**HefestosPHP: Simplificando desenvolvimento web para iniciantes (brasileiros?) em um framework MVC com PHP moderno**

Bruno Luis Gomes Grangeiro[[1]](#footnote-0), Jhonatta Pietro[[2]](#footnote-1)

**Resumo:** Copie e cole o seu resumo enviado e aprovado. SÓ ALTERE O QUE ESTÁ EM AMARELO. S saf sf asdf asdf asdf asdf asd faz dfas dfa sdf asdf asdf as fdas fda sdf asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfa sdf asdf asd faz dfa sdf asdf asdf as dfas df asdf asdf as dfas df asdf asd faz df asdf asd faz df asdf asd faz df asdf asdf as dfas df asdf as df asdf as dfa sdf asdf as dfa sdf asfd asdf asdf a sdfa sdf asdf asd fa sdf asdf as dfas dfa sdf. S saf sf asdf asdf asdf asdf asd faz dfas dfa sdf asdf asdf as fdas fda sdf asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfa. S saf sf asdf asdf asdf asdf asd faz dfas dfa sdf asdf asdf as fdas fda sdf asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfas df asdf asdf as dfa

**Palavras-Chave:** Palavra 1. Palavra 2. Palavra 3. Palavra 4. Palavra 5.

**Introdução**

A evolução do desenvolvimento *web* tem sido marcada por uma série de avanços e desafios ao longo das últimas décadas. O lançamento do primeiro *website* em 1991 por Tim Berners-Lee, o criador da web, abriu caminho para o que se tornaria uma das principais forças motrizes da tecnologia moderna. Desde então, a *web* tem sido transformada por diversas tecnologias e linguagens de programação, que permitiram o desenvolvimento de aplicações cada vez mais avançadas e sofisticadas, eventualmente culminando no que conhecemos hoje.

Nesse contexto, uma dessas linguagens de programação é o PHP: Hypertext Preprocessor, tendo mais de duas décadas de história e um impacto significativo no desenvolvimento *web* como conhecemos hoje. Criado em 1995, no Canadá, por Rasmus Lerdorf, o PHP rapidamente se tornou uma das linguagens mais populares para criar aplicativos *web*. No entanto, a simplicidade inicial da linguagem permitia que muitos iniciantes criassem códigos que, apesar de funcionais, eram despadronizados e de difícil leitura e manutenção. Junto a isso, até sua versão 5, a linguagem sofria com diversas inconsistências e falhas de segurança. Por esses motivos, o PHP ganhou certa má fama e, com o passar do tempo e o amadurecimento de outras linguagens, foi tido cada vez mais como uma linguagem ultrapassada, especialmente por detalhes menores de sintaxe.

Entretanto, o PHP também amadureceu: falhas de segurança foram corrigidas, inconsistências foram retificadas e grandes funcionalidades foram implementadas. Atualmente, o PHP é amplamente usado não só para criar aplicativos *web* complexos e seguros, como também é porta de entrada para muitos iniciantes no mundo do desenvolvimento *back-end, contando ainda* com uma enorme comunidade de desenvolvedores contribuindo ativamente para a evolução contínua da linguagem e seu ecossistema. Em outras palavras, ao contrário de uma popular opinião que muito é enviesada em versões antigas da linguagem, desenvolvimento *web* com PHP é uma opção tão viável quanto com qualquer outra linguagem.

