APIs Rest com framework Slim
Code Easy (Felipe Renan Vieira)

Vídeos: https://www.youtube.com/watch?v=vVkOUXpuuJg&list=PLZ8kYL6LBgg62kzIa6Io42Ccz_rWJBS-l

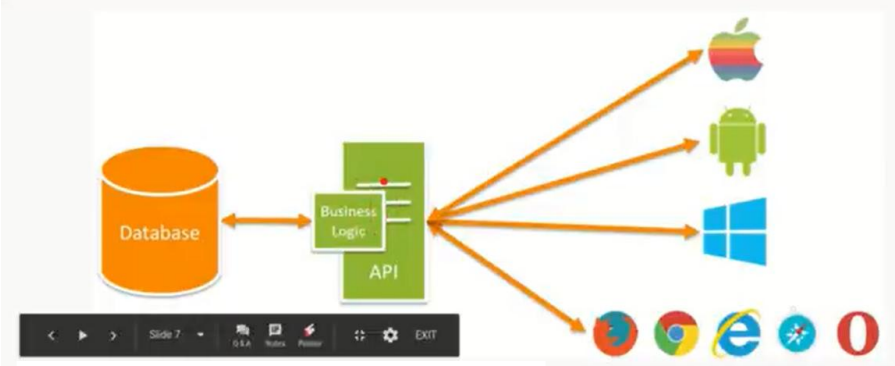
Aula 01 - Introdução

Requisitos

- Conhecer o PHP.
- Conhecer o básico de Programação Orientada a Objetos (POO).

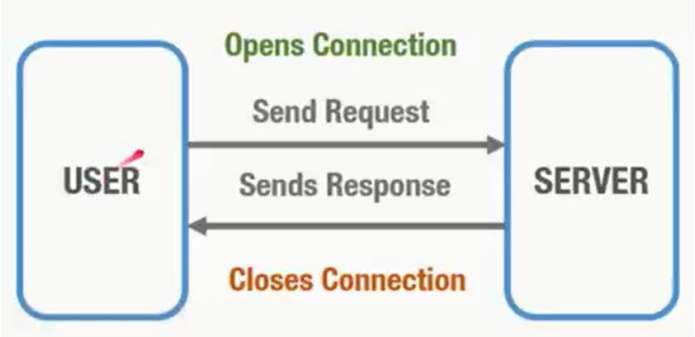
O que é uma API

- Application Program Interface (API) - Interface de Programação de Aplicações
- Conjunto de rotinas e padrões de uma aplicação que podem ser acessados por outra aplicação sem precisar conhecer os detalhes da implementação do software.



O que é uma API REST?

- Representational State Transfer (REST) - Estado de Transferência Representacional.
- Requisições HTTP
- Verbos HTTP: GET, POST, PUT, DELETE, PATCH, OPTIONS, HEAD, ...
- Formato de arquivo: JSON



Vantagens de usar uma API REST

- Separação do Back-end e do Front-end.
- Reutilização do Back-end em diferentes lugares.
- Fácil de conectar com qualquer linguagem e plataforma

Ferramentas

- PHP 7.2
- Composer
- MySQL
- Editor de texto ou IDE: Visual Studio Code | Atom | Sublime Text 3 | PHP Storm | Eclipse

Aulas

Etapas

1. Como o Slim funciona
2. Arquitetura do projeto
3. Banco de dados
4. Autenticação
5. Documentação
6. Conclusão

Aula 02 - Primeiro Projeto

- Crie a pasta do projeto:
curso_api_slim

Instalando o Slim com o Composer

php -v

```
C:\xampp\htdocs\curso_api_slim>php -v
PHP 7.4.29 (cli) (built: Apr 12 2022 20:21:18) ( ZTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.29, Copyright (c), by Zend Technologies
```

composer --version

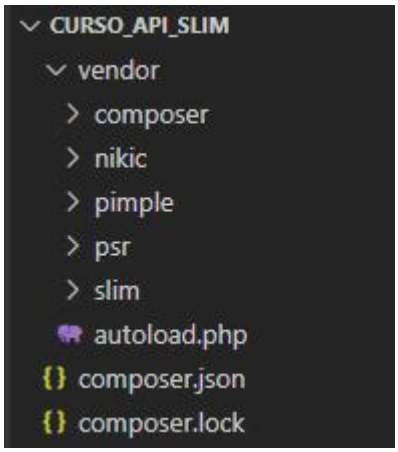
```
C:\xampp\htdocs\curso_api_slim>composer --version
Composer version 2.3.5 2022-04-13 16:43:00
```

.htaccess

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
Rewrite ^ index.php [QSA,L]

composer require slim/slim:"^3.0"

```
Roberto@DESKTOP-HGDUAQT MINGW64 /c/xampp/htdocs/curso_api_slim
$ composer require slim/slim:"^3.0"
./composer.json has been created
Running composer update slim/slim
Loading composer repositories with package information
https://repo.packagist.org could not be fully loaded (curl error 6 while downloading https://repo.packagist.org/packages.json: Could not
resolve host: repo.packagist.org), package information was loaded from the local cache and may be out of date
Updating dependencies
Lock file operations: 5 installs, 0 updates, 0 removals
  - Locking nikic/fast-route (v1.3.0)
  - Locking pimple/pimple (v3.5.0)
  - Locking psr/container (1.1.2)
  - Locking psr/http-message (1.0.1)
  - Locking slim/slim (3.12.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 5 installs, 0 updates, 0 removals
  - Installing psr/http-message (1.0.1): Extracting archive
  - Installing psr/container (1.1.2): Extracting archive
  - Installing pimple/pimple (v3.5.0): Extracting archive
  - Installing nikic/fast-route (v1.3.0): Extracting archive
  - Installing slim/slim (3.12.3): Extracting archive
Generating autoload files
```



Teste inicial

index.php

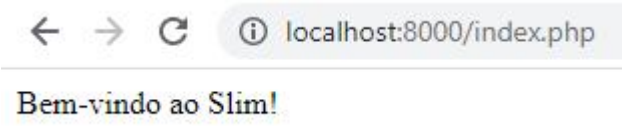
```
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->run();
```

Rodando o servidor do PHP

php -S localhost:8000

```
Roberto@DESKTOP-HGDUAQT MINGW64 /c/xampp/htdocs/curso_api_slim
$ php -S localhost:8000
[Fri May 27 03:30:32 2022] PHP 7.4.29 Development Server (http://localhost:8000) started
```

http://localhost:8000/



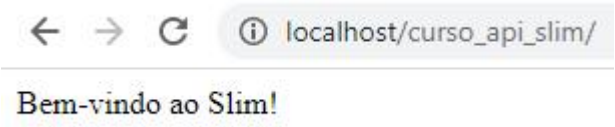
Usando o XAMPP como servidor

- Desabilite o servidor do PHP

- Habilite o servidor Apache do XAMPP

- No browser, entre com:

http://localhost/curso_api_slim/



Aula 03 - Método GET

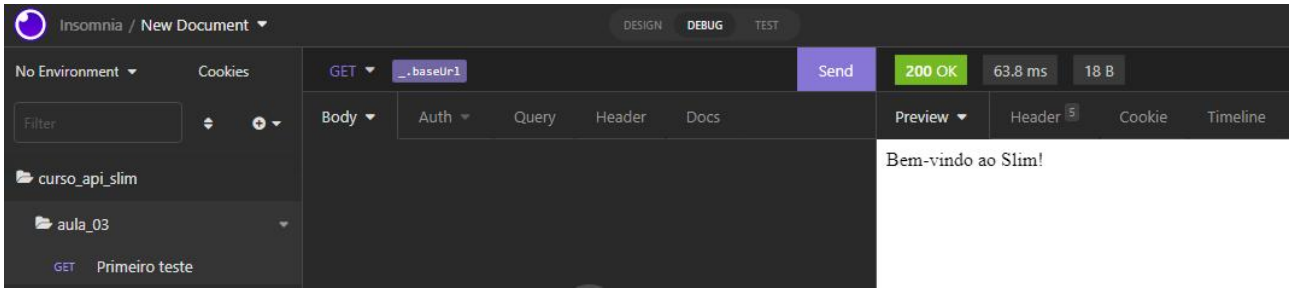
- Testes
- Como funciona
- Passando parâmetros pela URL
- Passando parâmetros pelo caminho da URL

Testes com o Insomnia

Configurando variável de ambiente para o servidor

```
{
  "baseUrl": "http://localhost/curso_api_slim/index.php"
}
```

Primeiro teste



Passando parâmetros

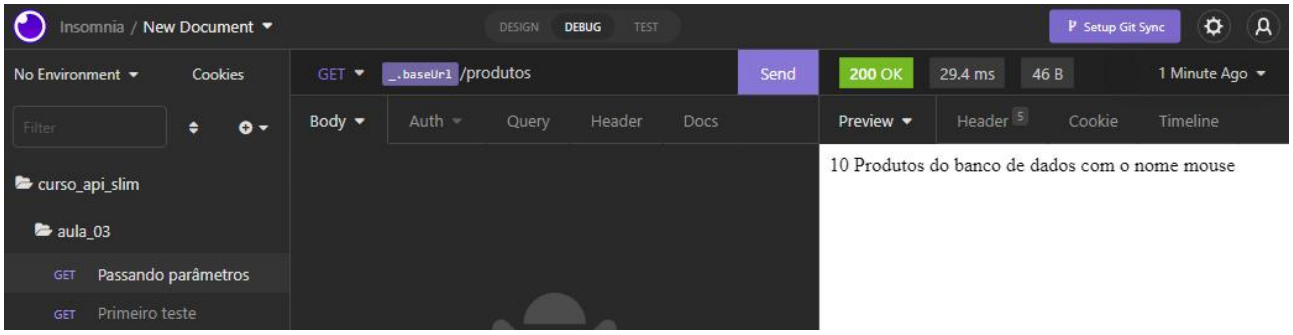
index.php

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->get('/produtos[/{nome}]', function (Request $request, Response $response, array $args) {
    $limit = $request->getQueryParams()['limit'] ?? 10;
    $nome = $args['nome'] ?? 'mouse';
    $response->getBody()->write("{$limit} Produtos do banco de dados com o nome {$nome}");
    return $response;
});

$app->run();
```

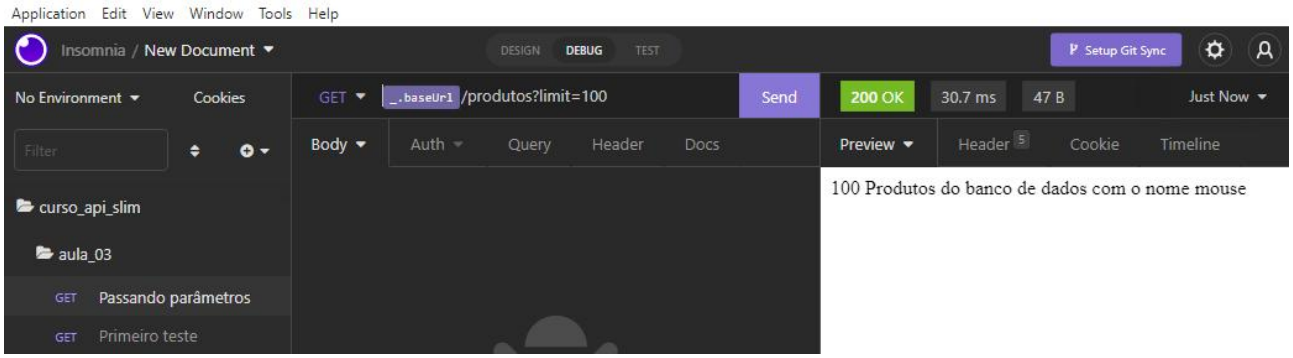
- No Insomnia:

Passando parâmetros

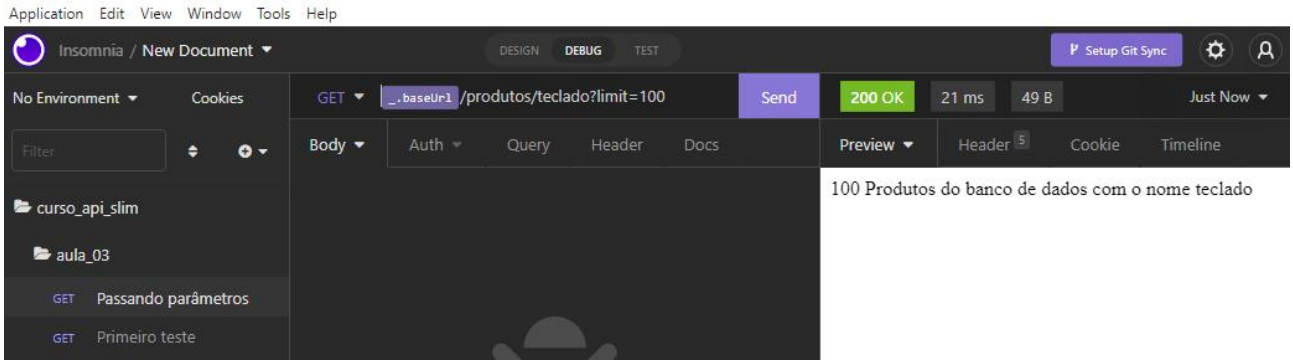
GET
{{ _.baseUrl }}/produtos



GET
{{ _.baseUrl }}/produtos?limit=100



GET
{{ _.baseUrl }}/produtos/teclado?limit=100



Aula 04 - Métodos POST, PUT e DELETE

Método POST

index.php

```
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->get('/produtos[/{nome}]', function (Request $request, Response $response, array $args) {
    $limit = $request->getQueryParams()['limit'] ?? 10;
    $nome = $args['nome'] ?? 'mouse';
    $response->getBody()->write("{$limit} Produtos do banco de dados com o nome {$nome}");
    return $response;
});

$app->post('/produto', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();
    print_r($data);
    die;
});

$app->run();
```

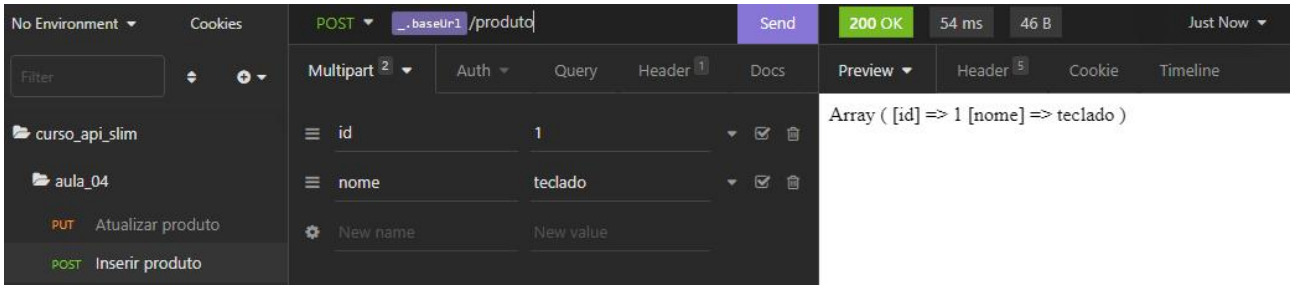
Observação:

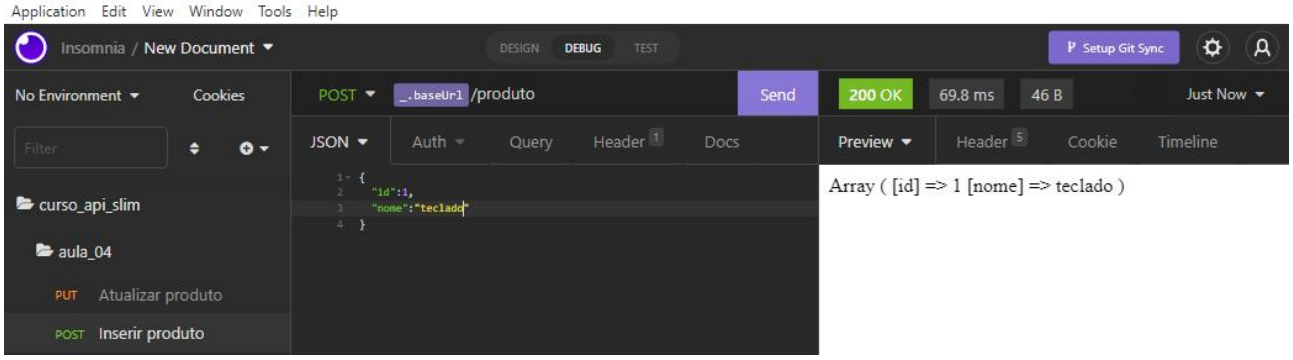
Quando se usa `$request->getParsedBody()` é retornado um array com todos os dados passados.

- No Insomnia:

Inserir produto

POST
{{ _.baseUrl }}/produto





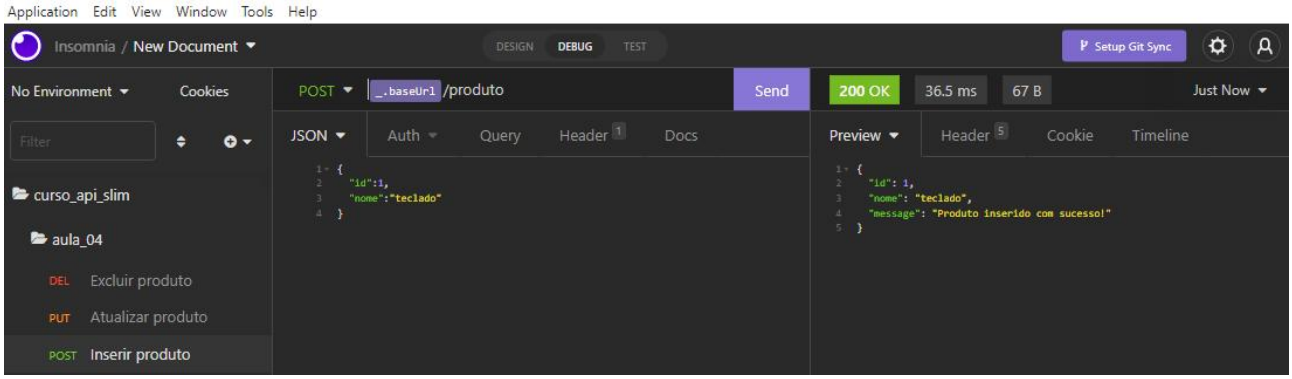
index.php

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->get('/produtos[/{nome}]', function (Request $request, Response $response, array $args) {
    $limit = $request->getQueryParams()['limit'] ?? 10;
    $nome = $args['nome'] ?? 'mouse';
    $response->getBody()->write("{$limit} Produtos do banco de dados com o nome {$nome}");
    return $response;
});

$app->post('/produto', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();

    $id = $data['id'] ?? '';
    $nome = $data['nome'] ?? '';

    $data['message'] = "Produto inserido com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
});

$app->run();
```

- No Insomnia:

Inserir produto

POST
{{ _.baseUrl }}/produto



Método PUT

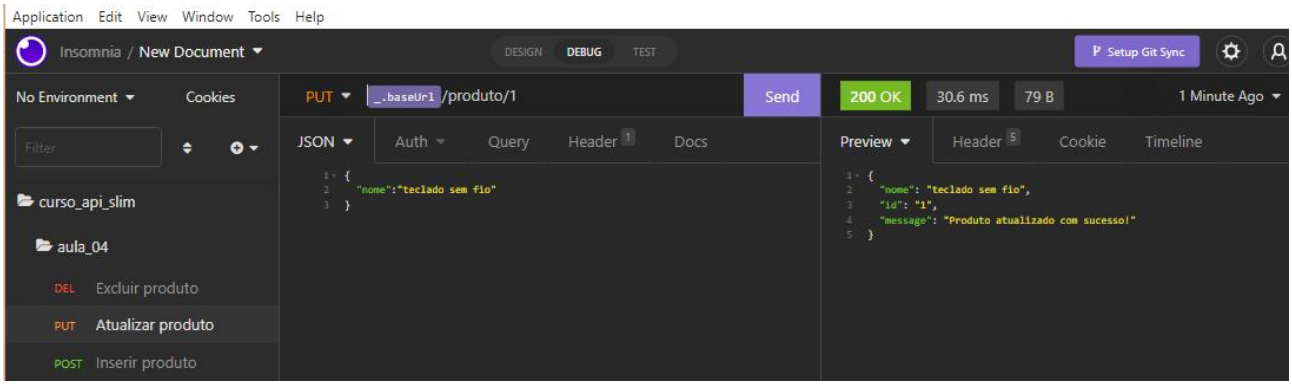
index.php

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->get('/produtos[/{nome}]', function (Request $request, Response $response, array $args) {
    $limit = $request->getQueryParams()['limit'] ?? 10;
    $nome = $args['nome'] ?? 'mouse';
    $response->getBody()->write("{$limit} Produtos do banco de dados com o nome {$nome}");
    return $response;
});

$app->post('/produto', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();

    $id = $data['id'] ?? '';
    $nome = $data['nome'] ?? '';

    $data['message'] = "Produto inserido com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
});

$app->put('/produto/{id}', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();

    $nome = $data['nome'] ?? '';

    $data['id'] = $args['id'];
    $data['message'] = "Produto atualizado com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
});

$app->run();
```

- No Insomnia:

Atualizar produto

PUT
{{ _.baseUrl }}/produto/1

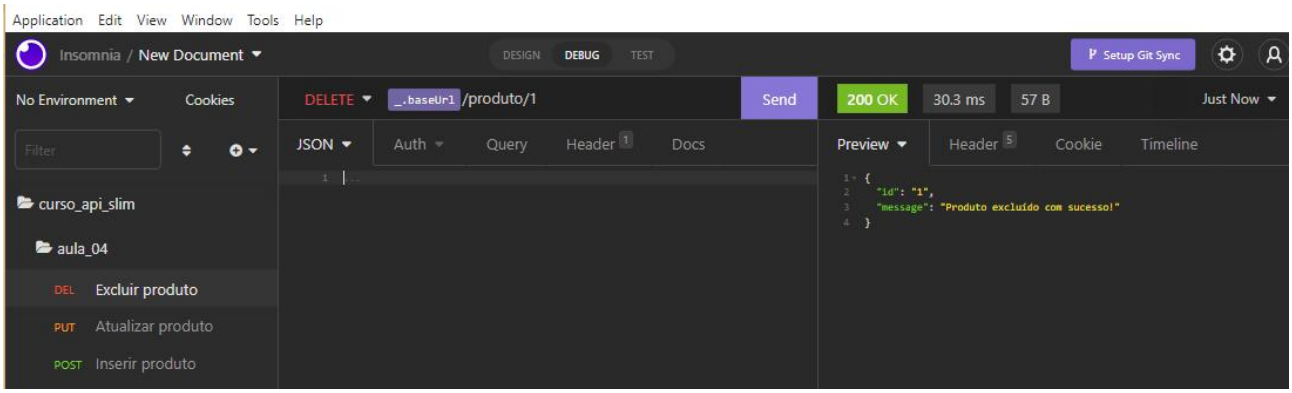


Método DELETE

index.php

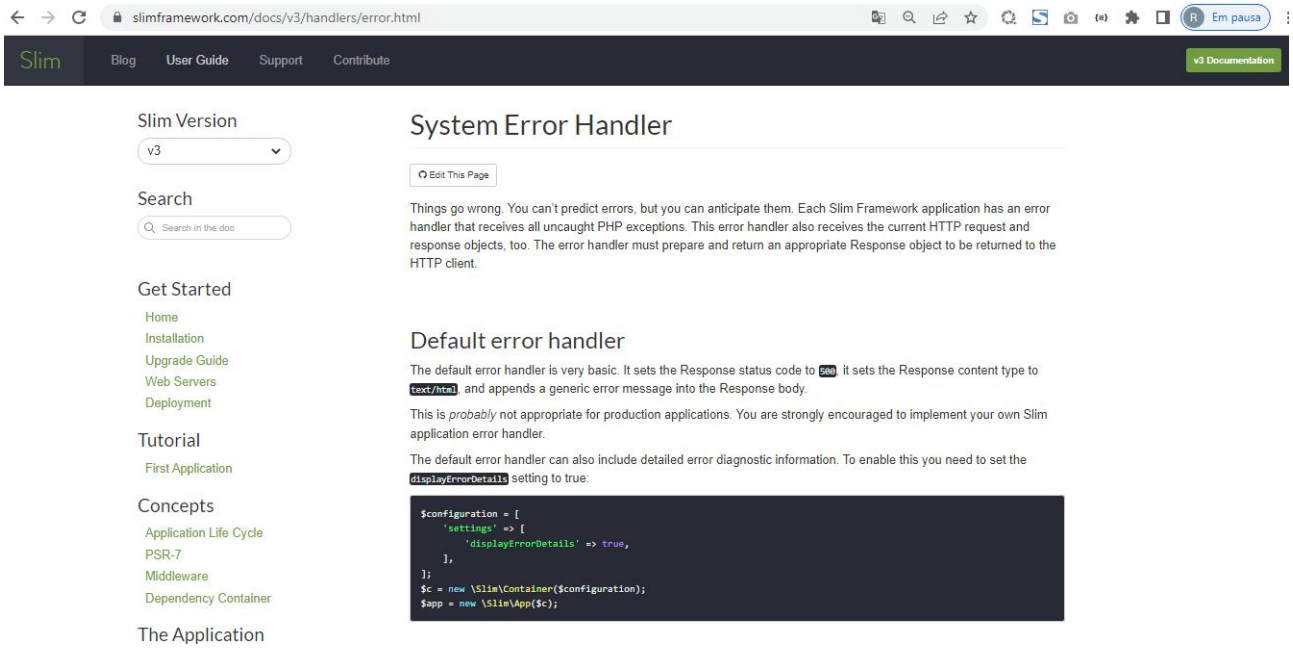
```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$app = new \Slim\App;

