INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

Desenvolvimento Web

Prof. Daniel Di Domenico

API REST com o Slim Framework

Objetivo das aulas

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

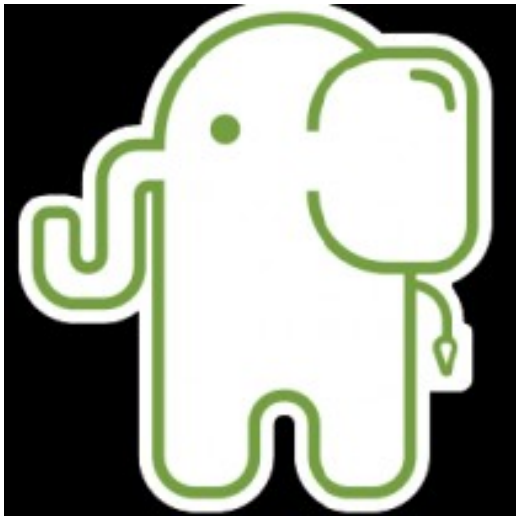
- Desenvolver um CRUD com API REST em PHP:
  - Ferramenta: **Slim Framework**
  - Arquitetura em camadas
  - Utilização do recurso de *namespaces* do PHP

- Ferramentas necessárias:
  - PHP, SlimFramework e Composer
  - ThunderClient, Postman ou Insomnia
  - Banco de dados MySQL
  - phpMyAdmin ou outro cliente com suporte a MySQL

Slim Framework

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Micro-framework para desenvolvimento em PHP:
  - Foco principal: **desenvolvimento de APIs REST**
  - Também suporta o desenvolvimento padrão de páginas Web
- Caraterísticas:
  - Facilidade de configuração
  - Suporte nativo a rotas HTTP
  - Flexibilidade de desenvolvimento
    - Não requer uma estrutura padrão



Slim Framework: instalação

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

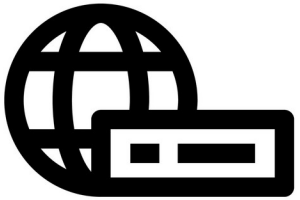
- Instalação:
  - Pode ser instalado pelo ***composer***
    - Gerenciador de dependências do PHP
    - https://getcomposer.org/download/
- Dependências (mínimas):
  - **Slim versão 4**:
    - É o micro-framework
  - **PSR-7 versão 6**:
    - Interface para mensagens HTTP utilizada pelo Slim
- https://www.slimframework.com/docs/v4/

Slim Framework: execução

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Servidor Web:
  - Slim requer o recurso de sobrescrita da URL para funcionar
    - Apache possui esse recurso inativado
- PHP Built-in Sever:
  - Recurso de sobrescrita de URL ativado por padrão
  - `php -S localhost:8082`
    - O servidor PHP será iniciado na porta 8082 (pode ser alterado)

PHP
Built-in Server



Slim Framework: dependências

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- **composer.json**:
  - Arquivo com as dependências do projeto

```
{
    "require": {
        "slim/slim": "4.11.0",
        "slim/psr7": "1.6"
    }
}
```

- Na pasta do projeto, atualizar as dependências (executar apenas um dos dois comandos):
  - `composer update`
  - `php composer.phar update`

Pode ser obtido no site do composer

- Após executar, será criado o diretório **vendor** com as dependências definidas para o projeto

Slim Framework: projeto base

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

```php
<?php
//Arquivo index.php

require __DIR__ . '/vendor/autoload.php';

use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Slim\Factory\AppFactory;

$app = AppFactory::create();
$app->setBasePath("/projeto_slim");

//Definir as rotas (endpoints)

$app->run();
```

Diretório do projeto onde está o index.php

Slim Framework: Olá Mundo!

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

As rotas devem seguir este formato de declaração

```
//Definição da rota raiz

$app->get('/', function (Request $request,
                         Response $response, $args) {
    $response->getBody()->write("Olá Mundo!");
    return $response;
});
```

Slim Framework: classes

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

```php
//Definição da rota: chamando o método de uma classe

$app→get('/aluno', AlunoController::class . ':getAluno');
```

```php
//Classe AlunoController: método getAluno() retornando um JSON
class AlunoController {

    public function getAluno(Request $request,
                             Response $response, $args) {
        $aluno = array('nome' => 'Carlos', 'age' => 40);
        $alunoJson = json_encode($aluno);

        $response->getBody()->write($alunoJson);
        return $response
              ->withHeader('Content-Type', 'application/json')
              ->withStatus(201);
    }
}
```

PHP: namespaces

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Namespaces:
  - Utilizados para agrupar os recursos da aplicação (classes, funções, constantes...)
    - Visa evitar conflitos de nomes entre os recursos
    - Permite estruturar a aplicação de forma semelhante aos diretórios de um sistema operacional
  - Vantagens:
    - Pode-se utilizar o mesmo nome de recurso em diferentes namespaces
    - Permite carregar as dependências do projeto utilizando o **autoload**

PHP: namespaces

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Declarar um namespace

```php
<?php

namespace App\Classe;

class Teste {

}
```

- Utilizar um namespace

```php
<?php

namespace App\Principal;

use App\Classe\Teste;

$teste = new Teste();
```

Classe **Teste** será carregada do namespace **App\Classe**

PHP: namespaces

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Pode-se carregar as classes do projeto utilizando namespaces
  - **Por padrão, o nome dos diretórios devem iniciar com letras maiúsculas (camelCase)**
  - Não é mais necessário utilizar **includes/requires** nas páginas (apenas o do arquivo autoload.php)

```
{
   "autoload": {
      "psr-4": {"App\\": "./app"}
   }
}
```

O namespace **App** será carregado do diretório **./app**

- Após alterar o arquivo composer.json, deve-se sempre executar o comando de *update*