Matheus Campos Fernandes

# **Linguagens de servidor:** uma abordagem prática com PHP

Senac

**Dados Internacionais de Catalogação na Publicação (CIP)**
**(Simone M. P. Vieira - CRB 8ª/4771)**

Fernandes, Matheus Campos
Linguagens de servidor : uma abordagem prática com PHP / Matheus Campos Fernandes. – São Paulo : Editora Senac São Paulo, 2021. (Série Universitária)

Bibliografia.
e-ISBN 978-65-5536-616-7 (ePub/2021)
e-ISBN 978-65-5536-617-4 (PDF/2021)

1. Linguagem de programação 2. PHP (linguagem interpretada) 3. Aplicações – Desenvolvimento 4. Aplicativos dinâmicos 5. Sistemas para internet I. Título. II. Série.

21-1323t

CDD – 005.13
BISAC COM051010

**Índice para catálogo sistemático**
**1. Linguagem de programação 005.13**

# LINGUAGENS DE SERVIDOR: UMA ABORDAGEM PRÁTICA COM PHP

Matheus Campos Fernandes

## Administração Regional do Senac no Estado de São Paulo

**Presidente do Conselho Regional**
Abram Szajman

**Diretor do Departamento Regional**
Luiz Francisco de A. Salgado

**Superintendente Universitário e de Desenvolvimento**
Luiz Carlos Dourado

## Editora Senac São Paulo

**Conselho Editorial**
Luiz Francisco de A. Salgado
Luiz Carlos Dourado
Darcio Sayad Maia
Lucila Mara Sbrana Sciotti
Luís Américo Tousi Botelho

**Gerente/Publisher**
Luís Américo Tousi Botelho (luis.tbotelho@sp.senac.br)

**Coordenação Editorial/Prospecção**
Dolores Crisci Manzano (dolores.cmanzano@sp.senac.br)

**Administrativo**
grupoedsadministrativo@sp.senac.br

**Comercial**
comercial@editorasenacsp.com.br

**Acompanhamento Pedagógico**
Otacília da Paz Pereira

**Designer Educacional**
Diogo Maxwell Santos Felizardo

**Revisão Técnica**
Luiz Fernando Albertin Bono Milan

**Preparação e Revisão de Texto**
Camila Lins

**Projeto Gráfico**
Alexandre Lemes da Silva
Emília Corrêa Abreu

**Capa**
Antonio Carlos De Angelis

**Editoração Eletrônica**
Sidney Foot Gomes

**Ilustrações**
Sidney Foot Gomes

**Imagens**
Adobe Stock Photos

**E-pub**
Ricardo Diana



Editora Senac São Paulo
Rua 24 de Maio, 208 – 3º andar
Centro – CEP 01041-000 – São Paulo – SP
Caixa Postal 1120 – CEP 01032-970 – São Paulo – SP
Tel. (11) 2187-4450 – Fax (11) 2187-4486
E-mail: editora@sp.senac.br
Home page: http://www.livrariasenac.com.br

# Sumário

Capítulo 1

# Aplicativos dinâmicos para web

Sites, websites, sítios da internet, portais on-line, aplicações web, aplicativos para a web, sistemas web. Todos esses termos foram usados nos últimos anos para descrever o mesmo conceito. A verdade é que, atualmente, os navegadores de internet se tornaram a principal forma de consumir software. Para um usuário médio, tudo parece funcionar como mágica, pois o entendimento geral do funcionamento de aplicações web é bastante nebuloso. Este capítulo pretende desmistificar isso.

Aqui, daremos início aos nossos estudos, a começar pela diferença entre as duas abordagens possíveis para o desenvolvimento de soluções web: as abordagens estática e dinâmica. Na sequência, iremos apresentar a linguagem que usaremos para explorar e praticar o desenvolvimento de aplicações dinâmicas: a linguagem PHP.

# 1 Aplicativo estático

A web, como conhecemos, surgiu como uma maneira de interligar documentos, especificamente documentos armazenados na forma de hipertexto. Um hipertexto nada mais é do que um texto que contém referências para outros documentos. Essas referências normalmente acontecem no formato de links, ou imagens, ou vídeos, etc.

