PRODERJ

Centro de Tecnologia da Informação e Comunicação do Estado do Rio de Janeiro

"Serva arcana tua."

Fragilidades nas aplicações Webm que devem ser tratadas antes de aplicar filtros ou middlewares pré-sistêmicos.

HTTP (Hypertext Transfer Protocol) e HTTPS (Hypertext Transfer Protocol Secure) são dois protocolos utilizados para a transferência de dados entre um navegador web e um servidor web. A principal diferença entre eles está na segurança dos dados transmitidos.

HTTP (Hypertext Transfer Protocol):
O HTTP é o protocolo padrão utilizado para transferir dados na web. No entanto, ele não fornece uma camada de segurança robusta para a transferência de informações sensíveis. Os dados transmitidos através do HTTP não são criptografados, o que significa que qualquer pessoa com acesso à rede entre o navegador e o servidor pode potencialmente interceptar e visualizar os dados.

HTTPS (Hypertext Transfer Protocol Secure):
O HTTPS é uma versão segura do HTTP. Ele utiliza uma camada adicional de segurança chamada SSL/TLS (Secure Sockets Layer/Transport Layer Security) para criptografar os dados transmitidos entre o navegador e o servidor. Isso significa que mesmo que alguém intercepte os dados, eles estarão criptografados e difíceis de serem decifrados sem a chave de criptografia correta. Além disso, os certificados SSL/TLS também autenticam a identidade do servidor, ajudando a garantir que você está realmente se comunicando com o site desejado e não com um impostor.

A instituição deve deliberar sobre ser ou não importante a diferença entre HTTP ou HTTPS e a segurança dos dados envolvidos. O HTTPS é amplamente recomendado para qualquer site que lida com informações sensíveis, como dados de login, informações pessoais e transações financeiras, porque oferece uma camada adicional de proteção contra interceptação e manipulação dos dados. A maioria dos navegadores modernos também exibe um ícone de cadeado ao lado da barra de endereço para indicar quando uma conexão HTTPS segura está sendo usada.

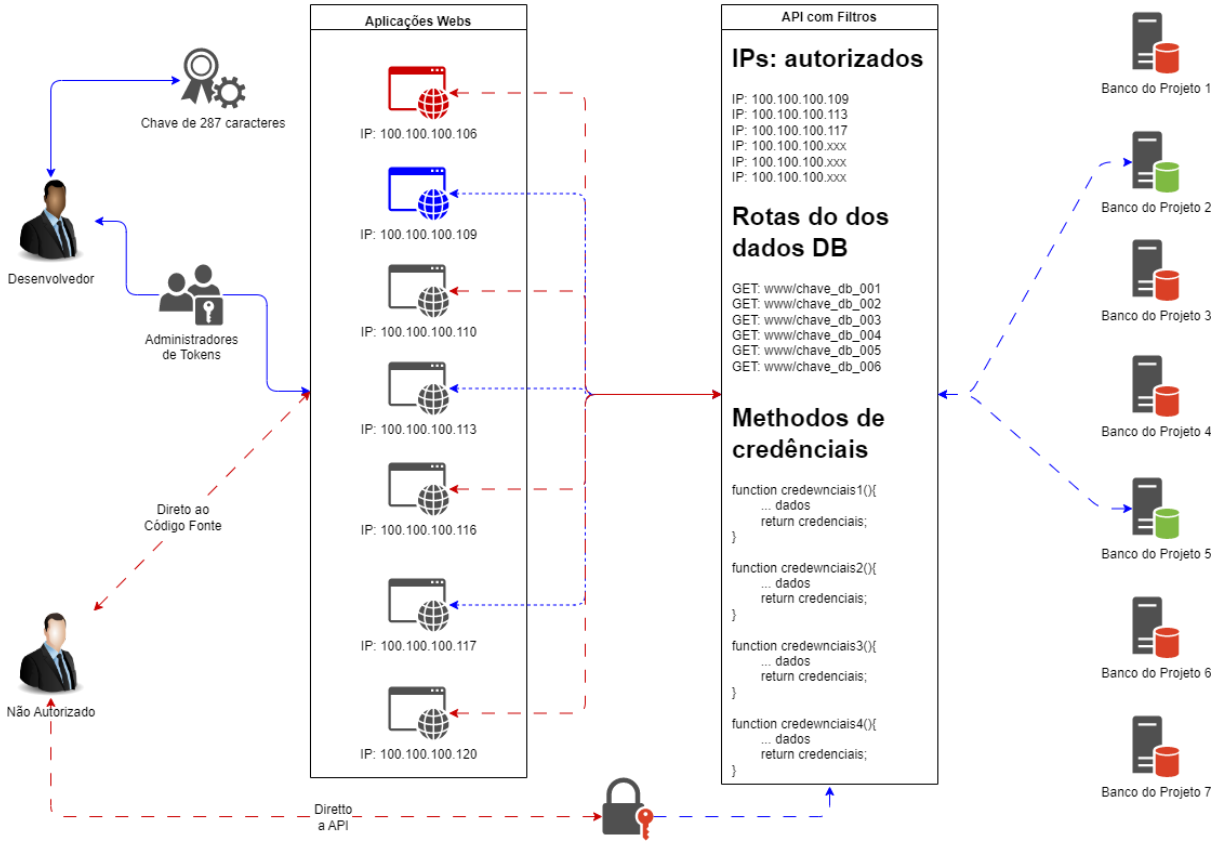
Versões de frameworks ou linguagem de programação possuem fragilidades conforme o "CVE Details", que é um banco de dados online que contém informações sobre vulnerabilidades de segurança conhecidas. O acrônimo "CVE" significa "Common Vulnerabilities and Exposures", e o site lista informações sobre várias vulnerabilidades de software, incluindo detalhes sobre as vulnerabilidades, suas classificações de gravidade, informações sobre patches e muito mais. Essas informações são úteis para profissionais de segurança cibernética, desenvolvedores de software e outros interessados em entender e mitigar riscos de segurança. Por favor, lembre-se de que a segurança cibernética é um assunto sensível e tratar essas informações com responsabilidade é importante. Todo código LEGADO que traz algum benefício para a instituição deve ter seu acesso o mais restrito possível respeitando as regras da segurança da Informação: Confidencialidade, Integridade, Disponibilidade e Autenticidade. Devendo estar em ambientes seguros sem acesso externo.

