

Apresentação do componente.

DESENVOLVIMENTO PARA SERVIDORES II (4)³⁰

Objetivos gerais. Desenvolver um site completo de e-commerce ou outro tipo de negócio na Internet usando uma linguagem apropriada a servidores, banco de dados e padrões de projeto.

Objetivos específicos. Implementar softwares do lado servidor e com uso de uma linguagem de programação e de padrões de projetos mais usuais como MVC, DAO, Composite, Singleton, entre outros.

Ementa. Conceitos e evolução das tecnologias de programação de servidores. Recursos da linguagem escolhida para servidores na Internet. Padrões de projetos. Integração com sistemas (Google Maps API, Twitter, entre outros)

Bibliografia básica

DEITEL, H; DEITEL, P. Java – Como Programar. São Paulo: Prentice-Hall do Brasil, 2010.

GAMMA, E et al. *Padrões de projeto*. Porto Alegre: Bookman, 2005.

MELO, A. A; LUCKOW, D. H. *Programação Java para a web*. São Paulo: Novatec, 2011.

Bibliografia complementar

DAIGNEAU, R. *Service design patterns*. Harlow (UK): Addison Wesley, 2011.

FREEMAN, E; FREEMAN, E. *Use a Cabeça! Padrões de Projetos*. 2, ed. Rio de Janeiro: Starlin Alta Consult, 2007.

HARTL, M. *Ruby on rails 3 tutorial*. Harlow (UK): Addison Wesley, 2011.

IERUSALIMSCHY, R; CELES, W; FIGUEIREDO, L. H. *LUA programming gems*. Rio de Janeiro: LUA Org, 2008.

RUBY, S; THOMAS, D; HANSSON, D. *Agile web development with rails*. New York: O'Reilly & Assoc. 2010.

1-) Uso de celulares:

Para o bom andamento das aulas, recomendo que utilizem os celulares em vibracall, para não atrapalhar o andamento da aula.

2-) Material das aulas:

A disciplina trabalha com NOTAS DE AULA que são disponibilizadas ao final de cada aula.

3-) Prazos de trabalhos e atividades:

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada, ou seja, se não for entregue até a data limite a mesma receberá nota 0, isso tanto para atividades entregues de forma impressa ou enviadas ao e-mail da disciplina. Qualquer problema que tenham, me procurem com antecedência para verificarmos o que pode ser realizado.

4-) Qualidade do material de atividades:

- Impressas ou manuscritas:
Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.
- Digitais:
Ao enviarem atividades para o e-mail da disciplina, SEMPRE no assunto deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail deverá ter o(s) nome(s) do(s) integrante(s) da atividade, sem estar desta forma a atividade será DESCONSIDERADA.

5-) Critérios de avaliação:

Cada trimestres teremos as seguintes formas de avaliação:

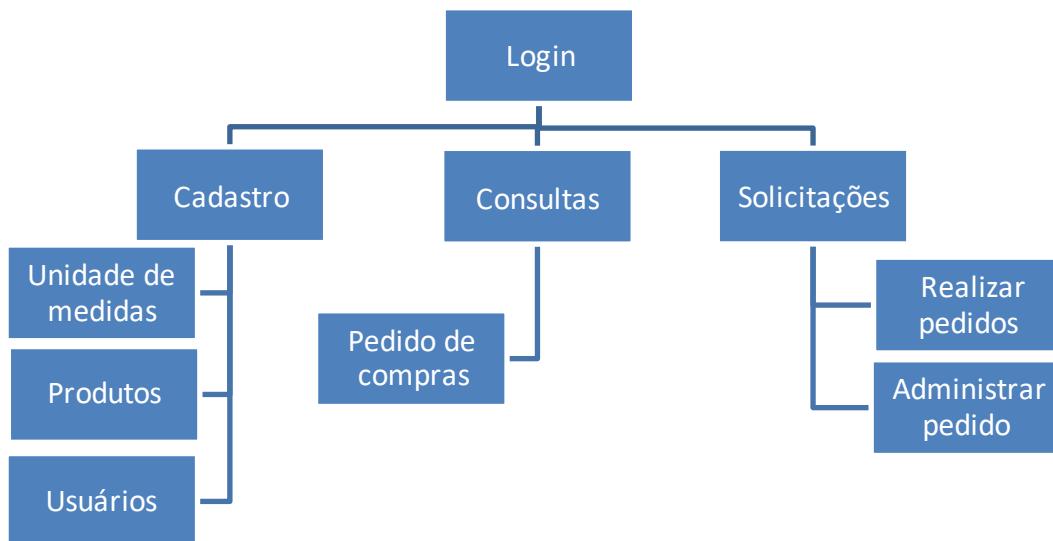
- Práticas em Laboratório;
- Assiduidade;
- Outras que se fizerem necessário.

1. Introdução

Nessa fase do curso iremos abordar a ideia da programação no lado do servidor, até dando continuidade aos conceitos vistos em Desenvolvimento para Servidores I, para isso, continuaremos com a linguagem PHP, nossa preocupação não será o **frontend**, no primeiro momento, mas após a fase de backend, iremos andar nesta parte refinando e melhorando os conceitos.

2. Mão à obra

Em nossas aulas utilizaremos a prática (prática), então a ideia agora, é montarmos um sistema de compras que terá a seguinte decomposição funcional, até mais simples que o sistema anterior que desenvolvemos, a pasta base do CodeIgniter, eu irei disponibilizar para vocês:



A cada aula nosso objetivo é montarmos a estrutura de cada módulo da ideia acima, portanto a frequência nas aulas é de suma importância para o bom andamento.

2.1 Tabela de login

O primeiro passo é criarmos o login de nosso sistema, assim iremos construir em um banco de dados MySql, uma tabela que armazene essas informações, abaixo temos a estrutura da mesma:

```

1      #Criação do banco de dados, denominamos compras
2 •  create database compras;
3
4      #Habilitamos a utilização do mesmo
5 •  use compras;
6
7      #Estrutura da tabela usuários
8 •  create table usuarios (
9          id_usuario integer not null auto_increment primary key,
10         usuario    varchar(15) not null,
11         senha      varchar(32) not null,
12         dtcria     datetime default now(),
13         estatus    char(01) default ''
14     );
15
16      #Vamos inserir um usuário padrão do sistema
17 •  insert into usuarios (usuario, senha)
18     values ('admin', md5('admin123'));
19
20      # Ao final fazemos um Select para verificar o registro inserido
21 •  select * from usuarios where senha = md5('admin123');
  
```

Ao fazer a `select` o resultado aparecerá assim:

	id_usuario	usuario	senha	dtcria	estatus
▶	1	admin	0192023a7bbd73250516f069df18b500	2021-08-26 11:45:14	

Vocês viram que usamos para inserir o usuário em nossa tabela uma função MD5, correto? Mas o que ela significa?

O MD5 é um dos algoritmos de criptografia mais conhecidos que funciona através de chamada de função como fizemos acima, e seu funcionamento consiste na geração de um valor hexadecimal de 32 dígitos a partir de uma *String*, por isso que criamos o campo senha com tamanho de 32. Esse processo, no entanto, é unidirecional, ou seja, uma vez aplicada, essa função criptográfica não pode ser revertida a fim de recuperar o valor original, ou seja, se o usuário esquecer essa senha, nem nós como administradores do banco de dados conseguiremos passar a senha para o usuário, o que força fazer o *reset* da mesma.

Isso nos ajuda nos quesitos de segurança (principalmente para salvar dados sensíveis como a senha do usuário), já que mesmo que os dados criptografados sejam hackeados, o invasor não terá como descriptografá-los. Para aplicar esse algoritmo no MySQL, dispomos da função MD5, que recebe como único parâmetro o texto a ser criptografado e retorna o valor mascarado. Assim, podemos utilizá-la diretamente ao inserir um registro na tabela, como fizemos anteriormente.

Bom, agora é com vocês, para a nossa próxima aula, configurem o *CodeIgniter* em vossas máquinas, o MySQL com o Workbench e criar o banco de dados com a primeira tabela com o registro do usuário *admin* com senha criptografada.