$app->get('/', function (Request $request, Response $response, array $args) {
    $response->getBody()->write("Bem-vindo ao Slim!");
    return $response;
});

$app->get('/produtos[/{nome}]', function (Request $request, Response $response, array $args) {
    $limit = $request->getQueryParams()['limit'] ?? 10;
    $nome = $args['nome'] ?? 'mouse';
    $response->getBody()->write("{$limit} Produtos do banco de dados com o nome {$nome}");
    return $response;
});

$app->post('/produto', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();

    $id = $data['id'] ?? '';
    $nome = $data['nome'] ?? '';

    $data['message'] = "Produto inserido com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
});

$app->put('/produto/{id}', function (Request $request, Response $response, array $args) {
    $data = $request->getParsedBody();

    $nome = $data['nome'] ?? '';

    $data['id'] = $args['id'];
    $data['message'] = "Produto atualizado com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
});

$app->delete('/produto/{id}', function (Request $request, Response $response, array $args) {
    $id = $args['id'];

    $data['id'] = $id;
```

```
    $data['message'] = "Produto excluído com sucesso!";

    $response->getBody()->write(json_encode($data));
});

$app->run();
```



Aula 05 - Middlewares

https://www.slimframework.com/docs/v3/handlers/error.html




```
$configuration = [
    'settings' => [
        'displayErrorDetails' => true,
    ],
];
$configuration = new \Slim\Container($configuration);
```

Detalhando um erro

index.php

```
$app->post('/produto', function (Request $request, Response $response, array $args):Response {
    $data = $request->getParsedBody();

    $id = $data['id'] ?? '';
    $nome = $data['nome'] ?? '';

    $data['message'] = "Produto inserido com sucesso!";

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

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

Setup Git Sync

No Environment Cookies

POST _.baseUrl /produto Send 500 Internal Server Error 20.8 ms 2.3 KB 4 Minutes Ago

Filter

curso_api_slim

aula_04

DEL Excluir produto

PUT Atualizar produto

POST Inserir produto

aula_03

api_rest_laravel8

cgitar

WireMock

Viagens

JSON Auth Query Header 1 Do

```
1 {
2   "id":1,
3   "nome":"teclado"
4 }
```

Preview Header 6 Cookie Timeline

Slim Application Error

The application could not run because of the following error:

Details

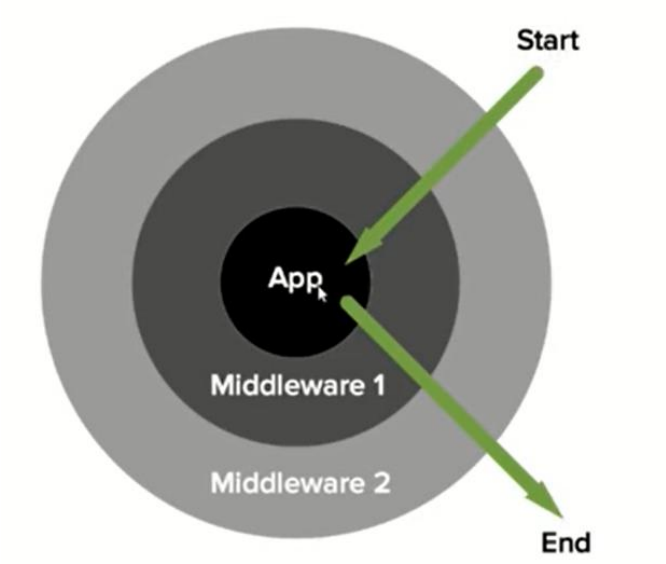
Type: TypeError
Message: Return value of Closure::{closure}() must implement interface Psr\Http\Message\ResponseInterface, int returned
File: C:\xampp\htdocs\curso_api_slim\index.php
Line: 46

Trace

```
#0 [internal function]: Closure->{closure}(Object(Slim\Http\Request), Object(Slim\Http\Response), Array)
#1 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\Handlers\Strategies\RequestResponse.php(40): cal
#2 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\Route.php(281): Slim\Handlers\Strategies\Request
#3 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\MiddlewareAwareTrait.php(117): Slim\Route->__inv
#4 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\Route.php(268): Slim\Route->callMiddlewareStack(
#5 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\App.php(503): Slim\Route->run(Object(Slim\Http\R
#6 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\MiddlewareAwareTrait.php(117): Slim\App->__invok
#7 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\App.php(392): Slim\App->callMiddlewareStack(Obje
#8 C:\xampp\htdocs\curso_api_slim\vendor\slim\slim\Slim\App.php(297): Slim\App->process(Object(Slim\Http
#9 C:\xampp\htdocs\curso_api_slim\index.php(71): Slim\App->run()
#10 {main}
```

Middlewares

https://www.slimframework.com/docs/v3/concepts/middleware.html



O middleware é um código intermediário que é executado entre rodar a aquisição e executar o código e executar o código e finalizar a execução.

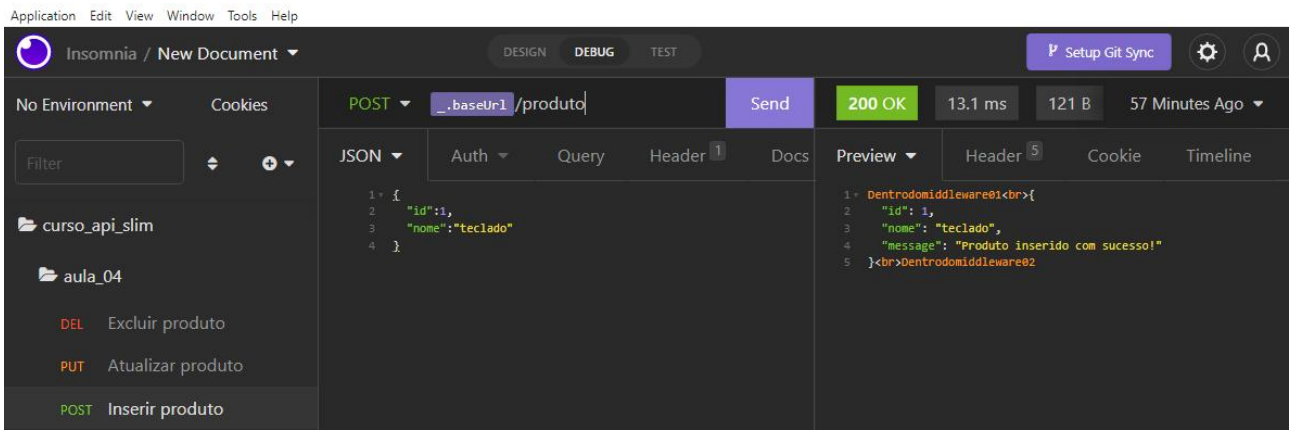
Um middleware pode ser utilizado para fazer uma autenticação.

Criando um middleware

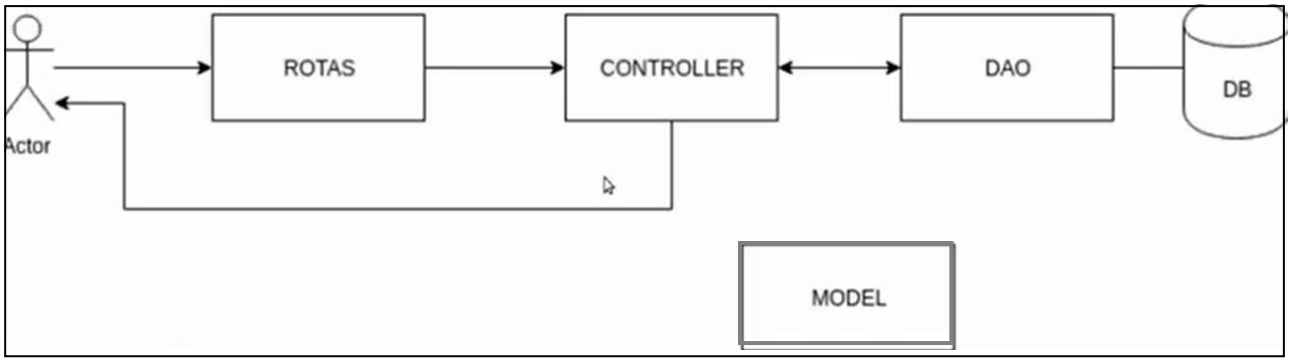
```
$mid01 = function(Request $request, Response $response, $next): Response{
  $response->getBody()->write("DENTRO DO MIDDLEWARE 01");
  $response = $response->$next($request, $response);
  $response->getBody()->write("DENTRO DO MIDDLEWARE 02");

  return $response;
}
```

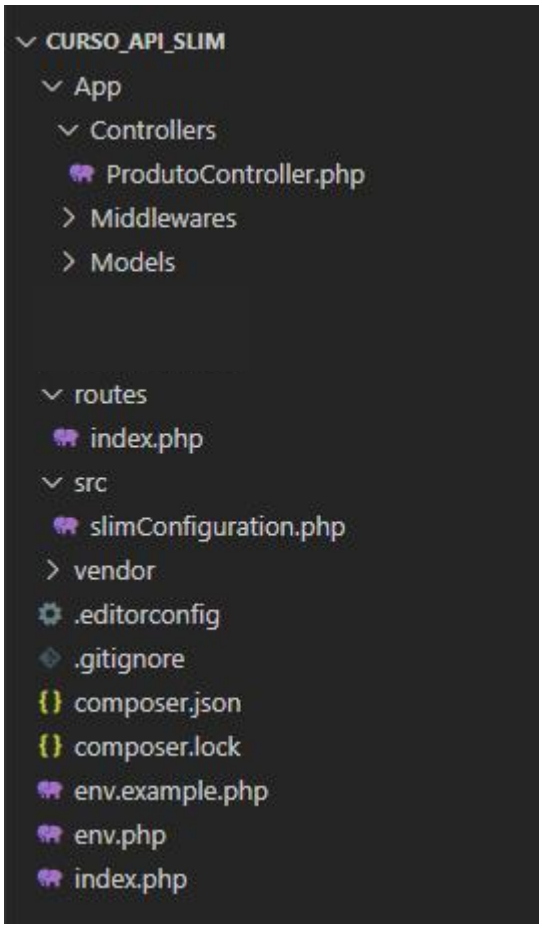
index.php

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require_once './vendor/autoload.php';

$configuration = [
    'settings' => [
        'displayErrorDetails' => true,
    ],
];
$c = new \Slim\Container($configuration);

$mid01 = function(Request $request, Response $response, $next): Response{
  $response->getBody()->write("Dentro do middleware 01");
  $response = $next($request, $response);
  $response->getBody()->write("Dentro do middleware 02");

  return $response;
};

$app = new \Slim\App($c);

$app->post('/produto', function (Request $request, Response $response, array $args): Response {
    $data = $request->getParsedBody();

    $id = $data['id'] ?? '';
    $nome = $data['nome'] ?? '';

    $data['message'] = "Produto inserido com sucesso!";

    $response->getBody()->write(json_encode($data));
    return $response->withHeader('Content-Type', 'application/json');
})->add($mid01);

$app->run();
```



Aula 06 - Arquitetura do Software





env.php

```
<?php

putenv('DISPLAY_ERRORS_DETAILS=' . true);
```

.gitignore

```
vendor/
.buildpath
.project
.settings/
composer.lock
env.php
composer.phar
.vscode/
```

index.php

```
<?php

require_once './vendor/autoload.php';
require_once './env.php';
require_once './src/slimConfiguration.php';
require_once './routes/index.php';
```

src\slimConfiguration.php

```
<?php

namespace src;

function slimConfiguration(): \Slim\Container
{
    $configuration = [
        'settings' => [
            'displayErrorDetails' => getenv('DISPLAY_ERRORS_DETAILS'),
        ],
    ];

    $container = new \Slim\Container($configuration);

    return $container;
}
```

Rotas

routes\index.php

```php
<?php

use function src\slimConfiguration;
use App\Controllers\ProdutoController;

$app = new \Slim\App(slimConfiguration());

// ========================================

$app->get('/', ProdutoController::class . ':getProdutos');

// ========================================

$app->run();
```

Controller

App\Controllers\ProdutoController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;

final class ProdutoController
{
    public function getProdutos(Request $request, Response $response, array $args): Response
    {
        $response = $response->withJson([
            "message" => "Hello World"
        ]);

        return $response;
    }
}
```

composer.json

```
{
    "require": {
      "slim/slim": "^3.0"
    },
    "autoload": {
      "psr-4": {
        "App\\": "App"
      }
    }
  }
```

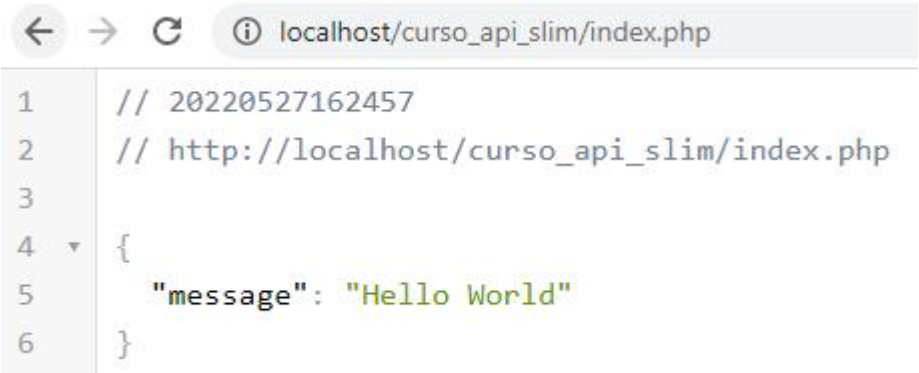
composer dump-autoload -o

```
C:\xampp\htdocs\curso_api_slim>composer dump-autoload -o
Generating optimized autoload files
Generated optimized autoload files containing 108 classes
```

vendor\composer\autoload_psr4.php

```
<?php

// autoload_psr4.php @generated by Composer

$vendorDir = dirname(__DIR__);
$baseDir = dirname($vendorDir);

return array(
    'Slim\\' => array($vendorDir . '/slim/slim/Slim'),
    'Psr\\Http\\Message\\' => array($vendorDir . '/psr/http-message/src'),
    'Psr\\Container\\' => array($vendorDir . '/psr/container/src'),
    'FastRoute\\' => array($vendorDir . '/nikic/fast-route/src'),
    'App\\' => array($baseDir . '/App'),
);
```

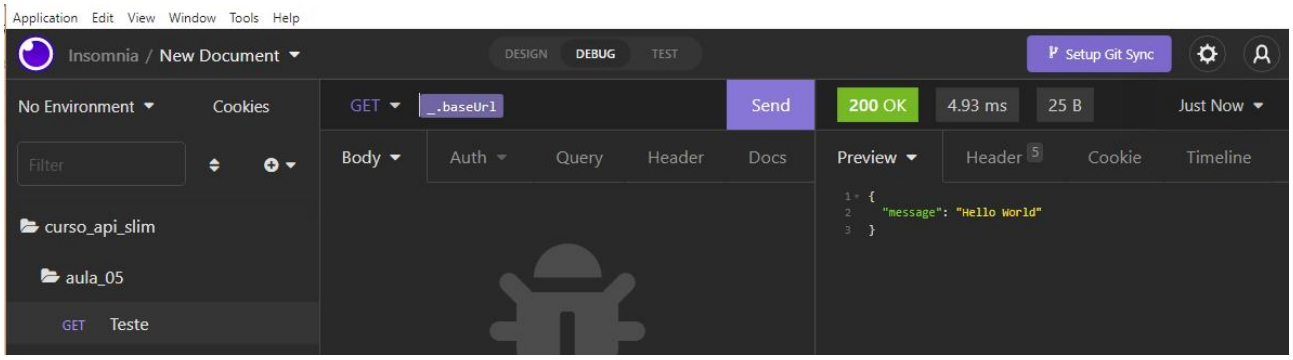
http://localhost/curso_api_slim/index.php



- No Insomnia:

Teste

GET
{{ _.baseUrl }}



Aula 07 - Construindo o banco de dados

- No MySQL Workbench:

Criando o BD e as tabelas

create_database_and_tables.sql

```
CREATE DATABASE codeeasy_gerenciador_de_lojas CHARACTER SET utf8 COLLATE utf8_general_ci;

USE codeeasy_gerenciador_de_lojas;

CREATE TABLE lojas (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    telefone VARCHAR(15) NOT NULL,
    endereco VARCHAR(200) NOT NULL,
    PRIMARY KEY(id)
);

CREATE TABLE produtos (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    loja_id INT UNSIGNED NOT NULL,
    nome VARCHAR(100) NOT NULL,
    preco DECIMAL(7,2) NOT NULL,
    quantidade INT UNSIGNED NOT NULL,
    PRIMARY KEY(id),
    CONSTRAINT fk_produtos_loja_id_lojas_id
    FOREIGN KEY (loja_id) REFERENCES lojas(id)
);
```

Inserindo dados nas tabelas

inserts.sql

```
USE codeeasy_gerenciador_de_lojas;

INSERT INTO lojas (nome, telefone, endereco)
VALUES ('EIPRO Tecnologia', '(12)3207-9133', 'Av. Dr. Nelson d\'Ávila, 1837');

INSERT INTO lojas (nome, telefone, endereco)
VALUES ('Continuum TI', '(11)97315-0491', 'R. Itaguaçu, 170');

INSERT INTO lojas (nome, telefone, endereco)
VALUES ('Digi Office Informatica', '(31)99908-1436', 'R. da Bahia, 1176');

INSERT INTO lojas (nome, telefone, endereco)
VALUES ('Infocentric Informatica', '(21)97467-7480', 'R. Cap. Cruz, 714');

INSERT INTO lojas (nome, telefone, endereco)
VALUES ('bitti - Soluções em TI', '(51)3664-3225', 'Av. Silva Jardim, 227');

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (1, 'teclado', 42.70, 10);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (1, 'mouse', 32.10, 15);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (1, 'cd-r gravável', 3.70, 35);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (1, 'mouse pad', 18.65, 12);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (2, 'pen-drive', 5.40, 60);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (2, 'dvd-r gravável', 3.70, 35);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (3, 'cd-r gravável', 3.70, 50);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (3, 'dvd-r gravável', 5.40, 10);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (3, 'pen-drive', 58.35, 5);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (4, 'mouse pad', 18.65, 10);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (4, 'teclado', 42.70, 8);
```

```
INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (5, 'mouse', 32.10, 10);

INSERT INTO produtos (loja_id, nome, preco, quantidade)
VALUES (5, 'dvd-r gravável', 5.40, 40);
```

Selecionando

selects.sql

SELECT * FROM lojas;

Result Grid | Filter Rows: | Edit: | Export/Import:

id	nome	telefone	endereco
1	EIPRO Tecnologia	(12)3207-9133	Av. Dr. Nelson d'Ávila, 1837
2	Continuum TI	(11)97315-0491	R. Itaguaçu, 170
3	Digi Office Informatica	(31)99908-1436	R. da Bahia, 1176
4	Infocentric Informatica	(21)97467-7480	R. Cap. Cruz, 714
5	bitti - Soluções em TI	(51)3664-3225	Av. Silva Jardim, 227
NULL	NULL	NULL	NULL

SELECT * FROM produtos;

Result Grid | Filter Rows: | Edit: | Export/Import:

id	loja_id	nome	preco	quantidade
1	1	teclado	42.70	10
2	1	mouse	32.10	15
3	1	cd-r gravável	3.70	35
4	1	mouse pad	18.65	12
5	2	pen-drive	5.40	60
6	2	dvd-r gravável	3.70	35
7	3	cd-r gravável	3.70	50
8	3	dvd-r gravável	5.40	10
9	3	pen-drive	58.35	5
10	4	mouse pad	18.65	10
11	4	teclado	42.70	8
12	5	mouse	32.10	10
13	5	dvd-r gravável	5.40	40
NULL	NULL	NULL	NULL	NULL

```
SELECT lojas.nome as loja,
produtos.nome as produto,
produtos.preco as preco,
produtos.quantidade as quantidade
FROM produtos
INNER JOIN lojas ON produtos.loja_id = lojas.id
WHERE produtos.nome = 'teclado'
ORDER BY produtos.nome;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

loja	produto	preco	quantidade
EIPRO Tecnologia	teclado	42.70	10
Infocentric Informatica	teclado	42.70	8

Atualizando dados

updates.sql

☐ Safe Updates (rejects UPDATEs and DELETEs with no restrictions)

```
UPDATE produtos
SET
  nome = 'mouse',
    preco = 27.50
WHERE
  nome = 'mouse';

SELECT * FROM produtos where nome='mouse';
```

Result Grid | Filter Rows: | Edit:

id	loja_id	nome	preco	quantidade
2	1	mouse	27.50	15
12	5	mouse	27.50	10
NULL	NULL	NULL	NULL	NULL

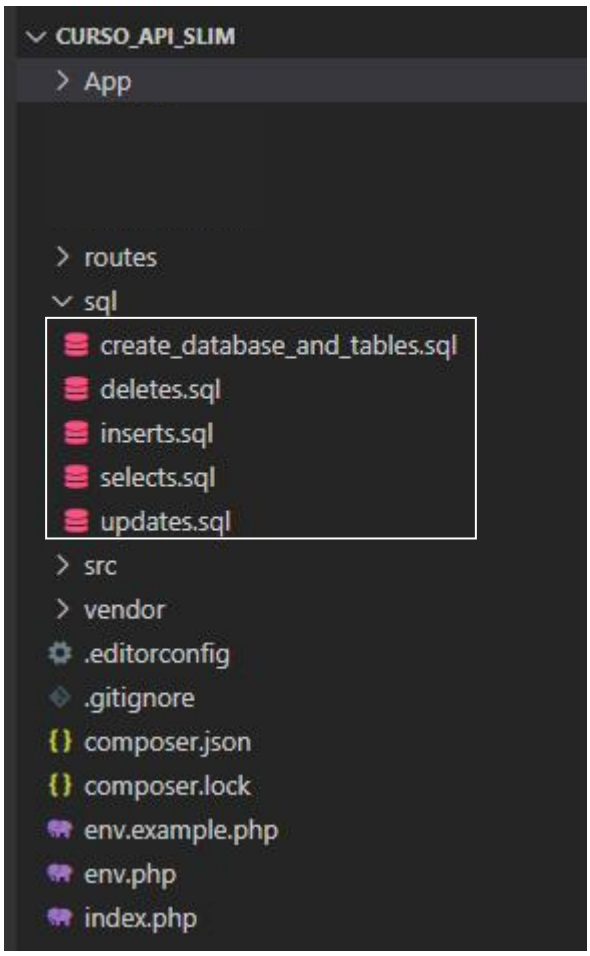
Excluindo um produto

deletes.sql

```
DELETE FROM produtos
WHERE nome = 'pen-drive' AND preco = 5.40

SELECT * FROM produtos where nome='pen-drive';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	loja_id	nome	preco	quantidade
9	3	pen-drive	58.35	5
NULL	NULL	NULL	NULL	NULL


