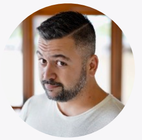
O universo PHP

Imergindo em uma linha de evolução da linguagem de programação mais utilizada no mundo.

https://fullstackphpdeveloper.com

upinside
vem aprender com a gente!

```
{
  "name": "UpInside/fsphp",
  "description": "Full Stack PHP Developer",
  "minimum-stability": "stable",
  "authors": [
    {
      "name": "Robson V. Leite",
        "email": "cursos@upinside.com.br"
        "homepage": "https://upinside.com.br",
        "role": "Developer"
    }
  ],
  "config": {
    "vendor-lib": "vendor"
  },
  "autoload": {
    "psr-4": {
      "Source\\": "source/"
    }
  }
}
```



Robson V. Leite

CEO UpInside Treinamentos
cursos@upinside.com.br
www.upinside.com.br

<?= "page X"; ?>

Hypertext Preprocessor

É uma linguagem interpretada livre que atua do lado do servidor, tem seu melhor uso no desenvolvimento de aplicações web dinâmicas.

Figura entre as primeiras linguagens passíveis de inserção em documentos HTML e é uma linguagem extremamente modularizada. O que a torna ideal para servidores web.

O PHP é utilizado em aplicações como Facebook, Drupal, Joomla, WordPress, Magento, Oscommerce, Laravel, Synfony, Zend Framework e muitos outros...

<?= Rasmus Lerdorf

Autor da primeira versão do PHP e co-autor das versões seguintes. ;?>

<?= Andi Gutmans and Zeev Suraski

Fundadores da Zend Technologies e principais responsáveis pelo desenvolvimento do PHP. ;?>

1994/1995
PHP 1 e 2

O PHP é criado por Rasmus para ser usado no desenvolvimento de sua página pessoal, era apenas um conjunto de scripts que foi chamado de **P**ersonal **H**ome **P**age Tools.

1997/1998
PHP 3

Andi e Zeev entram no time para reescrever a base do PHP como uma linguagem completa de programação e vários desenvolvedores passam a colaborar com o desenvolvimento do PHP.

2000/2003
PHP 4

A versão 3 é totalmente abandonada e o PHP 4 é desenvolvido com recursos de OO. O que atraiu milhares de desenvolvedores para a linguagem.

2004/2014
PHP 5

Ocorre a acensão do PHP com a versão 2 do zend engine, agora totalmente orientado a objetos, banco de dados com PDO, jSON nativo, namespaces, interfaces, traits e diversos outros recursos que solidificaram o PHP.

2015/20XX
PHP 7

Com performance surpreendente e fortificação em recursos de OO, o PHP é visto como uma linguagem madura presente em mais de 83.5% dos sites na internet sendo a mais utilizada linguagem de programação do mundo.

upinside

upinside

Mercado e desenvolvimento com PHP!

- ✓ Milhares de informações estão disponíveis para todos que querem aprender PHP.
- ✓ Diversos padrões de projetos e infinitas formas de desenvolver com PHP. É como programar sem restrições.
- ✓ Um mercado gigante formado pelo ecossistema, ter um projeto em PHP significa pertencer a evolução de maior potencial entre as linguagens.
- ✓ Com novas tendências de mercado sendo atendidas por poderosos recursos do PHP, surgem novas e incríveis possibilidades de implementação.
- ✓ Milhares de componentes, bibliotecas, APIs, sistemas comunicáveis começam a surgir.

- ✗ A grande maioria dessas informações trás recursos obsoletos e más práticas, gerando ainda mais confusão para antigos e novos programadores.
- ✗ Não existe compatibilidade e comunicação, cada sistema segue um padrão e usa recursos diferentes.
- ✗ Empresas, startups, micro serviços, APIs, marketplaces começam a enfrentar problemas para se adaptar e garantir o ciclo de vida da aplicação.
- ✗ Escala, rotatividade ou formação de equipe com alta curva de aprendizagem.
- ✗ Cresce a necessidade de ter um desenvolvimento interoperável no o PHP.

upinside

INTEROPERABILIDADE

É a capacidade de um sistema de se comunicar de forma transparente ou o mais próximo disso com outro (sistema, componente, API)…

…por meio de open standard ou ontologies.

$openStandard =

Interop. técnica

São **padrões abertos**, livres e disponíveis para acesso e implementação que independem de royalties, outras taxas ou discriminação de uso.

$ontologies =

Interop. semântica

São ontologias com modelos de dados que representam um conjunto de conceitos possuindo o domínio da aplicação e os relacionamentos entre ela.