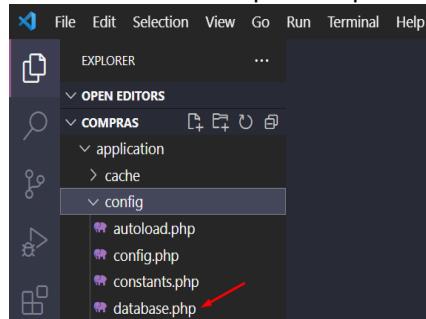
2.2. Configurando o *CodeIgniter* para nosso projeto

Nesse momento teremos que configurar nosso *framework* para a estrutura de nosso projeto, iremos configurar nesse alterando os seguintes arquivos ne nossa estrutura:

- database.php (configuraremos o banco de dados compras);
- autoload.php (configurarmos a chamada automática do banco de dados);
- routes.php (configuraremos a rota de nossa controller inicial do projeto);
- config.php (configuraremos nossa base_url());
- htaccess (URLs amigáveis).

2.2.1. Configurando o arquivo *databases.php*

Vamos acessar o arquivo na pasta *application/config*, assim:



Informaremos os dados de configuração, da seguinte forma:

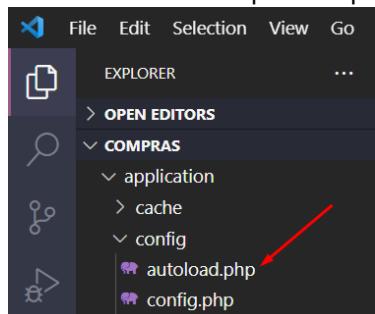
```

76 $db['default'] = array(
77     'dsn'      => '',
78     'hostname' => 'localhost',
79     'username' => 'root',
80     'password' => '',
81     'database' => 'compras',
82     'dbdriver' => 'mysqli',
83     'dbprefix' => '',
84     'pconnect' => FALSE,
85     'db_debug' => (ENVIRONMENT !== 'production'),
86     'cache_on' => FALSE,
87     'cachedir' => '',
88     'char_set' => 'utf8',
89     'dbcollat' => 'utf8_general_ci',
90     'swap_pre' => '',
91     'encrypt' => FALSE,
92     'compress' => FALSE,
93     'stricton' => FALSE,
94     'failover' => array(),
95     'save_queries' => TRUE
96 );

```

2.2.2. Configurando o arquivo *autoload.php*

Vamos acessar o arquivo na pasta *application/config*, assim:



E acrescentamos a chamada da biblioteca *database*, conforme abaixo:

```

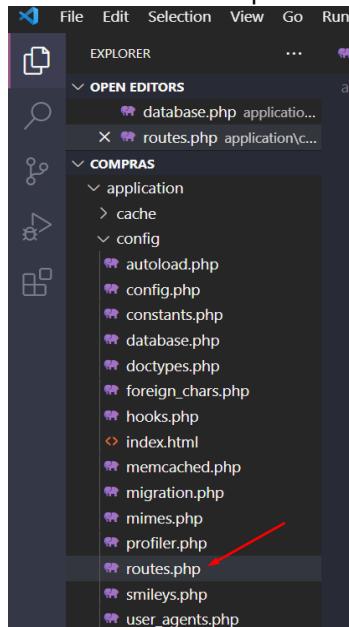
autoload.php X database.php config.php

application > config > autoload.php
56 | You can also supply an alternative library
57 | in the controller:
58 |
59 | $autoload['libraries'] = array('user_agent');
60 */
61 $autoload['libraries'] = array('database');
62

```

2.2.3. Configurando o arquivo *routes.php*

Vamos acessar o arquivo na pasta *application/config*, assim:



Informaremos a *controller* que dará início ao sistema, da seguinte forma:

```

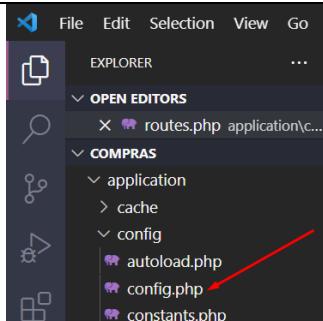
routes.php X

application > config > routes.php
48 |
49 | Examples: my-controller/index -> my_controller/index
50 | my-controller/my-method -> my_controller/my_method
51 */
52 $route['default_controller'] = 'login';
53 $route['404_override'] = '';
54 $route['translate_uri_dashes'] = FALSE;

```

2.2.4. Configurando o arquivo *config.php*

Vamos acessar o arquivo na pasta *application/config*, para vamos configurar a URL que será utilizada, assim:



Devemos informar o caminho de nosso projeto, conforme a seguir:

```
Run Terminal Help config.php - Compras - Visual Studio Code [Active]
config.php x
application > config > config.php
22 | If you need to allow multiple domains, remember that t
23 | a PHP script and you can easily do that on your own.
24 |
25 */
26 $config['base_url'] = 'http://localhost/compras/';
27
```

Também nesse mesmo arquivo vamos retirar o *index.php* para trabalharmos com URL amigáveis.

```
config.php x
application > config > config.php
35 | variable so that it is blank.
36 |
37 */
38 $config['index_page'] = '';
39
40 /*
41 |-----
```

2.2.5. Configurando o arquivo *htaccess* (Já irei disponibilizar a pasta base com ele feito)

O *CodeIgniter* por padrão já dá suporte a URL amigáveis, como os grandes frameworks em PHP, o grande problema é que ele ainda insiste em mostrar o arquivo *index.php* na URL o que deixa toda requisição horrível. Veja o padrão de URL na instalação padrão do *CodeIgniter*: <http://seusite.com.br/index.php/controller/method/parameter>

Como deveria ser: <http://seusite.com.br/controller/method/parameter>

Como estamos usando o Apache como nosso servidor *web*, criamos um arquivo *.htaccess* no caminho “C:\XAMP\HTDOCS\Compras” com o seguinte conteúdo (irei fornecer esse arquivo a vocês, basta colocar na pasta do projeto, é conforme abaixo):

```
.htaccess - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ /compras/index.php/$1 [L]

#RewriteEngine On
#RewriteBase /
#RewriteCond %{REQUEST_FILENAME} !-f
#RewriteCond %{REQUEST_FILENAME} !-d
##RewriteRule ^(.*)$ index.php?url=$1
#RewriteRule ^(.*)$ /compras/index.php?url=$1
```

Com isso terminamos a configuração do projeto Compras, vamos dizer que é a base para que o projeto “rode”.

2.3. Controller Login

Agora vamos implementar nossa *controller* para que a mesma possa receber os atributos usuário e senha, fazendo a solicitação para que a *Model* faça a consulta no banco de dados e

retorne se o usuário e a senha estão cadastrados no sistema, essa *controller* deverá ser salva em: C:\xampp\htdocs\compras\application\controllers.

Mas antes vamos ver dois comandos que teremos em nossa *controller Login*, os mesmos servem para receber dados no formato JSON e na sequência colocá-los em objetos que serão passados para atributos.

- **file_get_contents('php://input')**: Esse comando serve para ler os dados Brutos que geralmente são enviados por um JSON, nesse caso estamos fazendo essa leitura e colocando no atributo \$json.

- **json_decode**: Esse comando recebe um JSON e a transforme em um atributo PHP, em nosso caso um objeto.

Na pasta **helpers** de nosso projeto, está o arquivo que trata os campos que deverão ser informados pelo *frontend*.

Iremos também ver aqui o **try catch**:

As funções *try catch* são funções nativas do PHP. O PHP *try catch* são blocos de comandos que tem como principal objetivo tratar exceções que o programador não tem como prever que irão acontecer ou controlar. Como, por exemplo, erros de execução, ou ainda erros como o usuário perder a conexão com a internet, entre outros. Dessa forma, esses erros e falhas são tratados como exceções, de forma que é possível então compreender ou até manipular o comportamento da aplicação caso aconteça algo inesperado.

Portanto, o bloco PHP *try catch* serve para que, em um dado momento em que um código possa gerar um erro inesperado, o programador consiga manipular as possibilidades e exceções. Dessa forma, através do *try* ele irá tentar executar o código, caso não ocorra nenhum erro, o programa seguirá o seu fluxo normal. Porém, caso aconteça algum erro, ele passa a ser tratado como uma exceção, e dessa forma, utilizará o *Catch* para poder tratar esse erro.