Desse modo, o PHP está hoje entre as tecnologias mais populares para desenvolvimento web. A vista disso, não surpreende que esta linguagem seja uma das mais utilizadas nesse ambiente, inclusive no mercado profissional, o que gera uma grande demanda por programadores. Sua reputação de aprendizado fácil beneficia aqueles que estão começando, enquanto sua poderosa flexibilidade o torna também uma escolha valiosa para empreendimentos mais elaborados. Porém, qualquer aplicação desestruturada tende a se desfazer em uma bagunça incoerente como bem aponta Chris Pitt (2012) em seu livro Pro PHP MVC, sendo exatamente isso que a arquitetura MVC (Model-View-Controller) pretende solucionar ao definir diferentes camadas para a aplicação

Ao longo dos anos, a arquitetura MVC tornou-se quase um consenso no desenvolvimento *web,* independentemente da linguagem sendo utilizada, principalmente por sua capacidade de tornar a base de código mais organizada e de fácil manutenção. Isso se dá pois ela propõe separar a lógica de negócios da navegação e da interface do usuário, tendo sido originalmente concebida para aplicações *full stack* onde a interação do usuário é feita por meio de uma interface *web*. No entanto, o padrão MVC pode ser adaptado a diferentes tipos de aplicações: em APIs *RESTful* para servir dados a outras aplicações ou clientes, a separação clara entre a lógica de negócios (*Model*), o formato de saída dos dados (*View*) e o controle de fluxo (*Controller*) ainda faz sentido e pode ajudar a garantir a escalabilidade, manutenibilidade e qualidade do código. Dito isso, não é então nenhuma surpresa que seja também a arquitetura de escolha padrão dos principais *frameworks* do mercado, tanto no PHP como em outras linguagens.

Por conseguinte, um *framework* é uma ferramenta utilizada para simplificar tarefas repetitivas e padronizar o desenvolvimento de acordo com certas convenções, garantindo melhor organização e manutenção do projeto. Assim sendo, é evidente que empresas, em sua esmagadora maioria, optam por utilizar *frameworks* para prezar pela qualidade de seus sistemas. Desse modo, o aprendizado de tais ferramentas se mostra crucial para ingressar no mercado de trabalho, sendo esse o objetivo de muitos que estudam programação, seja em universidades ou mesmo sozinhos. Entretanto, o aprendizado dessas ferramentas por parte dos iniciantes pode ser um enorme desafio. Decerto, para alguém com experiência, é elementar compreender o fluxo de funcionamento de um *framework* e o seu papel no projeto para além da linguagem sem maiores complicações. Porém, um principiante no mundo da programação pode encontrar grandes dificuldades em entender esse fluxo, especialmente se levado em consideração que muitos brasileiros não estão tão familiarizados com o inglês, sendo este o idioma padrão de todas as grandes ferramentas do mercado.

Não obstante, esses *frameworks* normalmente dependem de diversas outras tecnologias. Desse modo, alguém inexperiente não só precisará se preocupar em aprender a linguagem de programação propriamente dita, como também as convenções de seu *framework* e de suas respectivas dependências. Como agravante, *frameworks* de grande porte tendem a conter uma quantidade excessiva de mecanismos internos desacoplados para satisfazer qualquer necessidade ao redor do mundo, gerando também uma sobrecarga de informações. Naturalmente, entender tudo isso pode ser muito confuso para um iniciante, especialmente quando considerada a possível barreira linguística. Entretanto, permanece essencial saber discernir esses conceitos para a devida compreensão da ferramenta e das partes que a compõem, levando a um melhor entendimento do projeto trabalhado como um todo. Ainda, essa complexidade pode se mostrar um desafio também para um professor, que possivelmente entenderá como muito conteúdo extra além do básico a ser passado para principiantes no curto tempo de um semestre letivo. Diante de todo o exposto, é notável o número de dificuldades com as quais um principiante pode se deparar na fase de aprendizado do desenvolvimento web padronizado.

