



<?= Imergindo no universo do PHP ;?>

FORMAÇÃO FULL STACK PHP DEVELOPER



www.upinside.com.br

```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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



```
DEFINE("CLASSNAME", "Imergindo no universo do PHP");
```

Uma linha de evolução:

Criada por Rasmus para monitorar visitas em seu currículo online, era apenas um conjunto de scripts que foi chamado de **Personal Home Page Tools**, em resumo eram ferramentas que ofereciam variáveis básicas com interpretação automática de formulários usando sintaxe com HTML embutido.

Em 1994: Rasmus, Andi e Zeev se uniram para reescrever e transformar o PHP em uma linguagem de programação completa com sintaxe consistente e suporte básico á orientação a objetos.

Em 1998: o PHP3 foi lanudo como **PHP: Hypertext Preprocessor** já oferecendo diversas extensões a banco de dados, protocolos e APIS.. isso atraiu muitos novos desenvolvedores e no final de 1998 o PHP já estava presente em mais de 10% dos servidores web!

Hoje o PHP é utilizado por mais de 83.5% dos sites no mundo e conta com uma gama completa de recursos.

Seu suporte é feito e garantido por dezenas de desenvolvedores de núcleo em todo mundo, padrões de desenvolvimento da comunidade garantem a interoperabilidade entre projetos e componentes graças ao **PHP Framework Interop Group** (PHP-FIG).

Namespaces, Composer, Git, uma comunidade enorme e milhares de componentes disponíveis libertam e entregam a você a mais completa e poderosa linguagem de programação web.



Robson V. Leite

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



```
<?= "page X"; ?>
```



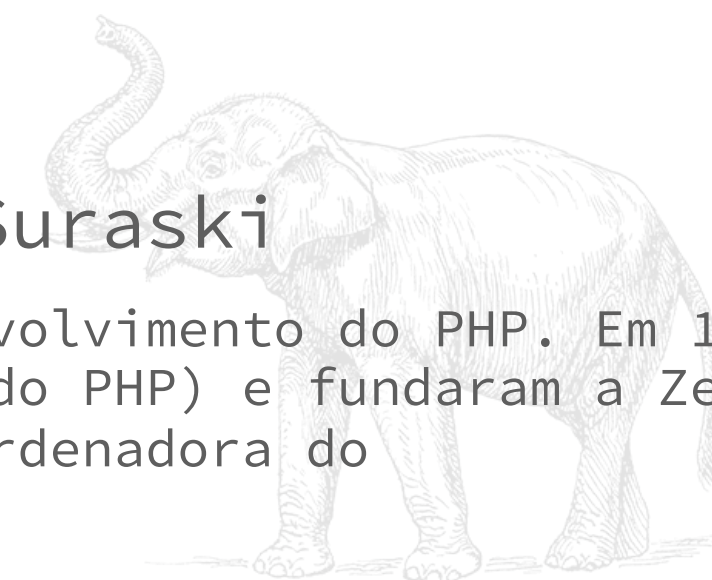
<?= Rasmus Lerdorf

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



<?= Andi Gutmans and Zeev Suraski

Os principais responsáveis pelo desenvolvimento do PHP. Em 1999 eles escreveram o Zend Engine (motor do PHP) e fundaram a Zend Technologies, que desde então é a coordenadora do desenvolvimento no PHP. ;?>





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

DEFINE("CLASSNAME", "Imergindo no universo do PHP");

Sobre frameworks:

Um Framework é uma abstração que une rotinas, componentes e códigos comuns entre diversas aplicações provendo funcionalidades genéricas do projeto. Diferente dos padrões de software, possuem domínio sobre sua aplicação, o que garante eficiência na resolução dos problemas e otimização de recursos.



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



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

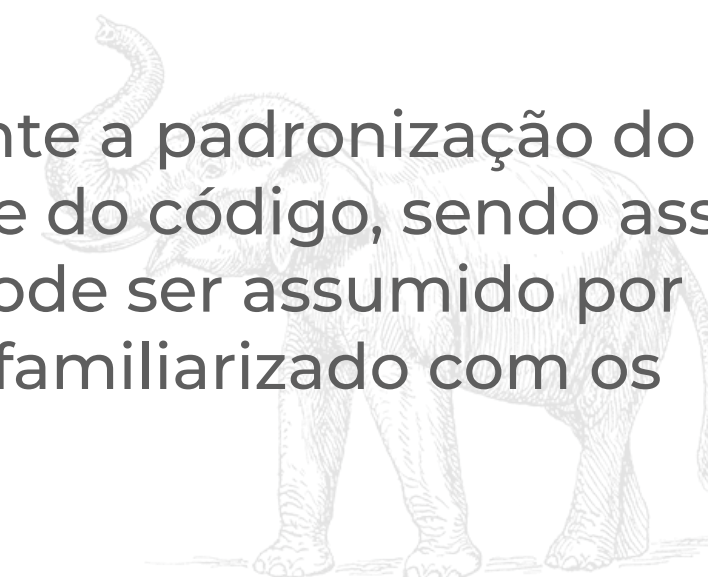


Criado para trabalhar com metodologias ágeis, focado em regras de negócio para aplicações mais robustas.

MITO: Muito se fala que uma das vantagens no uso dos FW é sua curva de aprendizagem, isso é um mito pois além de **saber PHP**, aprender a **regra do negócio**, você ainda terá que aprender a usar o Framework.

VERDADE: A melhor forma de garantir a manutenção de um projeto **até o PHP-FIG** era o uso de um framework uma vez que qualquer programador PHP que saiba usar o mesmo poderia assumir o projeto. E por isso existe um mercado imenso para frameworks, principalmente com relação ao Laravel por ser o mais usado no mundo.

A chegada do PHP-FIG garante a padronização do projeto e a interoperabilidade do código, sendo assim um projeto que siga a PSR pode ser assumido por qualquer programador PHP familiarizado com os padrões da comunidade.





```
DEFINE("CLASSNAME", "Imergindo no universo do PHP");
```

Sobre CMS`s:

Os CMS`s ou **sistemas de gerenciamento de conteúdo** tem como principal característica um painel de controle completo, diversos plugins e temas prontos. É uma aplicação que pode ser instalada, personalizada e entregue. **Os 3 principais e mais utilizados CMS`s do mundo são desenvolvidos com PHP.**



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

É preciso ter alguns cuidados com o desenvolvimento de soluções com CMS`s uma vez que são estruturados de forma genérica para atender ao maior número de funcionalidades possível e acabam sendo pouco otimizados por isso.



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

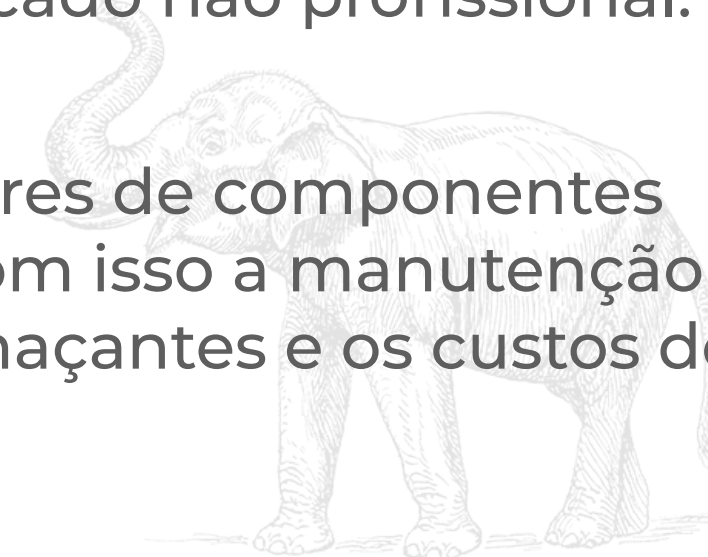
Além disso os CMS`s não atendem os padrões da comunidade deixando totalmente de lado a interoperabilidade. Uma vez feito com um CMS, o projeto raramente conseguirá se desvincular dele.



Uma combinação de CMS com APIs e módulos, o mais avançado entre eles, usado por quem (2.1%) quer ter mais controle sobre a aplicação.

MERCADO: Um dos maiores problemas de usar um CMS`s aberto como ferramenta de entrega está na concorrência desleal de mercado não profissional.

RECURSOS: Por terem milhares de componentes operam com maior carga. Com isso a manutenção e o monitoramento se tornam maçantes e os custos de hospedagem muito maiores.