De certa forma, podemos comparar com o laço de repetição *if* e *else*. Porém, o PHP *try catch* deve ser utilizado quando o desenvolvedor web não tem como garantir que aquele código será executado com sucesso. Ou seja, para tratar os comportamentos inesperados.

Podemos dizer que o bloco *try* é um bloco protegido, pois, caso ocorra algum problema, a execução do código será direcionado ao bloco *catch* correspondente. Dessa forma, se o usuário final realizar algo inadequado, ou houver uma perda de conexão, resultará em um erro e na quebra de execução do programa. Porém, podemos evitar um erro brusco ou queda brusca e tratar o erro como exceção da melhor forma possível.

Também percebam que iremos fazer as validações dos dados passados para a *Controller Login* **antes de enviarmos para a camada Model**, além disso padronizamos um array de retorno para padronizarmos todos os retornos que teremos, esse array é composto por um código de mensagem que é numérico e uma descrição desse código, observem o código a seguir:

```

1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3
4  class Login extends CI_Controller {
5      //Atributos provados da classe
6      private $usuario;
7      private $senha;
8
9      //Getters dos atributos
10     public function getUsuario()
11     {
12         return $this->usuario;
13     }
14
15     public function getSenha()
16     {
17         return $this->senha;
18     }
19

```

```

20 //Setters dos atributos
21 public function setUsuario($usuarioFront)
22 {
23     $this->usuario = $usuarioFront;
24 }
25
26 public function setSenha($senhaFront)
27 {
28     $this->senha = $senhaFront;
29 }
30
31 public function logar() {
32     /////////////////////////////////
33     //Recebimento via JSON o Usuário e senha
34     //Retornos possíveis:
35     //1 - Usuário e senha validados corretamente (Banco)
36     //2 - Faltou informar o usuário (FrontEnd)
37     //3 - Faltou informar a senha (FrontEnd)
38     //4 - Usuário ou senha inválidos (Banco)
39     //99 - Os campos vindos do FrontEnd não representam o método de login
40     ///////////////////////////////
41
42 try {
43     //Usuário e senha recebidos via JSON
44     //e colocados em atributos
45     $json = file_get_contents('php://input');
46     $resultado = json_decode($json);
47
48     //Array com os dados que deverão vir do Front
49     $lista = array(
50         "usuario" => '0',
51         "senha" => '0'
52     );
53
54     if (verificarParam($resultado, $lista) == 1) {
55         //Fazendo os setters
56         $this->setUsuario($resultado->usuario);
57         $this->setSenha($resultado->senha);
58
59         if (trim($this->getUsuario()) == ''){
60             $retorno = array('codigo' => 2,
61                             'msg' => 'Usuário não informado');
62         }elseif (trim($this->getSenha()) == ''){
63             $retorno = array('codigo' => 3,
64                             'msg' => 'Senha não informada');
65         }else{
66
67             //Realizo a instância da Model
68             $this->load->model('M_acesso');
69
70             //Atributo $retorno recebe array com informações
71             //da validação do acesso
72             $retorno = $this->M_acesso->validalogin($this->getUsuario(),
73                                         $this->getSenha());
74         }
75     } else {
76         $retorno = array(
77             'codigo' => 99,
78             'msg' => 'Os campos vindos do FrontEnd não representam
79             o método de login. Verifique.'
80         );
81     }
82 } catch (Exception $e) {
83     $retorno = array('codigo' => 0,
84                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
85                     '$e->getMessage()');
86 }
87
88 //Retorno no formato JSON
89 echo json_encode($retorno);
90 }
91 }
```

2.4. Model M_acesso

Para completar essa ponte, os atributos vindo da Controller que serão validados e consequentemente passados para a Model M_acesso onde faremos uma Query no banco de dados e retorna se o usuário e senha são válidos, essa Model deverá ser salva em: C:\xampp\htdocs\compras\application\models , vejam:

```

1  <?php
2  defined('BASEPATH') or exit('No direct script access allowed');
3
4  class M_acesso extends CI_Model {
5      public function validaLogin($usuario, $senha){
6          try{
7              //Atributo retorno recebe o resultado do SELECT
8              //realizado na tabela de usuários lembrando da função MD5()
9              //por causa da criptografia.
10             $retorno = $this->db->query("select * from usuarios
11                                         where usuario = '$usuario'
12                                         and senha = md5('$senha')
13                                         and estatus = ''");
14
15             //Verifica se a quantidade de linhas trazidas na consulta é superior a 0,
16             //isso quer dizer que foi encontrado o usuário e senha passados pela
17             //Controller.
18
19             //Criado um array com dois elementos para retorno do resultado
20             //1 - Código da mensagem
21             //2 - Descrição da mensagem
22
23             if($retorno->num_rows() > 0){
24                 $dados = array('codigo' => 1,
25                               'msg' => 'Usuário correto');
26
27             }else{
28                 $dados = array('codigo' => 4,
29                               'msg' => 'Usuário ou senha inválidos.');
30             }
31             //Envia o array $dados com as informações tratadas
32             //acima pela estrutura de decisão if
33
34             return $dados;
35         } catch (Exception $e) {
36             $dados = array('codigo' => 0,
37                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
38                           $e->getMessage(), "\n");
39         }
40     }
41 }
```

Com isso fechamos o ciclo View (atributos recebidos via POST) – Controller – Model (Requisição) e o passo inverso com retorno da validação.

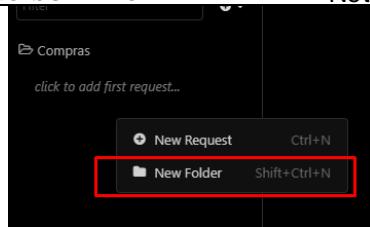
2.5. Colocando nosso primeiro método no INSOMNIA

Vamos configurar nosso ambiente para o projeto de nossas aulas, vamos criar uma pasta conforme imagem abaixo:

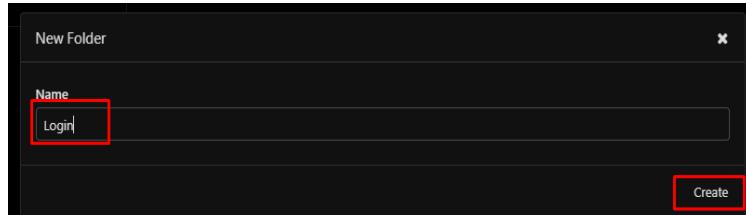
Definiremos uma pasta para guardar os ENDPOINTS



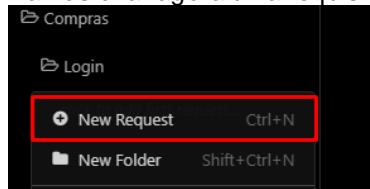
Dentro da pasta Compras criamos uma outra pasta para o objeto Login de nosso sistema.



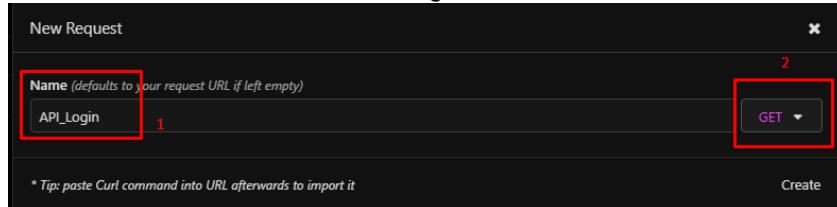
Criaremos uma pasta de *Login*, a ideia é armazenarmos todos os *Endpoints* relacionados a *Login* nesse local.



Vamos criar agora uma requisição, onde colocaremos o nosso primeiro método.