PADRÕES ABERTOS

upinside

✓ /Design Patterns

De criação	**Singleton**	**Factory**	**Prototype**	**and more...**
Estruturais	**Adapter**	**Facade**	**Decorator**	**and more...**
Comportamentais	**Memento**	**Observer**	**Strategy**	**and more...**

✗ /Standard Recomendation

As recomendações	**Livros**	**Cursos**	**Tutoriais**	**and more...**

Qual recomendação devo seguir para:

O PHP evoluiu com muitos padrões de projeto…

…mas sem um padrão de desenvolvimento e uma recomendação oficial de codificação.

- ✗ Escrever meu código? (Namespaces?, Classes?, **Variáveis??**)
- ✗ Estruturar meu sistema? (Comportamento?, Dependências?)
- ✗ Desenvolver micro serviços? (Padrão de projeto?, Abstração?)
- ✗ Garantir o ciclo de vida? (Implementação?, Manutenção?)
- ✗ Usar, criar, compartilhar? (Componentes?, Bibliotecas?, APIs?)

ONTOLOGIAS

Com o PHP sem recomendações oficiais de desenvolvimento, as ontologias ganharam força no mercado:

Manutenção garantida no domínio da ontologia.

Aprendizado focado em regra e não em tecnologia.

É interoperável em componentes no domínio da ontologia

Domínio total da aplicação dentro de um micro ambiente PHP

/FRAMEWORKS

Uma abstração que une rotinas, componentes e códigos comuns entre diversas aplicações provendo funcionalidades genéricas do projeto.

Laravel

Extremamente versátil e robusto, segue o padrão MVC e é considerado o framework PHP mais usado no mundo.

ZEND FRAMEWORK

Uma coleção de pacotes PHP 100% orientado a objetos apoiado pelo Google e MS e patrocinado pela Zend.

Symfony

Criado para trabalhar com metodologias ágeis, focado em regras de negócio para aplicações mais robustas.

/CMS`s

Um painel de controle completo, diversos plugins e temas prontos, é uma aplicação que pode ser instalada, personalizada e entregue.

WordPress

O WordPress é apontado como o maior CMS de toda a internet usado por mais de 31.1% dos sites hospedados.

Joomla!®

Não tão popular quanto o WP pela sua curva de gestão, mas mais personalizável, roda 3.1% dos sites hospedados.

Drupal™

Uma combinação de APIs e módulos. É o mais avançado entre eles por dar maior controle sobre a aplicação. (2.1%)

Lançamento do PHP 7
Falta de recomendações
Confusão no PHP

2015

Acensão dos Frameworks

Laravel considerado o Framework mais utilizado do mundo.

Publicação Oficial PHP-FIG
PHP Standard Recommendation
PHP do jeito certo

2016

Diversas ontologias lançadas no mercado sem interoperabilidade entre eles…

Popularização das PSRs
Acensão do PHP

2017

Confusão nos Frameworks

Frameworks de Mercado

Interop. Técnica

2018

PHP-FIG

php-fig.org

Os padrões recomendados do PHP começaram a surgir em 2009 com a criação do **PHP Framework Interop Group** (PHP-FIG) e ganhou força em 2016 com a publicação oficial das recomendações.

Formado por fundadores dos principais frameworks PHP, o objetivo é fornecer uma base técnica comum para a implementação de conceitos e ótimas práticas de programação afim de garantir a interoperabilidade técnica entre projetos, componentes, bibliotecas e APIs PHP.

PHP Standard Recommendations (PSRs) são especificações de desenvolvimento com PHP publicadas pelo PHP Framework Interop Group.

PHP
The Right Way

phptherightway.com

Um dos grandes problemas do PHP é a quantidade de informações obsoletas, inseguras e com más práticas disponíveis na web dadas pela popularidade do PHP.

Não existe uma maneira canônica de usar PHP, mas hoje temos uma referência rápida e fácil de ler, introduzindo desenvolvedores às melhores práticas, padrões de código e links para tutoriais competentes.

Esse é o PHP do Jeito Certo.

Criado por Josh Lockhart e publicado também em 2016, é uma iniciativa popular na comunidade PHP que incentiva boas práticas e dissemina informações confiáveis e de qualidade para desenvolvedores PHP.

Com a chegada dos padrões da comunidade e os novos recursos do PHP temos a liberdade de criar aplicações profissionais e personalizadas utilizando componentes poderosos, temos acesso a um ambiente produtivo e colaborativo, e ainda garantimos a interoperabilidade do projeto sem a necessidade do uso de frameworks ou CMS's.