A forma padrão que se assumiu na web para armazenar esses hipertextos foi a hypertext markup language (HTML). Um documento HTML é composto pelas chamadas tags, delimitadas entre <>. Um exemplo de um documento HTML é dado a seguir:

```
<html>
    <head>
        <title>Meu Documento</title>
    </head>
    <body>
        <h1>Cabeçalho principal do meu documento</h1>
        <h2>Cabeçalho secundário do meu documento</h2>
        <p>
            Meu parágrafo contendo um texto extremamente
grande.
            Para mais informações visite
            <a href="outro_documento.html">este outro
documento</a>
        </p>
        <img src="cachorro.jpg" alt="Cachorro" />
    </body>
</html>
```

No exemplo, o trecho *<title>Meu Documento</title>* delimita uma ocorrência da tag *title*, que indica qual o título do documento. Ela começa em *<title>* e termina em *</title>*, com o conteúdo sendo o texto "Meu Documento".

A tag *<a href="outro_documento.html">este outro documento</a>* tem o que chamamos de atributo. O atributo em questão é o *href*, e seu valor é *outro_documento.html*. Esta é uma tag que o texto "este outro documento" deve apontar para o documento *outro_documento.html*, caracterizando um hipertexto. Usualmente, esta tag é tratada como um link, que, ao receber um clique, levará o usuário para outro documento.

A tag *<img src="cachorro.jpg" alt="Cachorro" />* é uma tag que não pode ter conteúdo. Portanto, colocamos o fechamento dentro da própria tag de abertura, identificado pelo />. Ali temos dois atributos: *src*, indicando qual arquivo esta tag representa, e *alt,* indicando a descrição textual para essa imagem.

Se pareceu muita informação, fique tranquilo! Não iremos nos prender, no momento, à estrutura de um HTML. A especificação dos elementos HTML é extensa e será abordada de forma gradual ao longo deste livro.

**PARA SABER MAIS**

Caso você tenha interesse em se familiarizar com os elementos do HTML, uma lista didática pode ser acessada na referência das tags HTML no portal da W3Schools (W3SCHOOLS, [*s. d.*]).

Como vimos, além de apontar para um outro documento HTML e para um arquivo de imagem, nosso HTML pode apontar para outros tipos de arquivos de texto, assim como vídeos, áudios, binários e qualquer outro que você possa imaginar. Em especial, temos dois outros arquivos muito importantes que veremos mais adiante: os arquivos de Cascading Style Sheets (CSS) e JavaScript (JS).

Toda a parte de desenhar de forma gráfica esse HTML (processo chamado de *render*) é responsabilidade única do navegador (browser). A partir dessas informações textuais, o navegador interpreta o

documento e o renderiza em tela. Mas como o nosso navegador tem acesso a um documento HTML?

Ao digitarmos um endereço (URL – uniform resource locator), sinalizamos para o navegador que queremos "abrir uma página na internet". Esse processo é apenas, a princípio, a transferência (download) de um único arquivo HTML, que pode (ou não) indicar arquivos adicionais que precisam ser transferidos, gerando uma sequência de transferências. Todos esses arquivos, HTML ou de outro tipo, são transferidos usando-se o protocolo de transferência de hipertextos (HTTP – hypertext transfer protocol). Esse protocolo indica como o navegador deve pedir os documentos ao endereço e foi projetado de forma a facilitar a troca de hipertextos, como o próprio nome indica.

**NA PRÁTICA**

Atualmente, é muito comum usarmos HTTPS no lugar de HTTP. A letra "S" significa *secure* (seguro). De forma simples, é uma versão que integra criptografia de ponta a ponta ao HTTP. Dessa forma, caso um terceiro intercepte a requisição ou a resposta, o conteúdo dela estará "trancado", podendo apenas ser aberto pelo emissor e/ou pelo destinatário.