CURSO_API_SLIM
App
routes
sql
create_database_and_tables.sql
deletes.sql
inserts.sql
selects.sql
updates.sql
src
vendor
.editorconfig
.gitignore
composer.json
composer.lock
env.example.php
env.php
index.php

Aula 08 - Classe PDO

Como conectar a API ao banco de dados

env.php

```
<?php

putenv('DISPLAY_ERRORS_DETAILS=' . true);

putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_HOST=localhost');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_DBNAME=codeeasy_gerenciador_de_lojas');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_USER=root');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PASSWORD=');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PORT=3306');
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\Conexao.php

```
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

abstract class Conexao
{
    /**
     * @var \PDO
     */
    protected $pdo;

    public function __construct()
    {
        $host = getenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_HOST');
        $port = getenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PORT');
        $user = getenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_USER');
        $pass = getenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PASSWORD');
        $dbname = getenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_DBNAME');

        $dsn = "mysql:host={$host};dbname={$dbname};port={$port}";

        $this->pdo = new \PDO($dsn, $user, $pass);
        $this->pdo->setAttribute(
            \PDO::ATTR_ERRMODE,
            \PDO::ERRMODE_EXCEPTION
        );
    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\LojasDAO.php

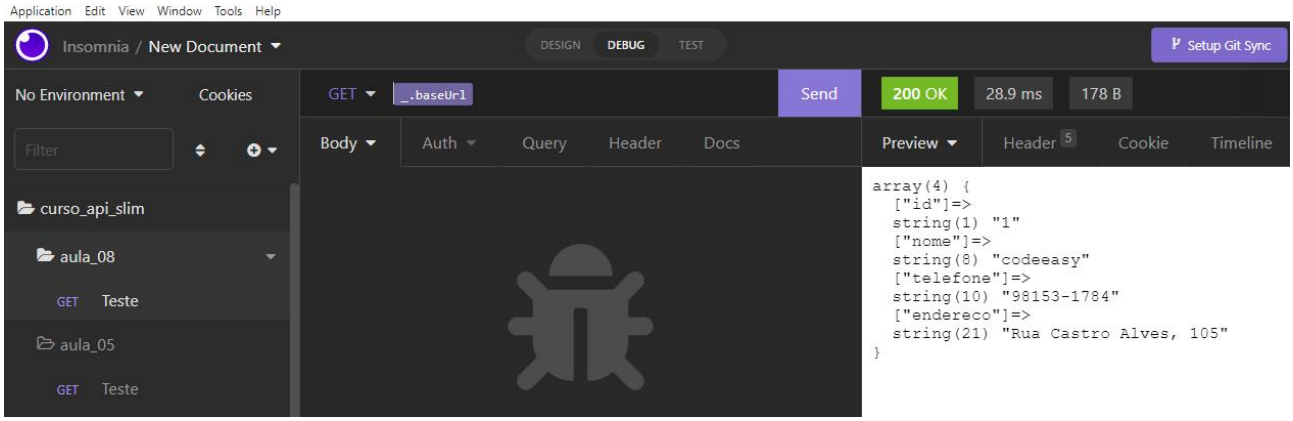
```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

class LojasDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function teste(){
        $lojas = $this->pdo
        ->query('SELECT * FROM lojas;')
        ->fetchAll(\PDO::FETCH_ASSOC);

        echo "<pre>";
        foreach($lojas as $loja) {
            var_dump($loja);
        }
        die;
    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\ProdutosDAO.php

```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

class ProdutosDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }
}
```

App\Controllers\ProdutoController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\LojasDAO;

final class ProdutoController
{
    public function getProdutos(Request $request, Response $response, array $args): Response
    {
        // $response->getBody()->write("Hello World!");

        $response = $response->withJson([
            "message" => "Hello World"
        ]);

        $lojasDAO = new LojasDAO();
        $lojasDAO->teste();

        return $response;
    }
}
```



Aula 09 - CRUD

routes\index.php

```
<?php

use function src\slimConfiguration;
use App\Controllers\ProdutoController;
use App\Controllers\LojaController;

$app = new \Slim\App(slimConfiguration());

// ==========================================

$app->get('/loja', LojaController::class . ':getLojas');
$app->post('/loja', LojaController::class . ':insertLoja');
$app->put('/loja', LojaController::class . ':updateLoja');
$app->delete('/loja', LojaController::class . ':deleteLoja');

$app->get('/produto', ProdutoController::class . ':getProdutos');
$app->post('/produto', ProdutoController::class . ':insertProduto');
$app->put('/produto', ProdutoController::class . ':updateProduto');
$app->delete('/produto', ProdutoController::class . ':deleteProduto');

// ==========================================

$app->run();
```

App\Controllers\LojaController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\LojasDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\LojaModel;
use Slim\Container;

final class LojaController
{

    public function getLojas(Request $request, Response $response, array $args)
    {
        $lojasDAO = new LojasDAO();
        $lojas = $lojasDAO->getAllLojas();
        $response = $response->withJson($lojas);

        return $response;
    }

    public function insertLoja(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $lojasDAO = new LojasDAO();
        $loja = new LojaModel();
        $loja->setNome($data['nome'])
            ->setEndereco($data['endereco'])
            ->setTelefone($data['telefone']);
        $lojasDAO->insertLoja($loja);

        $response = $response->withJson([
            'message' => 'Loja inserida com sucesso!'
        ]);

        return $response;
    }

    public function updateLoja(Request $request, Response $response, array $args): Response
    {

        $queryParams = $request->getQueryParams();
        $data = $request->getParsedBody();

        $id = (int)$queryParams['id'];

        $lojasDAO = new LojasDAO();
        $loja = new LojaModel();
        $loja->setId($id)
            ->setNome($data['nome'])
```

```
            ->setEndereco($data['endereco'])
            ->setTelefone($data['telefone']);

//          print_r($loja);
//          die;

        $lojasDAO->updateLoja($loja);

        $response = $response->withJson([
            'message' => 'Loja alterada com sucesso!'
        ]);

        return $response;
    }

    public function deleteLoja(Request $request, Response $response, array $args): Response
    {
        $queryParams = $request->getQueryParams();

        $lojasDAO = new LojasDAO();
        $id = (int)$queryParams['id'];
        $lojasDAO->deleteLoja($id);

        $response = $response->withJson([
            'message' => 'Loja excluída com sucesso!'
        ]);

        return $response;
    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\LojasDAO.php

```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

use App\Models\MySQL\CodeeasyGerenciadorDeLojas\LojaModel;

class LojasDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function getAllLojas(): array
    {
        $lojas = $this->pdo
            ->query('SELECT
                    id,
                    nome,
                    telefone,
                    endereco
                FROM lojas;')
            ->fetchAll(\PDO::FETCH_ASSOC);

        return $lojas;
    }

    public function insertLoja(LojaModel $loja): void
    {
        $statement = $this->pdo
            ->prepare('INSERT INTO lojas VALUES(
                null,
                :nome,
                :telefone,
                :endereco
            );');
        $statement->execute([
            'nome' => $loja->getNome(),
            'telefone' => $loja->getTelefone(),
            'endereco' => $loja->getEndereco()
        ]);
    }

    public function updateLoja(LojaModel $loja): void
    {

        $statement = $this->pdo
            ->prepare('UPDATE lojas SET
                    nome = :nome,
                    telefone = :telefone,
                    endereco = :endereco
                WHERE
```

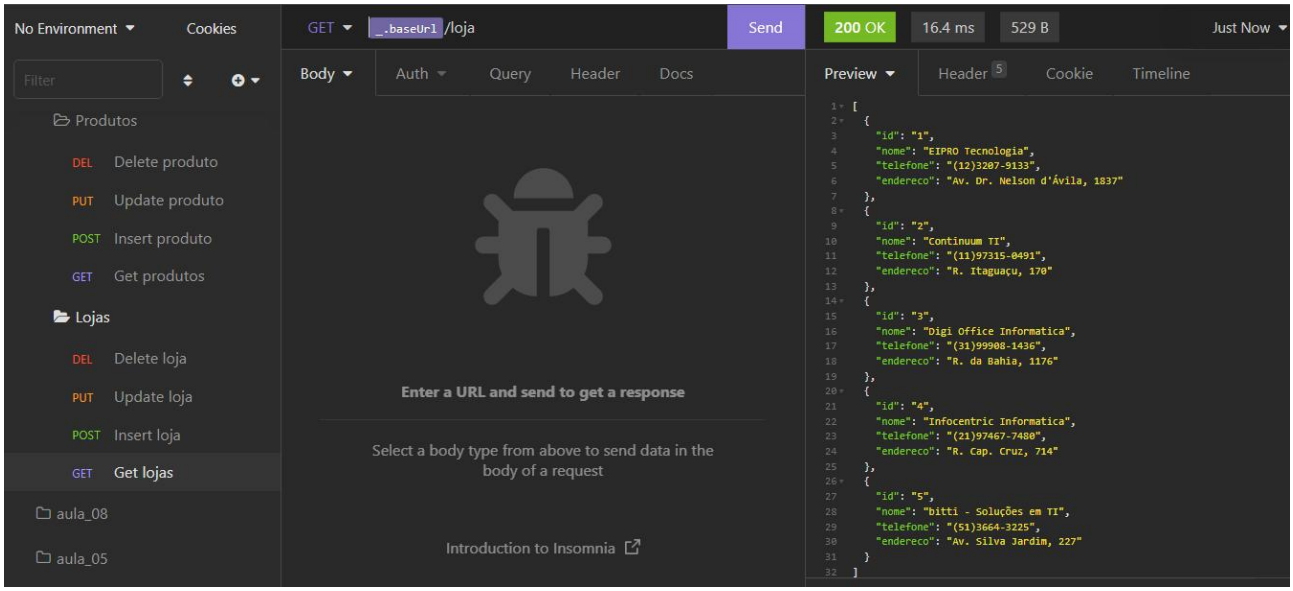
```
                id = :id
            ;');
        $statement->execute([
            'nome' => $loja->getNome(),
            'telefone' => $loja->getTelefone(),
            'endereco' => $loja->getEndereco(),
            'id' => $loja->getid()
        ]);
    }

    public function deleteLoja(int $id): void
    {
        $statement = $this->pdo
            ->prepare('DELETE FROM produtos WHERE loja_id = :id;
                DELETE FROM lojas WHERE id = :id;
            ');
        $statement->execute([
            'id' => $id
        ]);
    }
}
```

App\Models\MySQL\CodeeasyGerenciadorDeLojas\LojaModel.php

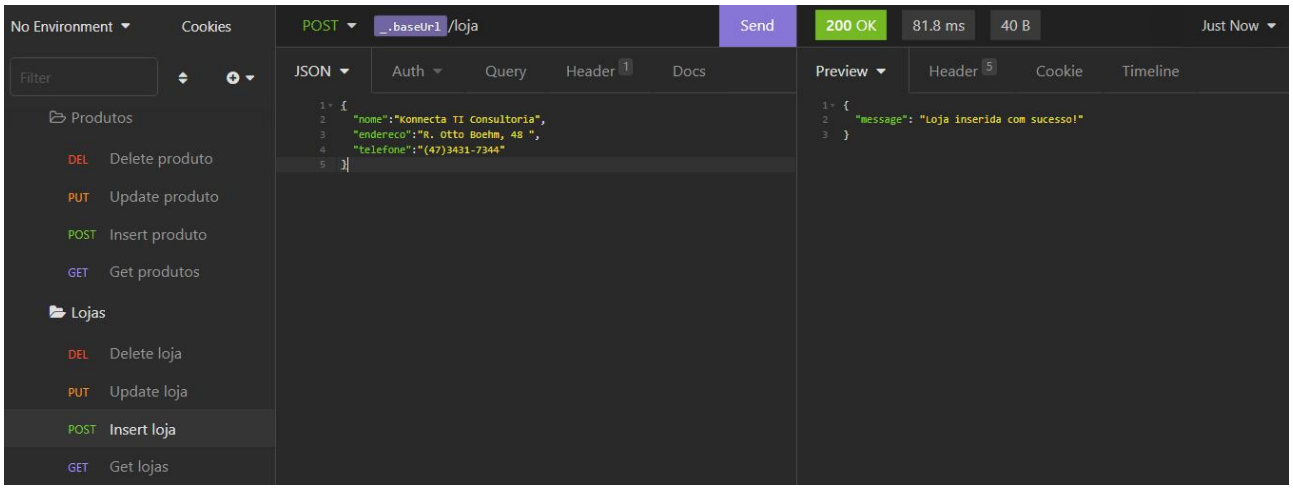
```php
<?php

namespace App\Models\MySQL\CodeeasyGerenciadorDeLojas;

final class LojaModel
{
    /**
     * @var int
     */
    private $id;
    /**
     * @var string
     */
    private $nome;
    /**
     * @var string
     */
    private $telefone;
    /**
     * @var string
     */
    private $endereco;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    public function setId(int $id): LojaModel
    {
        $this->id = $id;
        return $this;
    }

    /**
     * @return string
     */
    public function getNome(): string
    {
        return $this->nome;
    }
    /**
     * @param string $nome
     * @return LojaModel
     */
    public function setNome(string $nome): LojaModel
    {
        $this->nome = $nome;
```

```
        return $this;
    }

    /**
     * @return string
     */
    public function getTelefone(): string
    {
        return $this->telefone;
    }
    /**
     * @param string $telefone
     * @return LojaModel
     */
    public function setTelefone(string $telefone): LojaModel
    {
        $this->telefone = $telefone;
        return $this;
    }

    /**
     * @return string
     */
    public function getEndereco(): string
    {
        return $this->endereco;
    }
    /**
     * @param string $endereco
     * @return LojaModel
     */
    public function setEndereco(string $endereco): LojaModel
    {
        $this->endereco = $endereco;
        return $this;
    }
}
```

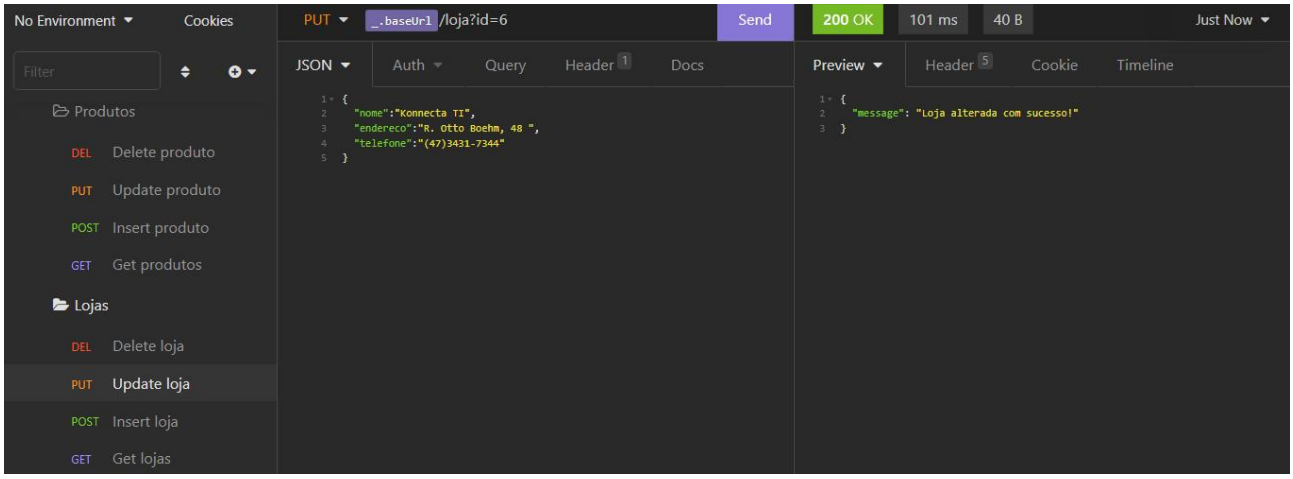
Get loja



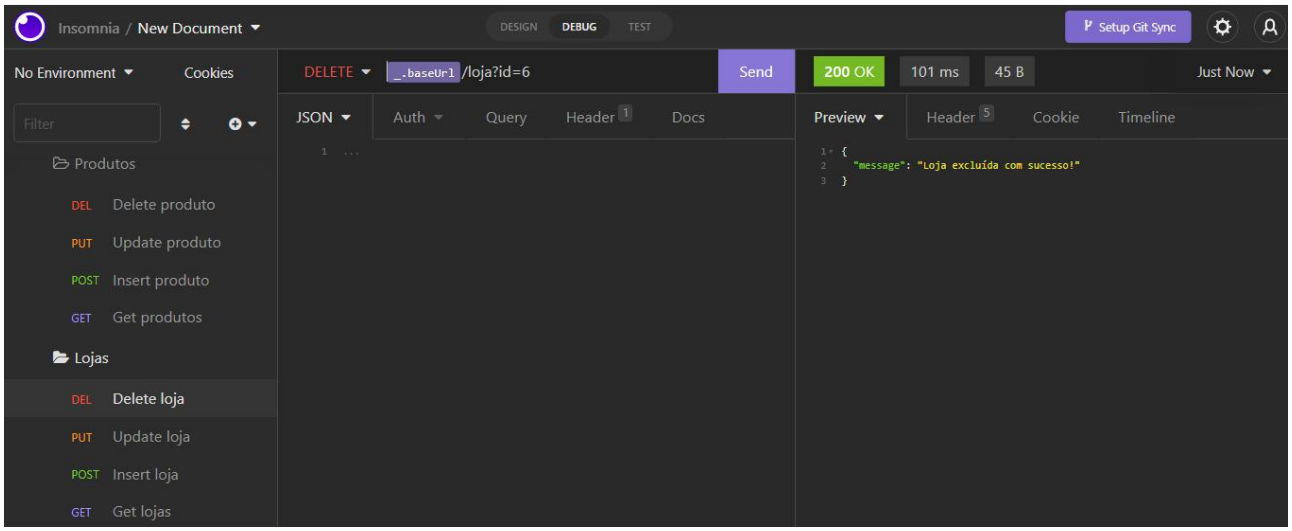
Insert loja



Update loja



Delete loja


Insomnia / New Document
DESIGN
DEBUG
TEST
Setup Git Sync
No Environment
Cookies
DELETE
_.baseUrl /loja?id=6
Send
200 OK
101 ms
45 B
Just Now
Filter
JSON
Auth
Query
Header 1
Docs
Preview
Header 5
Cookie
Timeline
Produtos
DEL Delete produto
PUT Update produto
POST Insert produto
GET Get produtos
Lojas
DEL Delete loja
PUT Update loja
POST Insert loja
GET Get lojas
{
"message": "Loja excluida com sucesso!"
}

App\Controllers\ProdutoController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\ProdutosDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\ProdutoModel;

final class ProdutoController
{
    public function getProdutos(Request $request, Response $response, array $args): Response
    {
        $queryParams = $request->getQueryParams();

        $produtosDAO = new ProdutosDAO();
        $id = (int)$queryParams['loja_id'];
        $produtos = $produtosDAO->getAllProdutosFromLoja($id);
        $response = $response->withJson($produtos);

        return $response;
    }

    public function insertProduto(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $produtosDAO = new ProdutosDAO();
        $produto = new ProdutoModel();
        $produto->setLojaId($data['loja_id']);
        $produto->setNome($data['nome']);
        $produto->setPreco($data['preco']);
        $produto->setQuantidade($data['quantidade']);

        $produtosDAO->insertProduto($produto);

        $response = $response->withJson([
            'message' => 'Produto inserido com sucesso!'
        ]);

        return $response;
    }

    public function updateProduto(Request $request, Response $response, array $args): Response
    {

        $queryParams = $request->getQueryParams();
        $data = $request->getParsedBody();

        $id = (int)$queryParams['id'];
```

```
        $produtosDAO = new ProdutosDAO();
        $produto = new ProdutoModel();
        $produto->setId($id)
            ->setLojaId($data['loja_id'])
            ->setNome($data['nome'])
            ->setPreco($data['preco'])
            ->setQuantidade($data['quantidade']);

        $produtosDAO->updateProduto($produto);

        $response = $response->withJson([
            'message' => 'Produto atualizado com sucesso!'
        ]);

        return $response;

    }

    public function deleteProduto(Request $request, Response $response, array $args): Response
    {

        $queryParams = $request->getQueryParams();

        $produtosDAO = new ProdutosDAO();
        $id = (int)$queryParams['id'];

        $produtosDAO->deleteProduto($id);

        $response = $response->withJson([
            'message' => 'Produto excluiído com sucesso!'
        ]);

        return $response;

    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\ProdutosDAO.php

```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

use App\Models\MySQL\CodeeasyGerenciadorDeLojas\ProdutoModel;

class ProdutosDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function getAllProdutosFromLoja(int $lojaId): array
    {
        $statement = $this->pdo
            ->prepare('SELECT
                    *
                FROM produtos
                WHERE
                    loja_id = :loja_id
            ;');
        $statement->bindParam(':loja_id', $lojaId, \PDO::PARAM_INT);
        $statement->execute();
        $produtos = $statement->fetchAll(\PDO::FETCH_ASSOC);

        return $produtos;
    }

    public function insertProduto($produto): void
    {
        $statement = $this->pdo
            ->prepare('INSERT INTO produtos VALUES(
                null,
                :loja_id,
                :nome,
                :preco,
                :quantidade
            );');
        $statement->execute([
            'loja_id' => $produto->getLojaId(),
            'nome' => $produto->getNome(),
            'preco' => $produto->getPreco(),
            'quantidade' => $produto->getQuantidade()
        ]);
    }

    public function updateProduto(ProdutoModel $produto): void
    {

        $statement = $this->pdo
            ->prepare('UPDATE produtos SET
```

```
                loja_id = :loja_id,
                nome = :nome,
                preco = :preco,
                quantidade = :quantidade
            WHERE
                id = :id
        ;');
    $statement->execute([
        'loja_id' => $produto->getLojaId(),
        'nome' => $produto->getNome(),
        'preco' => $produto->getPreco(),
        'quantidade' => $produto->getQuantidade(),
        'id' => $produto->getid()
    ]);
}

public function deleteProduto(int $id): void
{

    $statement = $this->pdo
        ->prepare('DELETE FROM produtos WHERE id = :id;
        ');
    $statement->execute([
        'id' => $id
    ]);
}

}
```

App\Models\MySQL\CodeeasyGerenciadorDeLojas\ProdutoModel.php

```php
<?php

namespace App\Models\MySQL\CodeeasyGerenciadorDeLojas;

final class ProdutoModel
{
    /**
     * @var int
     */
    private $id;
    /**
     * @var int
     */
    private $loja_id;
    /**
     * @var string
     */
    private $nome;
    /**
     * @var float
     */
    private $preco;
    /**
     * @var int
     */
    private $quantidade;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }
    /**
     * @param int $id
     * @return ProdutoModel
     */
    public function setId(int $id): ProdutoModel
    {
        $this->id = $id;
        return $this;
    }

    /**
     * @return int
     */
    public function getLojaId(): int
    {
        return $this->loja_id;
    }
    /**
```

```
 * @param int $loja_id
 * @return ProdutoModel
 */
public function setLojaId(int $loja_id): ProdutoModel
{
    $this->loja_id = $loja_id;
    return $this;
}

/**
 * @return string
 */
public function getNome(): string
{
    return $this->nome;
}
/**
 * @param int $nome
 * @return ProdutoModel
 */
public function setNome(string $nome): ProdutoModel
{
    $this->nome = $nome;
    return $this;
}

/**
 * @return float
 */
public function getPreco(): float
{
    return $this->preco;
}
/**
 * @param int $preco
 * @return ProdutoModel
 */
public function setPreco(float $preco): ProdutoModel
{
    $this->preco = $preco;
    return $this;
}

/**
 * @return int
 */
public function getQuantidade(): int
{
    return $this->quantidade;
}
/**
 * @param int $quantidade
 * @return ProdutoModel
 */
```

```
    public function setQuantidade(string $quantidade): ProdutoModel
    {
        $this->quantidade = $quantidade;
        return $this;
    }

}
```