Comunicação global de serviços entre aplicações.

Garante a rotatividade de desenvolvedores.

Possibilita escalar a implementação e manutenção do sistema.

Um ecossistema repleto de projetos e empresas a serem atendidas.

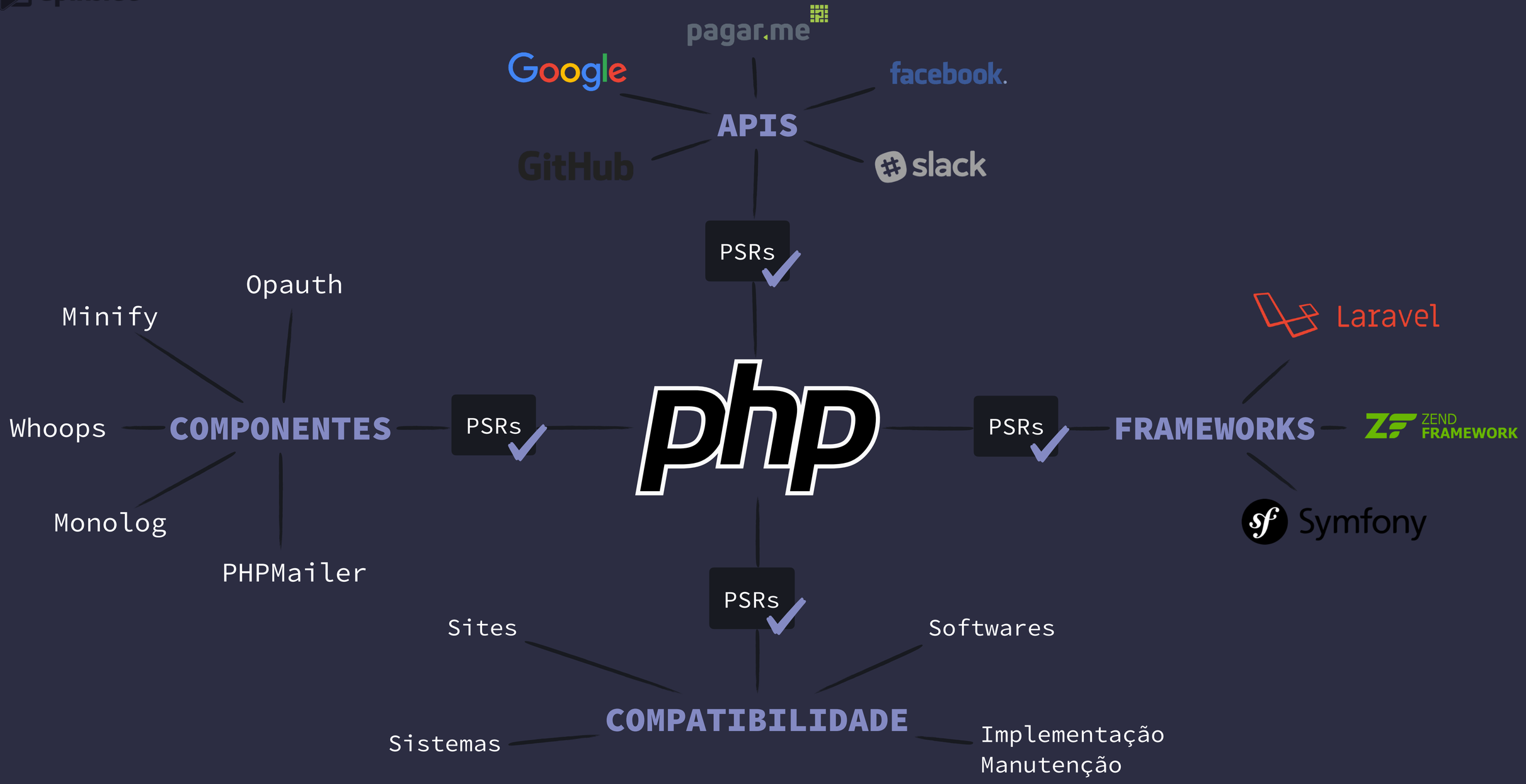
Um mercado com milhares de componentes disponíveis.