Digamos, por exemplo, que digitamos o endereço "example.com/sobre" na barra de endereço do nosso navegador. Hoje, por conveniência, o navegador preenche o restante do endereço com "http://", o que resulta em "http://example.com/sobre", sinalizando que provavelmente queremos transferir um documento de hipertexto desse endereço. Então, começa uma espécie de conversa entre o nosso navegador e o endereço que digitamos. Essa conversa acontece no formato requisição e resposta.

O nosso navegador (chamado de front-end) envia para o endereço fornecido uma requisição, como se fosse uma espécie de mensagem

ou pedido, dizendo: "Olá, *example*! Eu gostaria de acessar o arquivo 'sobre'. Poderia me informar qual o conteúdo dele?".

**Figura 1 – Exemplo de uma requisição HTTP**


Do lado que recebe a requisição, temos também um computador (chamado de back-end). Esse computador tem a tarefa de responder corretamente à nossa requisição. Existem algumas formas diferentes de criar uma resposta, como veremos.

Em resumo, uma requisição HTTP não é nada mais do que um computador requisitando um documento para outro computador. Para nós, usuários, tudo isso é feito de forma transparente ao digitarmos o endereço na barrinha do nosso navegador e apertarmos a tecla "Enter".

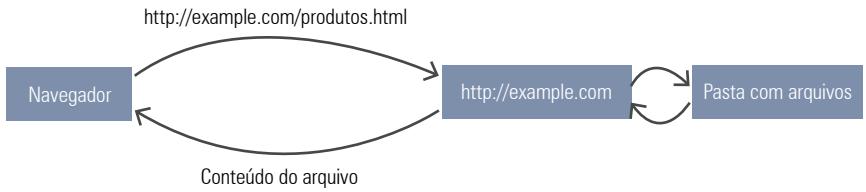
No computador do back-end, há sempre um programa em execução, que costumamos chamar de servidor HTTP. Em princípio, a única responsabilidade desse programa é ficar aguardando requisições e respondê-las adequadamente. A forma mais básica de fazer isso é simplesmente olhar em um diretório (pasta) com vários arquivos e devolver o arquivo adequado. Aplicações assim são chamadas de aplicações estáticas.

Por exemplo, digamos que o servidor de "example.com" esteja configurado dessa forma. Quando ele recebe uma requisição para "example.com/produtos.html", basta o servidor olhar no diretório que foi configurado, buscar pelo arquivo "produtos.html" e devolvê-lo. Nesse caso, a pessoa que desenvolve o site tem apenas a responsabilidade de escrever um arquivo HTML para cada página e colocá-lo no diretório adequado, e o servidor se encarrega de deixá-las corretamente "no ar".

## NA PRÁTICA

Dentre os softwares de servidor HTTP mais usados atualmente, podemos citar o Apache e o Nginx. No mundo Windows, também temos o IIS.