Dito isso, o presente trabalho propõe o desenvolvimento de um framework PHP, também utilizando arquitetura MVC, mas de modo simplificado e majoritariamente em português. Isso significa dizer que, apesar de completo com todas as funcionalidades essenciais e algumas além, o *framework* será desenvolvido tendo como prioridade a simplicidade, evitando sobrecarregar o usuário iniciante com informações e funcionalidades desnecessárias a princípio. Para tanto, HefestosPHP será desenvolvido do absoluto zero, possuindo nenhuma dependência externa de bibliotecas ou ferramentas de terceiros, mas compatível com elas se o usuário julgar necessário. Funcionalidades importantes para a arquitetura MVC, como roteamento, acesso ao banco de dados e componentes de interface, serão implementadas de maneira simplificada, mas seguindo as convenções dos grandes frameworks PHP, desse modo ainda familiarizando os usuários ao seu funcionamento em geral. Além disso, a implementação será feita de forma generalizada, evitando prender os usuários a detalhes de implementação específicos de cada framework e focando em passar a ideia geral, tornando o conhecimento adquirido em sua utilização reciclável e útil mesmo ao usar outra linguagem de programação no futuro.

Ademais, todas as funcionalidades dentro do framework estarão documentadas e declaradas em português. Ou seja, não apenas o usuário terá fácil acesso à documentação em português de cada função e método, especialmente se usado em conjunto com um bom editor de código, como também sua utilização torna o seu código mais naturalmente legível. Não apenas isso, mas também será desenvolvida uma documentação online, visando uma fácil navegação e organização lógica para auxiliar o usuário a entender toda a estrutura do framework e suas ferramentas. Dito isso, a utilização do português apenas não será aplicada em certas nomenclaturas, tendo em vista que a tradução de algumas delas podem vir a ser um detrimento maior que uma dádiva por serem convenções gerais. Porém, claro, a documentação também levará isso em conta, prontamente fornecendo descrições e comparações para entendimento do conceito geral. Dessa forma, os únicos pré-requisitos necessários para a utilização do framework serão noções de lógica de programação e o entendimento básico da linguagem PHP.

Assim, espera-se que este projeto possa ser utilizado em fase de aprendizado para ajudar iniciantes a compreenderem melhor o desenvolvimento web padronizado. Sua utilização cedo na fase do aprendizado irá evitar esse erro tão comum no PHP que é o código desestruturado e desordenado (como diz o chirs pitt de novo?). A exemplo, a implementação deste framework em contextos universitários, sob a orientação de um professor durante a criação de projetos acadêmicos, pode proporcionar uma valiosa oportunidade de adquirir experiência real. Sobretudo, espera-se que o presente projeto ajude a diminuir a curva de aprendizado dos grandes frameworks do mercado, de modo que se sintam habituados ao graduar para algo como Laravel, CodeIgniter e Symphony no PHP ou ainda qualquer outro framework com arquitetura de 3 camadas como Ruby on Rails, Django ou Express, assim facilitando a capacitação dos iniciantes e consequentemente seu ingresso no mundo profissional. (talvez citar o primeagen aqui sobre focar na ideia invés de na linguagem em si.

**1 Desenvolvimento web e PHP**

A Internet é, sem dúvidas, uma das mais significativas inovações do século XX. Desde a sua criação, ela tem servido como um catalisador para uma série de avanços tecnológicos contínuos, que continuam a moldar e revolucionar nossa forma de vida e interação social. Seu surgimento de fato pode ser traçado até a Advanced Research Projects Agency Network (ARPANET) no final da década de 60 como apontado em (<https://www.internetsociety.org/wp-content/uploads/2017/09/ISOC-History-of-the-Internet_1997.pdf>). No entanto, a web propriamente dita, como conhecemos hoje, começa com o trabalho de Tim Berners-Lee na CERN, no final da década de de 80 e começo da década de 90, com o sonho de proporcionar um lugar universal para troca de informações, como o mesmo bem aponta em seu artigo [The World Wide Web: A very short personal history](https://www.w3.org/People/Berners-Lee/ShortHistory.html).

Inicialmente, a proposta era simples: utilizar *Universal Document Identifiers, depois conhecidas como URLs*, para rotular todos os documentos de forma única e padronizada e torná-los acessíveis por qualquer computador na rede. Então, com avanço do projeto, passou a ser generalizado para qualquer tipo de recurso e, em juntamente ao *HyperText Markup Language* (HTML) e o *HyperText Transfer Protocol* (HTTP), foi publicado o primeiro servidor para promover uma ampla adoção e discussão da abordagem. Após isso, a web se popularizou imensamente e, ao se espalhar para um público maior, nasceu a demanda por recursos mais poderosos para o desenvolvimento de websites, impulsionando o desenvolvimento *web* como conhecemos hoje.