<?= COMPOSER:
O gerenciador de dependências do PHP permite usar e gerenciar os componentes da comunidade. **;?>**

<?= PACKAGIST:
É o repositório oficial do Composer onde você pode acessar milhares de componentes. **;?>**

<?= GIT:
Nos permite ter controle total sobre nossa aplicação de forma auditada para trabalhar sozinho ou em equipe. **;?>**

upinside


pagar.me
Google
facebook.
APIS
GitHub
slack
PSRs
Opauth
Minify
Laravel
Whoops
COMPONENTES
PSRs
php
PSRs
FRAMEWORKS
ZEND FRAMEWORK
Monolog
Symfony
PHPMailer
PSRs
Sites
Softwares
COMPATIBILIDADE
Sistemas
Implementação
Manutenção

2009

Curso em DVD Site Dinâmico com PHP

2011

Plataforma EAD Curso PRO PHP

2014

Work Series PHP Orientado a Objetos

php FULL STACK developer

Full Stack PHP Developer

Quarta geração de cursos PHP da UpInside, a formação Full Stack com core em PHP é desenvolvida em três pilares fundamentais que trazem conhecimento sem código legado ou adaptação de linguagem! Aplica fundamentos aderentes ao PHP Standard Recommendation desde a primeira linha de código em estudo:

/EXPAND

O primeiro pilar é onde você APRENDE o PHP do básico, mas já utilizando recursos modernos filtrando aquilo que você realmente precisa aprender para aplicar rotinas, resolver problemas e criar soluções.

/SYNTHESISE

O segundo pilar de conhecimento é onde você vai CONSTRUIR aplicações sintetizando tudo que aprendeu em bibliotecas e componentes para reutilizar sempre que necessário em qualquer projeto.

/DEVELOPMENT

O terceiro pilar é onde você vai DESENVOLVER na prática e montar esse quebra cabeças programando uma série de projetos práticos e reais.

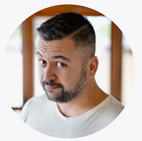
Full Stack Developer

A pilha de tecnologias de um profissional preparado para o mercado e desenvolvimento moderno.

https://fullstackphpdeveloper.com

upinside
vem aprender com a gente!

```
{
  "name": "UpInside/fsphp",
  "description": "Full Stack PHP Developer",
  "minimum-stability": "stable",
  "authors": [
    {
      "name": "Robson V. Leite",
        "email": "cursos@upinside.com.br"
        "homepage": "https://upinside.com.br",
        "role": "Developer"
    }
  ],
  "config": {
    "vendor-lib": "vendor"
  },
  "autoload": {
    "psr-4": {
      "Source\\": "source/"
    }
  }
}
```



Robson V. Leite

CEO UpInside Treinamentos
cursos@upinside.com.br
www.upinside.com.br

<?= "page X"; ?>

FULL-STACK DEVELOPER

É o profissional apto a trabalhar tanto no Back-End quanto no Front-End de uma aplicação, compreendendo de forma razoável toda a 'stack' necessária para desenvolver um projeto web desde o planejamento até a execução.

Core Stack: Um profissional full-stack não é um especialista em toda a pilha de tecnologias, mas as domina o suficiente para desenvolver ou liderar equipes de desenvolvimento. O core Stack é a especialidade deste profissional.

FS Web Developer: É o profissional Full-Stack com CORE stack em sistemas web.

FS PHP Developer: É o profissional Full-Stack com CORE stack em PHP.

/FRONT-END:

Responsável por trabalhar com a parte da aplicação que interage diretamente com a experiência do usuário.

HTML, CSS, Design Responsivo, e JS são as linguagens comuns na 'stack' de desenvolvimento do front-end, que deve interpretar o design e preparar a interface.

/BACK-END:

Responsável pela implementação da regra de negócio, PHP e SQL são as principais ferramentas para programar funcionalidades, componentes e desenvolver a aplicação.

O Back-End programa a interface, implementa e testa a aplicação para entregar ao usuário final.

UMA PILHA
FRONT-END

<
HTML ou linguagem de marcação de hipertexto é o formato padrão para criação de páginas e aplicações web.

Em conjunto com o CSS e JS formam as pedras principais para a World Wide Web
/>

{
CSS ou folha de estilo em cascata e um mecanismo para adicionar todos os estilos (cores, fontes, efeitos, etc.) a documentos web.

Assim como o HTML é uma das linguagens insubstituíveis.
}

@media(
Design responsivo é uma técnica de marcação e estilo que garante a web única (one web) ao permitir que se crie sites que se ajustem a qualquer dispositivo independentemente do tamanho da tela.
);

$(function(){
O jQuery é a biblioteca JavaScript multi-plataforma mais utilizada do mundo.

