip, coloque os exemplos online no infinityfree e abra a pasta raiz no **VS Code**.

- a) Acesse o exemplo **ex1-hello-ajax** no navegador e clique no botão **Carregar mais conteúdo**. Observe o novo conteúdo. Abra o exemplo no **VS Code** e adicione comentários explicando o código JavaScript;
- b) Acesse o exemplo **ex2-busca-cidade** no navegador e digite os CEPs indicados. Observe o resultado. Abra o exemplo no **VS Code** e adicione comentários explicando o código JavaScript e o respectivo código PHP.

Exercício 2

- a) No infinityfree, acesse **Accounts** → **Sua Conta** → **Control Panel** → **Alter PHP Config** → **Alter PHP Directives** e altere o valor da opção **Display Errors** para **off**;
- b) Abra novamente o exemplo **ex2-busca-cidade** no navegador e tecle F12 para abrir o ambiente de desenvolvimento;
- c) Insira o CEP 38400-100 e monitore a respectiva requisição Ajax (Network → Fetch/XHR → buscaCidade.php). Observe os cabeçalhos da **requisição HTTP** (Request Headers) e os cabeçalhos da **resposta HTTP** (Response Headers). Alterne entre a exibição formatada e a exibição original (raw) dos dados. Observe também o **corpo** da resposta HTTP (aba **Response**). Crie um arquivo de nome **respostasExercicio2.txt** na pasta do exemplo e responda:
 - i. Qual foi o código de status retornado?
 - ii. Qual o valor do cabeçalho **Content-Type** da resposta HTTP? Por quê?
 - iii. Qual o valor recebido no **corpo** da resposta HTTP (aba **Response**)?
 - iv. Qual o valor do cabeçalho **Server** da resposta HTTP?
 - v. A **requisição** HTTP possui o cabeçalho **Content-Type**? Por quê?
- d) Insira o CEP 38400-500 e observe novamente os cabeçalhos da respectiva resposta HTTP. Responda as perguntas enumeradas no item anterior, porém no contexto da nova requisição.
- e) Simule um erro no script PHP removendo o fecha parênteses do primeiro **if**. Envie o script com o erro para o servidor e repita a busca usando o CEP 38400-100. Qual foi o código de **status** retornado? E o conteúdo do **corpo** da resposta HTTP?
- f) No infinityfree, acesse **Accounts** → **Sua Conta** → **Control Panel** → **Alter PHP Config** → **Alter PHP Directives** e altere a opção **Display Errors** para **on**. Repita o teste realizado no item anterior e observe o resultado. O que mudou? Qual foi o código de status retornado? O que foi retornado no corpo da resposta HTTP? Em qual situação essa configuração seria útil? Em qual situação ela jamais deveria ser utilizada?

Exercício 3

- a) Abra o exemplo **ex3-recebendo-json1** no navegador, informe um dos CEPs sugeridos e observe o resultado. Adicione **comentários** explicando o código JavaScript e o respectivo código PHP;
- b) Abra novamente o exemplo **ex3-recebendo-json1** no navegador e analise a **resposta** HTTP utilizando o ambiente de desenvolvimento do navegador. Responda as perguntas solicitadas no Exercício 2, item c), porém considerando a requisição HTTP deste exemplo. As respostas das perguntas devem ser disponibilizadas em um arquivo de nome **respostasExercicio3.txt** na pasta do exemplo.

Exercício 4

- a) Abra o exemplo **ex4-enviando-json1** no navegador, informe o texto **123456** no campo **Chave de Acesso**, informe o CEP sugerido e observe o resultado. Recarregue a página e repita o procedimento informando um valor diferente no campo **Chave de Acesso**;
- b) Quais são as principais diferenças desse exemplo comparado ao **ex3-recebendo-json1**? Acrescente comentários explicando as seguintes linhas do código JavaScript:
 - i. `xhr.responseType = 'json';`
 - ii. `xhr.setRequestHeader("Content-Type", "application/json");`
 - iii. `xhr.send(JSON.stringify(objetoJS));`
- c) Abra novamente o exemplo no navegador, insira o CEP 38400-100 e monitore a requisição Ajax utilizando o ambiente de desenvolvimento. Observe os cabeçalhos da **requisição** HTTP (Request Headers). Qual o valor dos cabeçalhos **Content-Type** e **Content-Length**? O que é

apresentado na aba **Payload**? O que isso significa? As respostas das perguntas devem ser disponibilizadas em um arquivo de nome **respostasExercicio4.txt** na pasta do exemplo.