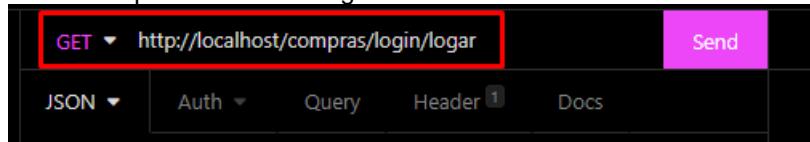


Iremos colocar no nome de API_Login

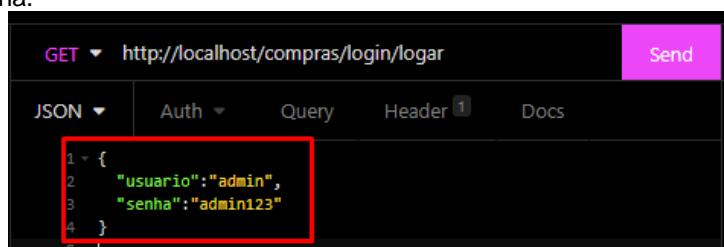


Sendo: 1 – Nome de nossa API; 2 – GET (Consulta)

Feito isso precisamos configurar a *url* da *Controller* com o método que vamos trabalhar:



Também precisamos parametrizar os dados usados nesse método, nesse caso é o usuário e a senha.



Após isso enviamos a requisição e observamos o retorno com a resposta.

A screenshot of a Postman API request. The URL is `http://localhost/compras/login/logar`. The method is `GET`. The request body is JSON with the following content:

```

1+ {
2  "usuario":"admin",
3  "senha":"admin123"
4 }
5

```

The response status is `200 OK`, with a response time of `83.9 ms` and a size of `41 B`. The response body is:

```

1+ {
2  "codigo": 1,
3  "msg": "Usuário correto"
4 }

```

Com isso, fazemos a verificação do nosso método e documentamos as *APIs* de nosso projeto, de acordo com o andamento de nossas aulas, iremos fazer a parte de *backend* dessa forma.

2.6. Interface de Usuário

Dando continuidade, agora vamos fazer o *CRUD(Create, Read, Update and Delete)* de nosso Usuário.

2.6.1. Criando o método de cadastro de usuário

Vamos agora criar o método de cadastro de usuário em nosso sistema, lembrando que faremos a parte de *backend*, para isso vamos criar uma nova *controller* para tratar usuários, assim teremos a `usuario.php`, iremos começar pelos métodos (*getters* e *setters*, inserir, consultar, alterar e desativar), e modificadores de acesso.

A screenshot of a code editor showing the `Usuario.php` controller file. The file is located at `application > controllers > Usuario.php`. The code defines a `Usuario` class that extends `CI_Controller`. It contains several `get` methods for retrieving user information and a `set` method for setting a user's front-end information. The code is as follows:

```

1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3
4  class Usuario extends CI_Controller {
5      //Atributos privados da classe
6      private $usuario;
7      private $senha;
8      private $nome;
9      private $tipoUsuario;
10     private $usuarioLogin;
11
12     //Getters dos atributos
13     public function getUsuario()
14     {
15         return $this->usuario;
16     }
17
18     public function getSenha()
19     {
20         return $this->senha;
21     }
22
23     public function getNome()
24     {
25         return $this->nome;
26     }
27
28     public function getTipoUsuario()
29     {
30         return $this->tipoUsuario;
31     }
32
33     public function getUsuarioLogin()
34     {
35         return $this->usuarioLogin;
36     }
37
38     //Setters dos atributos
39     public function setUsuario($usuarioFront)
40     {
41         $this->usuario = $usuarioFront;
42     }
43

```

```

44     public function setSenha($senhaFront)
45     {
46         $this->senha = $senhaFront;
47     }
48
49     public function setNome($nomeFront)
50     {
51         $this->nome = $nomeFront;
52     }
53
54     public function setTipoUsuario($tipoUsuario)
55     {
56         $this->tipoUsuario = $tipoUsuario;
57     }
58
59     public function setUsuarioLogin($usuarioLoginFront)
60     {
61         $this->usuarioLogin = $usuarioLoginFront;
62     }
63
64     public function inserir(){
65         //Usuário, senha, nome, tipo (Administrador ou Comum)
66         //recebidos via JSON e colocados em variáveis
67         //Retornos possíveis:
68         //1 - Usuário cadastrado corretamente (Banco)
69         //2 - Faltou informar o usuário (FrontEnd)
70         //3 - Faltou informar a senha (FrontEnd)
71         //4 - Faltou informar o nome (FrontEnd)
72         //5 - Faltou informar o tipo de usuário (FrontEnd)
73         //6 - Houve algum problema no insert da tabela (Banco)
74         //7 - Usuario do sistema não informado (Front)
75         //8 - Houve problema no salvamento do Log, mas o usuario foi inserido
76
77     try{
78         //Usuário e senha recebidos via JSON
79         //e colocados em atributos
80         $json = file_get_contents('php://input');
81         $resultado = json_decode($json);
82
83         //Array com os dados que deverão vir do Front
84         $lista = array(
85             "usuario" => '0',
86             "senha" => '0',
87             "nome" => '0',
88             "tipo_usuario" => '0',
89             "usuarioLogin" => '0'
90         );
91
92         if (verificarParam($resultado, $lista) == 1) {
93             //Fazendo os setters
94             $this->setUsuario($resultado->usuario);
95             $this->setSenha($resultado->senha);
96             $this->setNome($resultado->nome);
97             $this->setTipoUsuario(strtoupper($resultado->tipo_usuario));
98             $this->setUsuarioLogin($resultado->usuarioLogin);
99
100            //Faremos uma validação para sabermos se todos os dados
101            //foram enviados
102            if (trim($this->getUsuario()) == ''){
103                $retorno = array('codigo' => 2,
104                                'msg' => 'Usuário não informado');
105            }elseif (trim($this->getSenha()) == ''){
106                $retorno = array('codigo' => 3,
107                                'msg' => 'Senha não informada');
108            }elseif (trim($this->getNome()) == ''){
109                $retorno = array('codigo' => 4,
110                                'msg' => 'Nome não informado');
111        }
112    }
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
150
```

```

111     }elseif ((trim($this->getTipoUsuario()) != 'ADMINISTRADOR' &&
112             trim($this->getTipoUsuario()) != 'COMUM' ) ||
113             trim($this->getTipoUsuario()) == '') {
114         $retorno = array('codigo' => 5,
115                         'msg' => 'Tipo de usuário inválido');
116     }else if (trim($this->getUsuarioLogin()) == ''){
117         $retorno = array('codigo' => 7,
118                         'msg' => 'Usuário Logado no sistema não informado');
119     }else{
120         //Realizo a instância da Model
121         $this->load->model('M_usuario');
122
123         //Atributo $retorno recebe array com informações
124         //da validação do acesso
125         $retorno = $this->M_usuario->inserir($this->getUsuario(), $this->getSenha(),
126                                         $this->getNome(), $this->getTipoUsuario(),
127                                         $this->getUsuarioLogin());
128     }
129 }else {
130     $retorno = array(
131         'codigo' => 99,
132         'msg' => 'Os campos vindos do FrontEnd não representam
133                     o método de login. Verifique.'
134     );
135 }
136
137 }catch (Exception $e) {
138     $retorno = array('codigo' => 0,
139                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
140                     $e->getMessage());
141 }
142
143 //Retorno no formato JSON
144 echo json_encode($retorno);
145 }
```

No código acima temos validações em relação as informações enviadas pelo *frontend*, isso se fez necessário para que não enviamos variáveis sem conteúdo ou com dados que fujam a regra de nosso sistema, as regras são:

- Os dados de usuário, senha, nome não poderão estar vazios;
- O tipo de usuário deverá ser “ADMINISTRADOR” ou “COMUM”.

Mas os dados de nome e tipo de usuários não estão na tabela de usuários, correto?

Sim, para termos esses dados em nossa tabela, teremos que atualizar nosso banco de dados, assim:

```

#Alterando a tabela de usuários de acordo novas necessidades,
#inserindo os campos nome e tipo
alter table usuarios add column nome varchar(30) default '' after senha,
                    add column tipo varchar(20) default '' after estatus;
```

Com isso nossa tabela estará em conformidade para receber os dados vindos do *frontend*.