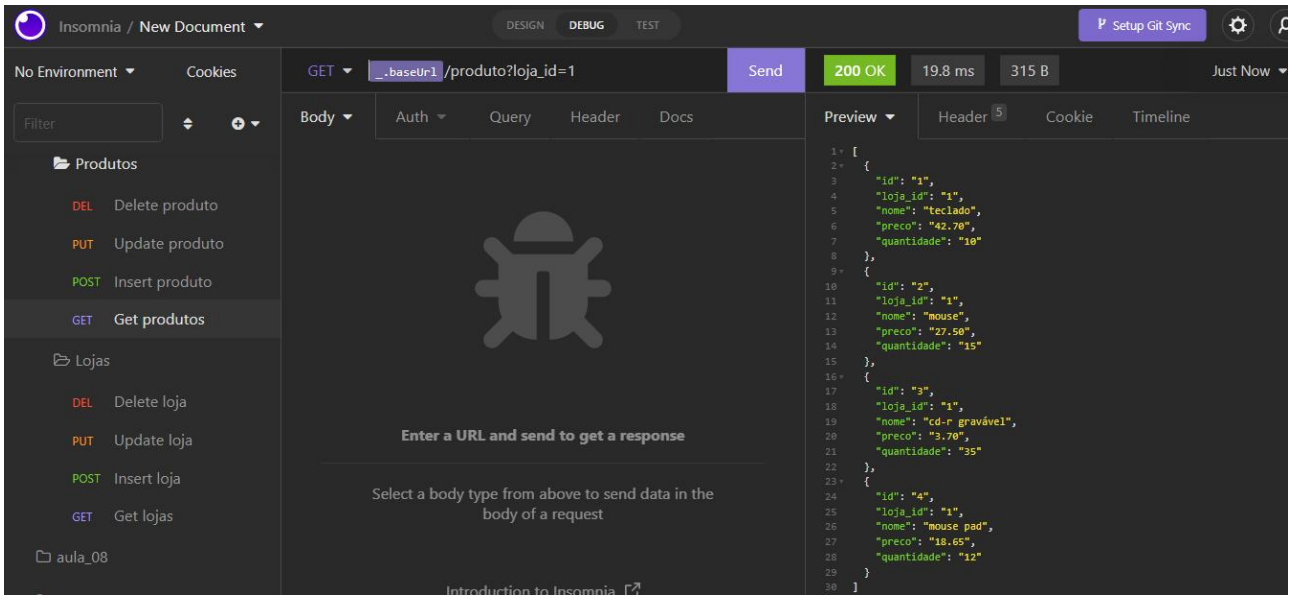
- No terminal:

composer dump-autoload -o

```
Roberto@DESKTOP-HGDUAQT MINGW64 /c/xampp/htdocs/curso_api_slim
$ composer dump-autoload -o
Generating optimized autoload files
Generated optimized autoload files containing 103 classes
```

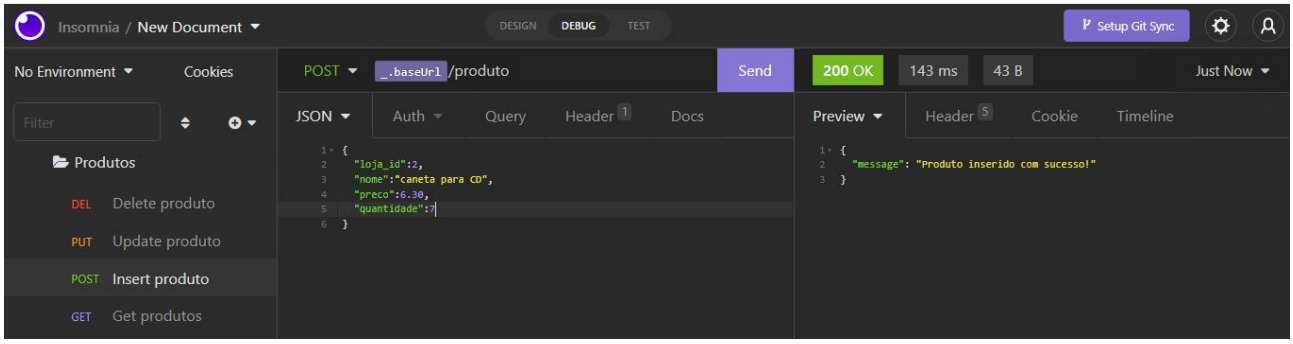
Get produtos

Produtos da loja de id =1



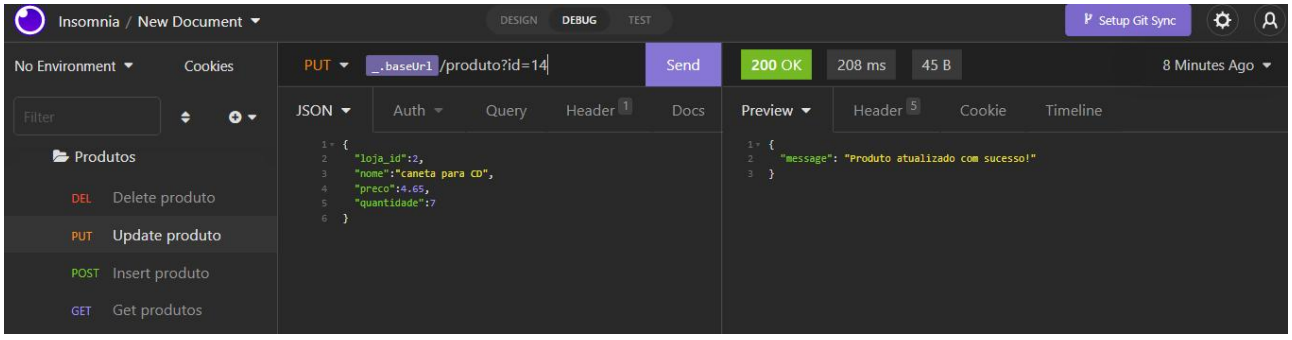
Insert produto

Inserindo um produto na loja de id = 2



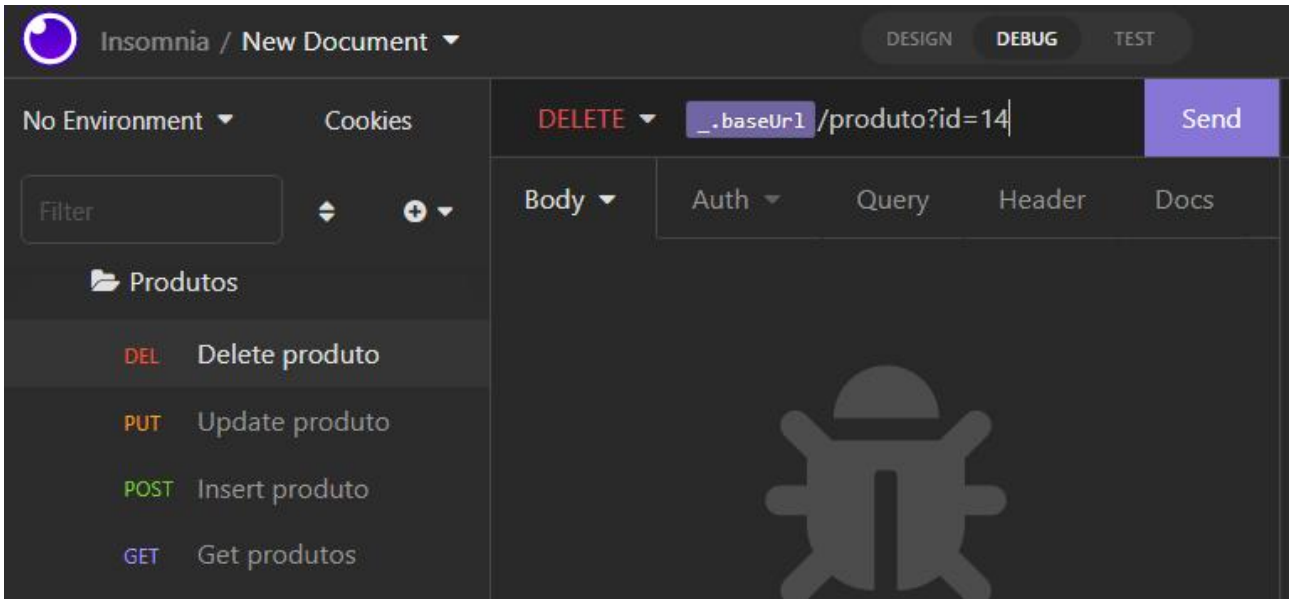
Update produto

Atualizando preço do produto de id = 14 da loja loja_id = 2:



Delete produto

Excluindo produto de id = 14



Aula 10 - Autenticação - Parte 1 - Basic Auth

Mika Tuupola

https://github.com/tuupola

https://github.com/tuupola/slim-basic-auth

Instalando o slim basic auth

- No terminal entre com o comando:

composer require tuupola/slim-basic-auth

```
Roberto@DESKTOP-HGDUAQT MINGW64 /c/xampp/htdocs/curso_api_slim
$ composer require tuupola/slim-basic-auth
Info from https://repo.packagist.org: #StandWithUkraine
Using version ^3.3 for tuupola/slim-basic-auth
./composer.json has been updated
Running composer update tuupola/slim-basic-auth
Loading composer repositories with package information
Updating dependencies
Lock file operations: 6 installs, 0 updates, 0 removals
  - Locking psr/http-factory (1.0.1)
  - Locking psr/http-server-handler (1.0.1)
  - Locking psr/http-server-middleware (1.0.1)
  - Locking tuupola/callable-handler (1.1.0)
  - Locking tuupola/http-factory (1.4.0)
  - Locking tuupola/slim-basic-auth (3.3.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 6 installs, 0 updates, 0 removals
  - Downloading tuupola/http-factory (1.4.0)
  - Downloading tuupola/callable-handler (1.1.0)
  - Downloading tuupola/slim-basic-auth (3.3.1)
  - Installing psr/http-factory (1.0.1): Extracting archive
  - Installing psr/http-server-handler (1.0.1): Extracting archive
  - Installing tuupola/http-factory (1.4.0): Extracting archive
  - Installing psr/http-server-middleware (1.0.1): Extracting archive
  - Installing tuupola/callable-handler (1.1.0): Extracting archive
  - Installing tuupola/slim-basic-auth (3.3.1): Extracting archive
Generating autoload files
3 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

composer.json

```
{
    "require": {
      "slim/slim": "^3.0",
        "tuupola/slim-basic-auth": "^3.3"
    },
    "autoload": {
      "psr-4": {
        "App\\": "App"
      }
    }
  }
```

src\basicAuth.php

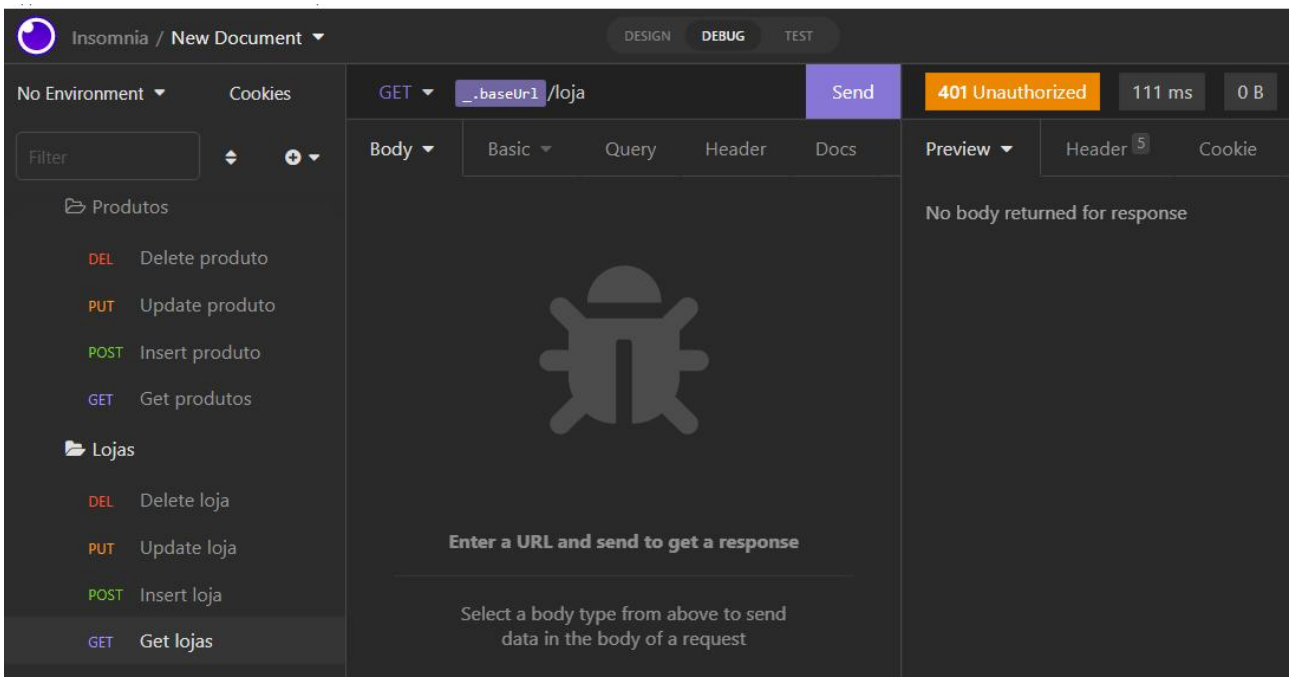
```
<?php

namespace src;

use Tuupola\Middleware\HttpBasicAuthentication;

function basicAuth(): HttpBasicAuthentication
{
    return new HttpBasicAuthentication([
        "users" => [
            "root" => ""
        ]
    ]);
}
```

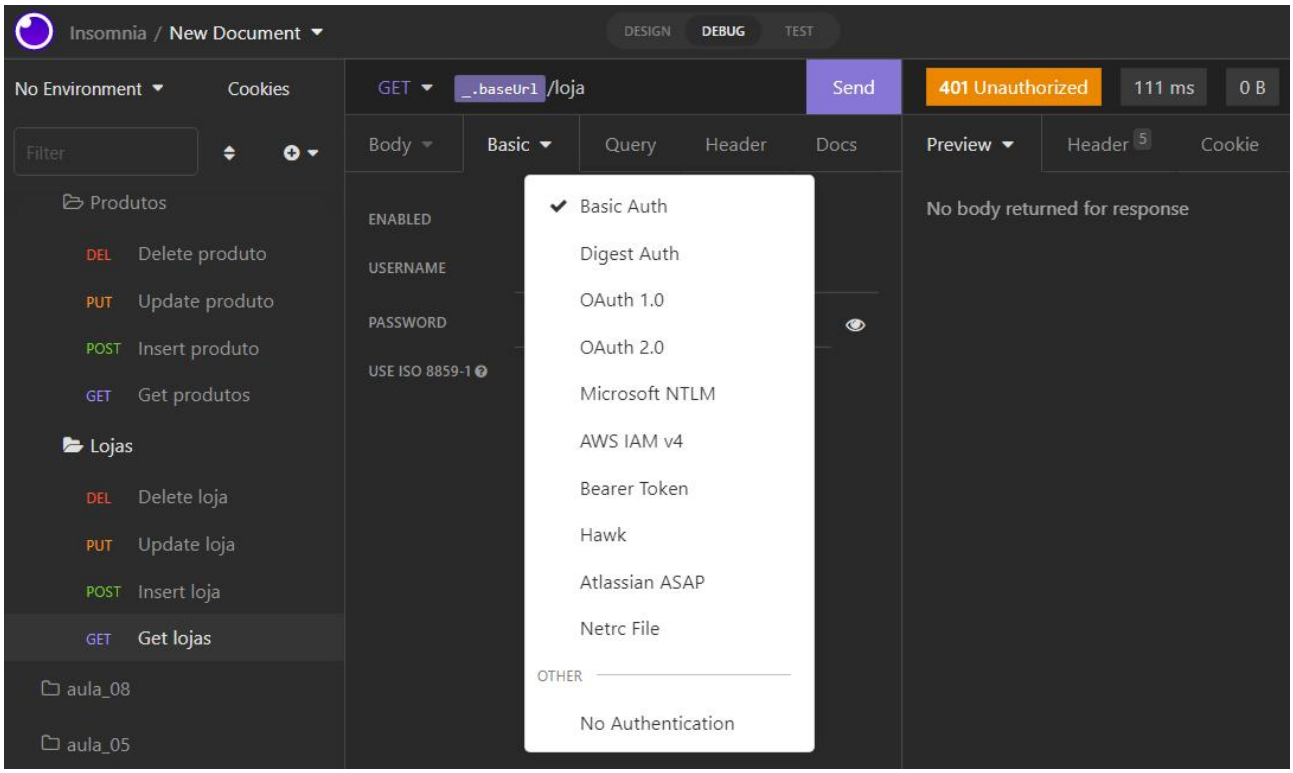
index.php

```
<?php

require_once './vendor/autoload.php';
require_once './env.php';
require_once './src/slimConfiguration.php';
require_once './src/basicAuth.php';
require_once './routes/index.php';
```

routes\index.php

```php
<?php

use function src\{
    slimConfiguration,
    basicAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
};

$app = new \Slim\App(slimConfiguration());

// ==========================================

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ==========================================

$app->run();
```


Insomnia / New Document
DESIGN
DEBUG
TEST
No Environment
Cookies
GET
_.baseUrl /loja
Send
401 Unauthorized
111 ms
0 B
Filter
Body
Basic
Query
Header
Docs
Preview
Header 5
Cookie
Produtos
DEL Delete produto
PUT Update produto
POST Insert produto
GET Get produtos
Lojas
DEL Delete loja
PUT Update loja
POST Insert loja
GET Get lojas
No body returned for response
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request


Insomnia / New Document
DESIGN
DEBUG
TEST
No Environment
Cookies
GET
_.baseUrl /loja
Send
401 Unauthorized
111 ms
0 B
Filter
Body
Basic
Query
Header
Docs
Preview
Header 5
Cookie
Produtos
DEL Delete produto
PUT Update produto
POST Insert produto
GET Get produtos
Lojas
DEL Delete loja
PUT Update loja
POST Insert loja
GET Get lojas
aula_08
aula_05
ENABLED
USERNAME
PASSWORD
USE ISO 8859-1
Basic Auth
Digest Auth
OAuth 1.0
OAuth 2.0
Microsoft NTLM
AWS IAM v4
Bearer Token
Hawk
Atlassian ASAP
Netrc File
OTHER
No Authentication
No body returned for response

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

Filter

Produtos

DEL Delete produto

PUT Update produto

POST Insert produto

GET Get produtos

Lojas

DEL Delete loja

PUT Update loja

POST Insert loja

GET Get lojas

aula_08

aula_05

GET _.baseUrl /loja Send

200 OK 76 ms 529 B

Body Basic Query Header Docs

ENABLED

USERNAME root

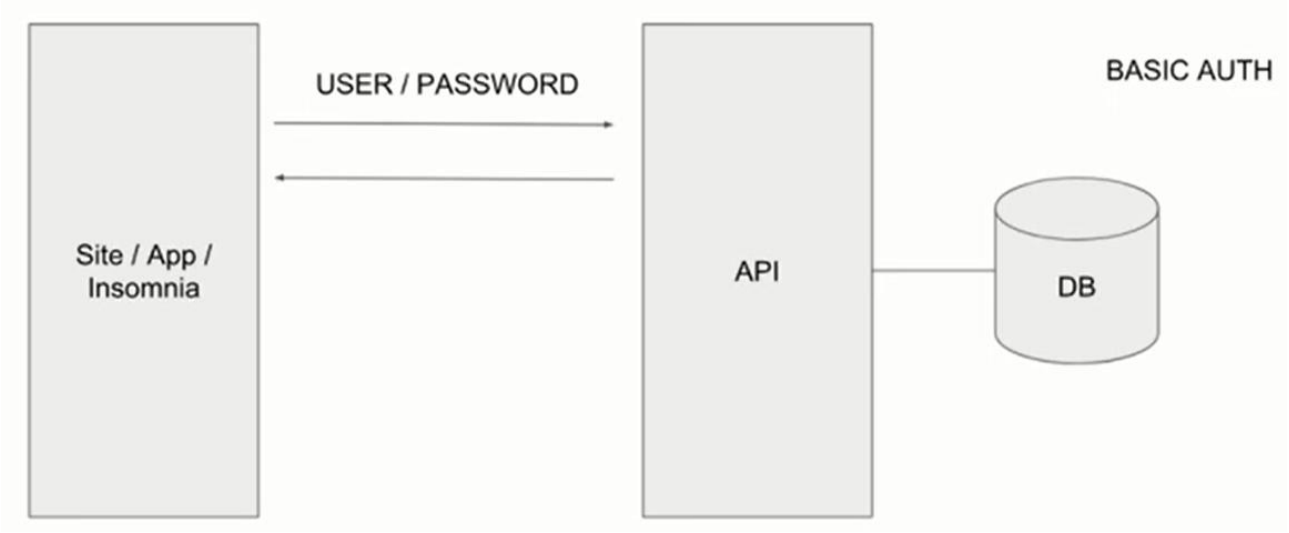
PASSWORD

USE ISO 8859-1

Preview Header 5 Cookie

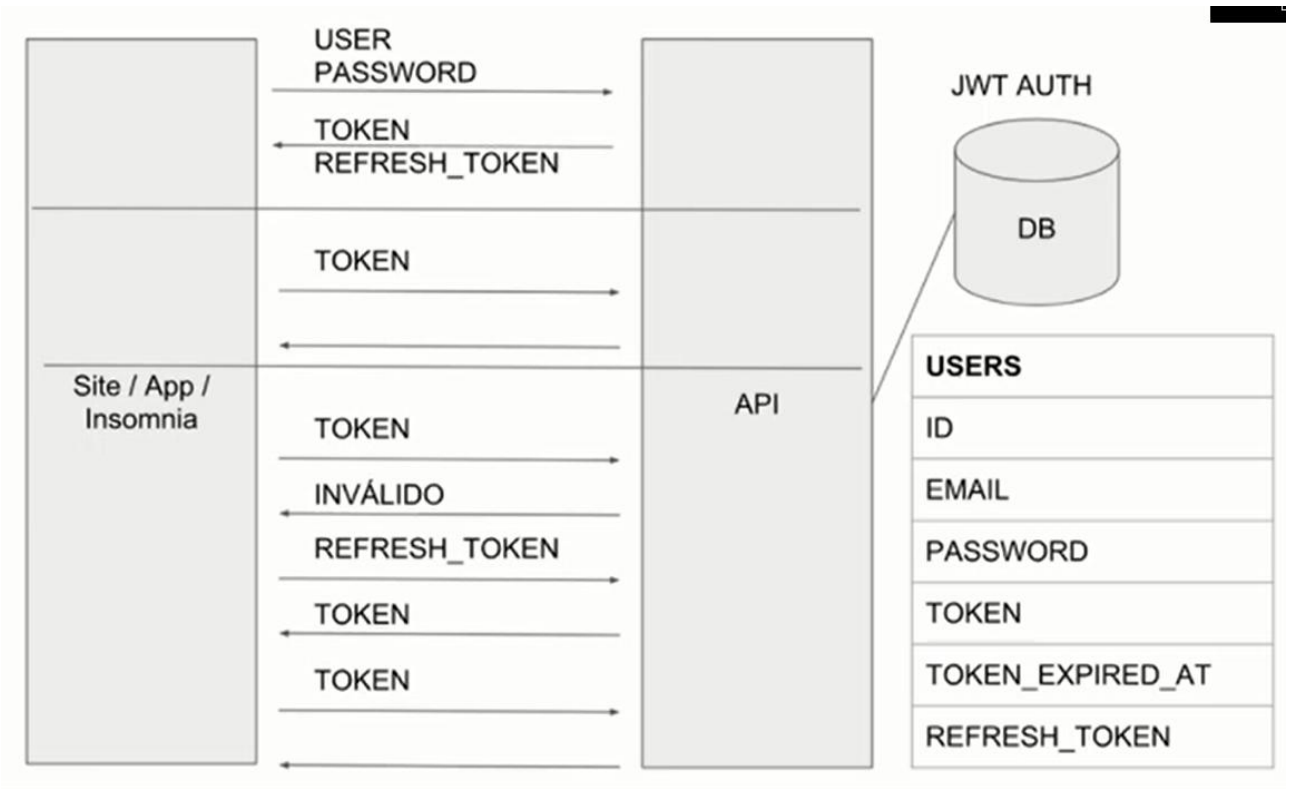

```
[
  {
    "id": "1",
    "nome": "EIPRO Tecnologia",
    "telefone": "(12)3207-9133",
    "endereco": "Av. Dr. Nelson d'Ávila, 1837"
  },
  {
    "id": "2",
    "nome": "Continuum TI",
    "telefone": "(11)97315-0491",
    "endereco": "R. Itaguaçu, 170"
  },
  {
    "id": "3",
    "nome": "Digi Office Informatica",
    "telefone": "(31)99908-1436",
    "endereco": "R. da Bahia, 1176"
  },
  {
    "id": "4",
    "nome": "Infocentric Informatica",
    "telefone": "(21)97467-7480",
    "endereco": "R. Cap. Cruz, 714"
  },
  {
    "id": "5",
    "nome": "bitti - Soluções em TI",
    "telefone": "(51)3664-3225",
    "endereco": "Av. Silva Jardim, 227"
  }
]
```

Aula 10 - Autenticação - Parte 2 - JWT Auth (REMAKE)



Sistema de autenticação por token

O JWT é uma ferramenta para construir tokens. O JWT é disponibilizado para várias linguagens, inclusive para o PHP.



https://jwt.io/

JWT

Debugger Libraries Introduction Ask

Crafted by auth0

JSON Web Tokens are an open, industry standard **RFC 7519** method for representing claims securely between two parties.

JWT.IO allows you to decode, verify and generate JWT.

LEARN MORE ABOUT JWT | SEE JWT LIBRARIES

Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.

Algorithm HS256

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```


token

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

Get the JWT Handbook for free! Download it now and get up-to-speed faster.

DOWNLOAD EBOOK

Um token é basicamente uma string de algo que foi criptografado.

Um token com JWT é composto de três partes:

1. O cabeçalho que indica o tipo de criptografia (algoritmo) usado.

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2. O payload que é o conteúdo que será enviado.

PAYLOAD: DATA

```
{
  "id": 10,
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

3. Uma chave de assinatura que é uma chave secreta. Toda vez que alguém enviar um token do front-end, se tiver com a chave de assinatura errada, não vai conseguir usar o token.