Exercício 5

- Certifique-se de que a tabela **Aluno** do banco de dados contenha os três registros seguintes:

```
INSERT INTO aluno VALUES ("Fulano", "123");
INSERT INTO aluno VALUES ("Ciclano", "456");
INSERT INTO aluno VALUES ("Beltrano", "789");
```
- Abra o exemplo **ex5-alunos-json** no **VS Code** e atualize os dados de conexão do arquivo **conexaoMysql.php**. Atualize o exemplo no servidor;
- Acesse o link **Exemplo 5 - Todos os alunos em JSON** do menu de opções no navegador e observe o resultado. Analise o código do arquivo **todos-alunos-json.php**;
- Acesse o link **Exemplo 5 - Aluno por telefone** no navegador e observe o resultado. Altere o parâmetro **telefone** na URL para o valor **456** e observe o resultado. Analise o código do arquivo **aluno-por-telefone-json.php**;

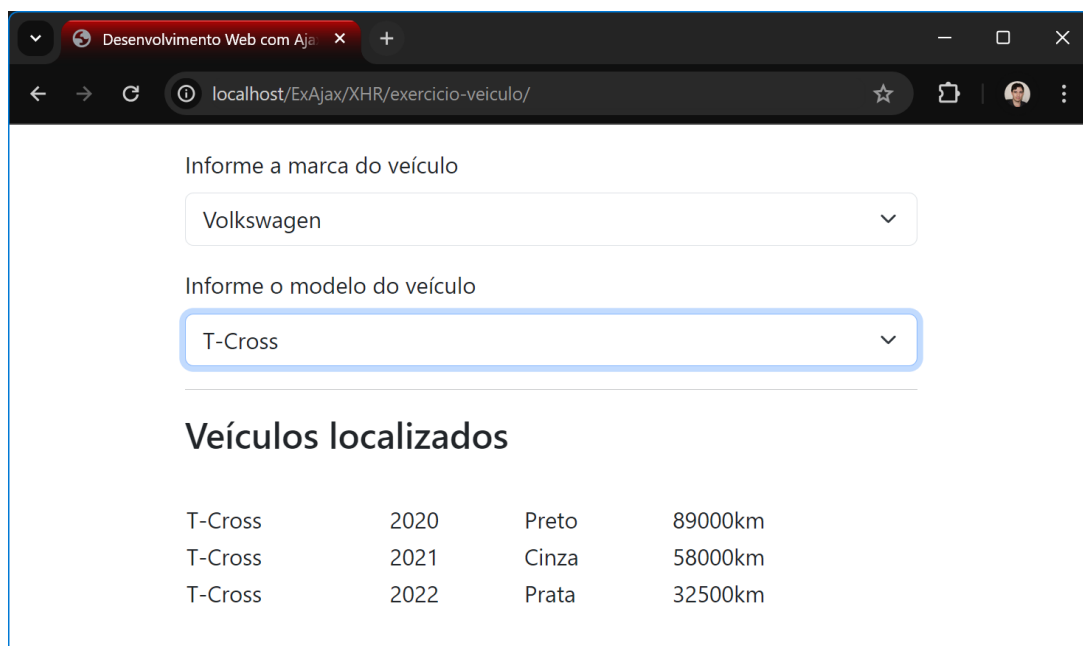
Exercício 6

Abra o exemplo **ex6-login** no navegador, insira o e-mail fulano@mail.com, a senha **123** e clique em **Entrar**. O formulário foi enviado da forma tradicional? Houve redirecionamento? Em seguida insira a senha **123456** e clique novamente em **Entrar**.

Acrescente comentários explicando as partes mais relevantes do código JavaScript e do código PHP. Inclua nos comentários as respostas das perguntas acima.

Exercício 7

Este exercício é **especialmente importante** no contexto da técnica Ajax e é uma oportunidade para testar o conhecimento adquirido nos exercícios anteriores. Seu objetivo principal é utilizar a técnica Ajax para implementar uma funcionalidade comum em aplicações web modernas. A página a ser criada permitirá ao usuário buscar por veículos cadastrados em um banco de dados conforme critérios fornecidos dinamicamente em campos do tipo **<select>**. A figura a seguir ilustra a funcionalidade na prática:



T-Cross	2020	Preto	89000km
T-Cross	2021	Cinza	58000km
T-Cross	2022	Prata	32500km

As opções do **primeiro <select>** (marcas dos veículos) são carregadas dinamicamente com Ajax assim que a página web termina de ser carregada.