Agora que precisamos criar a *MODEL* que receberá os dados para inserção na tabela usuários, para isso precisamos criar a M_usuario.

```
1 <?php
2 defined('BASEPATH') or exit('No direct script access allowed');
3
4 class M_usuario extends CI_Model {
5     public function inserir($usuario, $senha, $nome, $tipo_usuario){
6         try{
7             //Query de inserção dos dados
8             $this->db->query("insert into usuarios (usuario, senha, nome, tipo)
9                                values ('$usuario', md5('$senha'), '$nome', '$tipo_usuario')");
10
11            //Verificar se a inserção ocorreu com sucesso
12            if($this->db->affected_rows() > 0){
13                $dados = array('codigo' => 1,
14                               'msg' => 'Usuário cadastrado corretamente');
15
16            }else{
17                $dados = array('codigo' => 6,
18                               'msg' => 'Houve algum problema na inserção na tabela de usuários');
19            }
20        } catch (Exception $e) {
21            $dados = array('codigo' => 00,
22                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
23                           $e->getMessage(), "\n");
24        }
25        //Envia o array $dados com as informações tratadas
26        //acima pela estrutura de decisão if
27        return $dados;
28    }
29
30    public function consultar($usuario, $nome, $tipo_usuario){
31        //-----
32        //Função que servirá para quatro tipos de consulta:
33        // * Para todos os usuários;
34        // * Para um determinado usuário;
35        // * Para um tipo de usuário;
36        // * Para nomes de usuários;
37        //-----
38
39        try{
40            //Query para consultar dados de acordo com parâmetros passados
41            $sql = "select * from usuarios where estatus = '' ";
42
43            if(trim($usuario) != ''){
44                $sql = $sql . "and usuario = '$usuario' ";
45            }
46
47            if(trim($tipo_usuario) != ''){
48                $sql = $sql . "and tipo = '$tipo_usuario' ";
49            }
50
51            if(trim($nome) != ''){
52                $sql = $sql . "and nome like '%$nome%' ";
53            }
54
55            $retorno = $this->db->query($sql);
56
57            //Verificar se a consulta ocorreu com sucesso
58            if($retorno->num_rows() > 0){
59                $dados = array('codigo' => 1,
60                               'msg' => 'Consulta efetuada com sucesso',
61                               'dados' => $retorno->result());
62
63            }else{
64                $dados = array('codigo' => 6,
65                               'msg' => 'Dados não encontrados');
66            }
67        }
```

```

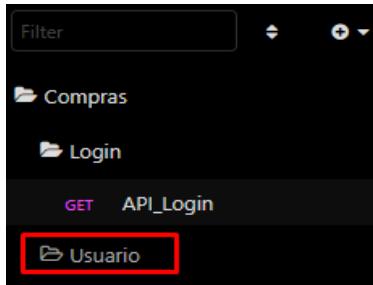
67     }catch (Exception $e) {
68         $dados = array('codigo' => 00,
69                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
70                     '$e->getMessage(), "\n");
71     }
72     //Envia o array $dados com as informações tratadas
73     //acima pela estrutura de decisão if
74     return $dados;
75 }
76
77 public function alterar($usuario, $nome, $senha, $tipo_usuario){
78     try{
79         //Query de atualização dos dados
80         $this->db->query("update usuarios set nome = '$nome', senha = md5('$senha'),
81                           nome = '$nome', tipo = '$tipo_usuario'
82                           where usuario = '$usuario'");
83
84         //Verificar se a atualização ocorreu com sucesso
85         if($this->db->affected_rows() > 0){
86             $dados = array('codigo' => 1,
87                           'msg' => 'Usuário atualizado corretamente');
88
89         }else{
90             $dados = array('codigo' => 6,
91                           'msg' => 'Houve algum problema na atualização na tabela de usuários');
92         }
93     }catch (Exception $e) {
94         $dados = array('codigo' => 00,
95                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
96                     '$e->getMessage(), "\n");
97     }
98     //Envia o array $dados com as informações tratadas
99     //acima pela estrutura de decisão if
100    return $dados;
101 }
102
103 public function desativar($usuario){
104     try{
105         //Query de atualização dos dados
106         $this->db->query("update usuarios set estatus = 'D'
107                           where usuario = '$usuario'");
108
109         //Verificar se a atualização ocorreu com sucesso
110         if($this->db->affected_rows() > 0){
111             $dados = array('codigo' => 1,
112                           'msg' => 'Usuário DESATIVADO corretamente');
113
114         }else{
115             $dados = array('codigo' => 6,
116                           'msg' => 'Houve algum problema na DESATIVAÇÃO do usuário');
117         }
118     }catch (Exception $e) {
119         $dados = array('codigo' => 00,
120                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
121                     '$e->getMessage(), "\n");
122     }
123     //Envia o array $dados com as informações tratadas
124     //acima pela estrutura de decisão if
125     return $dados;
126 }
127 }
```

Com essa parte do *backend* pronta, iremos colocar esse método no *INSOMNIA* para fazermos as validações necessárias.

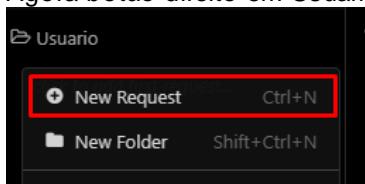
Vamos criar uma pasta dentro do projeto Compras para armazenar as *APIs* de Usuário.



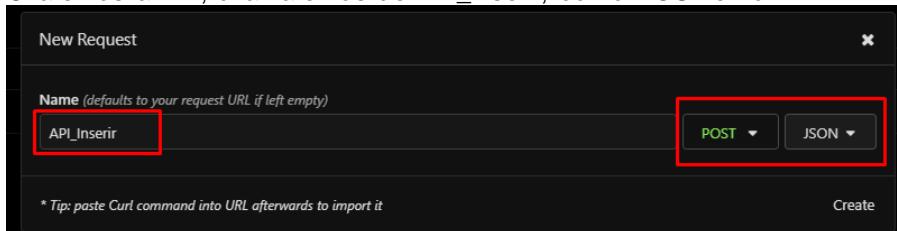
A estrutura ficará assim:



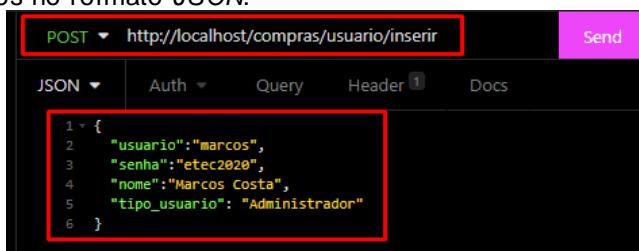
Agora botão direito em Usuario e New Request.



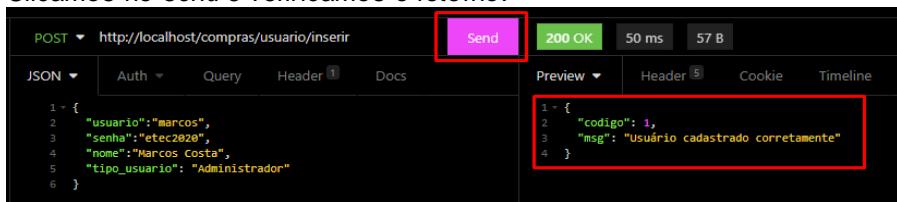
Criaremos a API, chamaremos de API_Inserir, como POST e Form



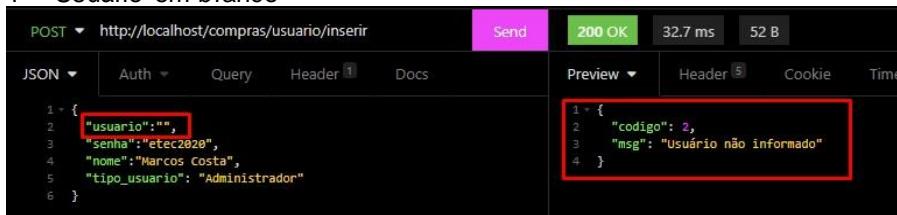
Agora informamos o caminho da controller que contém o método de inserção e atribuímos os dados no formato JSON.



Clicamos no send e verificamos o retorno.