**1.1 Desenvolvimento web e seu impacto no cenário da tecnologia**

Como aludido, a princípio a web funcionava apenas para acessar documentos por meio de URLs únicas para cada documento com, por exemplo, arquivos html. Eventualmente, com seu sucesso, mais e mais websites foram criados para os mais diferentes segmentos como empresariais, de comercialização de produtos, blogs e até mesmo para restaurantes. Então, conforme sites passaram de apenas arquivos e landing pages para se tornarem maiores e mais interativos, logo se viu a necessidade de tecnologias mais complexas atuando do lado do servidor, permitindo não apenas o simples acesso à arquivos nele guardados, mas também interfaces mais complexas e funcionalidades mais sofisticadas. Com isso, sistemas que não estavam presos às funcionalidades dos navegadores foram utilizados nos servidores para atender tais demandas, o que conhecemos hoje como o “back end” dos aplicativos web.

Desse modo, diversos mecanismos foram criados para tornar os websites mais acessíveis para os consumidores, como sistemas de roteamento que permitiam URLs amigáveis e assim evitando a utilização de extensões de arquivos nos links acessados. Não apenas isso, mas esse advento permitiu que sistemas complexos inteiros pudessem ser desenvolvidos tendo a web como interface principal, contrariamente ao padrão de software instalável que até então dominava o mercado.

Consequentemente, com o passar dos anos, a web não apenas é o maior conjunto de informações como sonhou Tim Berners-Lee, mas também base para sistemas modernos que movem indústrias inteiras. A vista disso, diversas ferramentas foram desenvolvidas para facilitar esse processo, desde tópicos mais básicos até sistemas complexos. Por exemplo, WordPress é um popular *Content Management System* (CMS) escrito em PHP que permite a criação de sites e blogs mesmo por leigos em programação, com mais de 60 milhões de projetos construídos nele ao longo dos anos, segundo seu site oficial. O setor bancário também entra nessa com bancos digitais, como as empresas Nubank ou Picpay, esta última que utiliza PHP com o relativamente novo *runtime “*Swoole” para enormes ganhos de performance. Ou ainda, WebAssembly, tecnologia usada pela empresa Autodesk para migrar sua *codebase* em C++ de mais de 30 anos do AutoCAD para a web como relatam aqui ([AutoCAD’s journey to the web, Dania El Hassan](https://www.youtube.com/watch?v=BfkL3WgOPdI)). Em resumo, a web hoje não apenas sedia novos negócios como também atrai os antigos, sendo um ponto central de toda a tecnologia e isso provavelmente não mudará em um futuro próximo.

**1.2 PHP: Origens, evolução e modernidade**

O PHP, como é conhecido hoje, na verdade, é o sucessor de um produto criado em 1994 por Rasmus Lerdorf, sendo apenas um conjunto simples de ferramentas escritas na linguagem de programação C. Originalmente usado para rastrear visitas ao seu currículo online, ele nomeou o conjunto de scripts de 'Personal Home Page Tools', ou 'PHP Tools’, como era mais frequentemente referido. Com o tempo, Rasmus desejou mais funcionalidades e reescreveu o PHP Tools, produzindo uma implementação muito maior e mais rica, capaz até mesmo de interagir com banco de dados, fornecendo um uma espécie de framework no qual os usuários poderiam desenvolver aplicativos web dinâmicos simples, como livros de visitas por exemplo. Em junho de 1995, Rasmus liberou o código-fonte das PHP Tools para o público, o que permitiu que outros desenvolvedores o usassem como bem quiserem. Com isso, não só foi possível como incentivado que os usuários fornecessem correções para bugs no código e o melhorassem de maneira geral, tal qual a maioria dos projetos de código aberto.