```
{
  "name": "Full Stack PHP Developer",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "authors": [
    {
      "name": "Robson V. Leite",
      "homepage": "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





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

DEFINE("CLASSNAME", "Imergindo no universo do PHP");

Padrões e componentes:

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 o uso de frameworks.

O que mudou?

O **PHP-FIG** pode ser visto como um manual completo de como desenvolver sua aplicação.



Com código aderente a esse manual você garante a manutenção do projeto por qualquer um que conheça as recomendações.

O MELHOR:

A PSR também possibilita que milhares de desenvolvedores criem componentes para seu projeto sem conhecer uma linha de código do mesmo.



<?= COMPOSER:

O gerenciador de dependências do PHP permite usar e gerenciar os componentes da comunidade. ;?>



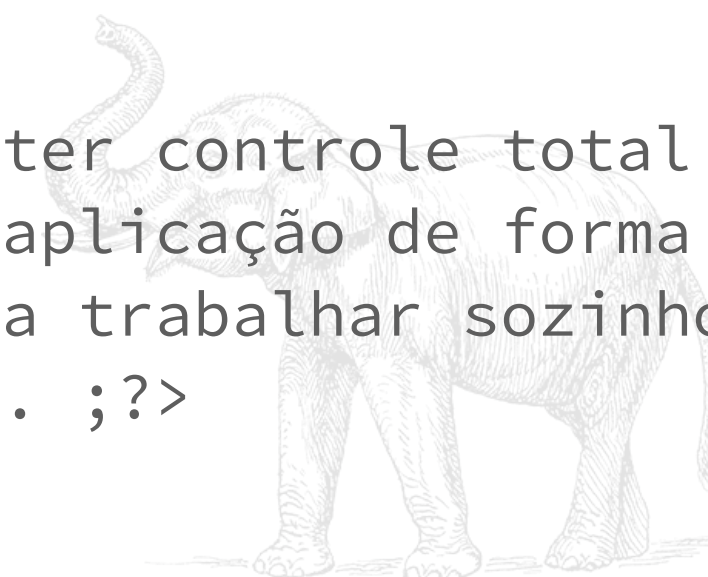
<?= PACKAGIST:

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. ;?>





Explorando o universo Full Stack

FORMAÇÃO FULL STACK PHP DEVELOPER



www.upinside.com.br



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

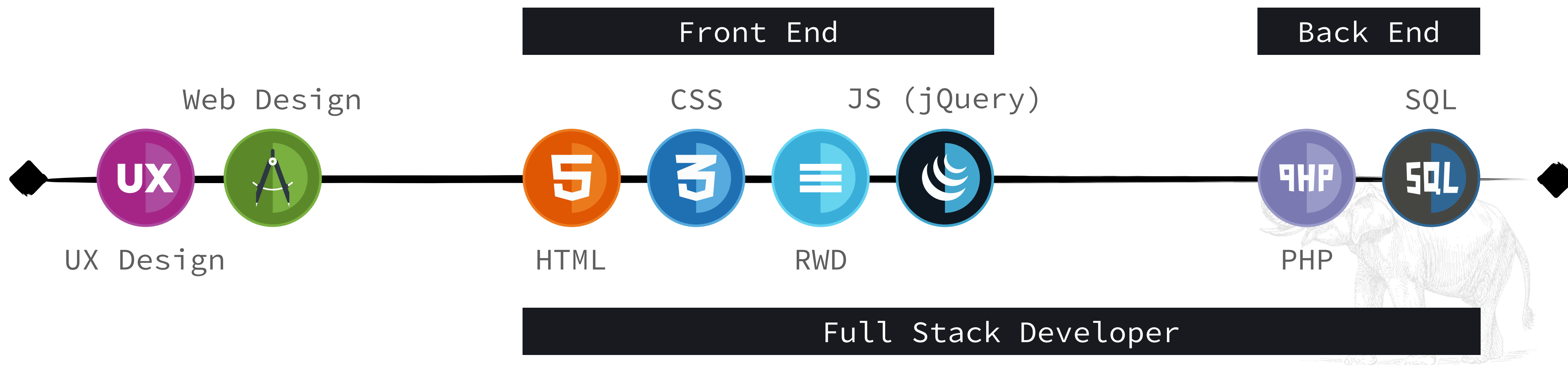
```
DEFINE("CLASSNAME", "Explorando o universo full stack");
```

Full Stack Developer:

Existe uma “PILHA DE TECNOLOGIAS” envolvidas no desenvolvimento de um projeto web, essas pilhas estão especializadas na parte que fala com o usuário (FRONT END) e na parte que resolve a regra da aplicação (BACK END). Uma aplicação web só existe quando unimos ambas...

/Full Stack O termo pilha completa não é tão segmentado quanto Front e Back mas ganhou mercado com a chegada das startups e tem sido cada vez mais requisitado no mercado pela capacidade de planejar, projetar e entregar projetos completos.

O Full Stack é o profissional que atua tanto no **Front** quanto no **Back End** e também costuma ocupar posições de liderança na equipe de desenvolvimento uma vez que está apto a planejar, acompanhar e auditar todas as etapas do projeto.





```
DEFINE("CLASSNAME", "Explorando o universo full stack");
```

Front End:

/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 pilha de desenvolvimento deste profissional que deve interpretar o design e preparar a interface.



< 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 estilo (cores, fontes, efeitos, espaçamentos, etc.) a documentos web.

Assim como o HTML é uma das linguagens insubstituíveis. }

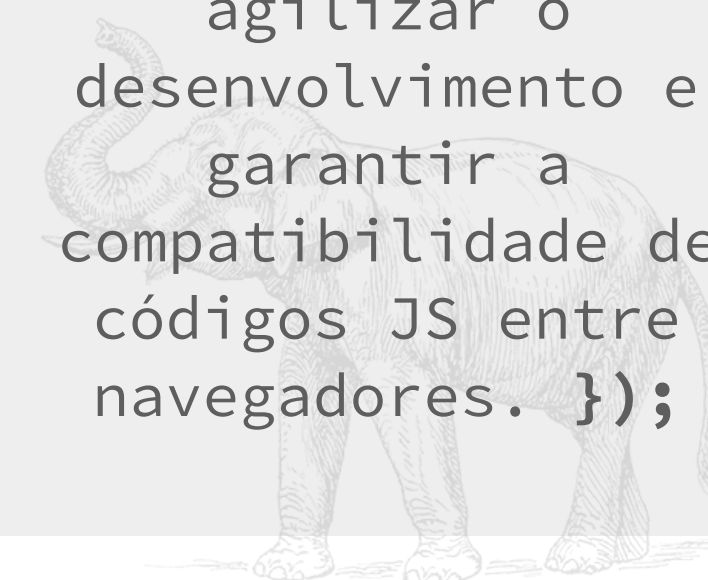


@media(Design responsivo é uma técnica de marcação e estilo que garante a web única (one web) ao permitir que se crie sites que se ajustem a qualquer dispositivo independentemente do tamanho tela.);



\$(function(){ 0 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. });



Robson V. Leite

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



<?= "page X"; ?>



```
DEFINE("CLASSNAME", "Explorando o universo full stack");
```

Back End:

/Back End responsável pela implementação da regra de negócio, PHP e SQL são suas ferramentas para programar funcionalidades, componentes e desenvolver a aplicação projetada no Front End. O Back End programa a interface, implementa e testa a aplicação para entregar ao usuário final.



`<?= "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.`

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



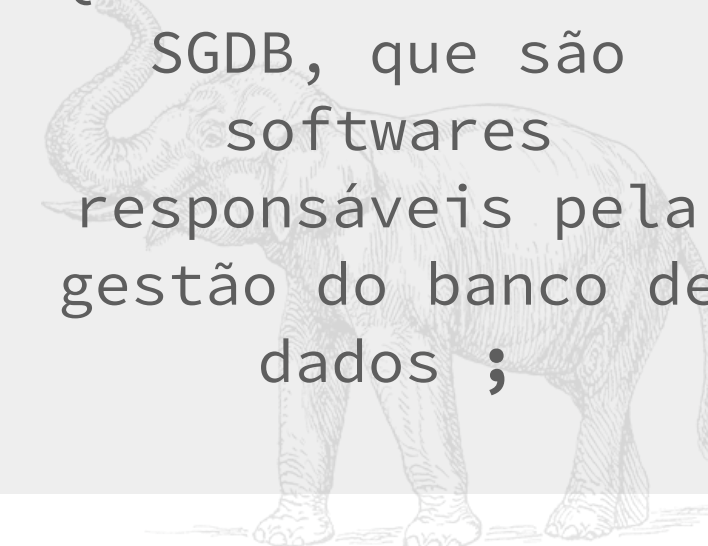
`<commit>` Poderoso sistema de controle de versões.