Link: https://www.cvedetails.com/vulnerability-list/

O presente estudo tem como maior propósito ocultar dados de conexão com os diversos bancos de dados utilizados pelo PRODERJ.

Conceito:

Fornecer APIs que serão implementadas pelos Desenvolvedores, utilizando uma rota estática, criptografada e um “JSON Web Token" para cada conexão com o banco de dados.



Antes de estabelecer uma camada de segurança, precisamos definir quais são os cenários de acesso ao código fonte e leitura não autorizada de credenciais registradas no mesmo código.

O Desenvolvedor que irá aplicar a segurança não deverá ter acesso aos seguintes parâmetros:

~~Host: mysql02-farm1.proderj.net via~~
~~Banco de dados: CorujaVermmelha (10.100.10.1000)~~
~~Usuário: HomemDeFerro~~
~~Senha: C@ch0eir4.2022~~

Para isso a API é convocada pelo recurso: Client URL (cURL) enviando um jwt de 286 caracteres e a rota criada pelo administrador de segurança.

No cenário correto, onde:
Responsáveis pela segurança geram as rotas de acesso (Endpoints das APIs);e
Entregam o JSON Web Token com 286 caracteres;
O Desenvolvedor configura a aplicação com o seguinte formato:

```php
$getAPI = myEndPoint($uriAPI, $tokenAPI);
if (
    isset($getAPI['result']['G4D9EEC8CB8BB4E2CB92336AA92FABD4B']) &&
    isset($getAPI['result']['MC8AB13AA245335BE277F3931D7BB8E7C']) &&
    isset($getAPI['result']['H7854A351CC9C8E6E1AE58CBB8382ADD2']) &&
    isset($getAPI['result']['HC94D087AC58D766725A51FCF84D5E199'])
) {
    $host = $getAPI['result']['G4D9EEC8CB8BB4E2CB92336AA92FABD4B'];
    $db   = $getAPI['result']['MC8AB13AA245335BE277F3931D7BB8E7C'];
    $user = $getAPI['result']['H7854A351CC9C8E6E1AE58CBB8382ADD2'];
    $pass = $getAPI['result']['HC94D087AC58D766725A51FCF84D5E199'];
} else {
    $host = '';
    $db = '';
    $user = '';
    $pass = '';
}
// Estabelecer conexão
$mysqli = new mysqli($host, $user, $pass, $db);
if ($mysqli->connect_error) {
    die('Erro ao se conectar');
}
print_r('Conexão bem-sucedida' . "<br>");
print_r(' Você está utilizando um Token com: ');
print_r(strlen($tokenAPI));
print_r(' Caracteres');
#
```

Tela que estará no código fonte.

No atual cenário (NÃO DESEJADO):
Abaixo segue a conexão com as credenciais gravadas no código fonte:

```php
<?php
$host = 'mysql02-farm5.proderj.net';
$db = 'GaloAzul';
$user = 'HomemAranha';
$pass = 'PneuT0rtO';

// Estabelecer conexão
$mysqli = new mysqli($host, $user, $pass, $db);
if ($mysqli->connect_error) {
die('Erro ao se conectar');
}
print_r('Conexão bem-sucedida' . "<br>");
?>

```

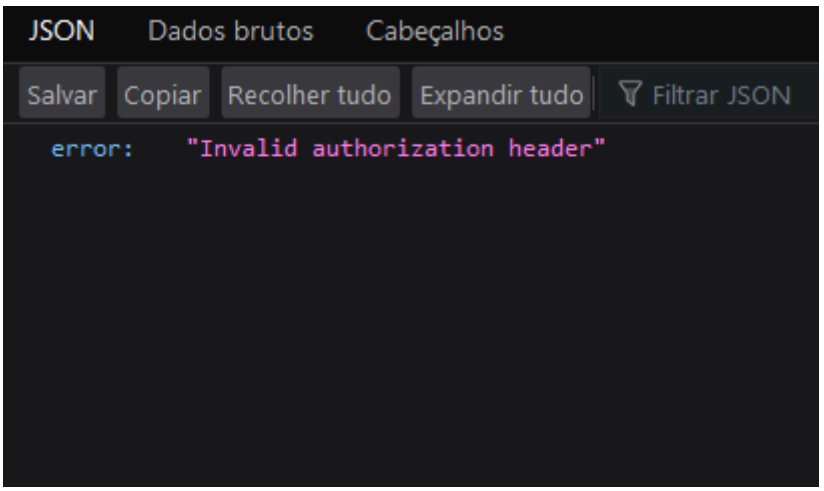
Tela com o código de conexão transparente

No cenário correto: A API deve ser chamada na aplicação do desenvolvedor e as variáveis deverão ser passadas após a validação do JSON Web Token e das variáveis.

Cenário Correto: Não é permitido antecipar a aplicação da API antes do DEPLOY. Somente no momento do Deploy o Dev deverá receber as credenciais e o JSON Web Token. as credenciais, dentro da aplicação do desenvolvedor, podem ser alteradas posteriormente em caso de detecção de ameaça.