Fazendo as validações que programamos na controller de usuários
1º - Usuário em branco



2º - Senha em branco

```
POST http://localhost/compras/usuario/inserir
{
  "usuario": "marcos",
  "senha": "",
  "nome": "Marcos Costa",
  "tipo_usuario": "Administrador"
}
```

200 OK | 39.3 ms | 45 B

```
1 + {
  2   "codigo": 3,
  3   "msg": "Senha não informada"
  4 }
```

3º - Nome em branco

```
POST http://localhost/compras/usuario/inserir
{
  "usuario": "marcos",
  "senha": "etec2020",
  "nome": "",
  "tipo_usuario": "Administrador"
}
```

200 OK | 25.1 ms | 44 B

```
1 + {
  2   "codigo": 4,
  3   "msg": "Nome não informado"
  4 }
```

4º - Tipo de usuário inválido ou em branco

```
POST http://localhost/compras/usuario/inserir
{
  "usuario": "marcos",
  "senha": "etec2020",
  "nome": "Marcos Costa",
  "tipo_usuario": ""
}
```

200 OK | 48.1 ms | 55 B

```
1 + {
  2   "codigo": 5,
  3   "msg": "Tipo de usuário inválido"
  4 }
```

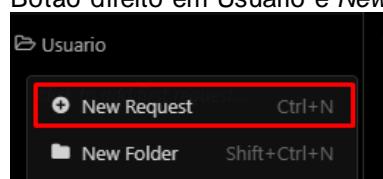
```
POST http://localhost/compras/usuario/inserir
{
  "usuario": "marcos",
  "senha": "etec2020",
  "nome": "Marcos Costa",
  "tipo_usuario": "Chefe"
}
```

200 OK | 34.1 ms | 55 B

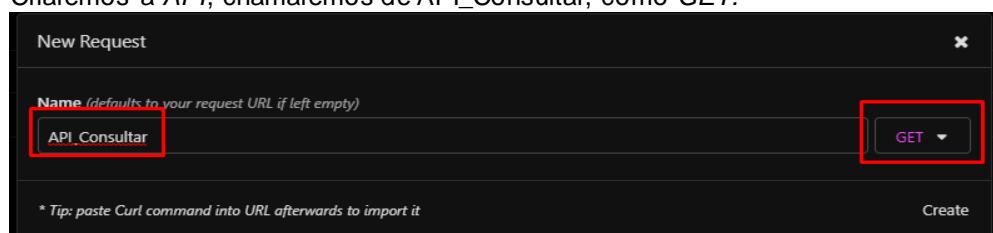
```
1 + {
  2   "codigo": 5,
  3   "msg": "Tipo de usuário inválido"
  4 }
```

Agora a API de Consulta:

Botão direito em Usuário e New Request.



Criaremos a API, chamaremos de API_Consultar, como GET.



Agora informamos o caminho da controller que contém o método de consulta e atribuímos o formato JSON.

```
GET ▼ http://localhost/compras/usuario/consultar
Send
```

JSON

```
1: {
  "usuario": "",
  "nome": "",
  "tipo_usuario": ""
}
```

Agora podemos ver as consultas sendo:

1º - Sem nenhum parâmetro, traz todos os registros da tabela de usuários:

```
GET ▼ http://localhost/compras/usuario/consultar
Send 200 OK 61 ms 374 B
```

JSON

```
1: {
  "usuario": "",
  "nome": "",
  "tipo_usuario": ""
}
```

Preview

```
1: {
  "codigo": 1,
  "msg": "Consulta efetuada com sucesso",
  "dados": [
    {
      "id_usuario": "1",
      "usuario": "admin",
      "senha": "0192023a7bbd73250516f069df18b500",
      "dcria": "2021-08-26 11:45:14",
      "estatus": "",
      "nome": "",
      "tipo": ""
    },
    {
      "id_usuario": "2",
      "usuario": "marcos",
      "senha": "f31dfa5c8efabaf683ae027f184857bc",
      "dcria": "2021-09-08 12:50:38",
      "estatus": "",
      "nome": "Marcos Costa",
      "tipo": "Administrador"
    }
  ]
}
```

2º - Somente o usuário:

```
GET ▼ http://localhost/compras/usuario/consultar
Send 200 OK 25.1 ms 230 B
```

JSON

```
1: {
  "usuario": "marcos",
  "nome": ,
  "tipo_usuario": ""
}
```

Preview

```
1: {
  "codigo": 1,
  "msg": "Consulta efetuada com sucesso",
  "dados": [
    {
      "id_usuario": "2",
      "usuario": "marcos",
      "senha": "f31dfa5c8efabaf683ae027f184857bc",
      "dcria": "2021-09-08 12:50:38",
      "estatus": "",
      "nome": "Marcos Costa",
      "tipo": "Administrador"
    }
  ]
}
```

3º - Somente o nome:

```
GET ▼ http://localhost/compras/usuario/consultar
Send 200 OK 47.1 ms 230 B
```

JSON

```
1: {
  "usuario": "",
  "nome": "Marcos Costa",
  "tipo_usuario": ""
}
```

Preview

```
1: {
  "codigo": 1,
  "msg": "Consulta efetuada com sucesso",
  "dados": [
    {
      "id_usuario": "2",
      "usuario": "marcos",
      "senha": "f31dfa5c8efabaf683ae027f184857bc",
      "dcria": "2021-09-08 12:50:38",
      "estatus": ,
      "nome": "Marcos Costa",
      "tipo": "Administrador"
    }
  ]
}
```

4º - Somente o tipo de usuário:

GET http://localhost/compras/usuario/consultar

200 OK

```

1+ {
2 "usuario":"",
3 "nome":"",
4 "tipo_usuario":"Administrador"
5 }
6
7

```

```

1+ {
2 "codigo": 1,
3 "msg": "Consulta efetuada com sucesso",
4+ "dados": [
5+
6 {
7 "id_usuario": "2",
8 "usuario": "marcos",
9 "senha": "f31dfa5c8efabaf683ae027f184857bc",
10 "dtcria": "2021-09-08 12:50:38",
11 "estatus": "",
12 "nome": "Marcos Costa",
13 "tipo": "Administrador"
14 ]
15 }

```

Caso o tipo seja diferente de vazio, administrador ou comum, também verifica:

GET http://localhost/compras/usuario/consultar

200 OK

```

1+ {
2 "usuario":"",
3 "nome":"",
4 "tipo_usuario":"Master"
5 }
6

```

```

1+ {
2 "codigo": 5,
3 "msg": "Tipo de usuário inválido"
4 }

```

5º - As combinações também trazem resultados de pesquisa.

GET http://localhost/compras/usuario/consultar

200 OK

```

1+ {
2 "usuario":"marcos",
3 "nome":"",
4 "tipo_usuario":"Administrador"
5 }
6
7

```

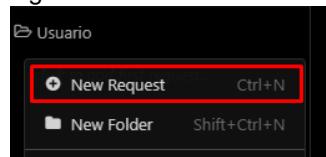
```

1+ {
2 "codigo": 1,
3 "msg": "Consulta efetuada com sucesso",
4+ "dados": [
5+
6 {
7 "id_usuario": "2",
8 "usuario": "marcos",
9 "senha": "f31dfa5c8efabaf683ae027f184857bc",
10 "dtcria": "2021-09-08 12:50:38",
11 "estatus": "",
12 "nome": "Marcos Costa",
13 "tipo": "Administrador"
14 ]
15 }

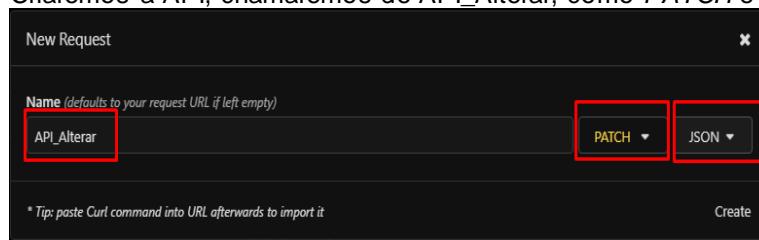
```

Agora a API de Alteração

Agora botão direito em Usuário e New Request.



Criaremos a API, chamaremos de API_Alterar, como PATCH e JSON.