GIT permite rastrear e auditar mudanças no projeto sempre visando a velocidade e 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.

Que é diferente do SGDB, que são softwares responsáveis pela gestão do banco de dados ;



Robson V. Leite

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



`<?= "page X"; ?>`



```
DEFINE("CLASSNAME", "Explorando o universo full stack");
```

Full Stack PHP:

/FSPHP: Uma posição única de mercado que nasceu pouco tempo depois dos padrões da comunidade serem difundidos, que já é bem popular e tem altos salários no mercado.

Um profissional full stack especializado em PHP e nos paradigmas que permitem desenvolver projetos aderentes a PSR está pronto a garantir a interoperabilidade, manutenção, escalabilidade e comunicação em um projeto com qualquer componente, API ou outros projetos que também sigam tais recomendações.

UMA VERDADE MAL INTERPRETADA: Existe um bandeirismo daqueles que não são full stack de que o mercado não aceita esse profissional por acreditar que ele não pode ser excelente em toda a pilha. Em **parte** verdadeiro, em **parte** falso.

TRUE: É sim muito raro um full stack especializado em toda a pilha, mas para ser um full stack você não precisa ser excelente em cada tecnologia, e sim compreendê-las suficientemente para implementar ou auditar cada uma delas;

FALSE: O mercado não só aceita como procura tais profissionais, que costumam crescer rápido por sua capacidade de execução do começo ao fim de qualquer projeto;

O FSPHP vai te preparar para compreender todas essas camadas e especializar você no PHP e nos padrões da comunidade.



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>



```
DEFINE("CLASSNAME", "Explorando o universo full stack");
```

O mercado:

Muitos desenvolvedores visam apenas dois segmentos de mercado. Atuar como freelancer/agência ou arrumar um emprego na área. Existe um universo maior para você explorar:

/CLT/Contrato

```
<?= "Como CLT trabalhando para uma empresa específica ou no modelo de contrato prestando serviços para uma ou mais empresas." ;?>
```

/MicroServiços

```
<?= "Existem diversas camadas de serviços em qualquer negócio que precisam de boas ferramentas, você pode desenvolvê-las." ;?>
```

/Freelancer/Agência

```
<?= "Desenvolvendo projetos sob encomenda em contato direto com o cliente final elaborando todas as etapas para entregar uma solução." ;?>
```

/Startups

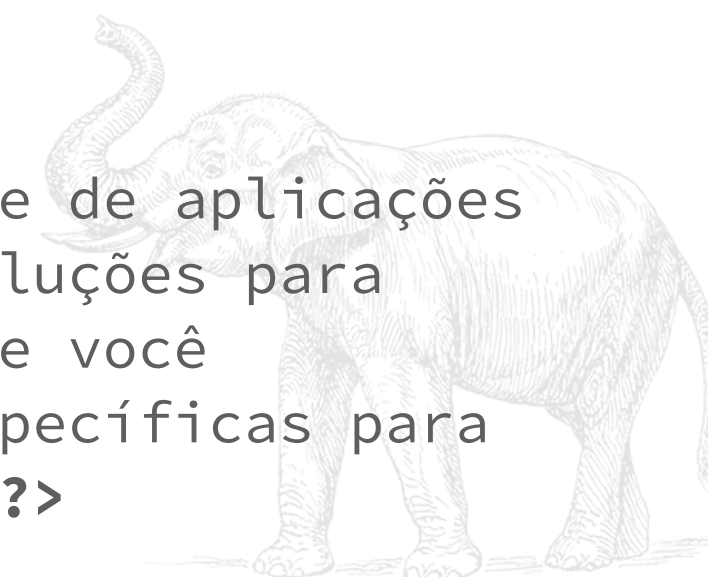
```
<?= "Idealizar, desenvolver e encubar um novo modelo de negócios que seja escalável e repetível com investimento próprio ou investimento-anjo." ;?>
```

/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 soluções específicas para outras plataformas." ;?>
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>
```





Os padrões da comunidade

```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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

FORMAÇÃO FULL STACK PHP DEVELOPER



www.upinside.com.br



```
DEFINE("CLASSNAME", "Os padrões da comunidade");
```

A interoperabilidade:

A interoperabilidade representa a capacidade que um sistema tem de se comunicar de forma transparente com outro sistema semelhante ou não. Um sistema interoperável deve trabalhar com padrões abertos, padrão de modelo de dados ou conjunto de conceitos.

A interoperabilidade serve para garantir o ciclo de vida de qualquer projeto uma vez que um desenvolvedor pode implementar e dar manutenção ao mesmo apenas conhecendo a regra de negócio e o padrão aplicado.

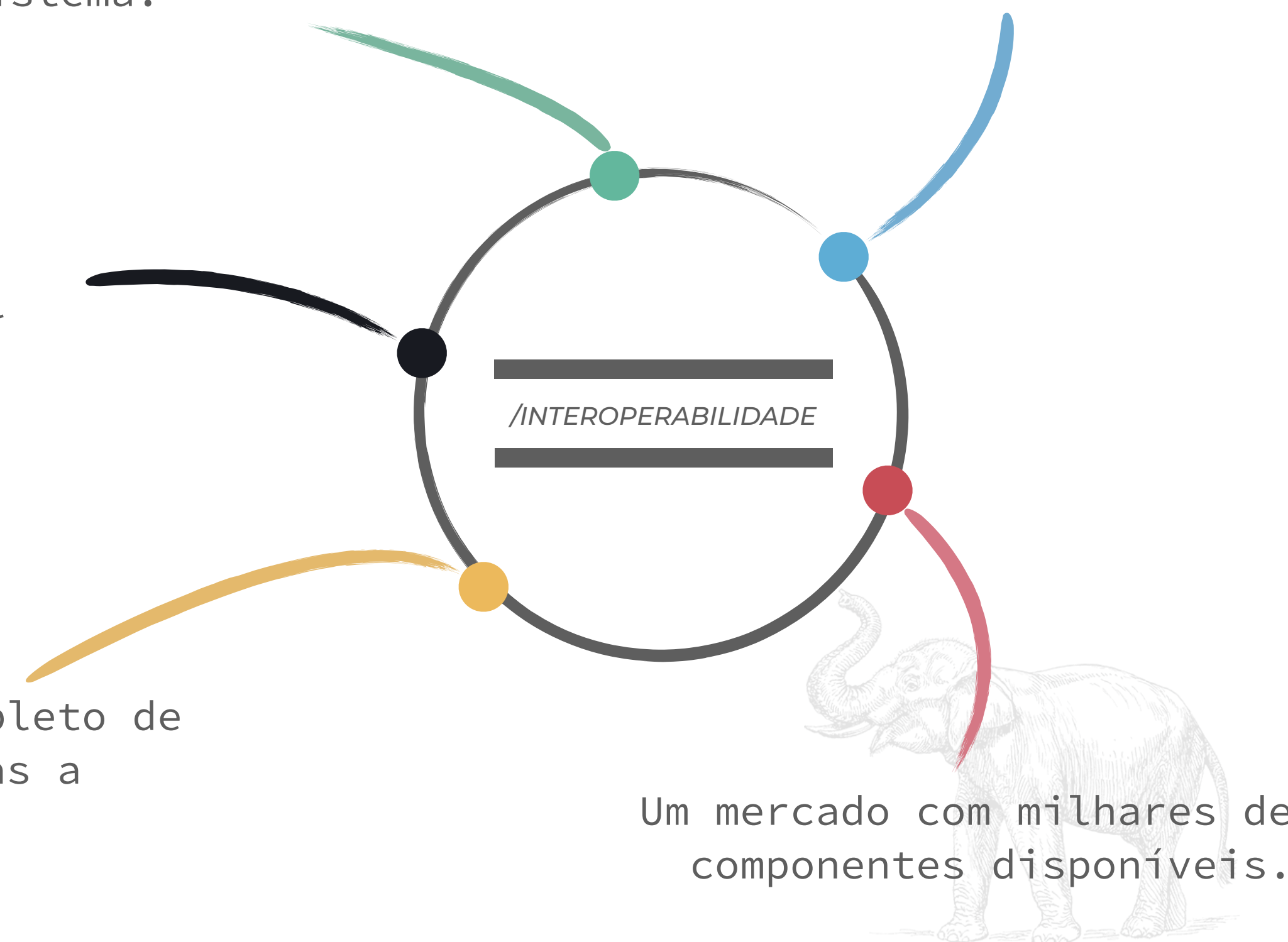
Possibilita escalar a implementação e manutenção do sistema.