Exemplo:

```
php -a
echo sha1("codeeasy");
```

```
Microsoft Windows [versão 10.0.16299.1087]
(c) 2017 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Roberto>php -a
Interactive shell

php > echo sha1("codeeasy");
c5ee2312f151d751c1c4950d8d2a614a80806170
```

c5ee2312f151d751c1c4950d8d2a614a80806170

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  c5ee2312f151d751c1c49
) ☐ secret base64 encoded
```

token gerado:


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JpZCI6MTAsInN1YiI6IjEyMzQ1Njc4OTAiLCJuY
W1lIjoiSm9obiBEb2UiLCJpYXQiOjE1MTYyMzkw
MjJ9.QF7xaGWjpD7F_O2Ef1uhiYor2POCBQTy67
pNBtAT1XI
```

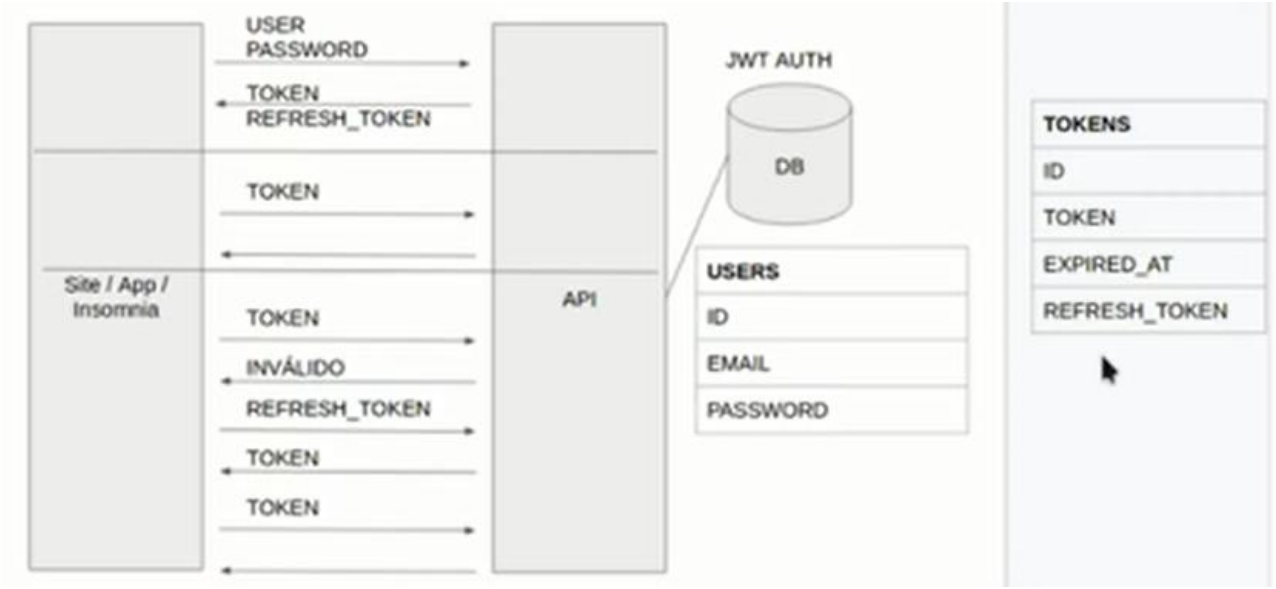

- Por segurança, não informe nenhum dado sensível (sigiloso) no token.

- A chave de assinatura não é decodificada (não é mostrada).

- A chave de assinatura é a parte mais importante a ser enviada do front-end para o back-end porque envolve a segurança. Se ela estiver errada, o token não será aceito.

- Nunca entregue essa chave de assinatura para ninguém.

Diagrama

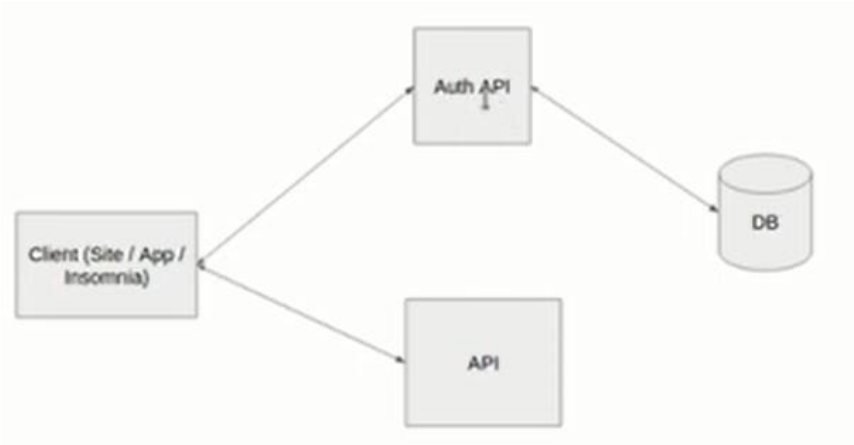


Um token possui:

1. ID;
2. token (o token em si);
3. expired_at (data de expiração do token);
4. refresh_token;

- O usuário envia para a API: o usuário (email) e a senha (password).
- A API irá verificar se esse usuário e senha existem no banco de dados e, em caso positivo, a API irá criar um token que será enviado ao usuário.
- Supondo que a data de expiração do token é de dois dias, passado dois dias o token não será mais válido.
- Nesse caso deverá ser usado o refresh_token para gerar um novo token com uma nova data de expiração.

- O tempo de expiração do token, normalmente é informado dentro do conteúdo (payload).

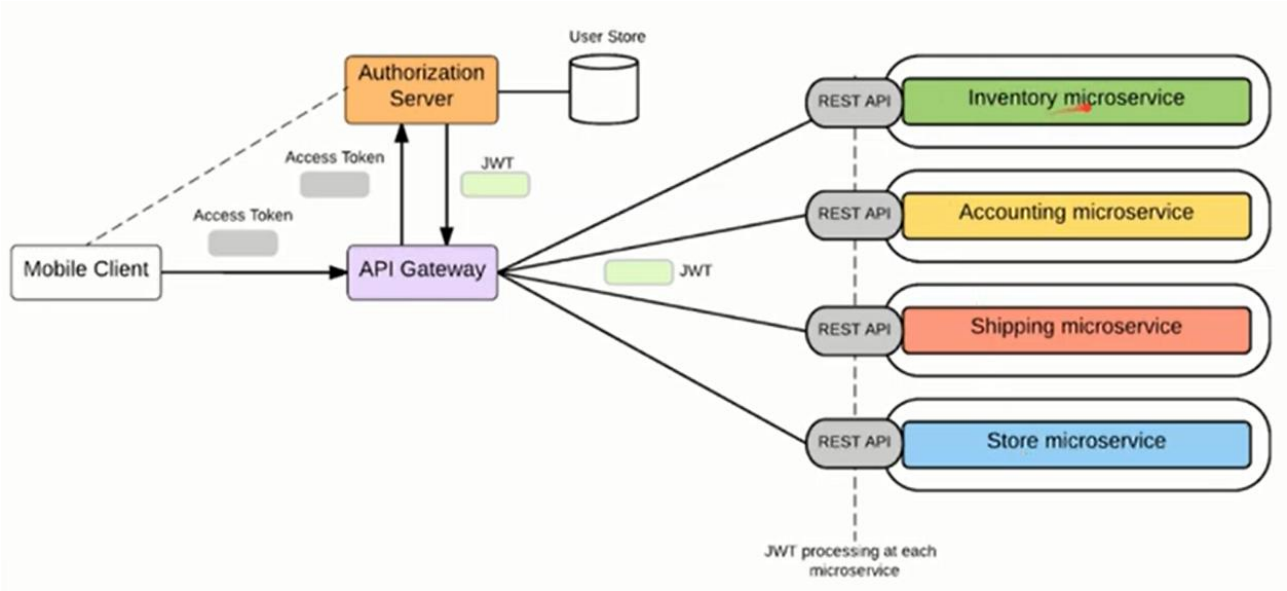


- Você poderá montar o sistema de autenticação em um servidor separado (em um projeto separado) e a API consegue fazer a autenticação mesmo sem estar conectado ao banco de dados.

- Todas as operações descritas no diagrama são executadas automaticamente sem o usuário sequer saber o que está sendo feito.

- Quando o sistema começar a ficar mais robusto, você poderá separar uma API só para autenticação em um servidor separado. Isso irá deixar a sua API mais limpa, que irá se preocupar apenas com a lógica de negócios (aquilo que deve ser feito).

Sistema mais completo (com microserviços)



Um microserviço permite separar a sua API em diversas API's para, se necessário, escalar o sistema. Isso trará uma melhoria na performance do sistema.

Inventory mcroservice: só para gerenciar inventários.
Accounting microservice: por exemplo, feita em Python, trabalha com coisas mais avançadas de cálculos.

Alguns microserviços são bem pequenos.

Vantagem: É possível criar equipes distintas para trabalhar em cada API.
Desvantagem: A desvantagem é que se acaba tendo um gasto maior utilizando mais servidores.

- Um microserviço tem um API Gateway que roteia para todas as API's (tem um ponto único de acesso).

Mobile Client é o front-end (por exemplo, o Insomnia).
Authorization Server: servidor para trabalhar só com autorização e autenticação de usuários.

Existe um sistema de autenticação mais robusto (OAuth) que além de trabalhar com token ele também trabalha com autenticação da aplicação.

Vamos construir tudo numa só API criando uma tabela para usuários (users) e uma tabela onde iremos salvar os tokens.

Um usuário pode ter muitos tokens (ele pode, por exemplo, fazer o login no notebook e no celular), porisso a necessidade da tabela de tokens.

Aula 10 - Autenticação - Parte 3 - JWT Auth

create_users_table.sql

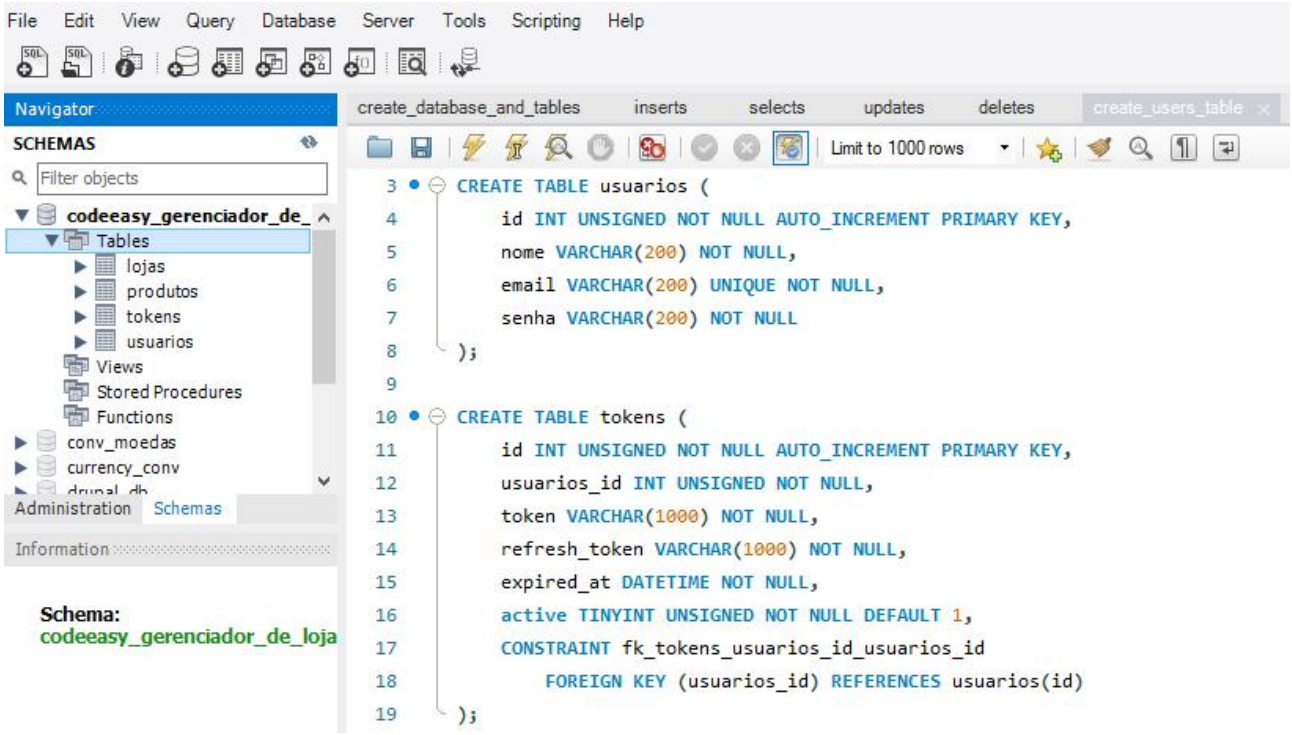
- Usando o MySQL Workbench construa as duas tabelas a seguir:

Construindo a tabela de usuários

```
CREATE TABLE usuarios (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   nome VARCHAR(200) NOT NULL,
   email VARCHAR(200) UNIQUE NOT NULL,
   senha VARCHAR(200) NOT NULL
);
```

Contruindo a tabela de tokens

```
CREATE TABLE tokens (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   usuarios_id INT UNSIGNED NOT NULL,
   token VARCHAR(1000) NOT NULL,
   refresh_token VARCHAR(1000) NOT NULL,
   expired_at DATETIME NOT NULL,
   active TINYINT UNSIGNED NOT NULL DEFAULT 1,
   CONSTRAINT fk_tokens_usuarios_id_usuarios_id
      FOREIGN KEY (usuarios_id) REFERENCES usuarios(id)
);
```



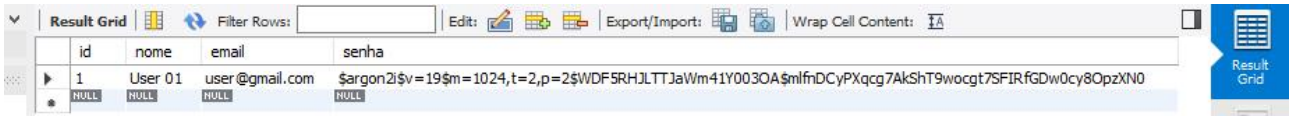
Inserindo dados na tabela de usuários (users)

insert_users.sql

```
INSERT INTO usuarios (
    nome,
    email,
    senha
) VALUES (
    'User 01',
    'user@gmail.com',
  '$argon2i$v=19$m=1024,t=2,p=2$WDF5RHJLTTJaWm41Y003OA$mlfnDCyPXqcg7AkShT9wocgt7SFIRfGDw
0cy8OpzXN0'
);
```

- Foi utilizado o novo sistema de geração de senhas de hash do PHP (Password Hash) implementado no PHP 7.2 com argon2i. E é o que vamos usar quando for criar um usuário no banco de dados.

- A senha utilizada foi teste123.



id	nome	email	senha
1	User 01	user@gmail.com	$argon2i$v=19$m=1024,t=2,p=2$WDF5RHJLTTJaWm41Y003OA$mlfnDCyPXqcg7AkShT9wocgt7SFIRfGDw0cy8OpzXN0
NULL	NULL	NULL	NULL

php -a

echo password_hash('teste123', PASSWORD_ARGON2I);

```
C:\Users\Roberto>php -a
Interactive shell

php > echo password_hash('teste123', PASSWORD_ARGON2I);
$argon2i$v=19$m=65536,t=4,p=1$aXZNLmpIS1libjNxVlhtWA$JC3mX1YseSKIhjR4I4K0aIlvjEkGGvNS14WMjhP4E6Q
php >
```

$argon2i$v=19$m=65536,t=4,p=1$aXZNLmpIS1libjNxVlhtWA$JC3mX1YseSKIhjR4I4K0aIlvjEkGGvNS14WMjh
P4E6Q

Readme

README.md

APIs REST com PHP 7 e Slim Framework

Material do curso

* Autor: Felipe Renan Vieira
* E-mail: feliperenanvieira@gmail.com
* Github: https://github.com/frv-dev
* Site: [https://codeeasy.com.br](https://www.codeeasy.com.br)
* Chat: https://gitter.im/frv-dev/CodeEasy

CONFIGURAÇÃO

Copie o arquivo `env.example.php` para `env.php` e preencha com as informações necessárias.

Rode no terminal `composer install` ou `php composer.phar install` dependendo de como você usa o composer.

Use os códigos no diretório `sql/` para criar o seu banco de dados trabalhando com o SGBD MySQL.

Etapas

ETAPAS.md

* [*] Construir as tabela no banco de dados
* [ ] Construir a rota de login
  * [ ] Verificar se o email existe
  * [ ] Verificar se a senha está correta
  * [ ] Construir o token
  * [ ] Salvar o token no banco de dados
  * [ ] Retornar o token para o usuário
* [ ] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

- [*] Construir as tabela no banco de dados
- [ ] Construir a rota de login
  - [ ] Verificar se o email existe
  - [ ] Verificar se a senha está correta
  - [ ] Construir o token
  - [ ] Salvar o token no banco de dados
  - [ ] Retornar o token para o usuário
- [ ] Validação das rotas através do token
- [ ] Construir a rota do refresh_token

Instalando a biblioteca slim-jwt-auth

https://github.com/tuupola/slim-jwt-auth

composer require tuupola/slim-jwt-auth

```
Roberto@DESKTOP-HGDUAQT MINGW64 /c/xampp/htdocs/curso_api_slim
$ composer require tuupola/slim-jwt-auth
Info from https://repo.packagist.org: #StandWithUkraine
Using version ^3.6 for tuupola/slim-jwt-auth
./composer.json has been updated
Running composer update tuupola/slim-jwt-auth
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
  - Locking firebase/php-jwt (v5.5.1)
  - Locking psr/log (1.1.4)
  - Locking tuupola/slim-jwt-auth (3.6.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
  - Downloading firebase/php-jwt (v5.5.1)
  - Downloading tuupola/slim-jwt-auth (3.6.0)
  - Installing psr/log (1.1.4): Extracting archive
  - Installing firebase/php-jwt (v5.5.1): Extracting archive
  - Installing tuupola/slim-jwt-auth (3.6.0): Extracting archive
1 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating autoload files
4 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
```

composer.json

```
{
   "require": {
     "slim/slim": "^3.0",
       "tuupola/slim-basic-auth": "^3.3",
       "tuupola/slim-jwt-auth": "^3.6"
   },
   "autoload": {
     "psr-4": {
       "App\\": "App"
     }
   }
 }
```

Construindo a rota de login

routes\index.php

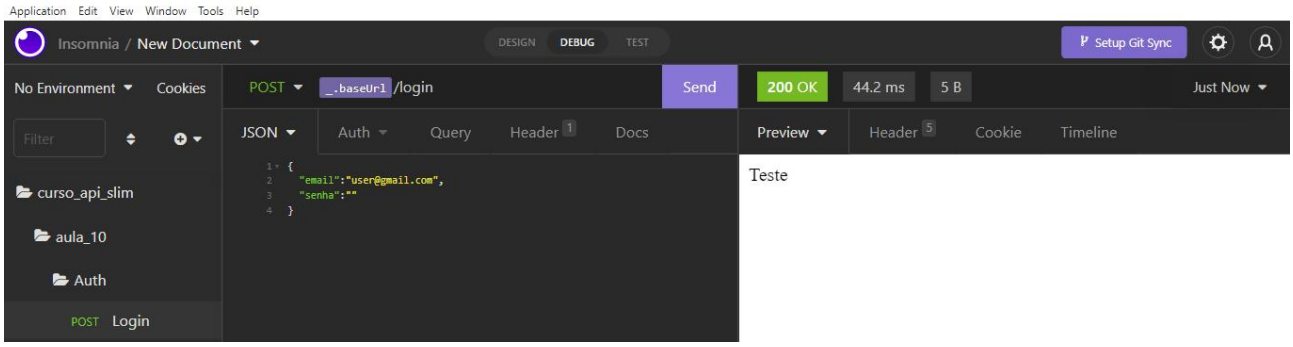
```
<?php

use function src\{
    slimConfiguration,
    basicAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController
};

$app = new \Slim\App(slimConfiguration());

// ========================================

$app->post('/login', AuthController::class . ':login');

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ========================================

$app->run();
```

App\Controllers\AuthController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $response->getBody()->write("Teste");

        return $response;
    }
}
```

- No Insomnia:

POST
{{ _.baseUrl }}/login



App\Models\MySQL\CodeeasyGerenciadorDeLojas\UsuarioModel.php

```php
<?php

namespace App\Models\MySQL\CodeeasyGerenciadorDeLojas;

final class UsuarioModel
{
    /**
     * @var int
     */
    private $id;
    /**
     * @var string
     */
    private $nome;
    /**
     * @var string
     */
    private $email;
    /**
     * @var string
     */
    private $senha;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @param int $id
     * @return self
     */
    public function setId(int $id): self
    {
        $this->id = $id;
        return $this;
    }

    /**
     * @return string
     */
    public function getNome(): string
    {
        return $this->nome;
    }

    /**
     * @param string $nome
     * @return self
```

```
     */
    public function setNome(string $nome): self
    {
        $this->nome = $nome;
        return $this;
    }

    /**
     * @return string
     */
    public function getEmail(): string
    {
        return $this->email;
    }

    /**
     * @param string $email
     * @return self
     */
    public function setEmail(string $email): self
    {
        $this->email = $email;
        return $this;
    }

    /**
     * @return string
     */
    public function getSenha(): string
    {
        return $this->senha;
    }

    /**
     * @param string $senha
     * @return self
     */
    public function setSenha(string $senha): self
    {
        $this->senha = $senha;
        return $this;
    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO.php

```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

use App\Models\MySQL\CodeeasyGerenciadorDeLojas\LojaModel;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\UsuarioModel;

class UsuariosDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function getUserByEmail(string $email): ?UsuarioModel
    {
        $statement = $this->pdo
            ->prepare('SELECT
                    id,
                    nome,
                    email,
                    senha
                FROM usuarios
                WHERE email = :email;
            ');
        $statement->bindParam('email', $email);
        $statement->execute();
        $usuarios = $statement->fetchAll(\PDO::FETCH_ASSOC);
        if(count($usuarios) === 0)
            return null;
        $usuario = new UsuarioModel();
        $usuario->setId($usuarios[0]['id'])
            ->setNome($usuarios[0]['nome'])
            ->setEmail($usuarios[0]['email'])
            ->setSenha($usuarios[0]['senha']);
        return $usuario;
    }
}
```