Projetada para simplificar e agilizar o desenvolvimento e garantir a compatibilidade de códigos JS entre navegadores.
});

upinside

UMA PILHA
BACK-END

<?= "PHP é o pré-processador de hipertexto mais utilizado do mundo.

Figura entre as primeiras linguagens de programação passíveis de inserção HTML capaz de gerar conteúdo 100% dinâmico.**" ;?>**

{ Composer é o gerenciador de componentes a nível de aplicação do PHP criado para o PHP.

Fornece um formato padrão e autômato para gerenciar e controlar todas as dependências de um projeto e/ou bibliotecas PHP. **}**

<commit> Poderoso sistema de controle de versões.

GIT permite rastrear e auditar mudanças no projeto sempre visando a velocidade individual ou em equipe, e a integridade de dados com suporte a fluxo de trabalho distribuído. **<git>**

SELECT * FROM SQL ou linguagem de consulta estruturada é a linguagem de pesquisa padrão de um banco de dados relacional.

Utilizada pelo PHP para cadastrar (Create), ler (Read), atualizar (Update) e deletar (Delete) dados de forma otimizada em um banco de dados **;**

upinside

DESENVOLVIMENTO

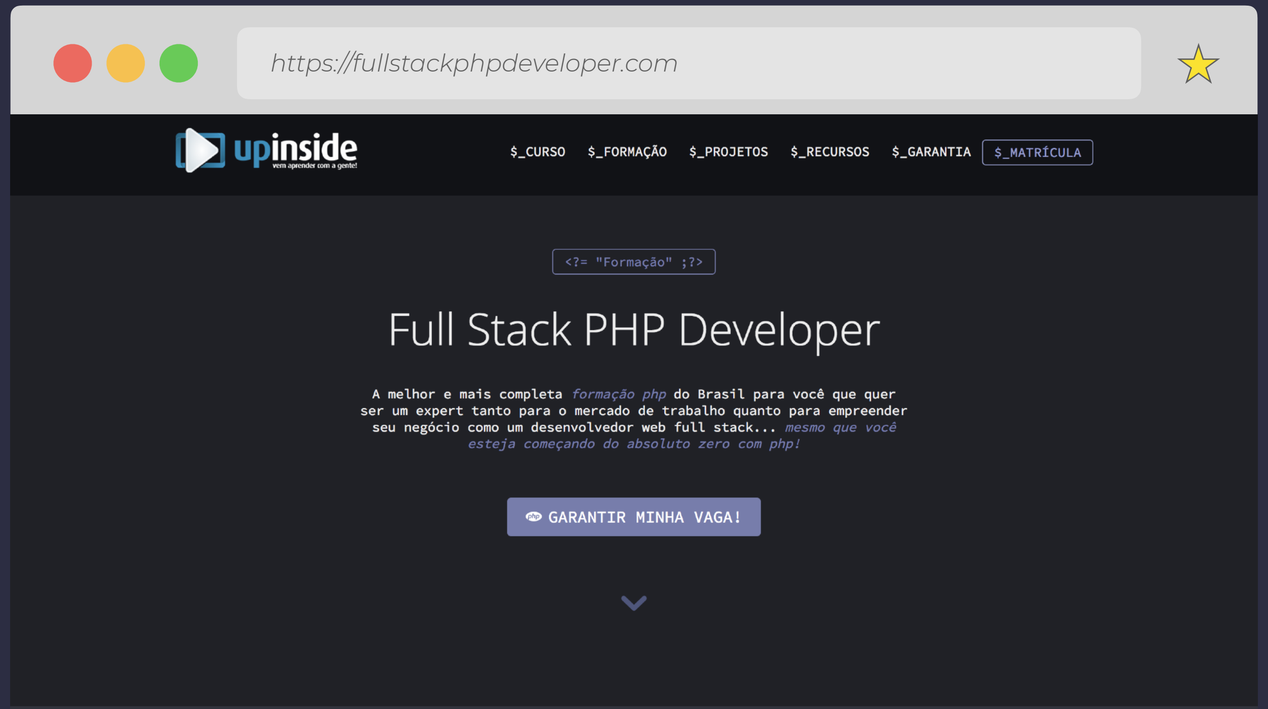
POR STACK

DESIGN E UX:

Prepara o padrão visual do website e projeta o design com base na experiência do usuário.

FRONT-END:

Realiza a marcação do HTML e do CSS, prepara os efeitos e eventos do website.