Comunicação global de serviços entre aplicações.

Um ecossistema repleto de projetos e empresas a serem atendidas.

Garante a rotatividade de desenvolvedores.

Um mercado com milhares de componentes disponíveis.



Robson V. Leite

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



<?= "page X"; ?>



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

DEFINE("CLASSNAME", "Os padrões da comunidade");

PHP antes do FIG:

/Interoperabilidade semântica e a ascensão dos Frameworks PHP:

O PHP demorou bastante para reconhecer e difundir padrões abertos, e com isso ocorreu a ascensão dos frameworks, que por meio de ontologias e domínio da aplicação garantem a **interoperabilidade semântica** entre projetos, desenvolvedores e componentes.

Com isso empresas passaram a adotar os frameworks como solução para seus negócios uma vez que poderiam contratar qualquer programador ***Laravel** para dar manutenção e implementar seu projeto ***Laravel** sem que ele precise estudar ou refazer o sistema.

Logo a ascensão dos Frameworks está mais relacionada a garantir a interoperabilidade e o ciclo de vida do projeto do que com ser a melhor solução para o mesmo.



O ecossistema do Framework mais utilizado do mundo.

Centenas de componentes e implementações disponibilizadas pela comunidade Laravel.

Um imenso mercado composto por diversas empresas que adotaram o Laravel como solução.

Milhares de desenvolvedores prontos para assumir projetos com Laravel.

Um ponto de comunicação interoperável e global entre aplicações Laravel.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

DEFINE("CLASSNAME", "Os padrões da comunidade");



php-fig.org

PHP
The Right Way

phptherightway.com

As recomendações para o desenvolvimento padrão com PHP surgiram em 2009 com a criação do **PHP Framework Interop Group** (PHP-FIG).

Um grupo formado por diversos fundadores de frameworks PHP onde o objetivo é fornecer uma base técnica comum para a implementação de conceitos e ótimas práticas de programação afim de permitir a interoperabilidade de componentes e projetos com PHP.

O PHP Standard Recommendation (PSR) é uma especificação do PHP publicada pelo PHP Framework Interop Group.

Um dos grandes problemas do PHP era a quantidade de informações obsoletas, inseguras e com más práticas disponíveis na web dada 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 é uma iniciativa popular na comunidade PHP que incentiva boas práticas e dissemina informações confiáveis e de qualidade para desenvolvedores PHP.

<?= Josh Lockhart

Criador do Slim Framework e autor e curador do PHP The Right Way. ;?>





```
DEFINE("CLASSNAME", "Os padrões da comunidade");
```

O PHP hoje e no futuro:

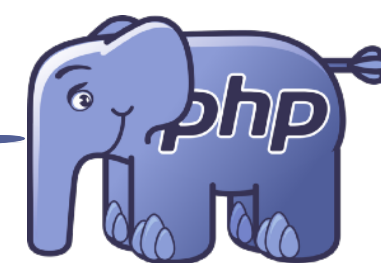
/Interoperabilidade técnica, a recuperação do domínio da aplicação projetada para a melhor solução e a ascensão do PHP.

De 2009 até aqui os padrões da comunidade ganharam muita força, e com isso a implementação de recursos importantes marcaram um novo universo PHP.

Hoje por meio desses padrões é possível garantir a interoperabilidade técnica entre projetos PHP sem o uso de frameworks.

O PHP se libertou do domínio dos Frameworks e entrega ao desenvolvedor um universo modular e componentizado onde as soluções são desenvolvidas com foco na regra e na lógica do negócio ao mesmo tempo que garantem a interoperabilidade.

No futuro, milhares serão milhões... de bibliotecas e componentes PHP.



O ecossistema da maior e mais utilizada linguagem de programação web do mundo.

Padrões da comunidade garantem a interoperabilidade técnica entre os projetos e componentes PHP.

O Composer dá acesso a milhares de componentes disponibilizados pela comunidade PHP.

Milhares de desenvolvedores PHP prontos para assumir qualquer projeto desenvolvido com PHP.

Um ponto de comunicação global e interoperável entre qualquer projeto PHP que aplique a PSR.

Baixa curva de aprendizagem e acesso a diversos Frameworks, CMS, APIs e aplicações feitas com PHP.

Domínio sobre a aplicação para desenvolver soluções personalizadas garantindo seu ciclo de vida.



Robson V. Leite

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



```
<?= "page X"; ?>
```



<?=

PHP standard recomendation

;?>

```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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



www.upinside.com.br

FORMAÇÃO FULL STACK PHP DEVELOPER



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "Os padrões da comunidade");
```

Fundamentos da PSR:

Tudo que você verá durante esta formação **mesmo que do básico** fará uso das recomendações de desenvolvimento padrão do PHP uma vez que não é possível aplicar esses fundamentos em separado nem mesmo para escrever uma simples variável.

Começaremos imergindo em uma revisão de tudo que você já sabe ou irá aprender agora sobre a linguagem e seus paradigmas. **Entre as principais recomendações temos:**

/GuiaDeEstiloDeCódigo:

A comunidade PHP é imensa e possui milhares de componentes e diversas bibliotecas e frameworks disponíveis.

O padrão básico (PSR-1) e o guia de estilo (PSR-2) fornecem a recomendação para garantir que desenvolvedores PHP possam misturar esses recursos em seus projetos.

O carregamento automático (PSR-4) fornece um controle automatizado de dependências, componentes e classes do projeto.

/Recomendações de base que devemos implementar em qualquer projeto:

PSR-1

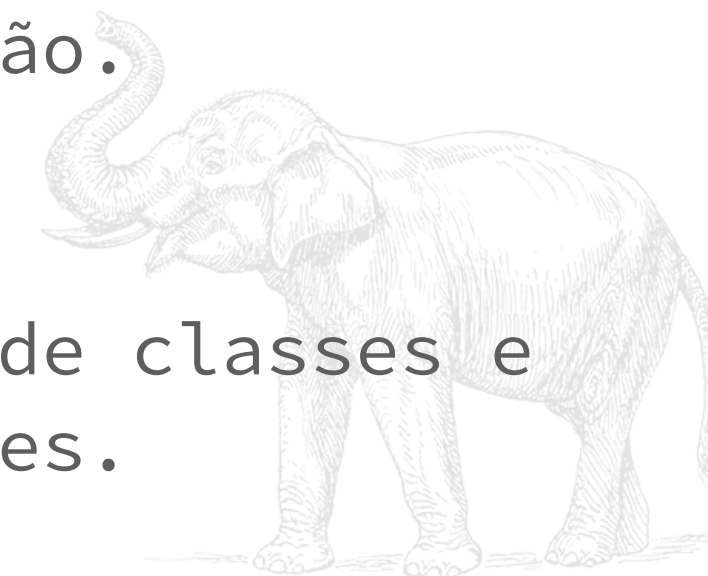
Padrão básico de codificação.

PSR-2

Guia de estilo de codificação.

PSR-4

Autoload de classes e componentes.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>
```

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-I: Padrão básico de codificação:

O que deve ser considerado como codificação padrão para garantir um alto nível de interoperabilidade técnica entre códigos compartilhados.

/Tags:

Use as tags `<?php ;?>` para abertura e fechamento e `<?= ;?>` para saída. Nunca use outras variações (ex: `<%, %>`, `<%=`)

```
file.php

<?php
    echo "Olá mundo!";
?>

<h1><?= "Olá mundo!"; ?></h1>
```

/Codificação de caracteres:

Arquivos devem usar apenas UTF-8 sem BOM para código PHP. (IDE, Charset, Etc.)

/Efeitos colaterais (side effects):

Você pode **(incluir um arquivo, conectar a um serviço, gerar uma saída) OU (declarar uma classe, criar uma função, definir uma constante)**. Mas nunca faça ambos no mesmo arquivo.