Verificar se o email existe

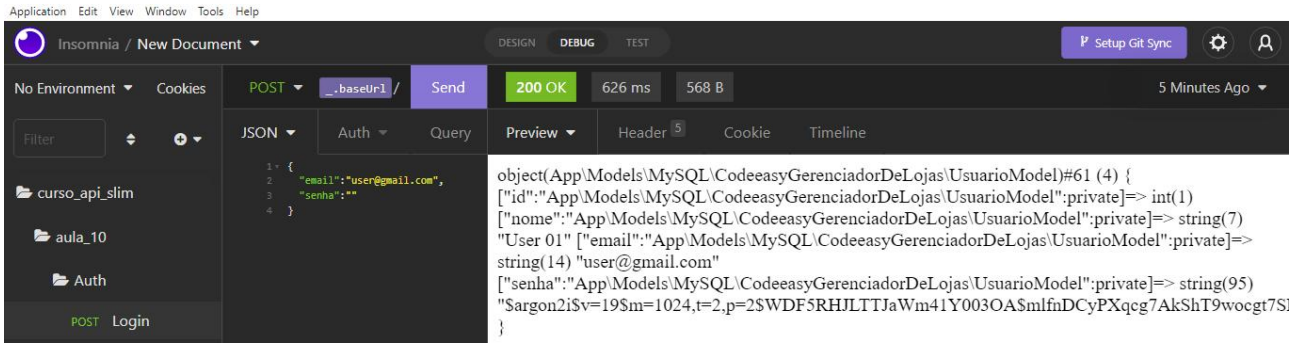
App\Controllers\AuthController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        var_dump($usuario);

        return $response;
    }
}
```

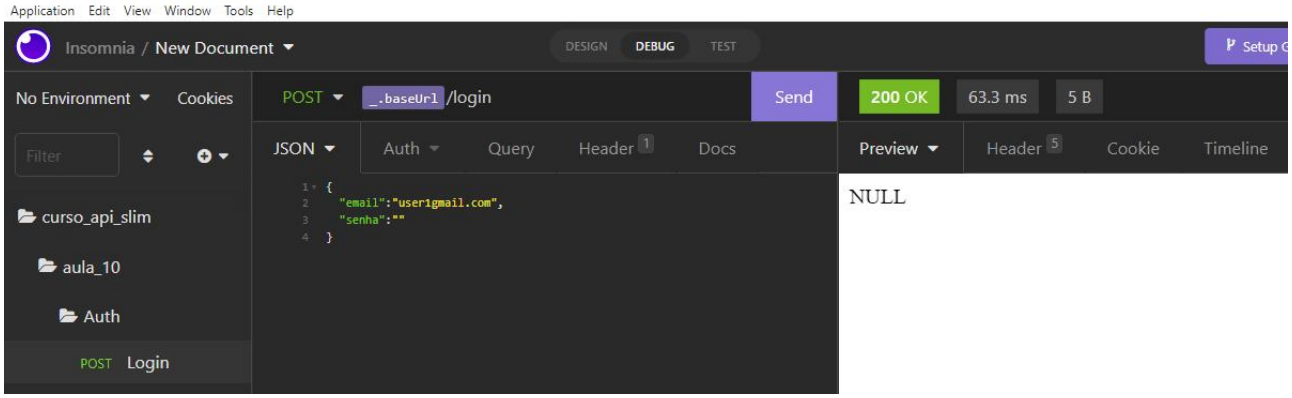
- No Insomnia:

Auth/Login

POST
{{ _.baseUrl }}/login



- Se for passado um email que não existe, retorna null.



ETAPAS.md

* [*] Construir as tabela no banco de dados
* [ ] Construir a rota de login
    * [*] Verificar se o email existe
    * [ ] Verificar se a senha está correta
    * [ ] Construir o token
    * [ ] Salvar o token no banco de dados
    * [ ] Retornar o token para o usuário
* [ ] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

Verificar se a senha está correta

App\Controllers\AuthController.php

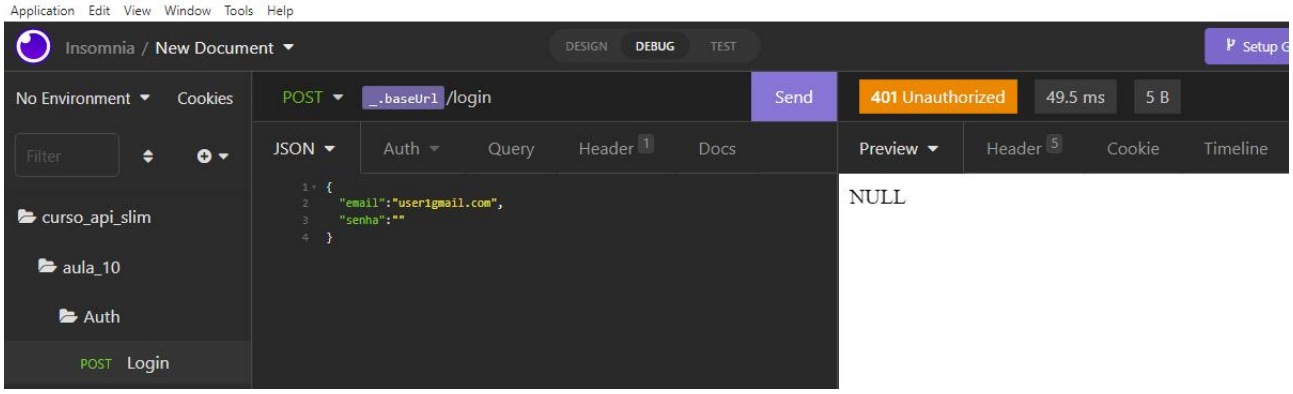
```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        var_dump($usuario);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        return $response;
    }
}
```

- No Insomnia:

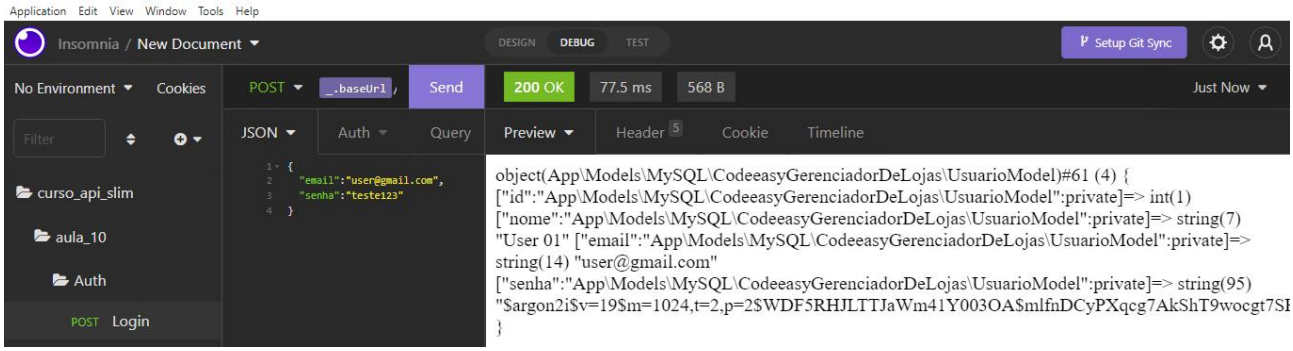
Auth/Login

POST
{{ _.baseUrl }}/login

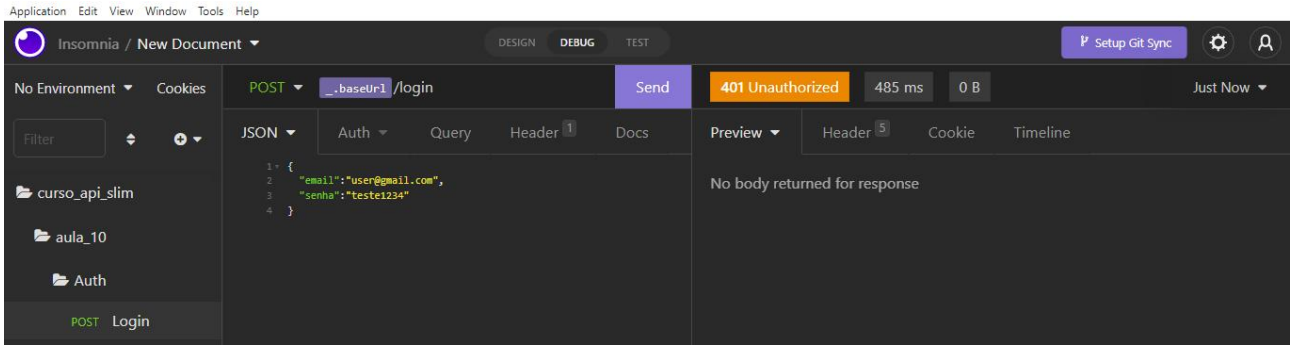
Senha não enviada:



Senha correta:



Senha inválida:



ETAPAS.md

* [*] Construir as tabela no banco de dados
* [ ] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [ ] Construir o token
    * [ ] Salvar o token no banco de dados
    * [ ] Retornar o token para o usuário
* [ ] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

Construir o token

github php jwt firebase

- Não é necessário instalar essa biblioteca porque ela já está incluída na biblioteca do Tuupola.

.env

```
<?php

putenv('DISPLAY_ERRORS_DETAILS=' . true);

putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_HOST=localhost');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_DBNAME=codeeasy_gerenciador_de_lojas');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_USER=root');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PASSWORD=');
putenv('CODEEASY_GERENCIADOR_DE_LOJAS_MYSQL_PORT=3306');

putenv('JWT_SECRET_KEY=codeeasy');
```

O valor codeeasy como chave de assinatura em um caso real, não deve ser usado. Em um caso real, é aconselhável usar uma chave de assinatura criptografada, que não deve ser informada a ninguém.

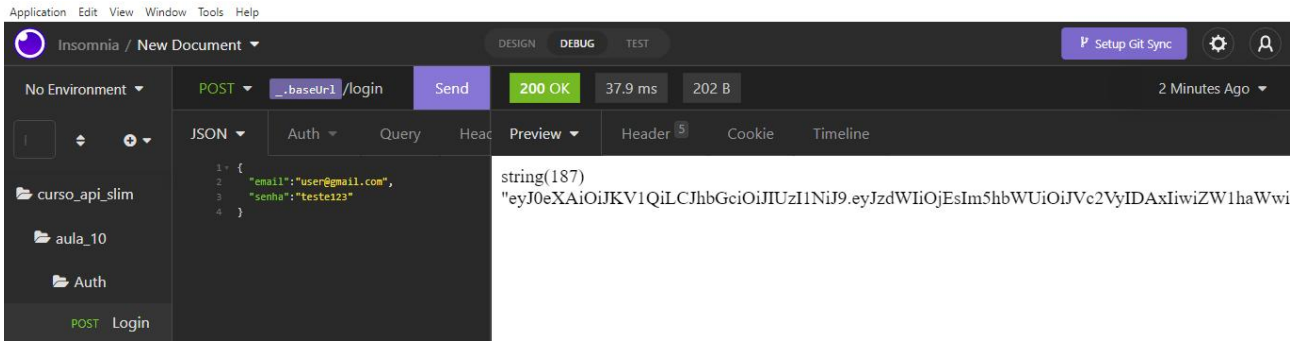
App\Controllers\AuthController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $expiredAt = (new \DateTime())->modify('+ 2 days')->format('Y-m-d H:i:s');

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'exp' => $expiredAt
        ];

        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));

        var_dump($token);

        return $response;
    }
}
```

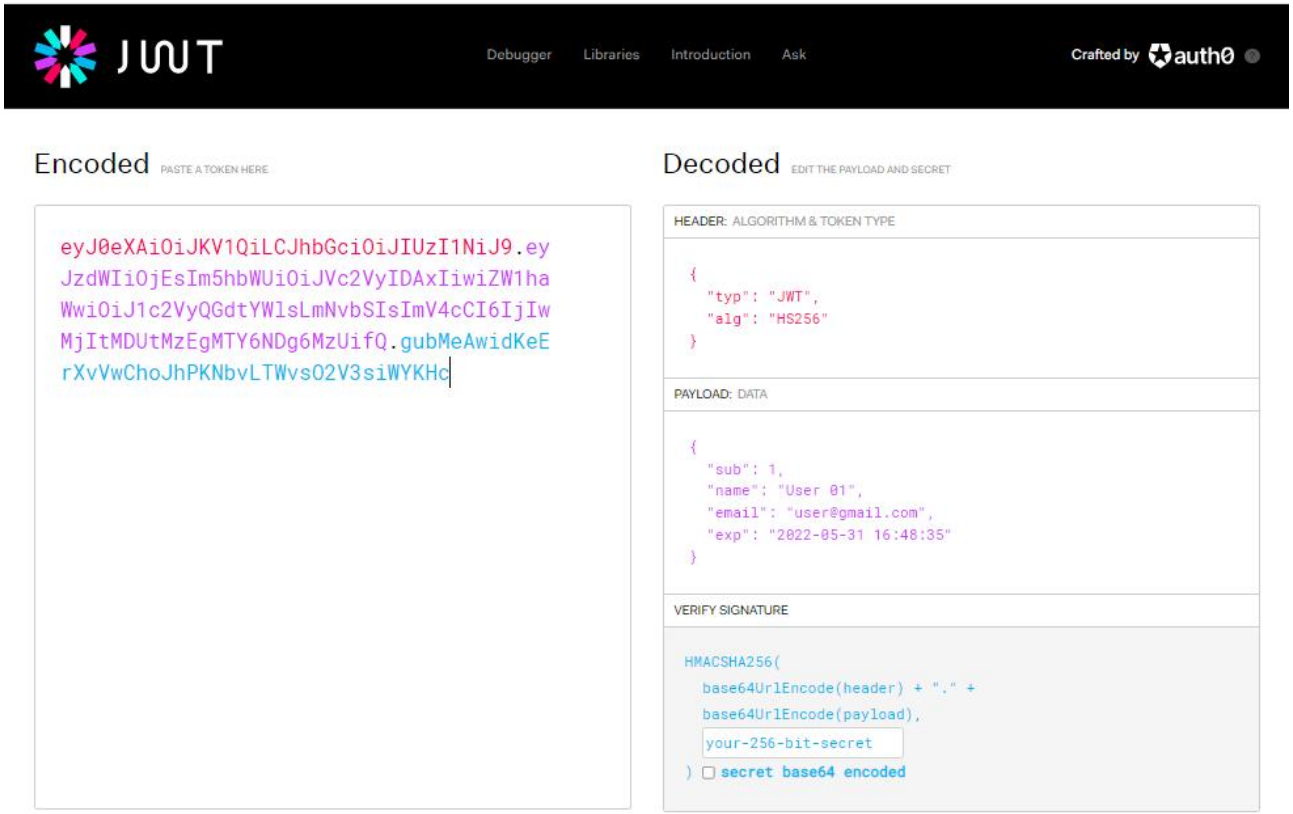
- No Insomnia:

Auth/Login

POST
{{ _.baseUrl }}/login



https://jwt.io/



- Inserindo o valor do token no campo Encoded, é exibido o tipo de algoritmo (criptografia) utilizado e o payload (com nome e email do usuário e a data de expiração do token).

- Dentro do payload nunca insira a senha.

App\Controllers\AuthController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];
        $expireDate = $data['expire_date'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'exp' => $expireDate
        ];

        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));

        var_dump($token);

        $refreshTokenPayload = [
            'email' => $usuario->getEmail(),
            'ramdom' => uniqid()
        ];
        $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));
        return $response;

    }
}
```

Encoded PASTE A TOKEN HERE

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjEsIm5hbWUiOiJVc2VyIDAxIiwiZW1haWwiOiJ1c2VyQGdtYWlsLmNvbSIsImV4cCI6IjIwMjItMDYtMDEgMDA6MDA6MDAifQ.RQ9eHQOQnaiX3yQrGHL_KaQVKaD7jfJRSiEeyc9dU8M

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

```
{
  "sub": 1,
  "name": "User 01",
  "email": "user@gmail.com",
  "exp": "2022-06-01 00:00:00"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

ETAPAS.md

* [*] Construir as tabela no banco de dados
* [ ] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [*] Construir o token
    * [ ] Salvar o token no banco de dados
    * [ ] Retornar o token para o usuário
* [ ] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

Salvar o token no banco de dados

App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel.php

```php
<?php

namespace App\Models\MySQL\CodeeasyGerenciadorDeLojas;

final class TokenModel
{
    /**
     * @var int
     */
    private $id;
    /**
     * @var string
     */
    private $token;
    /**
     * @var string
     */
    private $refresh_token;
    /**
     * @var string
     */
    private $expired_at;
    /**
     * @var int
     */
    private $usuarios_id;

    /**
     * @return int
     */
    public function getId(): int
    {
        return $this->id;
    }

    /**
     * @param int $id
     * @return self
     */
    public function setId(int $id): self
    {
        $this->id = $id;
        return $this;
    }

    /**
     * @return string
     */
```

```
public function getToken(): string
{
    return $this->token;
}

/**
 * @param string $token
 * @return self
 */
public function setToken(string $token): self
{
    $this->token = $token;
    return $this;
}

/**
 * @return string
 */
public function getRefresh_token(): string
{
    return $this->refresh_token;
}

/**
 * @param string $refresh_token
 * @return self
 */
public function setRefresh_token(string $refresh_token): self
{
    $this->refresh_token = $refresh_token;
    return $this;
}

/**
 * @return string
 */
public function getExpired_at(): string
{
    return $this->expired_at;
}

/**
 * @param string $expired_at
 * @return self
 */
public function setExpired_at(string $expired_at): self
{
    $this->expired_at = $expired_at;
    return $this;
}

/**
 * @return int
 */
```

```
    public function getUsuarios_id(): int
    {
        return $this->usuarios_id;
    }

    /**
     * @param int $usuarios_id
     * @return self
     */
    public function setUsuarios_id(int $usuarios_id): self
    {
        $this->usuarios_id = $usuarios_id;
        return $this;
    }
}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO.php

```php
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

class TokensDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function createToken(TokenModel $token): void
    {
        $statement = $this->pdo
            ->prepare('INSERT INTO tokens
                (
                    token,
                    refresh_token,
                    expired_at,
                    usuarios_id
                )
                VALUES
                (
                    :token,
                    :refresh_token,
                    :expired_at,
                    :usuarios_id
                );
            ');
        $statement->execute([
            'token' => $token->getToken(),
            'refresh_token' => $token->getRefresh_token(),
            'expired_at' => $token->getExpired_at(),
            'usuarios_id' => $token->getUsuarios_id()
        ]);
    }
}
```

App\Controllers\AuthController.php

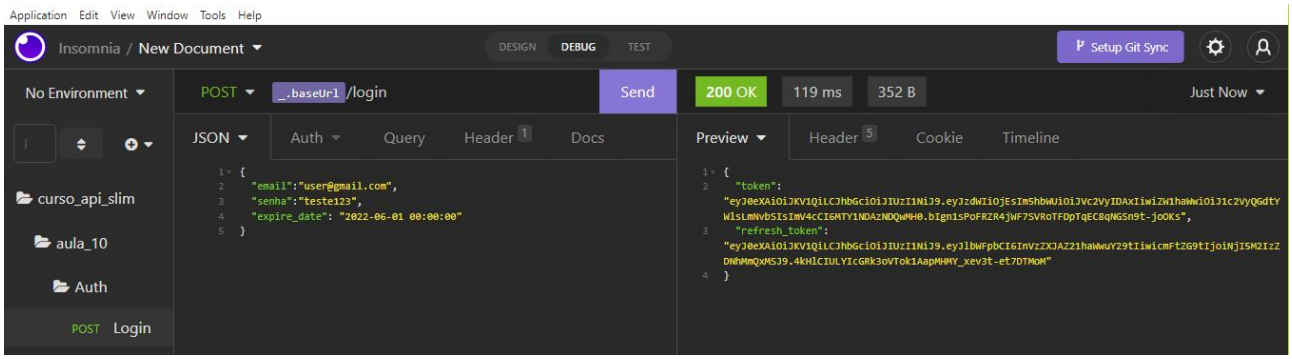
```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];
        $expireDate = $data['expire_date'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'exp' => (new \DateTime($expireDate))->getTimestamp()
        ];

        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
        $refreshTokenPayload = [
            'email' => $usuario->getEmail(),
            'ramdom' => uniqid()
        ];
        $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

        $tokenModel = new TokenModel();
        $tokenModel->setExpired_at($expireDate)
            ->setRefresh_token($refreshToken)
            ->setToken($token)
            ->setUsuarios_id($usuario->getId());
```

```
        $tokensDAO = new TokensDAO();
        $tokensDAO->createToken($tokenModel);

        $response = $response->withJson([
            "token" => $token,
            "refresh_token" => $refreshToken
        ]);

        return $response;
    }

}
```

- No Insomnia:

Auth/Login

POST
{{ _.baseUrl }}/login



token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjEsIm5hbWUiOiJVc2VyIDAxIiwiZW1haWwiOiJ1c2VyQGdtY
WlsLmNvbSIsImV4cCI6MTY1NDAzNDQwMH0.bIgn1sPoFRZR4jWF7SVRoTFDpTqEC8qNGSn9t-joOKs

refresh_token

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6InVzZXJAZ21haWwuY29tIiwicmFtZG9tIjoiNjI5M2IzZDN
hMmQxMSJ9.4kHlCIULYIcGRk3oVTok1AapMHMY_xev3t-et7DTMoM

Manage Environments

Base Environment

Sub Environments

Base Environment


```
{
  "servidor": "http://localhost:8089/api",
  "server": "165.227.93.41/cgitar",
  "baseUrl": "http://localhost/curso_api_slim/index.php",
  "token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjEsImShbWUiOiJVc2VyIDAxIiwiZW1haWwiOiJ1c2VyQGdtYWlsLmNvbSIsImV4cGlyZWRfYXQiOiIyMDIyLTA2LTAxIDAwOjAwOjAwIn0.7VhSI-rfcF5nCyHpFB747I0OdfACRDRWg-VDJ-n7-20",
  "refresh_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6InVzZXJAZ21haWwuY29tIiwicmFtZG9tIjoiNjI5M2IzZDNhMmQxMSJ9.4kHlCIULYIcGRk3oVTok1AapMHMY_xev3t-et7DTMoM"
}
```


* Environment data can be used for Nunjucks Templating in your requests

Done

ETAPAS.md

* [*] Construir as tabela no banco de dados
* [*] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [*] Construir o token
    * [*] Salvar o token no banco de dados
    * [*] Retornar o token para o usuário
* [ ] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

Observação:

SUB é um padrão do JWT para representar o id.

Validação das rotas através do token

Observação:

- Quando você envia o token, usando JWT, iremos verificar a validade do token sem precisar usar o banco de dados.

routes\index.php

```
<?php

use function src\{
    slimConfiguration,
    basicAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController
};

use Tuupola\Middleware\JwtAuthentication;

$app = new \Slim\App(slimConfiguration());

// ========================================

$app->post('/login', AuthController::class . ':login');

$app->get('/teste', function(){echo "oi";})
    ->add(function($request, $response, $next){
        $token = $request->getAttribute('jwt');
        echo "<pre>";
        var_dump($token);
        echo "</pre>";
        $response = $next($request, $response);
        return $response;
    })
    ->add(new Tuupola\Middleware\JwtAuthentication([
        "secret" => getenv('JWT_SECRET_KEY'),
        "attribute" => "jwt"
    ]));

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
```

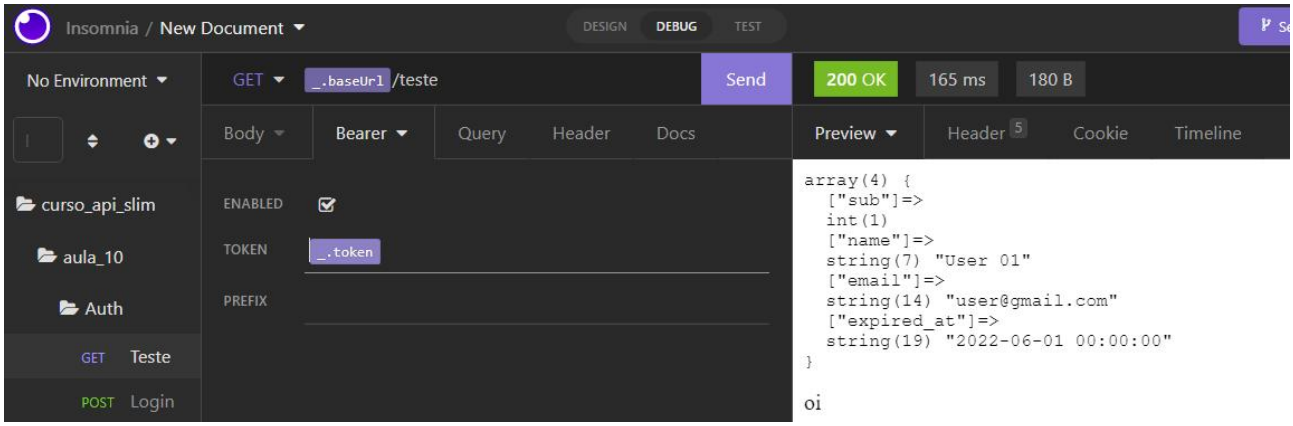
```
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ========================================