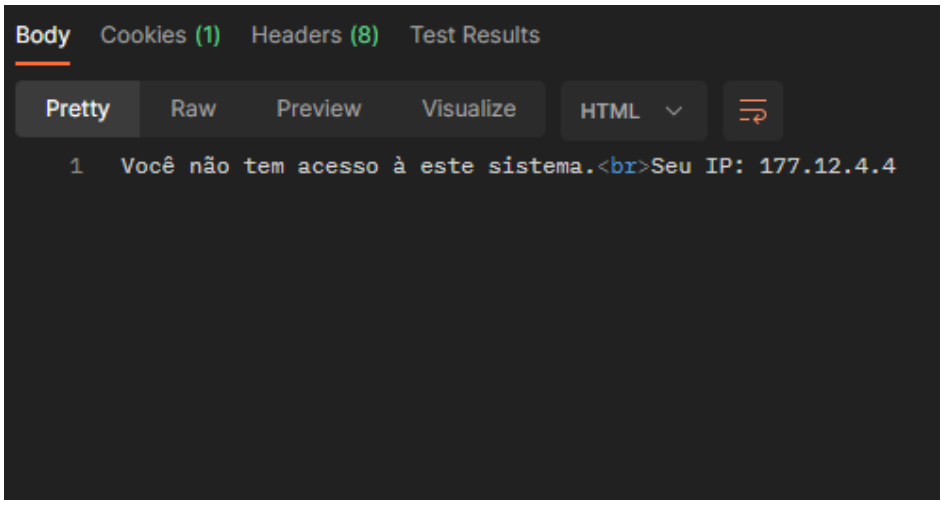
No cenário não permitido: A API não será acessada diretamente pelo desenvolvedor.

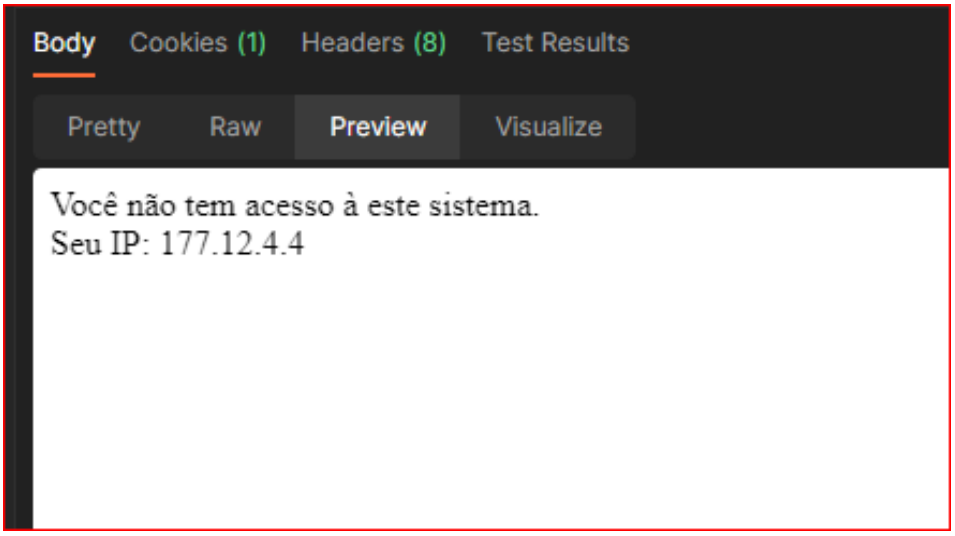
https://www/seguro/public/cdff3467e47eff0530a8465ddda62adf



No cenário correto Para que a API possa ser testada o desenvolvedor poderá utilizar um sistema de teste de APIs, acrescentar no endereço a credencial pública gerada apenas para aquela conexão entre sua aplicação e o banco de dados e o Token de autenticação com 287 caracteres. JSON Web Token pode ser alterado de acordo com os dados do Servidor em que a API se encontra ativa.

Após o Deploy, entre a aplicação do desenvolvedor e a API de segurança a mesma carregará um filtro interno com os IPs que deverão acessá-la.





A API de segurança foi desenvolvida para o diálogo entre servidores.

Não é recomendado a utilização da mesma em localhost ou máquinas que não estejam pré cadastradas dentro da API.

Segue anexos com os algoritmos da API:

Geração de Token:

```php
<?php

namespace App\Controllers;

use CodeIgniter\RESTful\ResourceController;
use CodeIgniter\API\ResponseTrait;
// use App\Models\NomeModel;
use Exception;

class Cea8409cb2bdb4db7a596696db5619ce extends ResourceController
{
    use ResponseTrait;
    private $ModelResponse;
    private $uri;

    public function __construct()
    {
        // $this->ModelResponse = new NomeModel();
        $this->uri = new \CodeIgniter\HTTP\URI(current_url());
        helper([
            'g13f8814e7d0fb0d0f3510d447673bac5'
        ]);
        return NULL;
    }

    # route POST /www/sigla/rota
    # route GET /www/sigla/rota
    # Informação sobre o controller
    # retorno do controller [JSON]
```

```php
    public function index($parameter = NULL)
    {
        try {
            $apiRespond = [
                'http' => array(
                    'header'  => 'Content-type: application/x-www-form-urlencoded',
                    'method'  => 'GET/POST',
                ),
                'message' => 'API loading data (dados para carregamento da API)',
                'date' => date('Y-m-d'),
                // 'method' => '__METHOD__',
                // 'function' => '__FUNCTION__',
                'page_title' => 'TITLE PAGE',
                'getURI' => $this->uri->getSegments(),
                'result' => array()
            ];
            $response = $this->response->setJSON($apiRespond, 201);
        } catch (\Exception $e) {
            $apiRespond = array(
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'TITLE PAGE',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
    }

    # route POST /www/sigla/rota
    # route GET /www/sigla/rota
    # Informação sobre o controller
    # retorno do controller [JSON]
    public function m30651056d1dfec84a0f3363168fa213c($parameter = NULL)
    {
        // exit($_SERVER['REMOTE_ADDR']);
        $c960116477798f1d0c060fca2472e500 = $this->applyFilter();
        $b0e43df1a1b37fecb7332a198df41495 = $_SERVER['REMOTE_ADDR'];
        if (!in_array($b0e43df1a1b37fecb7332a198df41495,
$c960116477798f1d0c060fca2472e500)) {
            print_r("Você não tem acesso à este sistema." . "<br>");
            exit("Seu IP: " . $b0e43df1a1b37fecb7332a198df41495);
        }
        $this->m1a8a5ed07e68228237069c9d74995629();
        try {
            $apiRequest = array(
                'G4D9EEC8CB8BB4E2CB92336AA92FABD4B' =>
m48fc9c2081401ee8bd4214a082916f74('bXlzcWwwMi1mYXJtMS5raW5naG9zdC5uZXQ='),
                'MC8AB13AA245335BE277F3931D7BB8E7C' =>
m48fc9c2081401ee8bd4214a082916f74('aGFiaWxpZGFkZTA2'),
                'H7854A351CC9C8E6E1AE58CBB8382ADD2' =>
m48fc9c2081401ee8bd4214a082916f74('aGFiaWxpZGEwNl9hZGQ0'),
                'HC94D087AC58D766725A51FCF84D5E199' =>
m48fc9c2081401ee8bd4214a082916f74('UHJvZGVyakAxMjIzMzM='),
            );
            $apiRespond = [
                'http' => array(
                    'header'  => 'Content-type: application/x-www-form-urlencoded',
                    'method'  => 'GET/POST',
```

```
                ),
                'message' => 'Customer origin (Origem do cliente): ' .
$b0e43df1a1b37fecb7332a198df41495,
                'date' => date('Y-m-d'),
                // 'method' => '__METHOD__',
                // 'function' => '__FUNCTION__',
                'page_title' => 'M59824F81215980A8CC5EA636D4A57A2A',
                'result' => $apiRequest,
            ];
            $response = $this->response->setJSON($apiRespond, 201);
        } catch (\Exception $e) {
            $apiRespond = array(
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'H711B0C50D7DA33816578FB043FB5D2B8',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
    }

    private function m1a8a5ed07e68228237069c9d74995629()
    {
        $response = "ERRO";
        require_once(APPPATH . 'Libraries/JWT/src/BeforeValidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/ExpiredException.php');
        require_once(APPPATH . 'Libraries/JWT/src/SignatureInvalidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/JWT.php');
        $key = H06BDB22FA5131A943A08FD13E410832F;
        $shelf_life = time() + (10 * 60 * 60); // 10 horas
        try {
            $token = array(
                "iss" => $_SERVER['REQUEST_SCHEME'] . $_SERVER['SERVER_NAME'],
                "aud" => $_SERVER['REQUEST_SCHEME'] . $_SERVER['SERVER_NAME'],
                "iat" => 1356999524,
                "nbf" => 1357000000,
                "data" => array(
                    "log" => strtoupper(md5(password_hash(time(), PASSWORD_DEFAULT)))
                )
            );
            $header = [
                "alg" => "HS256",
                "typ" => "JWT",
                "kid" => "key1"
            ];
            $jwt = \Firebase\JWT\JWT::encode($token, $key, 'HS256', null, $header);
            $response = setcookie('token', $jwt, [
                'expires' => $shelf_life,
                'path' => '/',
                'secure' => true, // cookies serão enviados apenas através de conexões
seguras
                'httponly' => true, // cookies não serão acessíveis através de scripts do
lado do cliente
                'samesite' => 'None' // os cookies serão enviados com requisições
cross-site
            ]);
        } catch (\Exception $e) {
            $apiRespond = array(
```

```
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'GE4E9B3E6DBA30CC2AEECDA8B4203FB63',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
    }

    private function applyFilter()
    {
        $list = [
            '::1',
            // '127.0.0.1',
            '177.12.4.4',
        ];

        return ($list);
    }
}
```