```
file.php

<?php

require "config.php"; /Arquivo com constantes
require "functions.php"; //Arquivo de funções

echo HELLO;
useMyFunction();

DEFINE("HELLO", "Olá mundo!");

mySecondeFunction(){
    //Function Body
}
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>
```

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-I: Padrão básico de codificação:

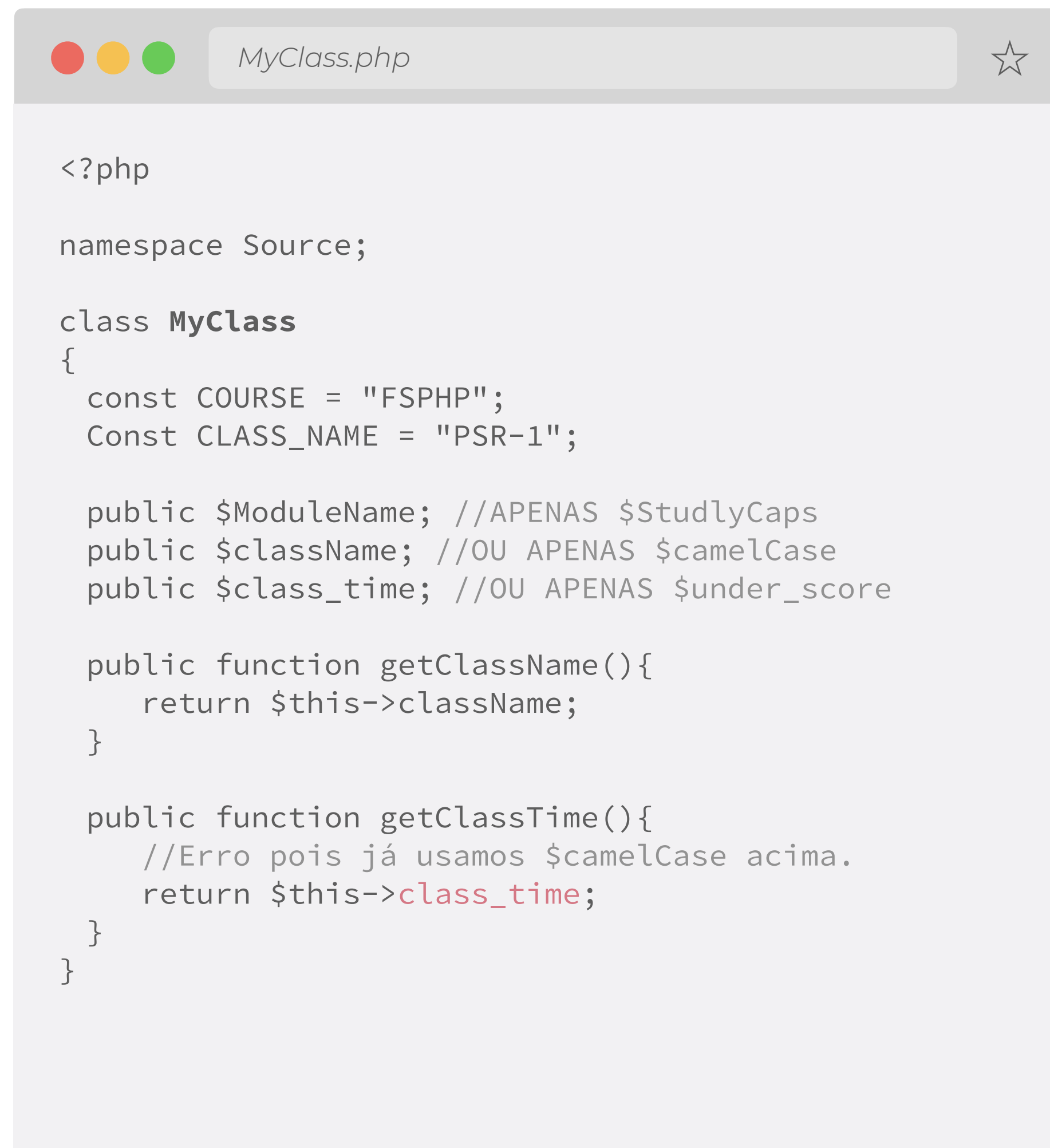
/Classes PHP:

Cada classe deve estar em seu próprio arquivo e ter pelo menos um nível de namespace (padrão PSR-4). O nome da classe deve ser declarada em **StudlyCaps**.

/Constantes da classe devem ser declaradas em maiúscula e quando preciso podem ser separadas por `under_score`.

/Propriedades da classe podem ser escritas em `$StudlyCaps`, `$camelCase` ou `$under_score`, não existe uma recomendação rígida para elas, mas é sempre importante escolher uma e usar sempre a mesma. Se for `$camelCase`, use sempre `$camelCase` e nunca as outras.

/Métodos da classe devem ser declarados sempre em `camelCase()`



```
<?php

namespace Source;

class MyClass
{
    const COURSE = "FSPHP";
    Const CLASS_NAME = "PSR-1";

    public $ModuleName; //APENAS $StudlyCaps
    public $className; //OU APENAS $camelCase
    public $class_time; //OU APENAS $under_score

    public function getClass_name(){
        return $this->className;
    }

    public function getClassTime(){
        //Erro pois já usamos $camelCase acima.
        return $this->class_time;
    }
}
```




```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recommendation");
```

PSR-2: Guia de estilo de codificação:

A intenção da PSR-2 é reduzir o atrito cognitivo ao escanear códigos de diferentes autores. Ele faz isso enumerando um conjunto compartilhado de regras e expectativas sobre como formatar o código PHP.

/Geral:

/Padrão básico de codificação: O código deve seguir todas as recomendações listadas na PSR-1.

/Arquivos: Sempre devem terminar com uma linha em branco (Unix linefeed).

/Somente PHP: Um arquivo somente com PHP deve omitir a tag `?>` de fechamento.

IDE /Linhas: Procure manter suas linhas com no máximo 80 caracteres, quando necessário você poderá usar até 120.

IDE /Espaços finais: Certifique-se que o último caractere da linha não é um espaço em branco.

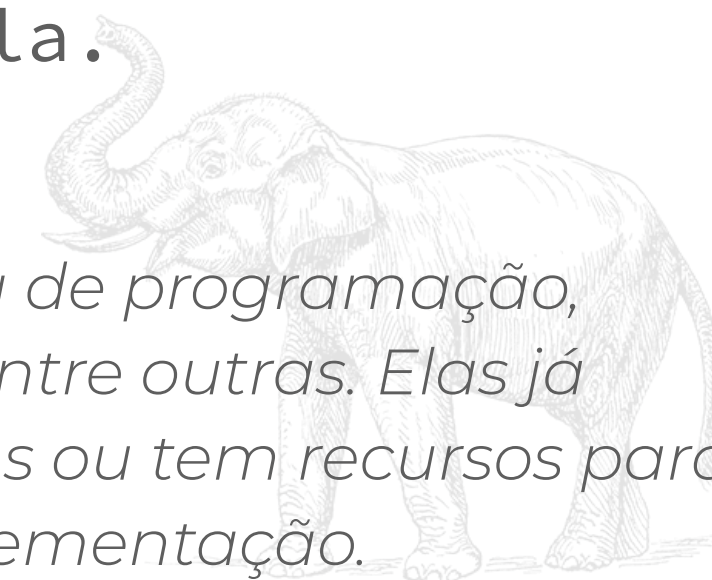
/Legibilidade: Pode adicionar linhas em branco para separar blocos de código.

IDE /Recuo: Use 4 espaços para indentar seu código, nunca o TAB.

IDE /Declaração: Não deve ter mais de uma por linha de código.

/true, false, null: São palavras-chave e constantes do PHP, devem ser escritas sempre em letra minúscula.

IDE *A IDE é sua ferramenta de programação, PHPStorm, netBeans entre outras. Elas já implementam as PSR's ou tem recursos para automatizar essa implementação.*





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-2: Guia de estilo de codificação:

/Namespaces e a declaração use:

/Namespaces: Após declarar um, sempre deixe uma linha em branco para então continuar seu código.

/use: Quando presentes devem ser declarados após os namespaces.

DEVE haver um use por declaração.

DEVE haver um espaço após o bloco de declaração do use.