Em seguida, Rasmus expandiu o PHP, implementando a hoje conhecida funcionalidade de código incorporado ao HTML, com sintaxe similar à do Perl, linguagem de programação popular na época, porém de forma simples e um tanto inconsistente. Em 1998, o código-fonte já havia sido reescrito mais duas vezes e, agora em sua versão 2.0 com suporte nativo para diversos tipos de bancos de dados, o PHP já estava instalado em mais de 60 mil servidores ao redor do mundo, como indicado por uma pesquisa da Netcraft em maio do mesmo ano. Paralelamente, Andi Gutmans e Zeev Suraski colaboravam com Rasmus e juntos decidiram desenvolver uma nova e independente linguagem para a versão 3.0, esta que foi então lançada sob o acrônimo recursivo de ‘PHP: Hypertext Preprocessor’. Com isso, a repaginação do PHP foi um tremendo sucesso por permitir a extensibilidade direta da linguagem, levando à criação de uma variedade de módulos por parte de contribuidores. Além disso, a linguagem agora dava suporte para orientação a objetos (ainda que em estados iniciais) e uma sintaxe mais consistente e poderosa, levando-a a ser instalado em aproximadamente 10% de todos os servidores web da internet.

Já nos anos 2000, foi lançado o PHP 4, dessa vez construído em cima da Zend Engine, desenvolvida por Zeev e Andi, que melhorou drasticamente a performance da linguagem. Além disso, a nova versão trouxe consigo funcionalidades importantes como buffering de saída, sessões HTTP e melhorias de segurança. Finalmente, em 2004, PHP 5 foi lançado, com um melhor suporte para orientação a objetos e dúzias de novas funcionalidades, sendo esta a versão mais conhecida da linguagem. Nesse ponto, PHP já era imensamente popular em todo o mundo, com um número de instalações por servidor bem acima dos 100 mil e projetos enormes como WordPress disparando em popularidade, sendo fortemente utilizado até os dias de hoje. Além disso, e por consequência, a linguagem conquistou uma gigantesca comunidade de desenvolvedores ao redor do globo, incluindo no Brasil, engajados em enriquecer seu ecossistema.

Todavia, com o passar da década, a sintaxe do PHP juntamente com seus ainda presentes problemas de inconsistência nas APIs internas fizeram com que a linguagem passasse a ser mal vista por diversos círculos de programadores. Não apenas alguns programadores PHP, descontentes com o estado atual da linguagem que tornava o código muito suscetível a erros e falhas de segurança, mas principalmente desenvolvedores de outras linguagens, como Python, que estavam também se popularizando na web. Por conta disso, o PHP até hoje sofre com a imagem de uma linguagem antiquada, ao ponto em que a opinião mais popular a respeito da linguagem é de que é ultrapassada e “morta”.

Entretanto, a verdade não poderia ser mais diferente. Ao longo da última década, o PHP recebeu diversas melhorias e atualizações, estando agora em sua versão 8, com performance 5 vezes melhor e inúmeros novos recursos: *Scalar types, return types* e *type hinting* receberam suporte, contribuindo para *type safety* na linguagem; Operadores como *null coalescing*  para valores padrões e *nullsafe* para encadeamento de métodos com segurança também receberam suporte; Propriedades e classes *readonly,* além das poderosas *match expressions,* também foram implementadas para código mais limpo; Até mesmo Just-In-Time (JIT) compiler foi adicionado em sua versão mais recente. Tudo isso culmina, portanto, em um código mais claro e aprazível que nunca, sendo que alguns desses recursos não estão disponíveis mesmo em linguagens mais “modernas”. Ademais, também não é de se estranhar que a comunidade tenha desenvolvido diversas abstrações para seus projetos, ou seja, funcionalidades do PHP foram encapsuladas em métodos e funções para proporcionar melhores e mais consistentes APIs, resolvendo até mesmo esse detalhe negativo da linguagem.