API de Credenciais de Banco de Dados:

```
<?php

namespace App\Controllers;

use CodeIgniter\RESTful\ResourceController;
use CodeIgniter\API\ResponseTrait;
// use App\Models\NomeModel;
use Exception;

class Cea8409cb2bdb4db7a596696db5619ce extends ResourceController
{
    use ResponseTrait;
    private $ModelResponse;
    private $uri;

    public function __construct()
    {
        // $this->ModelResponse = new NomeModel();
        $this->uri = new \CodeIgniter\HTTP\URI(current_url());
        helper([
            'g13f8814e7d0fb0d0f3510d447673bac5'
        ]);
        return NULL;
    }

    # route POST /www/sigla/rota
    # route GET /www/sigla/rota
    # Informação sobre o controller
    # retorno do controller [JSON]
    public function index($parameter = NULL)
    {
        try {
            $apiRespond = [
                'http' => array(
                    'header'  => 'Content-type: application/x-www-form-urlencoded',
```

```
                    'method'  => 'GET/POST',
                ),
                'message' => 'API loading data (dados para carregamento da API)',
                'date' => date('Y-m-d'),
                // 'method' => '__METHOD__',
                // 'function' => '__FUNCTION__',
                'page_title' => 'TITLE PAGE',
                'getURI' => $this->uri->getSegments(),
                'result' => array()
            ];
            $response = $this->response->setJSON($apiRespond, 201);
        } catch (\Exception $e) {
            $apiRespond = array(
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'TITLE PAGE',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
    }

    # route POST /www/sigla/rota
    # route GET /www/sigla/rota
    # Informação sobre o controller
    # retorno do controller [JSON]
    public function m30651056d1dfec84a0f3363168fa213c($parameter = NULL)
    {
        // exit($_SERVER['REMOTE_ADDR']);
        $c960116477798f1d0c060fca2472e500 = $this->applyFilter();
        $b0e43df1a1b37fecb7332a198df41495 = $_SERVER['REMOTE_ADDR'];
        if (!in_array($b0e43df1a1b37fecb7332a198df41495,
$c960116477798f1d0c060fca2472e500)) {
            print_r("Você não tem acesso à este sistema." . "<br>");
            exit("Seu IP: " . $b0e43df1a1b37fecb7332a198df41495);
        }
        $this->m1a8a5ed07e68228237069c9d74995629();
        try {
            $apiRequest = array(
                'G4D9EEC8CB8BB4E2CB92336AA92FABD4B' =>
m48fc9c2081401ee8bd4214a082916f74('bXlzcWwwMi1mYXJtMS5raW5naG9zdC5uZXQ='),
                'MC8AB13AA245335BE277F3931D7BB8E7C' =>
m48fc9c2081401ee8bd4214a082916f74('aGFiaWxpZGFkZTA2'),
                'H7854A351CC9C8E6E1AE58CBB8382ADD2' =>
m48fc9c2081401ee8bd4214a082916f74('aGFiaWxpZGEwNl9hZGQ0'),
                'HC94D087AC58D766725A51FCF84D5E199' =>
m48fc9c2081401ee8bd4214a082916f74('UHJvZGVyakAxMjIzMzM='),
            );
            $apiRespond = [
                'http' => array(
                    'header'  => 'Content-type: application/x-www-form-urlencoded',
                    'method'  => 'GET/POST',
                ),
                'message' => 'Customer origin (Origem do cliente): ' .
$b0e43df1a1b37fecb7332a198df41495,
                'date' => date('Y-m-d'),
                // 'method' => '__METHOD__',
                // 'function' => '__FUNCTION__',
```

```
                'page_title' => 'M59824F81215980A8CC5EA636D4A57A2A',
                'result' => $apiRequest,
            ];
            $response = $this->response->setJSON($apiRespond, 201);
        } catch (\Exception $e) {
            $apiRespond = array(
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'H711B0C50D7DA33816578FB043FB5D2B8',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
    }

    private function m1a8a5ed07e68228237069c9d74995629()
    {
        $response = "ERRO";
        require_once(APPPATH . 'Libraries/JWT/src/BeforeValidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/ExpiredException.php');
        require_once(APPPATH . 'Libraries/JWT/src/SignatureInvalidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/JWT.php');
        $key = H06BDB22FA5131A943A08FD13E410832F;
        $shelf_life = time() + (10 * 60 * 60); // 10 horas
        try {
            $token = array(
                "iss" => $_SERVER['REQUEST_SCHEME'] . $_SERVER['SERVER_NAME'],
                "aud" => $_SERVER['REQUEST_SCHEME'] . $_SERVER['SERVER_NAME'],
                "iat" => 1356999524,
                "nbf" => 1357000000,
                "data" => array(
                    "log" => strtoupper(md5(password_hash(time(), PASSWORD_DEFAULT)))
                )
            );
            $header = [
                "alg" => "HS256",
                "typ" => "JWT",
                "kid" => "key1"
            ];
            $jwt = \Firebase\JWT\JWT::encode($token, $key, 'HS256', null, $header);
            $response = setcookie('token', $jwt, [
                'expires' => $shelf_life,
                'path' => '/',
                'secure' => true, // cookies serão enviados apenas através de conexões
seguras
                'httponly' => true, // cookies não serão acessíveis através de scripts do
lado do cliente
                'samesite' => 'None' // os cookies serão enviados com requisições
cross-site
            ]);
        } catch (\Exception $e) {
            $apiRespond = array(
                'message' => array('danger' => $e->getMessage()),
                'page_title' => 'GE4E9B3E6DBA30CC2AEECDA8B4203FB63',
            );
            $response = $this->response->setJSON($apiRespond, 404);
        }
        return $response;
```

```
    }

    private function applyFilter()
    {
        $list = [
            '::1',
            // '127.0.0.1',
            '177.12.4.4',
        ];

        return ($list);
    }
}
```