BACK-END:

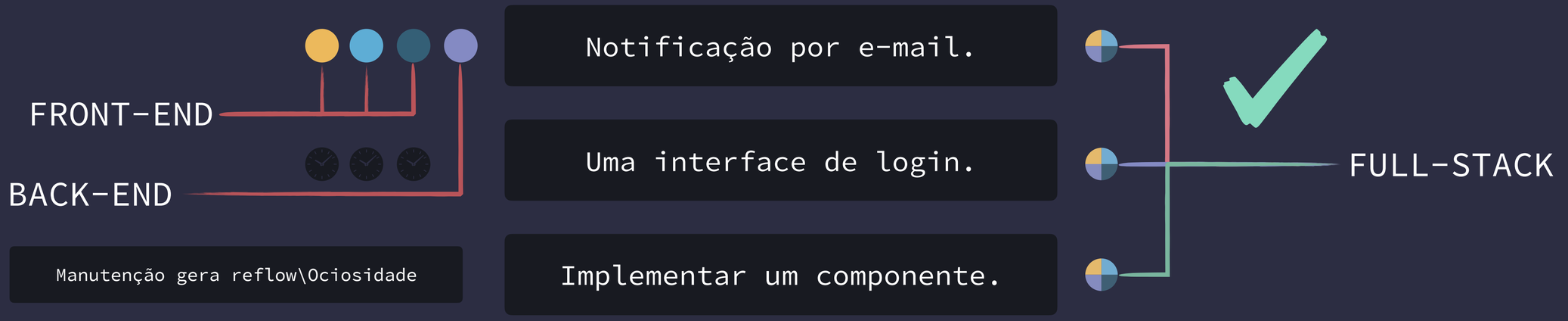
Programa as funcionalidades do website de acordo com a regra de negócios.

DESENVOLVIMENTO

POR COMPONENTE

DESIGN E UX:

Prepara o padrão visual do website e projeta o design com base na experiência do usuário.



Fluxo: Front-End prepara a interface e manda ao Back-End que programa, testa e coloca em produção

Fluxo: O Full-Stack programa a interface, implementa, testa e coloca em produção.

OPORTUNIDADES

DE MERCADO

/CLT/Contrato

Como CLT trabalhando para uma empresa específica ou no modelo de contrato prestando serviços para uma ou mais empresas como Back, Front ou Full.

/Freelancer/Agência

Desenvolvendo projetos sob encomenda em contato direto com o cliente final elaborando todas as etapas para entregar uma solução.

/Startups

Idealizar e desenvolver um novo modelo de negócios que seja escalável e repetível com investimento próprio ou investimento-anjo.

/MicroServiços

Existem diversas camadas de serviços em qualquer negócio que precisam de boas ferramentas, você pode desenvolvê-las.

/SaaS

Ferramenta entregue no modelo de assinaturas de software como serviço, que resolvem problemas específicos para pessoas ou empresas.

/Marketplaces

Um mercado gigante de aplicações criadas com base em soluções para aplicações maiores onde você desenvolve para outras plataformas.

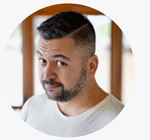
https://fullstackphpdeveloper.com

Um arquivo em PHP

Os fundamentos, boas práticas e arquitetura de um programa criado em um arquivo PHP.

upinside
vem aprender com a gente!

```
{
  "name": "UpInside/fsphp",
  "description": "Full Stack PHP Developer",
  "minimum-stability": "stable",
  "authors": [
    {
      "name": "Robson V. Leite",
        "email": "cursos@upinside.com.br"
        "homepage": "https://upinside.com.br",
        "role": "Developer"
    }
  ],
  "config": {
    "vendor-lib": "vendor"
  },
  "autoload": {
    "psr-4": {
      "Source\\": "source/"
    }
  }
}
```



Robson V. Leite

CEO UpInside Treinamentos
cursos@upinside.com.br
www.upinside.com.br

<?= "page X"; ?>

Um arquivo PHP:

Como sabemos o PHP é um pré-processador de hipertexto, um arquivo com a extensão PHP tem uma estrutura dinâmica incrível, capaz de interpretar e processar HTML, CSS, JavaScript, além é claro do próprio PHP.