```
<?php

namespace Source\Controller;

use Source\Model\MyClass, Source\Connection\MyClass;
use Source\Model\MyClass;
use Source\Connection\MyClass;

class MyClass
{
```

/Classes, propriedades e métodos:

Entenda classes como todas as classes, interfaces e traits.

/extends e implements devem SEMPRE ser declarados na mesma linha do nome da classe, se ambos, primeiro o extends.

IDÉ A chave de abertura da classe deve seguir na próxima linha após o nome e a chave de fechamento deve seguir na próxima linha após o corpo.

```
MyClass.php ☆

<?php

namespace Source\Controller;

use Source\Parent\ParentClass;
use Source\Parent\iClass;

class MyClass extends ParentClass implements iClass
{
    //Class Body
}
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recommendation");
```

PSR-2: Guia de estilo de codificação:

/Classes, propriedades e métodos:

/em uma lista de implements você pode declarar todas as interfaces em uma linha ou declarar uma por linha.

IDE Em uma por linha, cada linha subsequente deve ser recuada uma vez e o primeiro item deve estar na próxima linha e deve haver apenas uma interface por linha.

/Propriedades devem sempre declarar a visibilidade (public, protected, private) e nunca deve usar em sua declaração a palavra-chave **var**.

Não deve haver mais de uma propriedade declarada por linha e nunca use **underscore_** ou **_underscore** para declarar visibilidade protegida ou privada.

```
MyClass.php

<?php

namespace Source\Controller;

use Source\Parent\iClass

class MyClass implements
    iClass,
    \ArrayAccess,
    \Countable
{
    //Class Body
}
```

```
MyClass.php

<?php

namespace Source\Controller;

class Course
{
    //var $courseName;
    //private $courseName_;
    public $courseName;
    protected $courseAuthor = "Robson V. Leite";
    private $coursePrice;
}
```




```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-2: Guia de estilo de codificação:

/Classes, propriedades e métodos:

/Métodos assim como propriedades devem ter sua visibilidade declarada e não usar underscore_ ou _underscore.

Nomes de métodos não podem conter espaços nele ou após ele. A chave de abertura deve estar na mesma linha do nome, a de fechamento na próxima linha após o corpo e devem ser declarados com \$camelCase().

/Argumentos do método: Na lista de argumentos deve haver um espaço depois da vírgula mas não antes, argumentos com valores padrão devem ir ao final da lista.

A lista de argumentos também podem ser declaradas na próxima linha. Quando isso ocorrer cada argumento assim como os parênteses devem estar em uma linha subsequente e você deve adicionar um recuo para todos.

IDE

```
MyClass.php ☆

<?php

namespace Source\Controller;

class Course
{
    private $courseName;
    private $courseAuthor;
    private $coursePrice;

    public function setName($name){
        $this->courseName = $name;
    }

    public function setMore($course, $price = "997"){
        //Method Body
    }

    public function setAll(
        $course,
        $author,
        $price = "997"
    ){
        //Method Body
    }
}
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-2: Guia de estilo de codificação:

/abstract, final e static:

/abstract e final quando presentes devem ser declaradas ANTES da declaração de visibilidade.

Quando presente, a **static** deve ser declarada DEPOIS da declaração de visibilidade.

/chamada de métodos funções: Chamadas de métodos e funções não use espaços entre o nome declarado e os parênteses de abertura e fechamento.

Na lista de argumentos deve haver um espaço depois da vírgula mas não antes.

A lista de argumentos também pode ser dividida em várias linhas.

```
MyClass.php

<?php

namespace Source\Controller;

abstratic class Course
{
    private static $courseName;

    abstratic public function getCourse(){
        //Method Body
    }

    final public function getPrice(){
        //Method Body
    }
}
```

```
file.php

<?php

myFunction();
myFunction($arg, $argB);

$course->getCourse();
$course->setPrice("997");

$course->setAll(
    "FS PHP",
    "Robson V. Leite"
);
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-2: Guia de estilo de codificação:

/Regras gerais da PSR para estruturas de controle:

DEVE haver um espaço após a palavra-chave da estrutura de controle.

NÃO DEVE haver um espaço após o parêntese de abertura nem antes do parêntese de fechamento.

DEVE haver um espaço entre o parêntese de abertura e a chave de abertura.

O corpo da estrutura de controle **DEVE** ser recuado uma vez.

A chave de fechamento **DEVE** estar na próxima linha após o corpo da estrutura.

O corpo de qualquer estrutura **DEVE** estar entre chaves.

```
file.php ☆

<?php

if ($arg) {
    // if body
} elseif ($argB) {
    // elseif body
} else {
    // else body;
}

switch ($argTest) {
    case 0:
        //First Case
        break;
    default:
        //No tested In Case
        break;
}

try {
    // try body
} catch (ClassExceptionType $e) {
    // catch body
}
```




```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>
```

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

PSR-2: Guia de estilo de codificação:

/Closures:

DEVEM ser declaradas com um espaço depois de function e um espaço antes e depois de use.

A chave de abertura **DEVE** ir na mesma linha do nome, a chave de fechamento **DEVE** ir uma linha após o corpo.

NÃO DEVE haver espaço após o parêntese de abertura ou antes do parêntese de fechamento na lista de argumentos ou variáveis.

DEVE haver na lista de argumentos ou variáveis um espaço depois da vírgula mas nunca antes.

Argumentos com valor padrão **DEVEM** ir ao final da lista de argumentos.

```
file.php

<?php

$myClosure = function($argA, $argB) {
    //Closure Body
};

$myClosureB = function($argA) use ($varA) {
    //Closure B Body
};

$myLongClosureArgs = function(
    $argA,
    $argB,
    $argC
) {
    //Closure B Body
};

$myLongClosureArgsB = function(
    $argA,
    $argB
) use(
    $varA,
    $varB
) {
    //Closure B Body
};
```



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recommendation");
```

PSR-4: Carregamento automático:

Este PSR descreve uma especificação para o carregamento automático e interoperável das classes, assim como mostra onde colocar os arquivos em seu projeto.

/Especificação:

Entenda classes como todas as classes, interfaces e traits.

- 1) Um nome de classe totalmente qualificado deve seguir o seguinte formato:

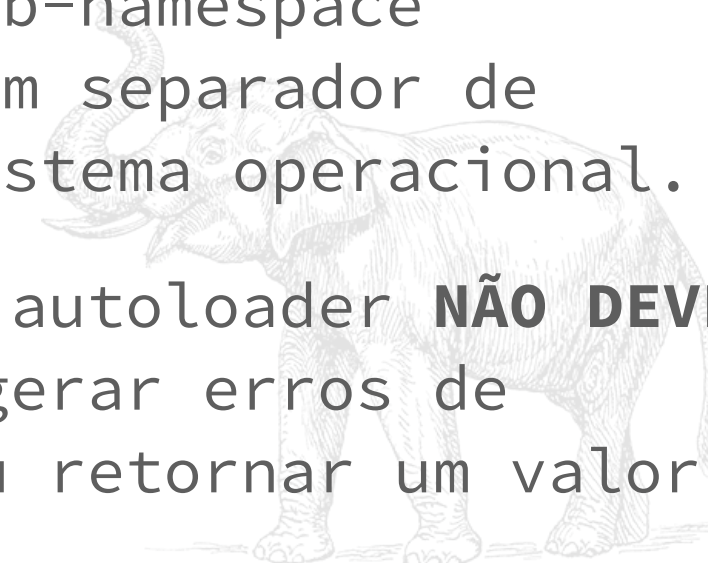
```
\<VendorNamespace>(\<SubNamespace>)*\<ClassName>
```

- 1) O namespace completo **DEVE** ter um nome de nível superior (Vendor).
- 2) O namespace **PODE** ter um ou mais sub-namespaces.
- 3) O namespace **DEVE** terminar com o nome da classe.
- 4) Underscore não tem qualquer efeito especial no namespace.
- 5) Caracteres alfabéticos **PODEM** ter qualquer combinação de minúsculas e maiúsculas.
- 6) Todos os namespaces **DEVEM** ser referenciados de forma única.

- 3) Ao carregar um arquivo que corresponde a um namespace:

- 1) O Vendor namespace e alguns dos primeiros níveis de sub-namespace **DEVEM** corresponder a um diretório base.
- 2) Cada sub-namespace seguinte deve corresponder a um sub-diretório dentro do diretório base, cada separador de sub-namespace corresponde a um separador de diretório no sistema operacional.

- 4) Implementações de autoloader **NÃO DEVEM** lançar exceções, gerar erros de qualquer nível ou retornar um valor.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

Um exemplo de autoload PSR-4:

autoload.php