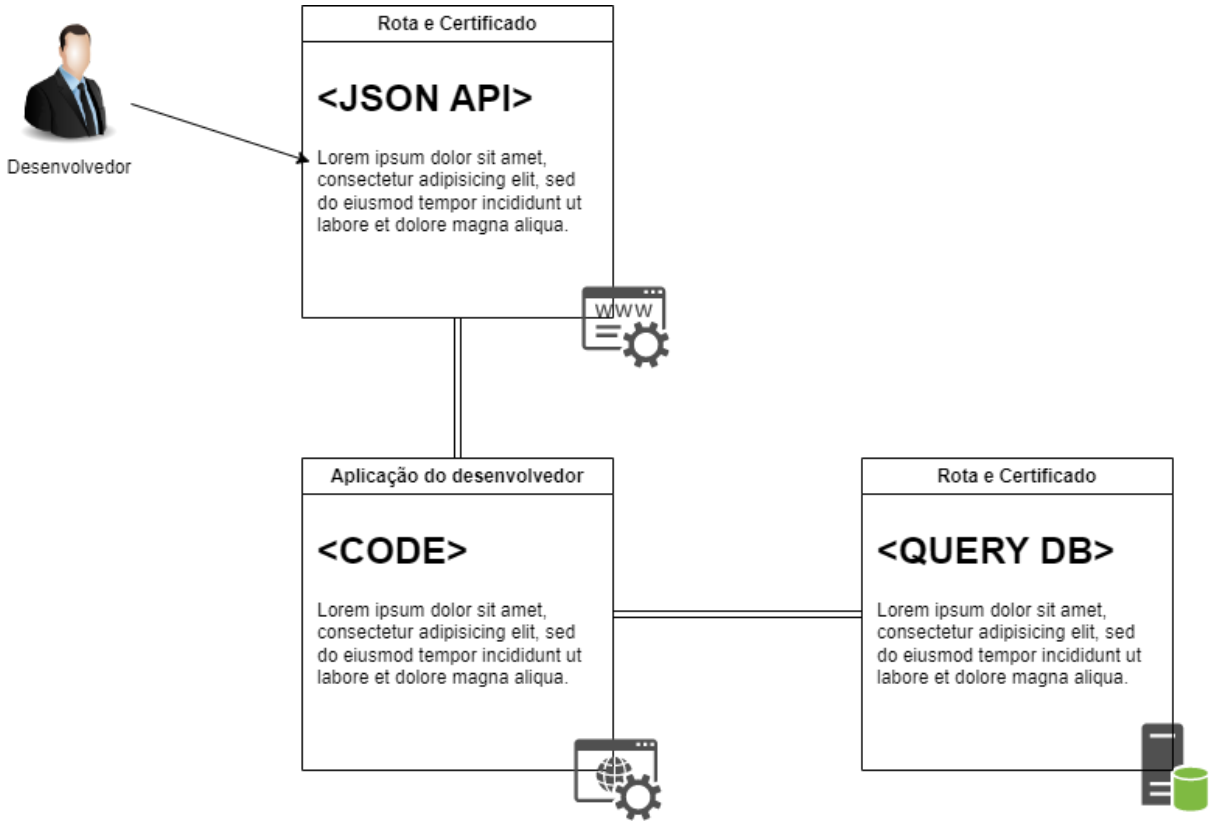
Middleware para filtro JwtFilter:
É utilizado antes do caregamento da API

```
<?php

namespace App\Filters;

use CodeIgniter\HTTP\RequestInterface;
use CodeIgniter\HTTP\ResponseInterface;
use CodeIgniter\Filters\FilterInterface;
use CodeIgniter\HTTP\Response;
use Config\Services;

class JwtFilter implements FilterInterface
{
    public function before(RequestInterface $request, $arguments = null)
    {
        // Incluindo os arquivos manualmente
        require_once(APPPATH . 'Libraries/JWT/src/BeforeValidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/ExpiredException.php');
        require_once(APPPATH . 'Libraries/JWT/src/SignatureInvalidException.php');
        require_once(APPPATH . 'Libraries/JWT/src/JWT.php');
        require_once(APPPATH . 'Libraries/JWT/src/Key.php');

        $keys = [
            "key1" => new \Firebase\JWT\Key(H06BDB22FA5131A943A08FD13E410832F, 'HS256')
        ];

        $authHeader = $request->getServer('HTTP_AUTHORIZATION');
        $arr = explode(' ', $authHeader);

        // Verifica se o cabeçalho de autorização está no formato correto
        if (count($arr) != 2) {
            $response = Services::response();
            return $response->setJSON(['error' => 'Invalid authorization
header'])->setStatusCode(ResponseInterface::HTTP_UNAUTHORIZED);
        }

        $token = $arr[1];

        if ($token) {
            try {
```

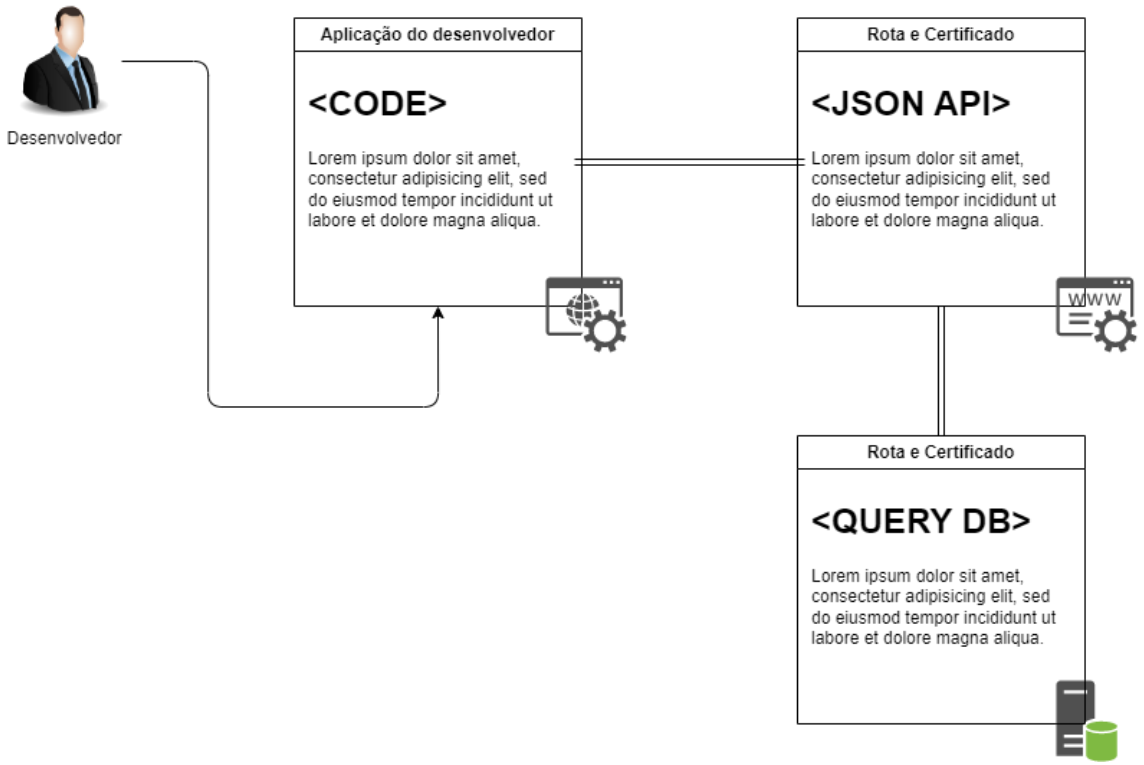
```
                $decoded = \Firebase\JWT\JWT::decode($token, $keys['key1']);
            } catch (\Exception $e) {
                $response = Services::response();
                return $response->setJSON(['error' =>
$e->getMessage()])->setStatusCode(ResponseInterface::HTTP_UNAUTHORIZED);
            }
        } else {
            $response = Services::response();
            return $response->setJSON(['error' => 'Token not
found'])->setStatusCode(ResponseInterface::HTTP_UNAUTHORIZED);
        }
    }

    public function after(RequestInterface $request, ResponseInterface $response,
$arguments = null)
    {
        // Do something here
    }
}
```