**Figura 2 – Exemplo de aplicação estática**

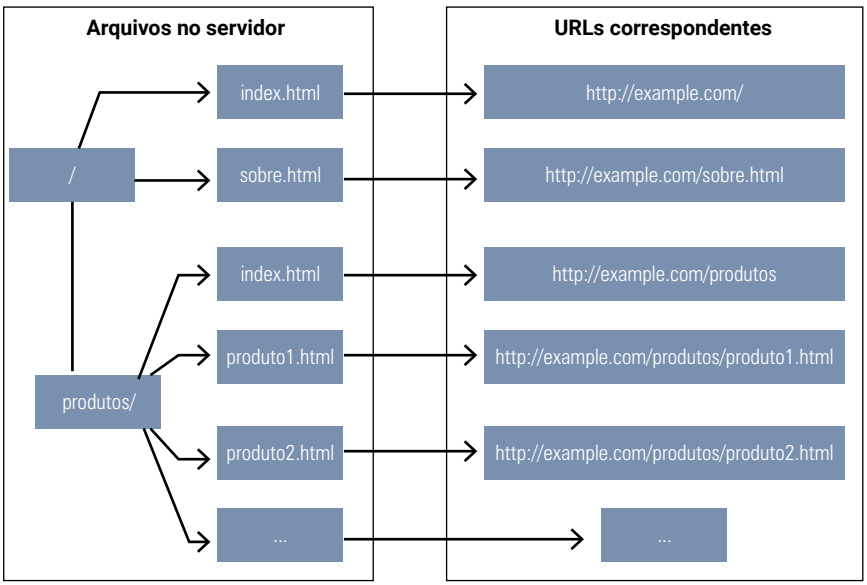



Repare que, nesta abordagem, o website é nada mais do que uma interface para interagir com uma pasta em outro computador. Assim, se quisermos que alguns dos nossos arquivos fiquem acessíveis de qualquer lugar, basta colocá-los em um desses servidores e acessá-los simplesmente navegando até o endereço correspondente.

Um caso especial é pensado para quando a requisição não pede nenhum arquivo em particular, como na requisição "example.com/". Nesse caso, há a convenção de termos um arquivo chamado "index.html", que corresponde a essa "requisição vazia". O arquivo recebe o nome de index (índice) pois era comum que fosse um documento que mostrasse um índice de todos os arquivos acessíveis, assim como um índice de um livro.

A imagem a seguir mostra um diagrama exemplificando como ficariam nossos endereços diante de determinada disposição de arquivos no servidor. Naturalmente, configurações adicionais podem ser realizadas no servidor para omitir a extensão ".html", tornando os endereços mais parecidos com aqueles com que estamos acostumados ("http://example.com/sobre", por exemplo).

**Figura 3 – Exemplo da estrutura de uma aplicação estática**




# 2 Aplicativo dinâmico

Há uma limitação na abordagem anterior. No nosso exemplo, tentamos montar uma página de e-commerce para vender produtos on-line. Como programadores, teríamos que escrever um HTML para a página inicial, um para cada página "fixa" (como o "sobre") e um para cada produto.

Para adicionar um produto novo, teríamos que criar um HTML novo para esse produto e atualizar a página principal para listá-lo também. Se houvesse uma seção de produtos relacionados, a cada novo produto teríamos que revisar todos os outros para checar se esse novo produto não se encaixa ali. Alterações em produtos – como mudar preço ou título ou mesmo marcar um item como fora de estoque – implicariam alterações em todas as páginas em que aquele produto aparece, seja na página do próprio produto, seja na página inicial ou nas seções de produtos relacionados de outros produtos. Qualquer alteração no layout da página, por menor que fosse, teria que ser feita em todos os HTMLs

existentes. Realizar promoções no site seria uma enorme dor de cabeça, pois teríamos que atualizar o layout para incluir os banners corretos e modificar os produtos para refletir o preço novo, além de termos que reverter tudo quando a promoção terminasse. E o mais importante de tudo: uma página de busca, por exemplo, não poderia existir, pois seria impossível criarmos um HTML para cada combinação de busca existente. Que trabalho, não é mesmo?

Felizmente, existe uma abordagem que resolve todos esses problemas. Nós simplesmente deixamos o nosso servidor "mais inteligente", tornando-o capaz de gerar esses HTMLs sob demanda. Para isso, é necessário que o servidor tenha acesso a instruções sobre como montar esses HTMLs de forma automática. Uma sequência de instruções, como você certamente sabe, é um algoritmo, que pode ser escrito em uma linguagem de programação.

**IMPORTANTE**

Em termos simples, usamos uma linguagem de programação para explicar ao nosso servidor como montar o HTML de resposta para cada requisição.

Assim, podemos expressar a nossa página de produtos de forma genérica, algo no seguinte sentido, em pseudocódigo:

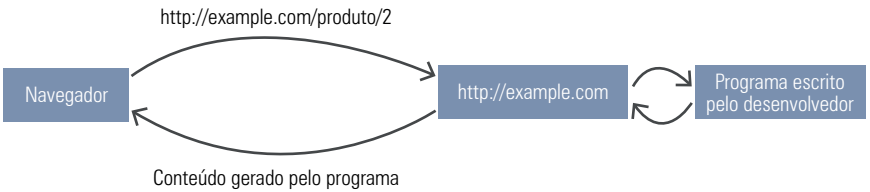
```
<h1>TÍTULO DO PRODUTO</h1>
<img src="FOTO DO PRODUTO" />

SE O PRODUTO ESTIVER EM ESTOQUE:
    <h2>R$ PREÇO DO PRODUTO</h2>
CASO CONTRARIO
    <h2>O produto não está em estoque</h2>

<p>DESCRIÇÃO DO PRODUTO</p>
```