```
<?php
```

```
autoload_register(function ($class) {
```

```
    //Define o Vendor Namespace e a pasta raiz.
```

```
    $prefix = 'Source\\';
```

```
    $baseDir = __DIR__ . '/source/';
```

```
    //Verifica o namespace vendor na classe
```

```
    $len = strlen($prefix);
```

```
    if (strncmp($prefix, $class, $len) !== 0) {
```

```
        return;
```

```
    }
```

```
    //Obtém o nome da classe e substitui o namespace por diretório.
```

```
    $relativeClass = substr($class, $len);
```

```
    $file = $baseDir . str_replace('\\', '/', $relativeClass) . '.php';
```

```
    //Carrega o arquivo de classe se ele existir.
```

```
    if (file_exists($file)) {
```

```
        require $file;
```

```
    }
```

```
});
```




```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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

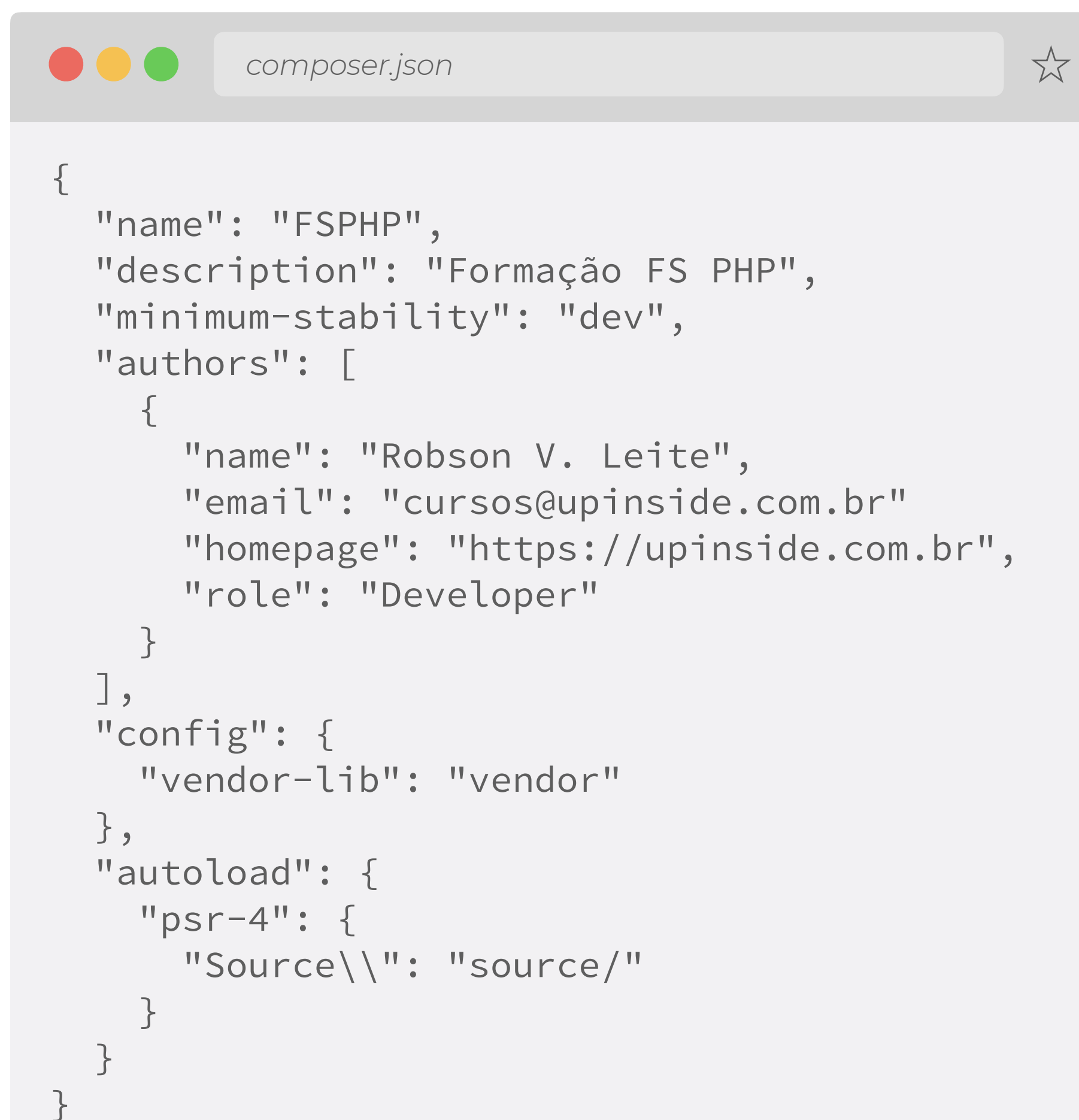


<?= "page X"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

Autoload PSR-4 com Composer:

Uma forma mais prática, inteligente e interoperável de criar seu autoload usando o Composer para gerenciar todas as dependências do projeto:



```
composer.json

{
  "name": "FSPHP",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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/"
    }
  }
}
```

O exemplo ao lado mostra o arquivo de configuração `composer.json` do Composer.

Com ele o gerenciador de dependências PHP poderá automatizar todo o processo de carregamento de classes e componentes para você.

A configuração autoload:

```
"autoload": {
  "psr-4": {
    "Source\\": "source/"
  }
}
```

Neste ponto estamos informando que nossas classes estão no namespace fornecedor **Source**, e dentro da pasta raiz **source** do projeto.

Depois de rodar o Composer, basta invocar o autoload:

```
require __DIR__."/vendor/autoload.php";
```





<?= A estrutura de um arquivo PHP ;?>

```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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



www.upinside.com.br

FORMAÇÃO FULL STACK PHP DEVELOPER



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

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.

```
index.php

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

Muitas bandeiras são levantadas afirmando que não deve misturar os códigos PHP com HTML, **esse é o maior dos mitos** pois 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 como conseguimos ser produtivos com isso. 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.
- * Arquivos e trechos que precisam ser reutilizados também precisam ser separados.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

O que você não deve fazer:

O PHP é permissivo ao extremo e isso permite que você construa qualquer coisa boa na mesma medida que ruim. Vejamos algumas práticas a serem evitadas:

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.

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.

MATERIALIZAÇÃO: Na correria de um projeto é comum ligar o piloto automático e aplicar uma programação funcional ao extremo ignorando a regra para aplicar uma funcionalidade.

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

```
echo "<heade>
    //heade Content...
</header>";
```

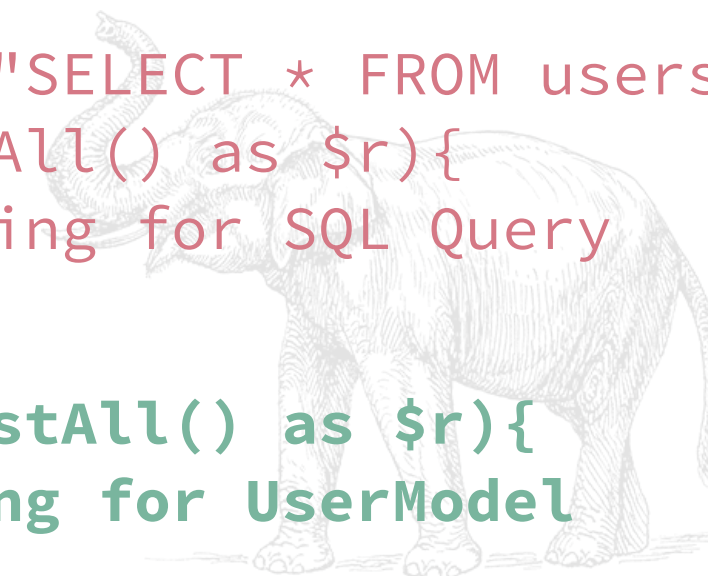
```
require "header.php";
```

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

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

```
$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
}
```





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recomendation");
```

Um mundo ideal para arquivos PHP:

Em um mundo ideal aplicamos o DRY para otimizar todo o processo de desenvolvimento, com isso separamos tudo que pode e precisa ser reutilizado, e usamos o arquivo PHP para controlar e processar todos os recursos da aplicação.