Os **modelos** de veículos apresentados no **segundo <select>** também são carregados dinamicamente, assim que o usuário escolhe a **marca** do veículo no **primeiro <select>**.

Ao selecionar o **modelo** do veículo no **segundo <select>**, são apresentados os dados **modelo, ano, cor** e **quilometragem** de todos os veículos daquele modelo cadastrados no banco de dados.

- a) Abra a pasta **exercicio-veiculo** no **VS Code**, atualize os dados de conexão com o banco de dados e observe o conteúdo dos arquivos **index.html** e **sql-tabelas.sql**. Utilize o código SQL fornecido para criar e povoar a tabela **veiculo** no banco de dados;
- b) Escreva o código dos scripts PHP:
 1. O script **get-marcas.php** deve retornar, no formato JSON, um array contendo os nomes distintos de todas as **marcas** de veículos existentes na tabela **veículo**;
 2. O script **get-modelos.php** deve receber uma **marca** de veículo pela **URL** e retornar, no formato JSON, um array contendo os nomes distintos de **modelos** de veículos cadastrados para a marca em questão;
 3. O script **get-veiculos.php** deve receber um **modelo** de veículo pela **URL** e retornar, no formato JSON, um array de objetos contendo os dados **modelo, ano, cor** e **quilometragem** de todos os veículos do **modelo em questão** cadastrados.
- c) Abra o arquivo **index.html** e complete o código da função JavaScript *carregaMarcasDistintas* para permitir que o **primeiro <select>** seja carregado assim que a página for apresentada ao usuário. O código JavaScript deve fazer uma requisição Ajax ao script **get-marcas.php**;
- d) No arquivo **index.html**, complete o código da função *carregaModelosDaMarca* para que o **segundo <select>** seja carregado com base na marca escolhida no **primeiro <select>**. A função deve fazer uma requisição Ajax ao script **get-modelos.php**. Adicione o código JavaScript para que a função seja chamada na ocorrência do evento apropriado;
- e) Por fim, complete a função JavaScript *carregaVeiculosDoModelo* para permitir que os dados de todos os veículos do modelo escolhido sejam listados em uma tabela HTML, conforme apresentado na figura anterior. A função deve fazer uma requisição Ajax ao script **get-veiculos.php**. Ao receber os dados do servidor, a função *carregaVeiculosDoModelo* deve repassá-los à função *exibeVeiculos*, a qual deverá inserí-los dinamicamente na tabela HTML (manipulando a árvore DOM e utilizando métodos como *document.createElement*).

OBS: quando o usuário **modificar** algum **<select>**, resultados anteriores não devem aparecer juntamente com novos dados. Para limpar o **<select>** com JavaScript pode-se utilizar o código *meuSelect.innerHTML = ""*. A mesma ideia pode ser utilizada para limpar a tabela HTML.

Disponibilização Online

O trabalho deve entregue pelo sistema SAAT e disponibilizado online utilizando o subdomínio gratuito registrado em site de hospedagem. Como este trabalho consiste de modificações dos arquivos de exemplo, não é necessário criar subpastas para cada exercício. Ao acessar o endereço a seguir, deverá abrir a página principal contendo o **menu de opções** modificado:

seusubdominio.com/trabalho11

Entrega

Além da disponibilização online, a pasta raiz deve ser compactada no formato **zip** e enviada pelo Sistema Acadêmico de Aplicação de Testes (SAAT) até a data limite indicada pelo professor em sala de aula.

Adicione também um arquivo de nome **link.txt**, na pasta raiz, contendo a URL do trabalho online (para a pasta raiz do trabalho).

Sobre Eventuais Plágios

Este é um trabalho individual. Os alunos envolvidos em qualquer tipo de plágio, total ou parcial, seja entre equipes ou de trabalhos de semestres anteriores ou de materiais disponíveis na Internet (exceto os materiais de aula disponibilizados pelo professor), serão duramente penalizados (art. 196 do Regimento Geral da UFU). Todos os alunos envolvidos terão seus **trabalhos anulados** e receberão **nota zero**.