Quando o servidor receber a requisição para mostrar a página do Produto 2, por exemplo, ele simplesmente seguirá as instruções apresentadas para montar o HTML na hora. Em praticamente todos os casos, as informações sobre os produtos ficam armazenadas em alguma forma de banco de dados, que o servidor acessa para preencher as lacunas do HTML.

**Figura 4 – Exemplo de uma requisição em uma aplicação dinâmica**




Note que, naturalmente, podemos misturar as duas abordagens. Isso, inclusive, é extremamente comum e acontece em quase todos os casos. Embora muitas vezes faça sentido o HTML ser dinâmico, imagens e outros arquivos são quase sempre estáticos, e podemos ter até HTMLs inteiramente estáticos, como uma página de "sobre" que contém apenas texto. Os servidores em geral já são preparados para lidar com esse tipo de situação.

A grande maioria das linguagens de programação, para não dizer praticamente todas, possui alguma forma de programar uma aplicação web dinâmica, como Java, C#, JavaScript, Go, Python, Haskell, Ruby, etc. Qualquer linguagem que o leitor possa vir a imaginar provavelmente já foi usada comercialmente para o desenvolvimento de sites em algum projeto. Cada linguagem, naturalmente, apresenta suas vantagens e desvantagens. Ao longo deste livro, usaremos a linguagem de programação PHP para escrever os programas que serão executados no nosso servidor, a qual será apresentada ao final deste capítulo.

# 3 Comparação entre as abordagens

Como pudemos ver, em geral, a abordagem dinâmica é mais poderosa e robusta se comparada à abordagem estática. Mas nem sempre é fácil enxergar de maneira simples os motivos disso. Por isso, apresentamos um resumo com as diferenças-chave entre as duas abordagens:

**Quadro 1 – Comparação entre as abordagens estática e dinâmica**

| SITES ESTÁTICOS | SITES DINÂMICOS |
|---|---|
| Não precisam de uma linguagem de programação | Dependem de uma linguagem de programação |
| Respondem às requisições com arquivos pré-armazenados | Respondem às requisições com arquivos gerados em tempo real |
| Não conseguem acessar bancos de dados | Costumam buscar informações em um banco de dados para gerar as páginas |
| Número fixo de páginas, determinado pela quantidade de arquivos no servidor | Número potencialmente infinito de páginas |
| Todos os visitantes do site terão acesso à mesma versão de determinada página | Podem fornecer uma versão diferente da página para cada situação |
| Não é possível reutilizar trechos de HTML (por exemplo, para compartilhar cabeçalhos, menus e rodapés) | Podemos dividir os trechos de HTML como bem entendermos. Já que as páginas serão geradas, basta escrever instruções para incluir um trecho de HTML aqui ou ali |

# 4 História do PHP e evolução

PHP é um software gratuito e de código aberto usado principalmente para o desenvolvimento de aplicações web. Entre 1994 e 1995, Rasmus Lerdorf criou e disponibilizou a primeira versão do que ele chamou de personal home page tools (PHP-tools, ou ferramentas de página principal pessoal, em tradução livre). Aos poucos, a linguagem foi se tornando popular, ganhando novas funcionalidades e se transformando na linguagem que hoje está em sua sétima versão e ainda continua com

uma comunidade ativa. Com o tempo, a sigla foi ressignificada para se tornar o acrônimo recursivo "PHP: Hypertext Preprocessor". Hoje, PHP é a linguagem responsável por aproximadamente 4 de cada 5 websites (W3TECHS, [2020b]).