```
index.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";

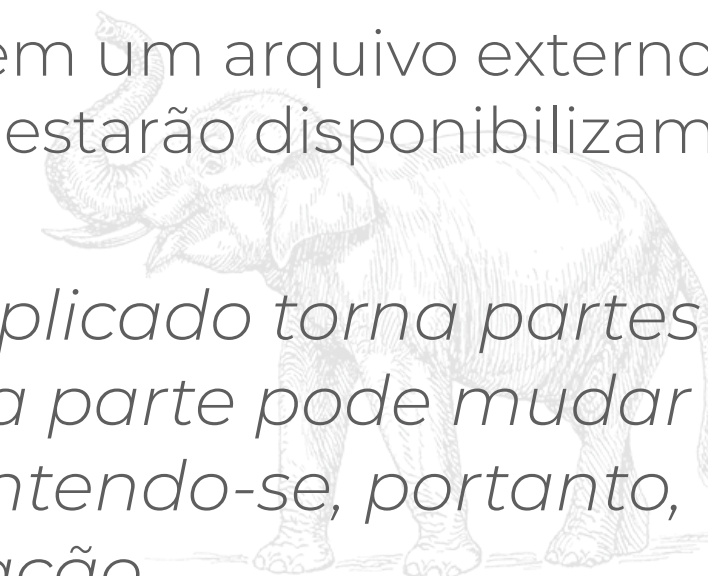
    /* Uma lógica aplicada afim de decidir o conteúdo
    * a ser carregado. Podendo interagir com a lógica
    * e regra de negócios para criar as visões */

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

Esse é um arquivo **index.php** comum e coeso 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** completando a estrutura base do site.
- * Por fim, ele requer o **scripts.js** em um arquivo externo como o CSS, onde todos os eventos estarão disponibilizados.

Don't Repeat Yourself: 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.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "PHP standard recommendation");
```

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
 * vires, headers, sideras e footers.
 */
```




<?= Como aproveitar melhor o curso ;?>

FORMAÇÃO FULL STACK PHP DEVELOPER



www.upinside.com.br

```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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



```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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

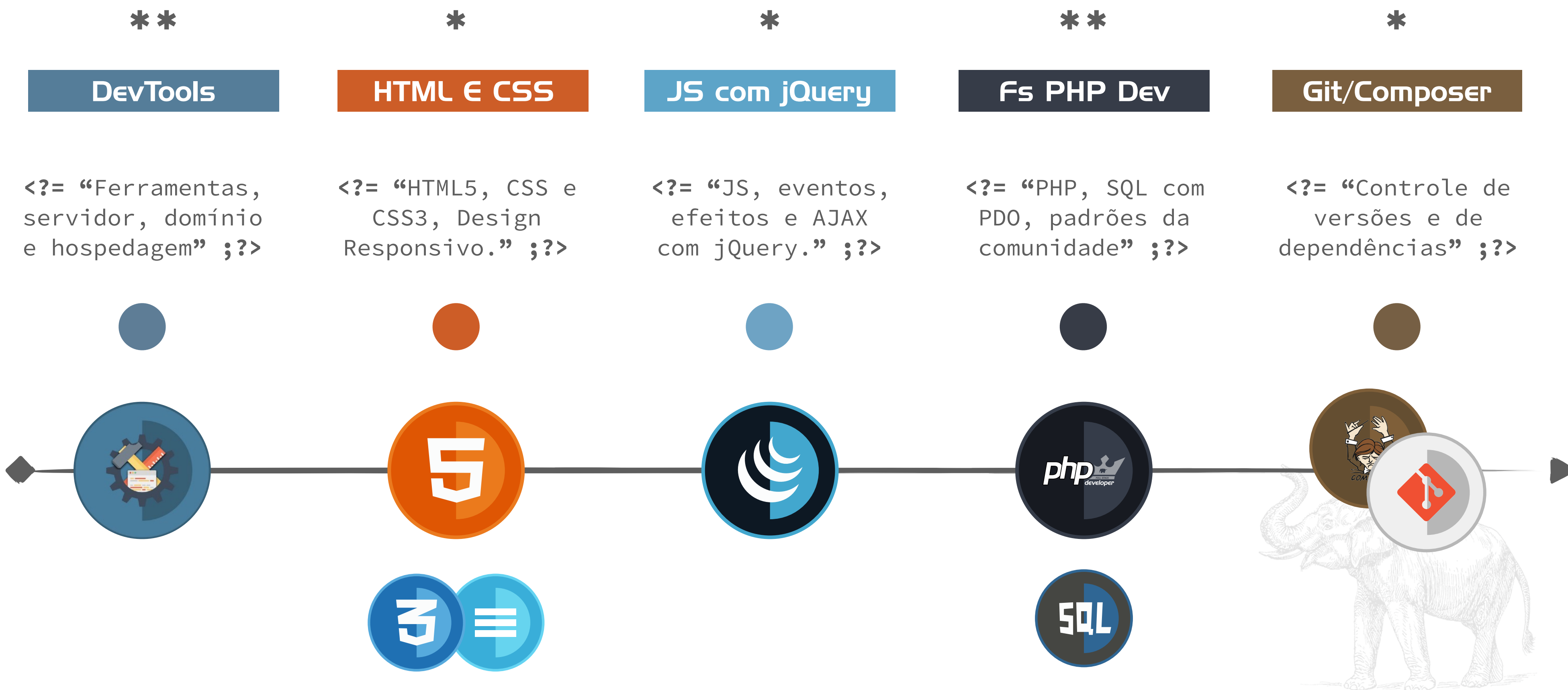


<?= "page X"; ?>

DEFINE("CLASSNAME", "Como aproveitar melhor o curso");

Formação FSPHP:

O plano passo a passo para você adquirir todos os conhecimentos necessários para se formar um full stack php developer nesta formação.





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "Como aproveitar melhor o curso");
```

Como estudar as aulas:

A 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 onde você aprende o fundamento enquanto executa o exercício.

/DicaDeComoEstudar

1º = Contexto: Apenas assista todas as aulas do módulo prestando atenção no conteúdo para entender o contexto e os resultados desejados.

2º = Exercício: Volte a primeira aula do módulo e venha fazendo junto, pausando a aula se e quando preciso para desenvolver o exercício.

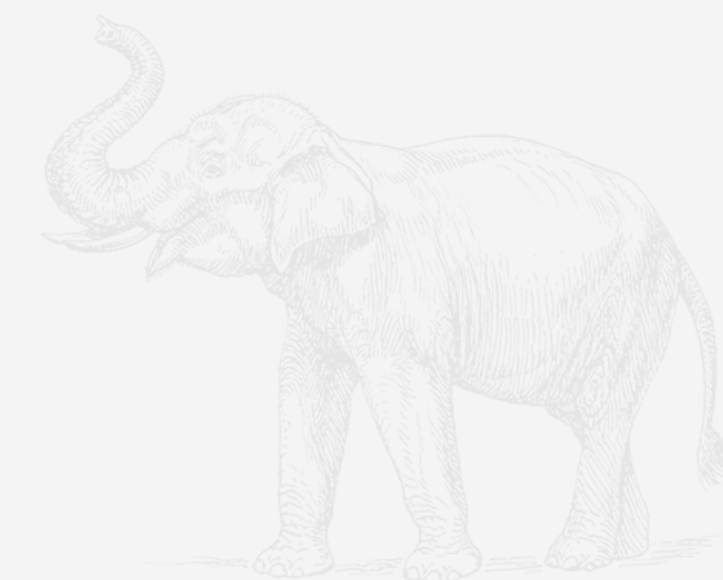
3º = Documentação: Faça uma terceira vez sem usar tanto a aula para testar na prática o que aprendeu.



www.upinside.com.br



Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos, e vem sendo utilizado desde o século XVI





```
{
  "name": "FSPHP/Source",
  "description": "Formação FS PHP",
  "minimum-stability": "dev",
  "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"; ?>

```
DEFINE("CLASSNAME", "Como aproveitar melhor o curso");
```

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 ou dia seguinte)

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." ;?>

/Conquista

<?= "Ao concluir os 6 cursos da formação e emitir seus certificados você recebe a conquista Full Stack PHP Developer." ;?>

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



<?= "page X"; ?>

/EstruturaDeExemplos:

/Curso

/Módulo

/Material

Aula

✎ /Exercícios FSPHP

📁 /01-ola-mundo-vamos-comecar

📁 /01-08

📁 /source

📁 /assets

📄 index.php

✎ /Projetos FSPHP

📁 /source

📁 /vendor

📁 /assets

📄 index.php