https://fullstackphpdeveloper.com

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="…">

   <title>Para isso funcionar a extensão deste arquivo deve
   ser .php</title>

    <style>
        /* Seu CSS funciona aqui! */
    </style>
</head>
<body>
<?php
   //Seu PHP funciona aqui, e em qualquer lugar deste
   arquivo!
?>
<script>
   // Seu JavaScript funciona aqui!
</script>
</body>
</html>
```

Muito se fala que não devemos misturar códigos PHP com HTML, é um mito!

HTML é hipertexto marcado e PHP um pré-processador de hipertexto que foi criado exatamente com esse objetivo.

Trabalhar com ambos faz total sentido e é incrível a produtividade obtida assim.

Mas claro, você precisa seguir as boas práticas de codificação.

Quando separar?

* Arquivos JS e CSS devem ser separados sempre, essa é uma ÓTIMA regra de boas práticas.

* Regras e lógicas de negócio devem ser separadas de visões e interfaces visuais.

* Arquivos e trechos que precisam ser reutilizados também precisam ser separados.

O que você não deve fazer:

O PHP é permissivo ao extremo, com isso você pode construir qualquer coisa boa na mesma medida que ruim. Vejamos algumas práticas a serem evitadas:

```
echo "<header>
   //header Content…
</header>";

require "header.php";
```

REPETIÇÃO:

Qualquer projeto web terá inevitavelmente repetição de estruturas a serem apresentadas. Mas repetir a estrutura não significa repetir o código.

Substitua repetição por reuso em códigos que serão utilizados mais de uma vez na aplicação.

```
function showName($name) {
   return "<p>{$name}</p>";
}
echo showName("Robson V. Leite");

require "functions.php";
echo showName("Robson V. Leite");
```

DECLARAÇÃO COM SAÍDA:

Declarar uma funcionalidade ou configuração em um arquivo com saída impede de reutilizar essa funcionalidade em outras camadas da aplicação.

Declare todas as configurações e funcionalidade em arquivos separados, sem saída e que possam ser requeridos.

```
$q = $pdo->query("SELECT * FROM users");
foreach($q->fetchAll() as $r) {
   //Results looping for SQL Query
}

foreach($user->listAll() as $r){
  //Results looping for UserModel
}
```

MATERIALIZAÇÃO:

Na correria de um projeto é comum ligar o piloto automático e aplicar uma programação funcional ignorando a regra para aplicar funcionalidades.

Prefira criar modelos e padrões reutilizáveis para ter um ponto de acesso único para implementação e manutenção.

Um arquivo PHP:

Don't **R**epeat **Y**ourself: Ao ser aplicado torna partes do sistema independentes. Cada parte pode mudar de forma previsível e uniforme, mantendo-se, portanto, sincronizadas com toda a aplicação.

https://fullstackphpdeveloper.com

```php
<?php
  require __DIR__."/config.php";
  require __DIR__."/vendor/autoload.php";
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    //head data
   <link rel="stylesheet" href="style.css"/>
</head>
<body>
<?php
require "header.php";

/* Uma lógica aplicada afim de decidir o conteúdo
 * a ser carregado. Podendo interagir com a lógica
 * e regra de negócios para criar as visões */

require "footer.php";
?>
<script src="scripts.js"></script>
</body>
</html>
```

Esse é um arquivo index.php comum em aplicações web, contendo apenas a regra de construção visual e se comunicando com tudo que precisa para processar e entregar a aplicação para o usuário.

* Ele começa requerendo as configurações globais do projeto config.php e faz uso do autoload.php de classes do Composer para ter todos os recursos disponibilizados.

* O CSS vem do arquivo externo style.css possibilitando formatar o visual em um ponto único na aplicação.

* Ele requer o header.php, abre um bloco lógico para processar o conteúdo e requer o footer.php.

* Por fim, ele requer o scripts.js em um arquivo externo como o CSS, onde todos os eventos estarão disponibilizados.

Outros arquivos PHP:

Em um projeto web com PHP, diversos arquivos serão criados com diferentes finalidades. Todos eles fazem parte da sua aplicação web. Como por exemplo:

Classe.php

```
/**
 * Um arquivo de classe PHP,
 * cada classe deve ter seu
 * próprio arquivo.
 */
```

Trait.php

```
/**
 * Os traits em PHP servem para
 * definir comportamentos e
 * recursos comuns em classes.
 */
```

Interface.php

```
/**
 * A interface é um contrato de
 * implementação da classe.
 */
```

Classes, traits e interfaces possuem a mesma estrutura base.

functions.php

```
/**
 * Um arquivo de funções que pode
 * ser incluído em qualquer outro
 * que precise usar essas funções.
 */
```

config.php

```
/**
 * Configurações gerais como
 * ini_sets e constantes que podem
 * ser incluídas globalmente.
 */
```

Includes.php

```
/**
 * Trechos de código que serão
 * reutilizados no projeto, como
 * headers, sidebars e footers.
 */
```

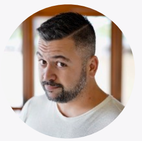
Seu guia de estudos

Como aprender os fundamentos na prática e acelerar seu aprendizado no Full Stack PHP Developer

https://fullstackphpdeveloper.com

upinside
vem aprender com a gente!