Sendo assim, mesmo que alguns recursos ainda sigam convenções antigas, por compatibilidade com versões anteriores ou quaisquer outros motivos, o problema é facilmente resolvido com abstrações, estas que são comumente fornecidas pelos frameworks da linguagem. A exemplo disso, a funcionalidade de sessões citada anteriormente pode ser bem complicada a princípio e, apesar de ser sim interessante saber seu funcionamento por baixo dos panos, no mundo real o programador dificilmente necessitará lidar com isso diretamente uma vez que esta é abstraída por funções auxiliares do framework utilizado no projeto. Isso permite, então, que o programador não precise se preocupar em decifrar todo o funcionamento de todas as funcionalidades desde o começo mas sim conforme necessário, privilégio esse que pode ser muito útil na fase de aprendizado. Ou seja, muitas das complexidades da linguagem são reduzidas ou eliminadas pelos frameworks que, normalmente, serão sempre utilizados em projetos reais.

Talvez uma imagem mostrando o exemplo da session em mais detalhes e o contraste com a simplicidade da abstração de um framework em relação ao overhead que podem ser causado principalmente quando acumulado com outros de outras origens.

Notavelmente, é por seu ecossistema que o PHP realmente brilha. A linguagem possui uma enorme comunidade ativa, desenvolvendo bibliotecas, encontrando soluções e promovendo conteúdo dos mais diversos tipos. Além disso, frameworks poderosíssimos como Laravel e Symphony dominam o mercado, mas com razão, sendo extremamente completos e interoperacionais entre si, recebendo louvor mesmo por desenvolvedores de outras linguagens. Claro, tanto poder vem com uma curva de aprendizado um pouco mais acentuada, podendo sobrecarregar principiantes, mas mesmo esses, se habituados ao seu funcionamento, logo perceberão o quanto é possível executar com essas ferramentas. Concluindo, o PHP não é uma ótima escolha para desenvolvimento web mesmo 2023 por ser a linguagem mais nova ou popular, pois não é, mas por ser tão conveniente para fazer tanta tarefa em tão pouco tempo, como aponta Brent Roose, developer advocate na JetBrains, em seu vídeo ([Why PHP?](https://www.youtube.com/watch?v=x7OsH3bH6DA)).

**2 Título do Assunto 2**

Discorra sobre o Assunto 2. Coloque quantos tópicos forem necessários. Converse isso com seu orientador.

**Conclusão**

Digite aqui suas conclusões alcançadas e ao mesmo tempo fazendo um resgate do seu Problema de Pesquisa, dizendo se o resolveu bem como os Objetivos Gerais e Específicos, se os alcançou.

**Referências** (Colocar em ordem alfabética crescente. Coloquei apenas exemplos)

CUSIN, C. A.; BACHINI, C.; FLATSCHART, F. **Open Web Platform**. São Paulo: Brasport, 2013.

MOREIRA, A. **Desenvolvendo Sites Acessíveis**. 2020. Disponível em: http://www.w3c.br/sitesacessíveis.html. Acesso em: 11 set. 2020.

1. Bruno Luis Gomes Grangeiro. Acadêmico do Curso de Sistemas de Informação. Centro Universitário Paraíso (UniFAP). Rua da Conceição, 1228 - São Miguel - CEP: 63.010-465 - Juazeiro do Norte, CE. E-mail: bruno.gomes@aluno.fapce.edu.br [↑](#footnote-ref-0)
2. Jhonatta Pietro. Pós-graduado em desenvolvimento web. Professor do Curso de Sistemas de Informação. Centro Universitário Paraíso (UniFAP). Rua da Conceição, 1228 - São Miguel - CEP: 63.010-465 - Juazeiro do Norte, CE. E-mail: jhonatta.pietro@ fapce.edu.br [↑](#footnote-ref-1)