$app->run();
```

- O primeiro middleware verifica se a chave de assinatura está correta.
- Se estiver, o segundo middleware verifica a data de expiração do token. Se não estiver vencida executa o código.

- No Insomnia:

Teste

GET
{{ _.baseUrl }}/teste



- Exibe o conteúdo do token na tela.

App\Controllers\AuthController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];
        $expireDate = $data['expire_date'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'expired_at' => $expireDate
        ];

        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
        $refreshTokenPayload = [
            'email' => $usuario->getEmail(),
            'ramdom' => uniqid()
        ];
        $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

        $tokenModel = new TokenModel();
        $tokenModel->setExpired_at($expireDate)
            ->setRefresh_token($refreshToken)
            ->setToken($token)
            ->setUsuarios_id($usuario->getId());
```

```
    $tokensDAO = new TokensDAO();
    $tokensDAO->createToken($tokenModel);

    $response = $response->withJson([
        "token" => $token,
        "refresh_token" => $refreshToken
    ]);

    return $response;
}

public function refreshToken(Request $request, Response $response, array $args): Response
{
    $data = $request->getParsedBody();
    $refreshToken = $data['refresh_token'];
    $expireDate = $data['expire_date'];

    $refreshTokenDecoded = JWT::decode(
        $refreshToken,
        getenv('JWT_SECRET_KEY'),
        ['HS256']
    );

    $tokensDAO = new TokensDAO();
    $refreshTokenExists = $tokensDAO->verifyRefreshToken($refreshToken);
    if (!$refreshTokenExists) {
        return $response->withStatus(401);
    }
    $usuariosDAO = new UsuariosDAO();
    $usuario = $usuariosDAO->getUserByEmail($refreshTokenDecoded->email);
    if (is_null($usuario)) {
        return $response->withStatus(401);
    }

    $tokenPayload = [
        'sub' => $usuario->getId(),
        'name' => $usuario->getNome(),
        'email' => $usuario->getEmail(),
        'expired_at' => $expireDate
    ];

    $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
    $refreshTokenPayload = [
        'email' => $usuario->getEmail(),
        'ramdom' => uniqid()
    ];
    $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

    $tokenModel = new TokenModel();
    $tokenModel->setExpired_at($expireDate)
        ->setRefresh_token($refreshToken)
        ->setToken($token)
        ->setUsuarios_id($usuario->getId());
```

```
        $tokensDAO = new TokensDAO();
        $tokensDAO->createToken($tokenModel);

        $response = $response->withJson([
            "token" => $token,
            "refresh_token" => $refreshToken
        ]);

        return $response;
    }

}
```

Organizando melhor

src\jwtAuth.php

```
<?php

namespace src;

use Tuupola\Middleware\JwtAuthentication;

function jwtAuth(): JwtAuthentication
{
    return new JwtAuthentication([
        'secret' => getenv('JWT_SECRET_KEY'),
        'attribute' => 'jwt'
    ]);
}
```

index.php

```
<?php

require_once './vendor/autoload.php';
require_once './env.php';
require_once './src/slimConfiguration.php';
require_once './src/basicAuth.php';
require_once './src/jwtAuth.php';
require_once './routes/index.php';
```

App\Middlewares\JwtDateTimeMiddleware.php

```
<?php

namespace App\Middlewares;

use Psr\Http\Message\{
  ServerRequestInterface as Request,
  ResponseInterface as Response
};

final class JwtDateTimeMiddleware
{

  public function __invoke(Request $request, Response $response, callable $next): Response
  {

        $token = $request
            ->getAttribute('jwt');
        $expireDate = new \DateTime($token['expired_at']);
        $now = new \DateTime();
```

```
        if($expireDate < $now)
            return $response
                ->withStatus(401);
        $response = $next($request, $response);
        return $response;
    }
}
```

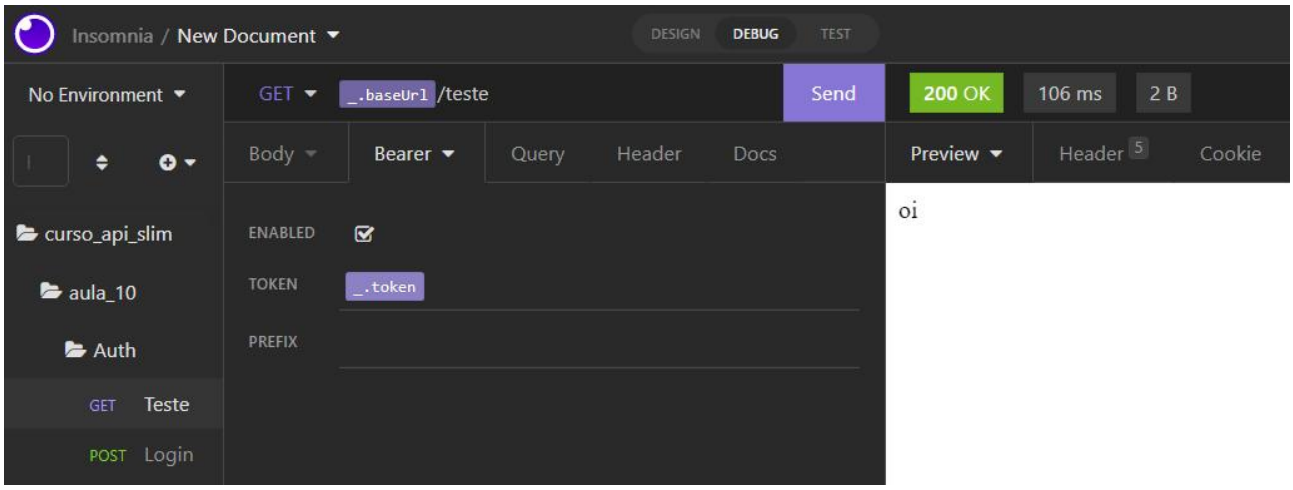
routes\index.php

```
<?php

use function src\{
    slimConfiguration,
    basicAuth,
    jwtAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController
};

use Tuupola\Middleware\JwtAuthentication;
use App\Middlewares\JwtDateTimeMiddleware;

$app = new \Slim\App(slimConfiguration());

// ==========================================

$app->post('/login', AuthController::class . ':login');

$app->get('/teste', function(){echo "oi";})
    ->add((new JwtDateTimeMiddleware()))
    ->add(jwtAuth());

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ==========================================

$app->run();
```

- No Insomnia:

Teste

GET
{{ _.baseUrl }}/teste



ETAPAS.md

* [*] Construir as tabela no banco de dados
* [*] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [*] Construir o token
    * [*] Salvar o token no banco de dados
    * [*] Retornar o token para o usuário
* [*] Validação das rotas através do token
* [ ] Construir a rota do refresh_token

Construir a rota do refresh_token

routes\index.php

```php
<?php

use function src\{
    slimConfiguration,
    basicAuth,
    jwtAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController
};

use Tuupola\Middleware\JwtAuthentication;
use App\Middlewares\JwtDateTimeMiddleware;

$app = new \Slim\App(slimConfiguration());

// ==========================================

$app->post('/login', AuthController::class . ':login');
$app->get('/refresh-token', AuthController::class . ':refreshToken')
    ->add(jwtAuth());

$app->get('/teste', function(){echo "oi";})
    ->add((new JwtDateTimeMiddleware()))
    ->add(jwtAuth());

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ==========================================

$app->run();
```

App\Controllers\AuthController.php

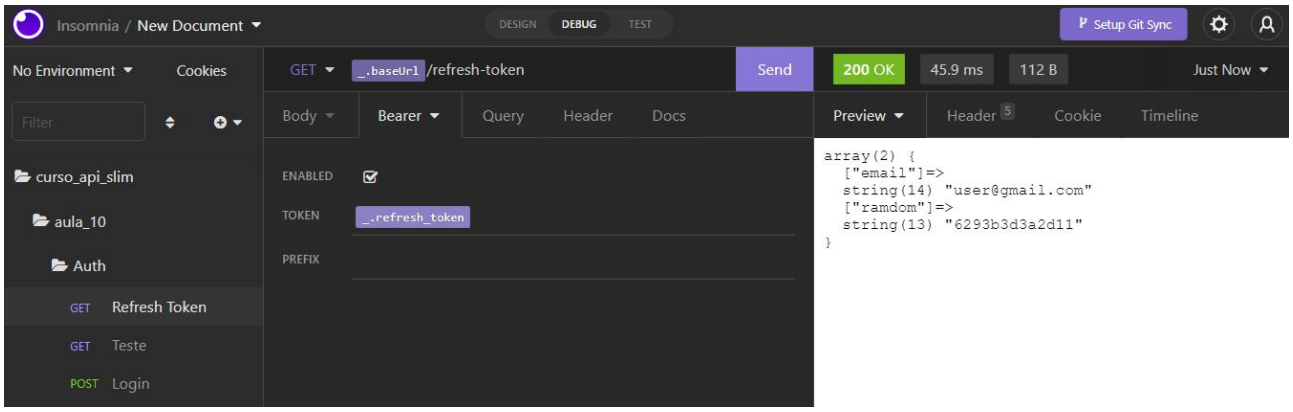
```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];
        $expireDate = $data['expire_date'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'expired_at' => $expireDate
        ];

        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
        $refreshTokenPayload = [
            'email' => $usuario->getEmail(),
            'ramdom' => uniqid()
        ];
        $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

        $tokenModel = new TokenModel();
        $tokenModel->setExpired_at($expireDate)
            ->setRefresh_token($refreshToken)
            ->setToken($token)
            ->setUsuarios_id($usuario->getId());
```

```
        $tokensDAO = new TokensDAO();
        $tokensDAO->createToken($tokenModel);

        $response = $response->withJson([
            "token" => $token,
            "refresh_token" => $refreshToken
        ]);

        return $response;
    }

    public function refreshToken(Request $request, Response $response, array $args): Response
    {

        $token = $request->getAttribute('jwt');
        echo "<pre>";
        var_dump($token);
        echo "</pre>";

        return $response;
    }

}
```

- No Insomnia:

Refresh Token

GET
{{ _.baseUrl }}/refresh-token



ETAPAS.md

* [*] Construir as tabela no banco de dados
* [*] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [*] Construir o token
    * [*] Salvar o token no banco de dados
    * [*] Retornar o token para o usuário
* [*] Validação das rotas através do token
* [ ] Construir a rota do refresh_token
    * [ ] Ver se o token existe na tabela de tokens
    * [ ] Capturar o usuário da tabela de usuários
    * [ ] Criar um novo token
    * [ ] Salvar o token no banco de dados
    * [ ] Retornar o token para o usuário

Ver se o token existe na tabela de tokens

routes\index.php

```
<?php

use function src\{
    slimConfiguration,
    basicAuth,
    jwtAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController
};

use Tuupola\Middleware\JwtAuthentication;
use App\Middlewares\JwtDateTimeMiddleware;

$app = new \Slim\App(slimConfiguration());

// ========================================

$app->post('/login', AuthController::class . ':login');
$app->post('/refresh-token', AuthController::class . ':refreshToken');

$app->get('/teste', function(){echo "oi";})
    ->add((new JwtDateTimeMiddleware()))
    ->add(jwtAuth());

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
```

```
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ========================================

$app->run();
```

App\Controllers\AuthController.php

```
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\UsuariosDAO;
use Firebase\JWT\JWT;
use App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO;
use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

final class AuthController
{
    public function login(Request $request, Response $response, array $args): Response
    {
        $data = $request->getParsedBody();

        $email = $data['email'];
        $senha = $data['senha'];
        $expireDate = $data['expire_date'];

        $usuariosDAO = new UsuariosDAO();
        $usuario = $usuariosDAO->getUserByEmail($email);

        if (is_null($usuario)) {
            return $response->withStatus(401);
        }

        if (!password_verify($senha, $usuario->getSenha())) {
            return $response->withStatus(401);
        }

        $tokenPayload = [
            'sub' => $usuario->getId(),
            'name' => $usuario->getNome(),
            'email' => $usuario->getEmail(),
            'expired_at' => $expireDate
        ];
```

```
    $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
    $refreshTokenPayload = [
        'email' => $usuario->getEmail(),
        'ramdom' => uniqid()
    ];
    $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

    $tokenModel = new TokenModel();
    $tokenModel->setExpired_at($expireDate)
        ->setRefresh_token($refreshToken)
        ->setToken($token)
        ->setUsuarios_id($usuario->getId());

    $tokensDAO = new TokensDAO();
    $tokensDAO->createToken($tokenModel);

    $response = $response->withJson([
        "token" => $token,
        "refresh_token" => $refreshToken
    ]);

    return $response;
}

public function refreshToken(Request $request, Response $response, array $args): Response
{
    $data = $request->getParsedBody();
    $refreshToken = $data['refresh_token'];
    $expireDate = $data['expire_date'];

    $refreshTokenDecoded = JWT::decode(
        $refreshToken,
        getenv('JWT_SECRET_KEY'),
        ['HS256']
    );

    $tokensDAO = new TokensDAO();
    $refreshTokenExists = $tokensDAO->verifyRefreshToken($refreshToken);
    if (!$refreshTokenExists) {
        return $response->withStatus(401);
    }
    $usuariosDAO = new UsuariosDAO();
    $usuario = $usuariosDAO->getUserByEmail($refreshTokenDecoded->email);
    if (is_null($usuario)) {
        return $response->withStatus(401);
    }

    $tokenPayload = [
        'sub' => $usuario->getId(),
        'name' => $usuario->getNome(),
        'email' => $usuario->getEmail(),
        'expired_at' => $expireDate
    ];
```

```
        $token = JWT::encode($tokenPayload, getenv('JWT_SECRET_KEY'));
        $refreshTokenPayload = [
            'email' => $usuario->getEmail(),
            'ramdom' => uniqid()
        ];
        $refreshToken = JWT::encode($refreshTokenPayload, getenv('JWT_SECRET_KEY'));

        $tokenModel = new TokenModel();
        $tokenModel->setExpired_at($expireDate)
            ->setRefresh_token($refreshToken)
            ->setToken($token)
            ->setUsuarios_id($usuario->getId());

        $tokensDAO = new TokensDAO();
        $tokensDAO->createToken($tokenModel);

        $response = $response->withJson([
            "token" => $token,
            "refresh_token" => $refreshToken
        ]);

        return $response;
    }

}
```

App\DAO\MySQL\CodeeasyGerenciadorDeLojas\TokensDAO.php

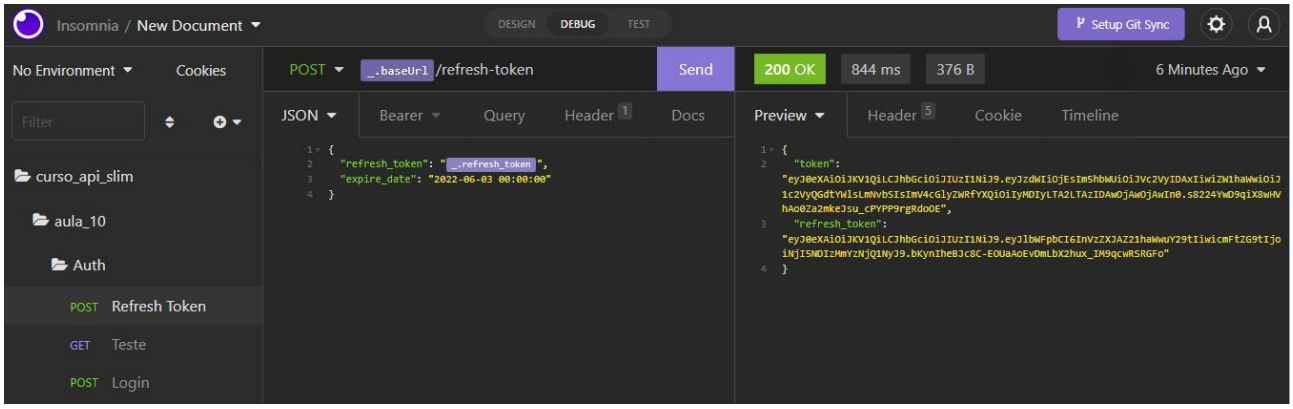
```
<?php

namespace App\DAO\MySQL\CodeeasyGerenciadorDeLojas;

use App\Models\MySQL\CodeeasyGerenciadorDeLojas\TokenModel;

class TokensDAO extends Conexao
{
    public function __construct()
    {
        parent::__construct();
    }

    public function createToken(TokenModel $token): void
    {
        $statement = $this->pdo
            ->prepare('INSERT INTO tokens
                (
                    token,
                    refresh_token,
                    expired_at,
                    usuarios_id
                )
                VALUES
```

```
            (
                :token,
                :refresh_token,
                :expired_at,
                :usuarios_id
            );
        ');
    $statement->execute([
        'token' => $token->getToken(),
        'refresh_token' => $token->getRefresh_token(),
        'expired_at' => $token->getExpired_at(),
        'usuarios_id' => $token->getUsuarios_id()
    ]);
}

public function verifyRefreshToken(string $refreshToken): bool
{
    $statement = $this->pdo
        ->prepare('SELECT
            id
        FROM tokens
        WHERE refresh_token = :refresh_token;
        ');
    $statement->bindParam('refresh_token', $refreshToken);
    $statement->execute();
    $tokens = $statement->fetchAll(\PDO::FETCH_ASSOC);
    return count($tokens) === 0 ? false : true;
}
}
```

- No Insomnia:

Refresh Token

GET
{{ _.baseUrl }}/refresh-token



A mostrar registos de 0 - 1 (2 total, A consulta demorou 0,0017 segundos.)

```
SELECT * FROM `tokens`
```

Perfil [Editar em linha] [Editar] [Explicar SQL] [Criar código PHP] [Actualizar]

Mostrar tudo | Número de registos: 25 Filtrar registos: Pesquisar esta tabela Ordenar pela chave: Nenhum

+ Opções

←T→				id	usuarios_id	token	refresh_token	expired_at	active
Editar	Copiar	Apagar		1	1	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjEsI...	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6I...	2022-06-01 00:00:00	1
Editar	Copiar	Apagar		2	1	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOjEsI...	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6I...	2022-06-03 00:00:00	1

ETAPAS.md

* [*] Construir as tabela no banco de dados
* [*] Construir a rota de login
    * [*] Verificar se o email existe
    * [*] Verificar se a senha está correta
    * [*] Construir o token
    * [*] Salvar o token no banco de dados
    * [*] Retornar o token para o usuário
* [*] Validação das rotas através do token
* [*] Construir a rota do refresh_token
    * [*] Ver se o token existe na tabela de tokens
    * [*] Capturar o usuário da tabela de usuários
    * [*] Criar um novo token
    * [*] Salvar o token no banco de dados
    * [*] Retornar o token para o usuário

Aula 11 - Tratamento de Exceções

routes\index.php

```
<?php

use function src\{
    slimConfiguration,
    basicAuth,
    jwtAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController,
    ExceptionController
};

use Tuupola\Middleware\JwtAuthentication;
use App\Middlewares\JwtDateTimeMiddleware;

$app = new \Slim\App(slimConfiguration());

// ==========================================

$app->get('/exception-test', ExceptionController::class . ':test');

$app->post('/login', AuthController::class . ':login');
$app->post('/refresh-token', AuthController::class . ':refreshToken');

$app->get('/teste', function(){echo "oi";})
    ->add((new JwtDateTimeMiddleware()))
    ->add(jwtAuth());

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ==========================================

$app->run();
```

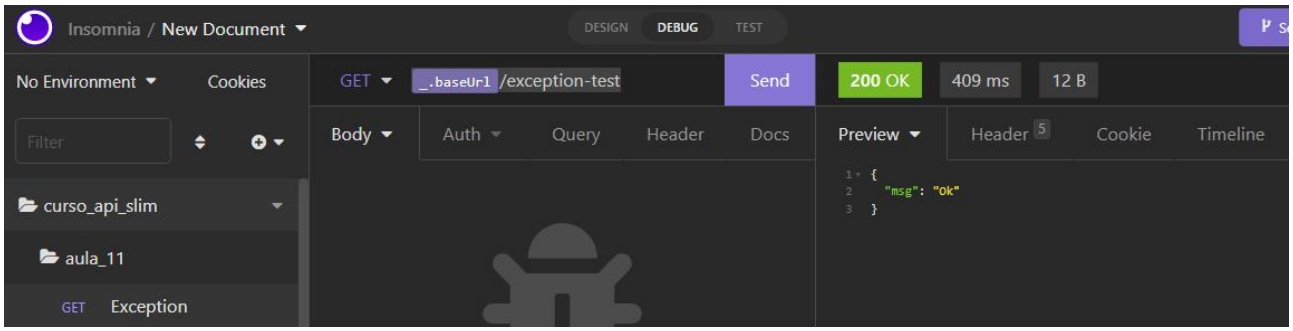
App\Controllers\ExceptionController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\Exceptions\TestException;

final class ExceptionController
{
    public function test(Request $request, Response $response, array $args): Response
    {
        return $response->withJson(['msg' => 'Ok']);
    }
}
```

- No Insomnia:

Exception

GET
{{ _.baseUrl }}/exception-test



Lançando uma exceção

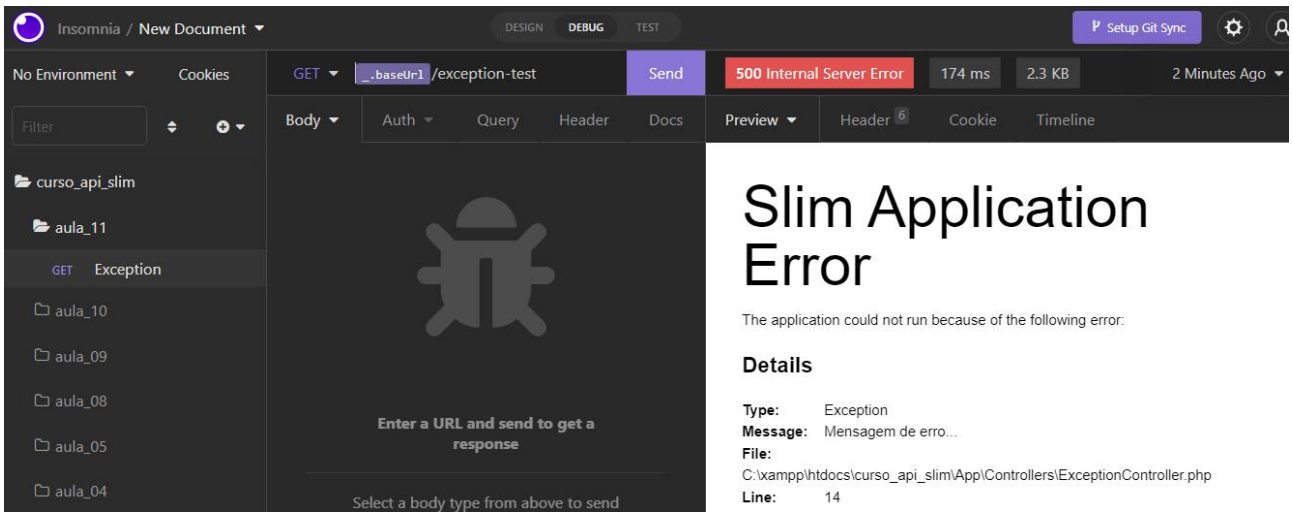
App\Controllers\ExceptionController.php

```
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\Exceptions\TestException;

final class ExceptionController
{
    public function test(Request $request, Response $response, array $args): Response
    {

        throw new \Exception("Mensagem de erro.");
        return $response->withJson(['msg' => 'Ok']);