```
{
  "name": "UpInside/fsphp",
  "description": "Full Stack PHP Developer",
  "minimum-stability": "stable",
  "authors": [
    {
      "name": "Robson V. Leite",
        "email": "cursos@upinside.com.br"
        "homepage": "https://upinside.com.br",
        "role": "Developer"
    }
  ],
  "config": {
    "vendor-lib": "vendor"
  },
  "autoload": {
    "psr-4": {
      "Source\\": "source/"
    }
  }
}
```



Robson V. Leite

CEO UpInside Treinamentos
cursos@upinside.com.br
www.upinside.com.br

<?= "page X"; ?>

Formação Full Stack PHP Developer

O 'stack full' disponível para você estudar, aprender e se certificar em todas as áreas de forma individual e completa.

**

DevTools

<?= “Ferramentas, servidor, domínio e hospedagem” ;?>

*

HTML E CSS

<?= “HTML5, CSS e CSS3, Design Responsivo.” ;?>

*

JS com jQuery

<?= “JS, eventos, efeitos e AJAX com jQuery.” ;?>

*

Fs PHP Dev

<?= “PHP, SQL com PDO, padrões da comunidade” ;?>

*

Git/Composer

<?= “Controle de versões e de dependências” ;?>

Como estudar no curso

Uma forma comum que tem acelerado o processo de nossos alunos garantindo o melhor aproveitamento no curso e o melhor resultado no aprendizado.

/EstruturaDoCurso

/Módulo 1
  * Aula 1
  * Aula 2
  * Aula 3
  * Aula 4

/MÓDULO: É como são organizados os conteúdos do curso levando em consideração o seu processo de evolução.

Um módulo nunca terá pendências de conteúdo antes dele, mas pode ter depois, por esse motivo você nunca deve pular módulos.

/AULA: Assim como os módulos as aulas também não devem ser puladas pois são projetadas em um passo a passo de evolução, em sua grande maioria 100% práticas.

É onde você aprende o fundamento enquanto executa o exercício.

/ComoEstudar

1

Contexto: Apenas assista todas as aulas do módulo prestando atenção no conteúdo para entender o contexto e os resultados desejados.

2

Exercício: Volte a primeira aula do módulo e venha fazendo junto, pausando a aula se e quando preciso para desenvolver o exercício.

3

Documentação: Faça uma terceira vez sem usar tanto a aula para testar na prática o que aprendeu.

Suporte e certificados:

Saiba como usar o suporte um a um para tirar suas dúvidas ao mesmo tempo que colabora com a turma FSPHP e veja mais sobre os certificados e a conquista da formação.

```
Use Source\Model\Course
```

/AbraUmTicket

Abaixo de cada aula você encontra o canal de suporte um a um. Basta enviar sua dúvida sobre o conteúdo da aula e aguardar a resposta em até 48 horas úteis. (geralmente feita no mesmo dia)

IMPORTANTE: Antes de abrir um ticket verifique se o mesmo é sobre a aula, use uma boa escrita e dê detalhes sobre o problema.

/PorQueRecusamos

Consideramos os tickets como conteúdo adicional de estudo do curso, ao seguir essas recomendações você também colabora com o restante da turma.

/Certificados

Os certificados da UpInside são válidos em território nacional para qualquer empresa ou instituição, e tem reconhecimento em mais de 17 países da America Latina pelos prêmios LAQI.

Seu certificado pode ser emitido assim que o requisito de estudo for concluído. Por esse motivo é importante assistir todas as aulas pelo menos uma vez, tendo todas as tarefas marcadas como concluídas.

Ao concluir todos cursos da formação e emitir seus certificados você recebe a conquista Full Stack Developer da UpInside.

```
$course = new Course();
```

/EstruturaDeExemplos:

/Curso
  /Módulo
    /Material
    Aula

/Exercícios FSPHP
  /01-ola-mundo-vamos-comecar
    /01-06-iniciando-um-projeto
      /source
      /assets
      index.php

/Projetos FSPHP
  /source
  /vendor
  /assets
  index.php