Cenário desejados:

Cenário 1: O Desenvolvedor recebe o endpoint, conecta em sua aplicação, chama as variáveis de segurança em seu código, prossegue para as consultas em banco de dados.

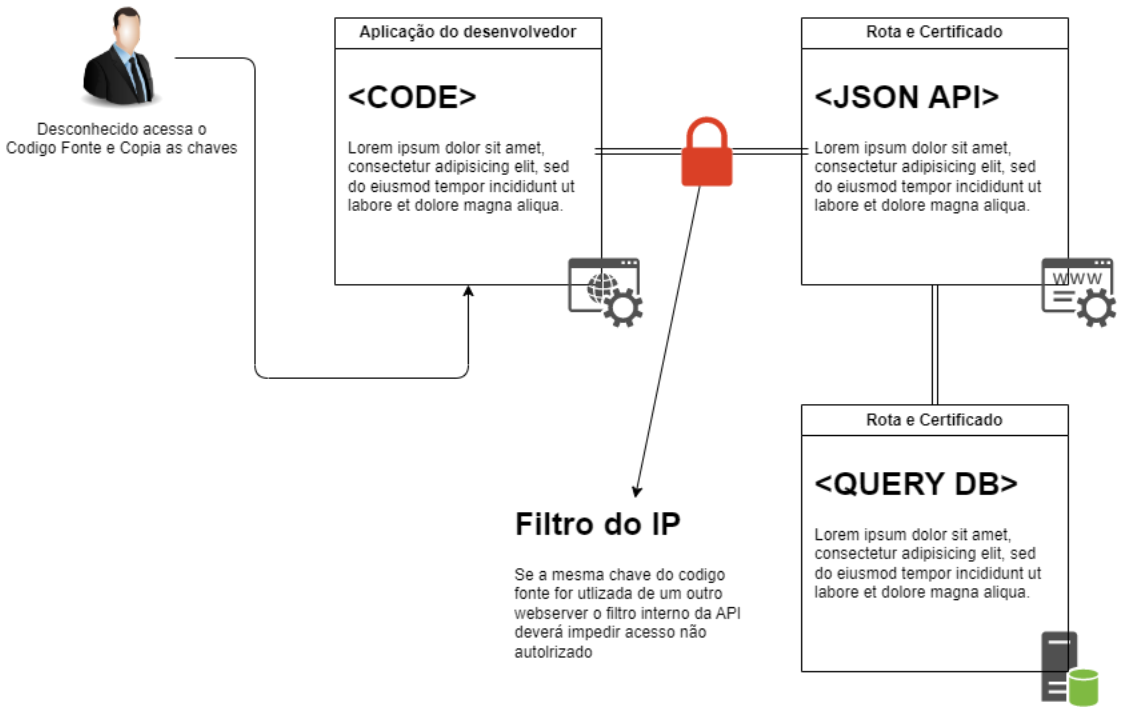


Cenário 2: A Aplicação do desenvolvedor vai até a API, recebe as credenciais e segue ao banco com as credenciais recebidas:

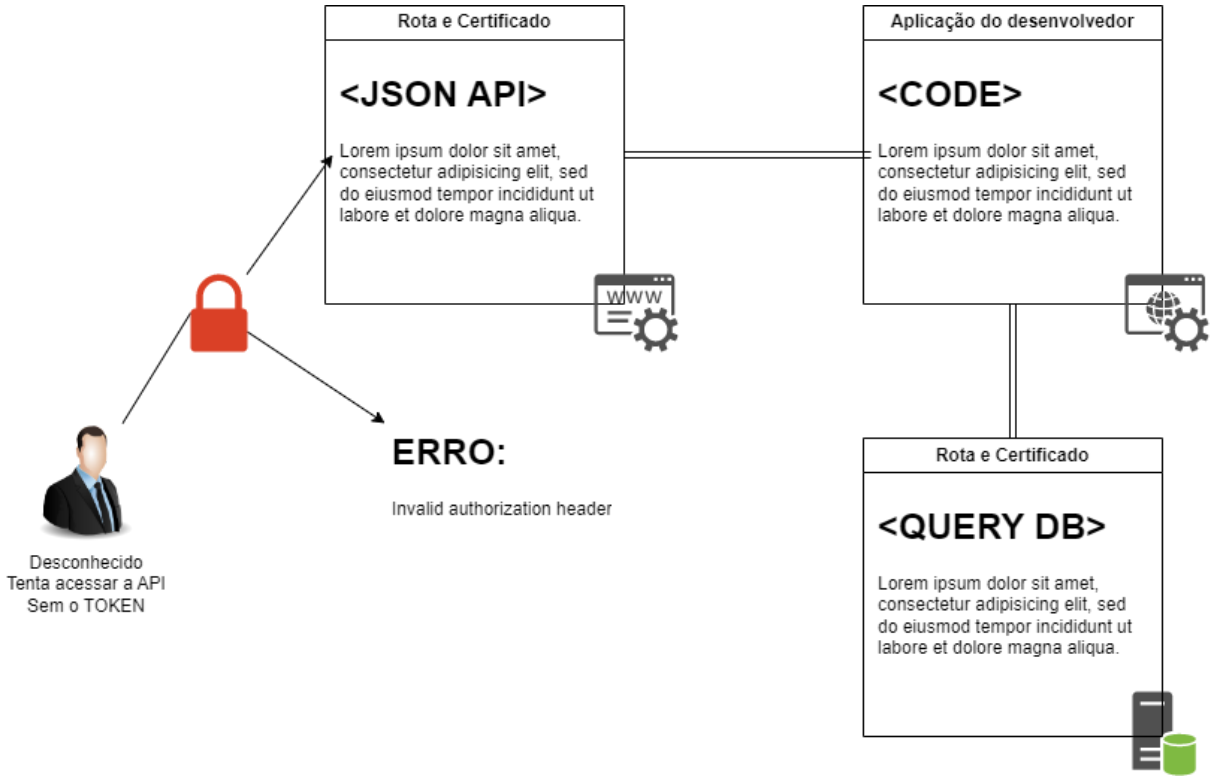


Cenários indesejados:

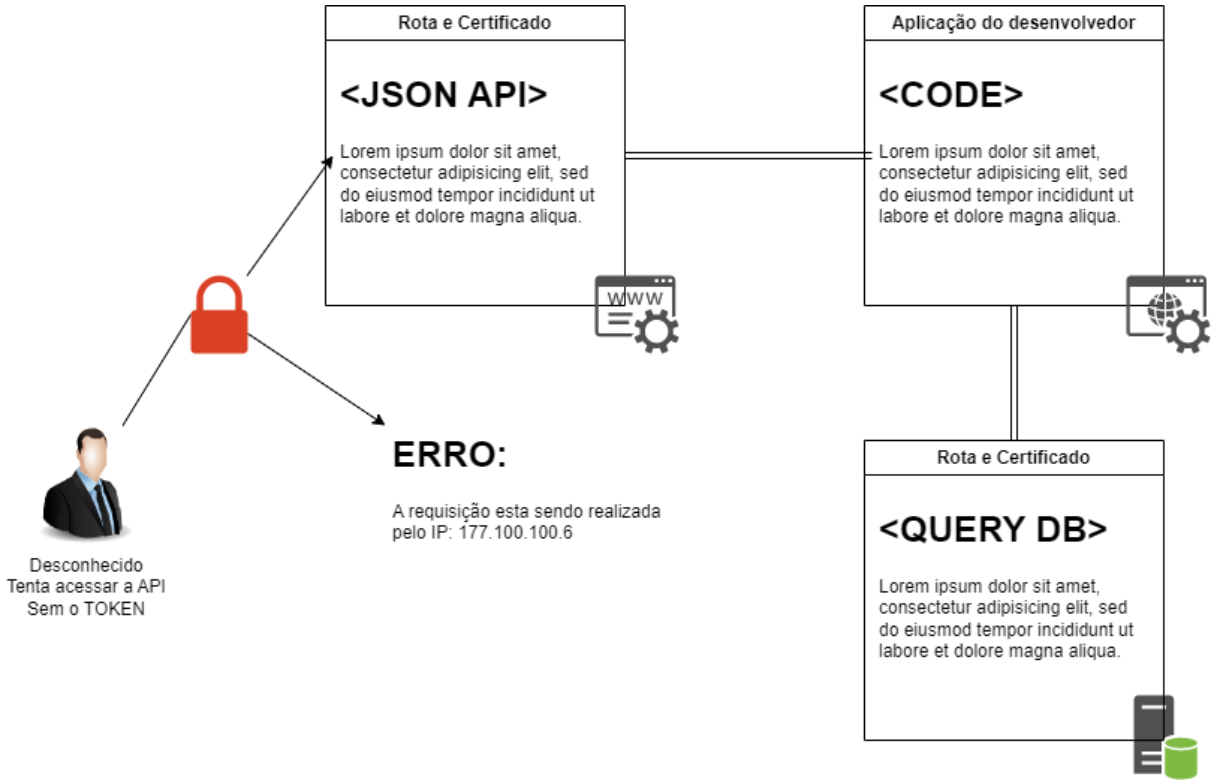
Cenário: 3: Desconhecido toma posse do arquivo de configuração e credenciais do JSON Web Token e tenta rodar o código em outro servidor web.



Cenário 4: Desconhecido tenta realizar uma conexão direta farejando a rede ou através de um BOT, mas não possui as credenciais: JSON Web Token.



Cenário 5: O Desconhecido consegue recuperar as credenciais do JSON Web Token porem tenta acessar de um IP não autorizado pelo filtro da API.



FIM

GUSTAVO DE MELO HAMMES
GFS FÁBRICA DE SOFTWARE
EDS
gustavo.hammes@extreme.digital
Whatsapp: +55 (21) 9 8055-8545