    }
}
```

- No Insomnia:

Exception

GET
{{ _.baseUrl }}/exception-test



Tratando exceções

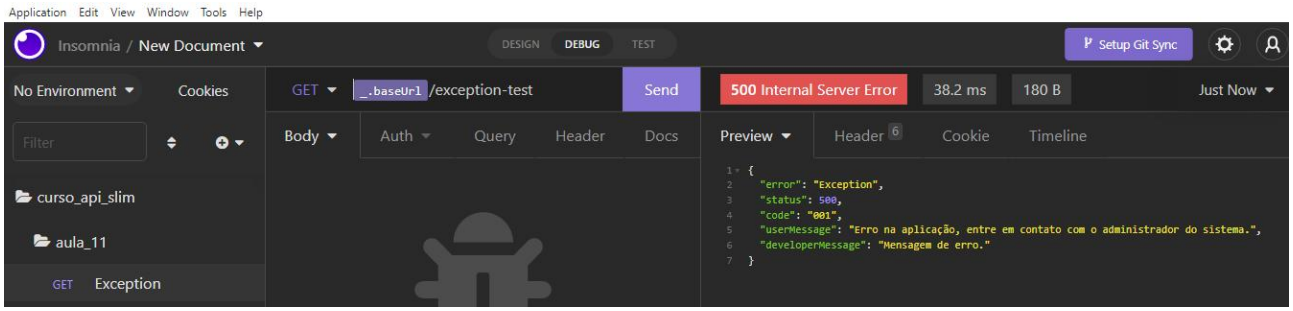
App\Controllers\ExceptionController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\Exceptions\TestException;

final class ExceptionController
{
    public function test(Request $request, Response $response, array $args): Response
    {
        try {
            throw new \Exception("Mensagem de erro.");
            return $response->withJson(['msg' => 'ok']);
        } catch(\Exception | \Throwable $ex) {
            return $response->withJson([
                'error' => \Exception::class,
                'status' => 500,
                'code' => '001',
                'userMessage' => "Erro na aplicação, entre em contato com o administrador do sistema.",
                'developerMessage' => $ex->getMessage()
            ], 500);
        }
    }
}
```

- No Insomnia:

Exception

GET
{{ _.baseUrl }}/exception-test



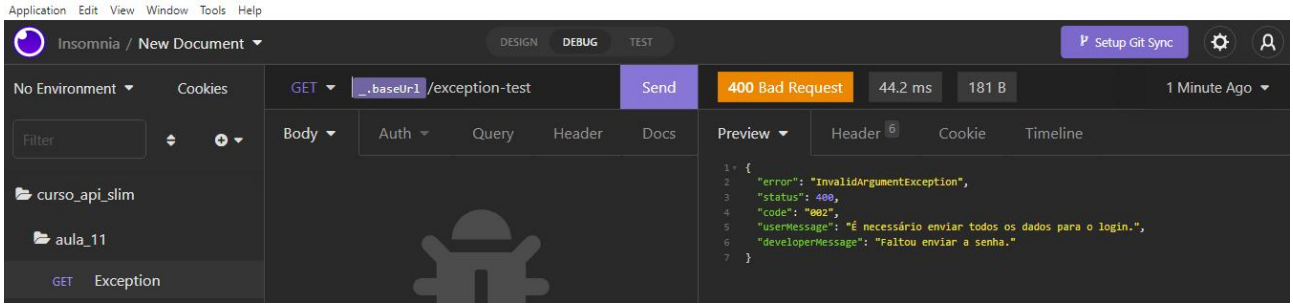
App\Controllers\ExceptionController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\Exceptions\TestException;

final class ExceptionController
{
    public function test(Request $request, Response $response, array $args): Response
    {

        try {
            // Código...
            throw new \InvalidArgumentException("Faltou enviar a senha.");
            return $response->withJson(['msg' => 'ok']);
        } catch(\InvalidArgumentException $ex) {
            return $response->withJson([
                'error' => \InvalidArgumentException::class,
                'status' => 400,
                'code' => '002',
                'userMessage' => "É necessário enviar todos os dados para o login.",
                'developerMessage' => $ex->getMessage()
            ], 400);
        } catch(\Exception | \Throwable $ex) {
            return $response->withJson([
                'error' => \Exception::class,
                'status' => 500,
                'code' => '001',
                'userMessage' => "Erro na aplicação, entre em contato com o administrador do sistema.",
                'developerMessage' => $ex->getMessage()
            ], 500);
        }
    }
}
```

- No Insomnia:

Exception

GET
{{ _.baseUrl }}/exception-test



Criando as nossas próprias exceptions

App\Exceptions\TestException.php

```php
<?php

namespace App\Exceptions;

class TestException extends \Exception
{
    public function __construct($message, $code = 0, Exception $previous = null)
    {
        parent::__construct($message, $code, $previous);
    }

    public function __toString(): string
    {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }
}
```

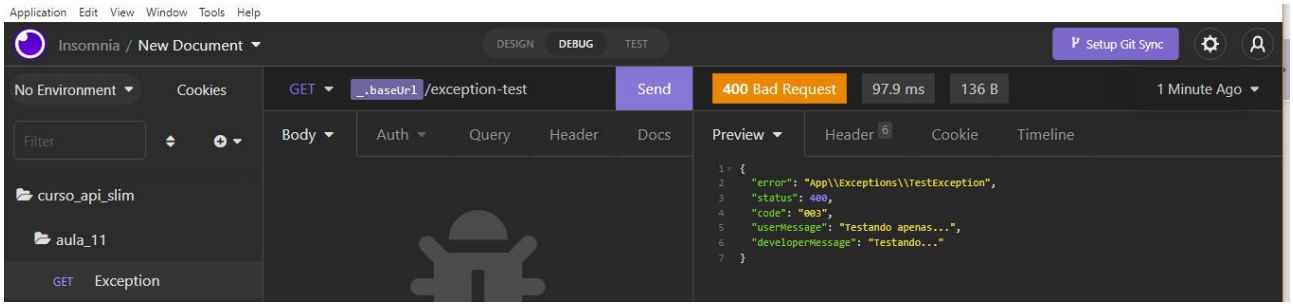
App\Controllers\ExceptionController.php

```php
<?php

namespace App\Controllers;

use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as Response;
use App\Exceptions\TestException;

final class ExceptionController
{
    public function test(Request $request, Response $response, array $args): Response
    {
        try {
            // Código...
            throw new TestException("Testando...");
            return $response->withJson(['msg' => 'ok']);
        } catch(TestException $ex) {
            return $response->withJson([
                'error' => TestException::class,
                'status' => 400,
                'code' => '003',
                'userMessage' => "Testando apenas...",
                'developerMessage' => $ex->getMessage()
            ], 400);
        } catch(\InvalidArgumentException $ex) {
            return $response->withJson([
                'error' => \InvalidArgumentException::class,
                'status' => 400,
                'code' => '002',
                'userMessage' => "É necessário enviar todos os dados para o login.",
                'developerMessage' => $ex->getMessage()
            ], 400);
        } catch(\Exception | \Throwable $ex) {
            return $response->withJson([
                'error' => \Exception::class,
                'status' => 500,
                'code' => '001',
                'userMessage' => "Erro na aplicação, entre em contato com o administrador do sistema.",
                'developerMessage' => $ex->getMessage()
            ], 500);
        }
    }
}
```

- No Insomnia:

Exception

GET
{{ _.baseUrl }}/exception-test



Aula 12 - Versionamento da API

Versionando uma API

routes\index.php

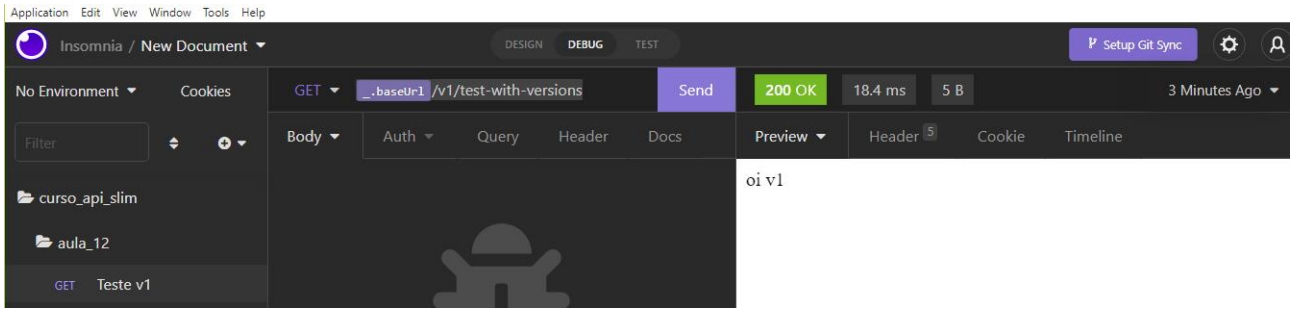
```
<?php

use function src\{
    slimConfiguration,
    basicAuth,
    jwtAuth
};

use App\Controllers\{
    ProdutoController,
    LojaController,
    AuthController,
    ExceptionController
};

use Tuupola\Middleware\JwtAuthentication;
use App\Middlewares\JwtDateTimeMiddleware;

$app = new \Slim\App(slimConfiguration());

// ========================================

$app->group('/v1', function () use ($app) {
    $app->get('/test-with-versions', function () {
        return "oi v1";
    });
});

$app->group('/v2', function () use ($app) {
    $app->get('/test-with-versions', function () {
        return "oi v2";
    });
});

$app->get('/exception-test', ExceptionController::class . ':test');

$app->post('/login', AuthController::class . ':login');
$app->post('/refresh-token', AuthController::class . ':refreshToken');

$app->get('/teste', function(){echo "oi";})
    ->add((new JwtDateTimeMiddleware()))
    ->add(jwtAuth());

$app->group('', function () use ($app) {
    $app->get('/loja', LojaController::class . ':getLojas');
```

```
    $app->post('/loja', LojaController::class . ':insertLoja');
    $app->put('/loja', LojaController::class . ':updateLoja');
    $app->delete('/loja', LojaController::class . ':deleteLoja');

    $app->get('/produto', ProdutoController::class . ':getProdutos');
    $app->post('/produto', ProdutoController::class . ':insertProduto');
    $app->put('/produto', ProdutoController::class . ':updateProduto');
    $app->delete('/produto', ProdutoController::class . ':deleteProduto');
})->add(basicAuth());

// ========================================

$app->run();
```

- No Insomnia:

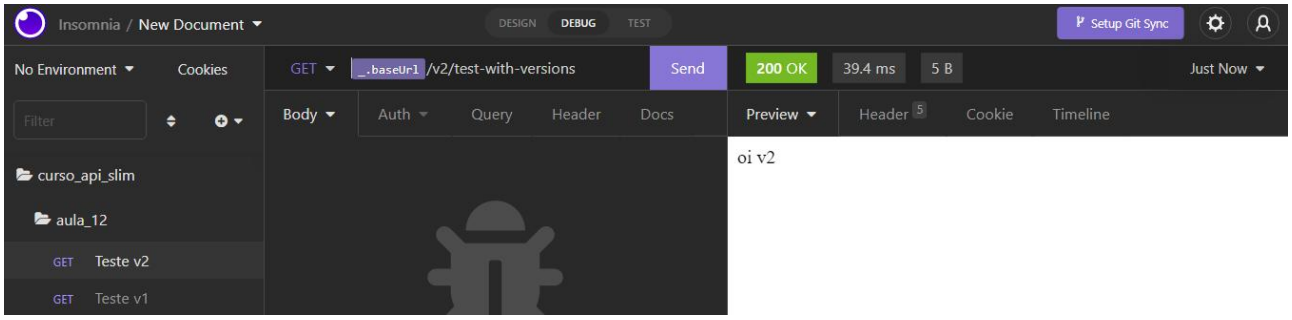
Teste v1

GET
{{ _.baseUrl }}/v1/test-with-versions



Teste v2

GET
{{ _.baseUrl }}/v2/test-with-versions



Aula 13 - Conclusão

Como continuar?

- Documentação - RAML, Swagger
- Outras arquiteturas de software
- OAuth 2.0
- Vá além...

Documentação de API's

Um usuário envia dados para um endpoint e a API retorna os dados para o usuário.

O usuário não precisa saber como a API funciona.

Montar uma documentação informando o que o usuário pode enviar, e tudo que pode ser retornado, inclusive os erros é extremamente importante.

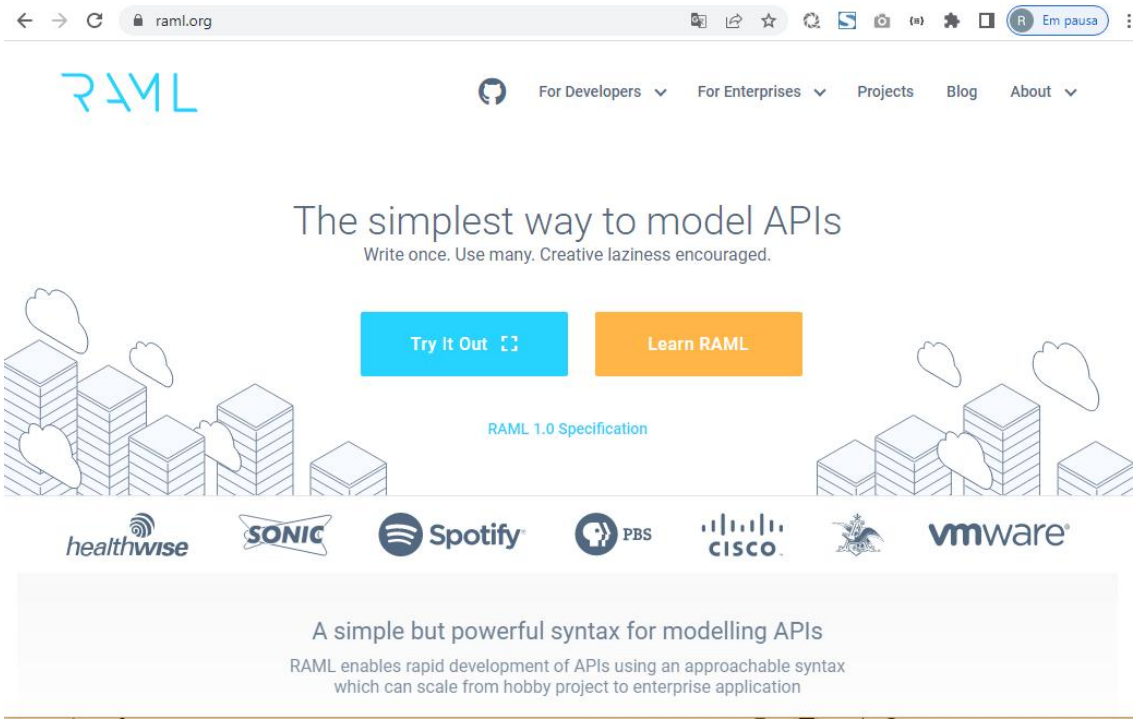
Algumas ferramentas para gerar a documentação de uma API são:

1. RAML (RESTFull API Modeling Language)
2. Swagger

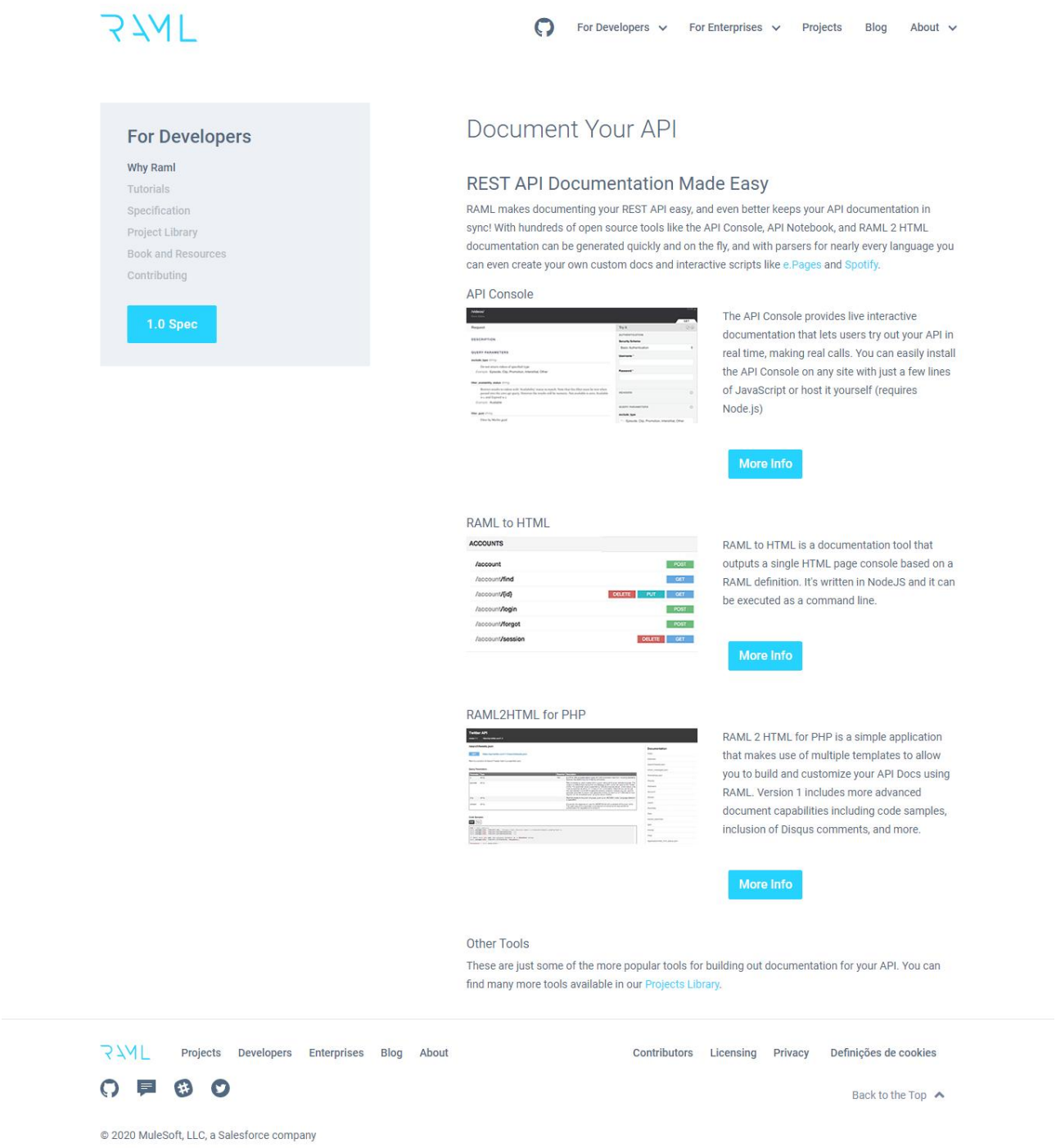
RAML

É uma linguagem para modelar API's.

https://raml.org/



https://raml.org/developers/document-your-api



O API Console cria um site onde é possível ver e testar a documentação da API.

O RAML to HTML converte o código para HTML permitindo que você tenha uma documentação completa.

https://github.com/raml2html/raml2html

RAML version support

raml2html 4 and higher only support RAML 1.0 files. Please stick with raml2html 3.x for RAML 0.8 support.

Install

```
npm i -g raml2html
```

Themes

raml2html ships with a default theme, but you can install more from NPM. For example, to render RAML to Markdown, you can install the raml2html-markdown-theme theme:

```
npm i -g raml2html-markdown-theme
```

Search NPM for the "raml2html-theme" keyword (or use this link) to find more themes.

Usage

As a command line script

```
raml2html --help
raml2html example.raml > example.html
raml2html --theme raml2html-markdown-theme example.raml > example.html
raml2html --template my-custom-template.nunjucks -i example.raml -o example.html
```

As a library

Using the default theme, different themes, or your own Nunjucks templates

```
const raml2html = require('raml2html');
const configWithDefaultTheme = raml2html.getConfigForTheme();
const configForDifferentTheme = raml2html.getConfigForTheme('raml2html-markdown-theme');
const configWithCustomTemplate = raml2html.getConfigForTemplate('~/path/to/my-custom-template.nunjucks');

// source can either be a filename, url, or parsed RAML object
raml2html.render(source, configWithDefaultTheme).then(function(result) {
  // Save the result to a file or do something else with the result
}, function(error) {
  // Output error
});
```

Using your own processing function, for full control over the whole rendering process

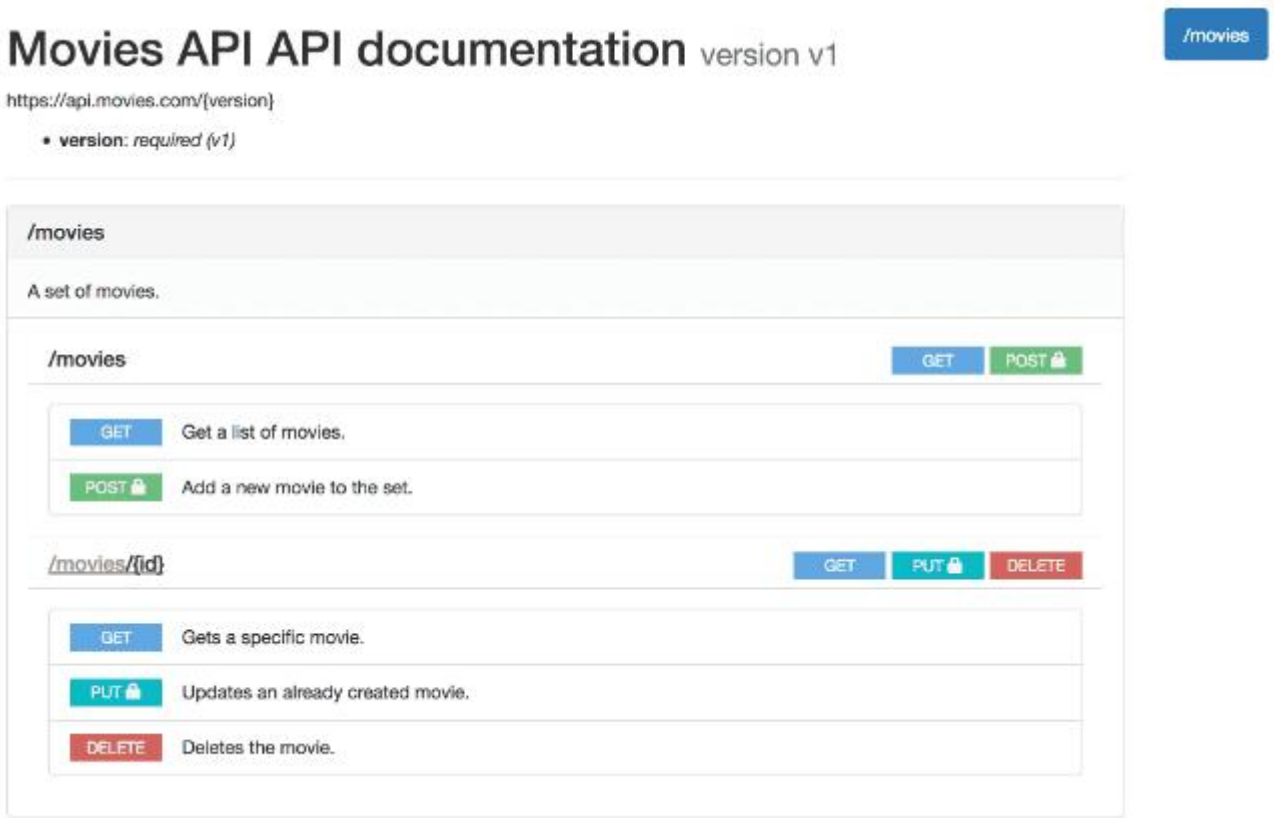
```
/**
 * config should be an object with at least an `processRamlObj` property which is a function that receives th
 * object and must return a promise with the result. You can do whatever you want in this function.
 *
 * You can also supply a postProcessHtml function that can for example minify the generated HTML.
 *
 * You can also supply a writeOutput function that takes over writing the output (to disk for example).
 *
 * You can also supply a setupNunjucks function that takes the env as its only parameter.
 */
raml2html.render(source, config).then(function(result) {
  // Save the result to a file or do something else with the result
}, function(error) {
  // Output error
});
```

See also examples/script.js for multiple examples of using raml2html as a library.

Example output

Please see the following links for live examples:

- https://rawgit.com/raml2html/default-theme/master/examples/helloworld.html
- https://rawgit.com/raml2html/default-theme/master/examples/world-music-api.html



Swagger

É uma ferramenta muito utilizada para trabalhar com API's. Algumas partes dessa ferramenta são pagas, outras são gratuitas.

Outras arquiteturas de software

Para microserviços não compensa montar uma arquitetura tão elaborada.

OAuth 2.0

É um outro sistema de autenticação que existe. Possui uma segurança maior, tem uma etapa a mais de verificação.