A linguagem PHP é a base do CMS (content management system, ou sistema de gerenciamento de conteúdo) mais usado no mundo: o WordPress. Segundo o W3Techs ([2020a]), 38,1% de todos os sites do mundo usam o WordPress. Considerando apenas os sites que usam CMS, 63,5% usam o WordPress, com o segundo colocado (Shopify) tendo apenas 4,6%.

A principal vantagem da linguagem PHP, que leva à sua adoção tão ampla, é ter sido criada especialmente para o desenvolvimento de aplicações web. Ou seja, a linguagem não tem a pretensão de ser uma linguagem de programação de propósito geral e foca em prover a experiência mais simples possível para criar um site. Assim, é muito fácil e direto obter um site simples, bastando gerar um arquivo com a extensão ".php" para cada página. Esse arquivo tem uma sintaxe parecida com o exemplo dado anteriormente (pseudocódigo no tópico "Aplicativo dinâmico"), pois um arquivo PHP consiste em basicamente HTML com "enxertos" de programação. A sintaxe específica da linguagem será abordada em profundidade ao longo do livro.

Embora tenha surgido como uma linguagem bastante simples, o PHP evoluiu para adotar o paradigma orientado a objetos e tem mostrado sinais de adoção do paradigma funcional, permitindo a criação de arquiteturas de software cada vez mais complexas. Em especial, houve um salto notável de desempenho entre as versões 5.x e 7.x (a versão 6 do PHP nunca foi lançada oficialmente). A adoção cada vez maior da linguagem também se deve ao surgimento de frameworks como CakePHP, CodeIgniter e Laravel, que tornaram bem mais fácil o desenvolvimento de sistemas web robustos.

**PARA SABER MAIS**

Mais informações sobre a história do PHP podem ser encontradas na página comemorativa dos 25 anos da linguagem, no site da Jet Brains (JET BRAINS, [2020]).

Neste livro, não focaremos em um framework ou CMS específico. Aqui, o intuito é aprender os fundamentos da programação de servidores em geral, e para isso utilizaremos a linguagem PHP de forma "pura", como ferramenta. A ideia é que os assuntos abordados possam ser facilmente transportados para qualquer linguagem de programação. Evidentemente, para tanto, será necessário aprender também os fundamentos da linguagem PHP, quase como uma consequência. Então, caso você tenha interesse em continuar os estudos de PHP, este livro também proverá bagagem suficiente para prepará-lo para a busca de mais informações sobre os frameworks de preferência, seja em documentações, seja em blogs, tutoriais na internet ou mesmo outros livros.

# Considerações finais

Este capítulo apresentou uma visão geral de como uma aplicação web funciona.

Aprendemos como um documento é representado na web e como um navegador obtém esse documento ao se comunicar com um servidor.

Vimos que uma possível abordagem para o servidor responder a uma requisição é simplesmente através de uma estrutura de pastas contendo arquivos, nas quais o servidor busca o arquivo que corresponde à requisição feita.

Notamos que essa abordagem, embora simples, é limitada. Concluímos que, para resolver essa questão, teríamos que instruir o

nosso servidor sobre a maneira de montar uma resposta, e para isso temos que fazer uso de alguma linguagem de programação.

Por fim, apresentamos a linguagem que usaremos ao longo do curso: o PHP, responsável pela maior parte das aplicações web da atualidade.

# Referências


JET BRAINS. 25 years of PHP History. **Jet Brains**, [2020]. Disponível em: https://www.jetbrains.com/lp/php-25/. Acesso em: 18 ago. 2020.

W3SCHOOLS. HTML Element Reference. **W3Schools**, [2020]. Disponível em: https://www.w3schools.com/TAGS/default.ASP. Acesso em: 18 ago. 2020.

W3TECHS. Usage statistics of content management systems. **W3Techs**, [2020a]. Disponível em: https://w3techs.com/technologies/overview/content_management. Acesso em: 18 ago. 2020.

W3TECHS. Usage statistics of server-side programming languages for websites. **W3Techs**, [2020b]. Disponível em: https://w3techs.com/technologies/overview/programming_language. Acesso em: 18 ago. 2020.