Agora informamos o caminho da controller que contém o método de consulta e atribuímos as variáveis.

```

PATCH ▾ http://localhost/compras/usuario/alterar
Send

JSON ▾ Auth ▾ Query Header 1 Docs
1 + {
2   "usuario": "marcos",
3   "nome": "Marcos Costa",
4   "senha": "admin123",
5   "tipo_usuario": "Comum"
6 }
7
  
```

Agora podemos ver as alterações sendo:

1º - Passando os parâmetros:

```

PATCH ▾ http://localhost/compras/usuario/alterar
Send 200 OK 44.8 ms 57 B

JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1 + {
2   "usuario": "marcos",
3   "nome": "Marcos Costa",
4   "senha": "admin123",
5   "tipo_usuario": "Comum"
6 }

1 + {
2   "codigo": 1,
3   "msg": "Usuário atualizado corretamente"
4 }
  
```

2º - Críticas para que a atualização aconteça:

- Usuário precisa ser informado:

```

PATCH ▾ http://localhost/compras/usuario/alterar
Send 200 OK 22.9 ms 52 B

JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie
1 + {
2   "usuario": "",
3   "nome": "Marcos Costa",
4   "senha": "admin123",
5   "tipo_usuario": "Comum"
6 }

1 + {
2   "codigo": 2,
3   "msg": "Usuário não informado"
4 }
  
```

- Nome precisa ser informado:

```

PATCH ▾ http://localhost/compras/usuario/alterar
Send 200 OK 26.1 ms 44 B

JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1 + {
2   "usuario": "marcos",
3   "nome": "",
4   "senha": "admin123",
5   "tipo_usuario": "Comum"
6 }

1 + {
2   "codigo": 3,
3   "msg": "Nome não informado"
4 }
  
```

- Senha precisa ser informada:

```

PATCH ▾ http://localhost/compras/usuario/alterar
Send 200 OK 45.6 ms 52 B

JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1 + {
2   "usuario": "marcos",
3   "nome": "Marcos Costa",
4   "senha": "",
5   "tipo_usuario": "Comum"
6 }

1 + {
2   "codigo": 4,
3   "msg": "Senha não pode estar vazia"
4 }
  
```

- Tipo de usuário precisa estar como Administrador, Comum ou vazio:

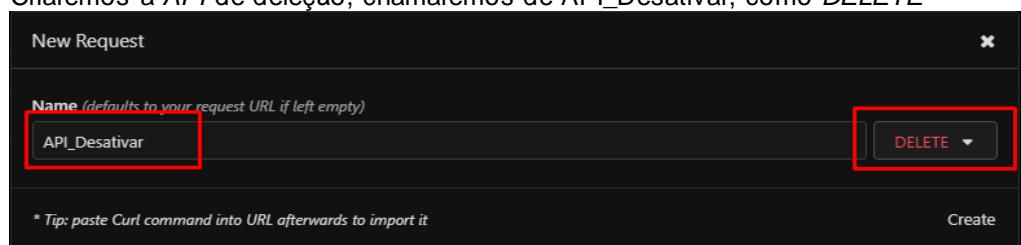
```

PATCH ▾ http://localhost/compras/usuario/alterar
Send 200 OK 55.4 ms 55 B

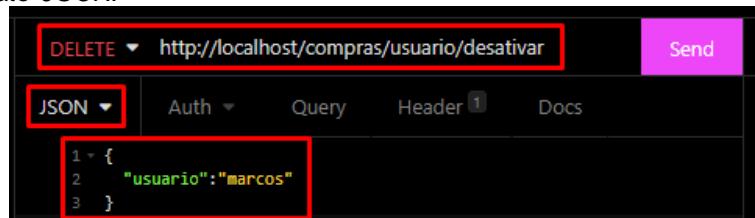
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1 + {
2   "usuario": "marcos",
3   "nome": "Marcos Costa",
4   "senha": "admin123",
5   "tipo_usuario": "Chefia"
6 }

1 + {
2   "codigo": 5,
3   "msg": "Tipo de usuário inválido"
4 }
  
```

Criaremos a API de deleção, chamaremos de API_Desativar, como *DELETE*

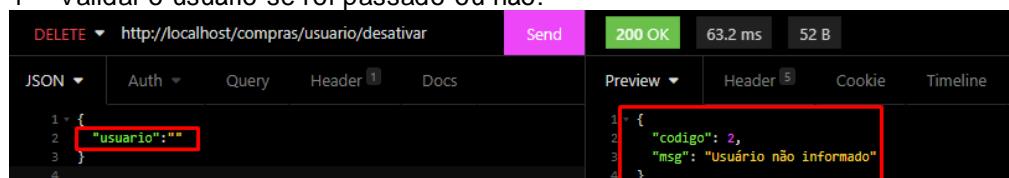


Agora informamos o caminho da controller que contém o método de desativar e atribuímos o formato JSON.

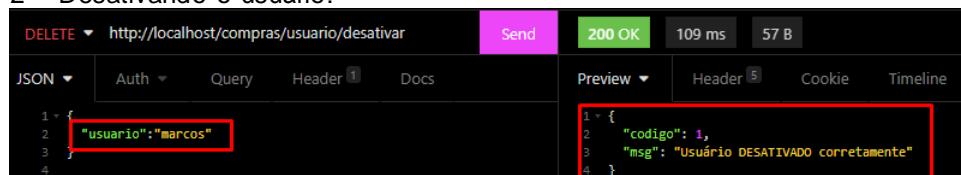


Agora podemos ver a desativação, mas antes:

1º - Validar o usuário se foi passado ou não:



2º - Desativando o usuário:

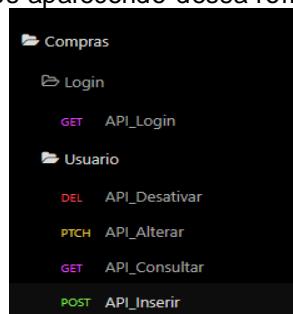


Com isso o usuário recebe no campo estatus o caractere "D". Vejam:

1 • `select * from usuarios`

	id_usuario	usuario	senha	dtcria	estatus	nome	tipo
▶	1	admin	0192023a7bbd73250516f069df18b500	2021-08-26 11:45:14			
▶	2	marcos	0192023a7bbd73250516f069df18b500	2021-09-08 12:50:38	D	Marcos Costa	Comum

Com isso, finalizamos as operações básicas de um cadastro, no INSOMNIA ficará com os métodos aparecendo dessa forma:



Agora é com vocês, implementem essas funcionalidades no sistema de vocês para que possamos dar continuidade ao conteúdo.

2.7. Melhoria e atividade prática para entrega, nossa P1.

Iremos agir agora como um líder técnico que precisará que nosso time de desenvolvedores atualizem o sistema desenvolvido até o momento, assim teremos as seguintes tarefas a considerar:

1º - No método **validaLogin()**, na classe **M_acesso**, precisamos que retorne o *status* do usuário, ou seja, se ele estiver deletado (logicamente), deveremos informar esta parte;

2º - No método **alterar()**, na classe **Usuario** e **M_Usuario** verifiquem as atualizações necessárias, para que só atualize os dados que o usuário pedir, um exemplo, quer atualizar o nome, não precisa ser obrigatório passar os outros campos, lembrando que hoje, nosso projeto "trava" se não passarmos todos os atributos.

3º - No método **desativar()**, na **M_usuario** façam uma verificação antes de efetivar a ação, se este usuário já não se encontra desativado, caso esteja, informe a situação.

4º - No mesmo método, não permitam, que um usuário do tipo "**Comum**" desative usuário, só "**Administradores**" poderão realizar tal ação.

5º - Crie no **Insomnia**, todos os **endpoints** necessários, conforme notas de aula.

Realizem as alterações, irei verificar de forma individual, ou seja, vocês terão que me relatar a solução adotada por vocês, prazo máximo para esta atividade será dia **03/09/2024** e lembrando, **será nossa P1**.

2.7.1 Correção da atividade prática, nossa P1.

Abaixo iremos representar uma solução que atenderia aos requisitos pedidos na avaliação, vale ressaltar, que poderemos ter outras formas de representar e resolver os problemas propostos, está somente é uma delas.

2.7.1.1 Correção do 1º exercício.

Neste primeiro exercício, o desafio era modificar algo que já está desenvolvido e funcionando corretamente, esta é uma das práticas que abordamos para mostrar ao aluno, situações reais em um time de desenvolvimento de software.

Para isso precisamos modificar o método **validaLogin()** na model **M_acesso**, podemos observar no código a seguir, que primeiramente a query não tem o filtro do **estatus** do usuário **(1)**, nós selecionamos um usuário e senha válida, somente isso, após isso, verificamos o **estatus** isoladamente **(2)**, e aí sim, fazemos os direcionamentos necessários para o front, sendo, retorno 4 – Usuário ou senha inválidos **(3)**, retorno 5 – Usuário desativado na base de dados **(4)** e retorno 1 – Usuário correto **(5)**, ou seja, usuário e senha validados corretamente em nosso banco de dados, veja a seguir:

```

5   public function validaLogin($usuario, $senha){
6       try{
7           //Atributo retorno recebe o resultado do SELECT
8           //realizado na tabela de usuários lembrando da função MD5()
9           //por causa da criptografia, e sem status pois teremos que validar
10          //para verificar se está deletado virtualmente ou não.
11          $retorno = $this->db->query("select * from usuarios
12                                     where usuario = '$usuario'
13                                     and senha = md5('$senha')");
14
15          //Verifica se a quantidade de linhas trazidas na consulta é superior a 0,
16          //Vinculamos o resultado da query para tratarmos o resultado do status
17          $linha = $retorno->row();
18

```

```

19 //Criado um array com dois elementos para retorno do resultado
20 //1 - Código da mensagem
21 //2 - Descrição da mensagem
22 if($retorno->num_rows() == 0){
23     $dados = array('codigo' => 4,
24                   'msg' => 'Usuário ou senha inválidos.');
25 }else{
26     if(trim($linha->estatus) == "D"){
27         $dados = array('codigo' => 5,
28                       'msg' => 'Usuário DESATIVADO NA BASE DE DADOS!');
29     }else{
30         $dados = array('codigo' => 1,
31                       'msg' => 'Usuário correto');
32     }
33 }
34 } catch (Exception $e) {
35     $dados = array('codigo' => 00,
36                   'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
37                   $e->getMessage(), "\n");
38 }
39
40 return $dados;
41 }
```

E submetendo novamente o **endpoint** de logar no **Insomnia**, ficamos assim:

GET ▾ http://localhost/compras/usuario/logar Send 200 OK 55.3 ms 43 B A Minute Ago

JSON	Auth	Query	Header 1	Docs	Preview	Header 6	Cookie	Timeline
1 { 2 "usuario": "admin", 3 "senha": "admin123" 4 } 5					1 { 2 "codigo": 1, 3 "msg": "Usuário correto" 4 }			

Com este mesmo usuário com o **estatus** de desativado na base de dados:

	id_usuario	usuario	senha	dtria	estatus	nome	tipo
▶	1	admin	0192023a7bbd73250516f069df18b500	2024-08-06 11:39:21	D	Marcos Costa	ADMINISTRADOR
▶	2	arthur	827ccb0eea8a706c4c34a16891f84e7b	2024-09-08 17:18:38	A	Arthur Prado Costa	COMUM

GET ▾ http://localhost/compras/usuario/logar Send 200 OK 31.5 ms 64 B Just Now

JSON	Auth	Query	Header 1	Docs	Preview	Header 6	Cookie	Timeline
1 { 2 "usuario": "admin", 3 "senha": "admin123" 4 } 5					1 { 2 "codigo": 5, 3 "msg": "Usuário DESATIVADO NA BASE DE DADOS!" 4 }			

Com o usuário com **estatus** ativo:

	id_usuario	usuario	senha	dtria	estatus	nome	tipo
▶	1	admin	0192023a7bbd73250516f069df18b500	2024-08-06 11:39:21	A	Marcos Costa	ADMINISTRADOR
▶	2	arthur	827ccb0eea8a706c4c34a16891f84e7b	2024-09-08 17:18:38	A	Arthur Prado Costa	COMUM

GET ▾	http://localhost/compras/usuario/logar	Send	200 OK	51.6 ms	43 B	Just Now ▾
JSON ▾	Auth ▾	Query	Header 1	Docs	Preview ▾	Header 6
1 <code>+</code> { 2 "usuario": "admin", 3 "senha": "admin123" 4 } 5					1 <code>+</code> { 2 "codigo": 1, 3 "msg": "Usuário correto" 4 }	

Assim finalizamos o primeiro exercício.

2.7.1.2 Correção do 2º exercício.

Com certeza, este era o exercício que demandava mais análise e trabalho para nós, precisaremos fazer algumas atividades nele. Primeiramente na **Controller** Usario modificamos a mesma, onde agora, precisamos validar que o usuário somente é obrigatório sua passagem, quanto aos outros elementos, como nome, senha, tipo de usuário, seria facultativo, mas, pelo menos um deles precisariam ser passado, onde satisfaz o enunciado do exercício passado. Vejam o código:

```

181 public function alterar(){
182     //Usuário, nome, senha e tipo (Administrador ou Comum)
183     //recebidos via JSON e colocados
184     //em variáveis
185     //Retornos possíveis:
186     //1 - Dado(s) alterado(s) corretamente (Banco)
187     //2 - Usuario em Branco ou Zerado
188     //3 - Nenhum parâmetro de alteração informado.
189     //4 - Tipo de usuário inválido (FrontEnd)
190     //5 - Dados não encontrados (Banco)
191
192     try{
193         $json = file_get_contents('php://input');
194         $resultado = json_decode($json);
195
196         //Array com os dados que deverão vir do Front
197         $lista = array(
198             "usuario" => '0',
199             "senha" => '0',
200             "nome" => '0',
201             "tipo_usuario" => '0'
202         );
203
204         if (verificarParam($resultado, $lista) == 1) {
205             //Fazendo os setters
206             $this->setUsuario($resultado->usuario);
207             $this->setSenha($resultado->senha);
208             $this->setNome($resultado->nome);
209             $this->setTipoUsuario(strtoupper($resultado->tipo_usuario));
210
211             //Validação para tipo de usuário que deverá ser ADMINISTRADOR, COMUM ou VAZIO
212             if (trim($this->getTipoUsuario()) != 'ADMINISTRADOR' &&
213                 trim($this->getTipoUsuario()) != 'COMUM' &&
214                 trim($this->getTipoUsuario()) != '') {
215
216                 $retorno = array('codigo' => 4,
217                                 'msg' => 'Tipo de usuário inválido');
218
219             }elseif(trim($this->getUsuario()) == ''){
220                 $retorno = array('codigo' => 2,
221                                 'msg' => 'Usuário não informado');
222
223             //Nome, Senha ou Tipo de Usuário, pelo menos 1 deles precisa ser informado.
224             }elseif(trim($this->getNome()) == '' & trim($this->getSenha()) == ''
225                     && $this->getTipoUsuario() == ''){
226                 $retorno = array('codigo' => 3,
227                                 'msg' => 'Pelo menos um parâmetro precisa ser passado
228                                         para atualização');
229             }else{
230                 //Realizo a instância da Model
231                 $this->load->model('M_usuario');
```

```
231 //Atributo $retorno recebe array com informações  
232 //da alteração dos dados  
233 $retorno = $this->M_usuario->alterar($this->getUsuario(), $this->getNome(),  
234 //                                            $this->getSenha(), $this->getTipoUsuario());  
235  
236     }  
237 }else {  
238     $retorno = array(  
239         'codigo' => 99,  
240         'msg' => 'Os campos vindos do FrontEnd não representam  
241             o método de login. Verifique.'  
242     );  
243 }  
244 }catch (Exception $e) {  
245     $retorno = array('codigo' => 0,  
246                      'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',  
247                      $e->getMessage());  
248 }  
249 //Retorno no formato JSON  
250 echo json_encode($retorno);  
251 }
```

Após isso, temos que modificar a **Model M_usuario**, vejam o código a seguir:

```
77 public function alterar($usuario, $nome, $senha, $tipo_usuario){  
78  
79     //Início a query para atualização  
80     $query = "update usuarios set ";  
81  
82     try{  
83         //Vamos comparar os items  
84         if($nome !== ''){  
85             $query .= "nome = '$nome', ";  
86         }  
87  
88         if($senha !== ''){  
89             $query .= "senha = md5('$senha'), ";  
90         }  
91  
92         if($tipo_usuario !== ''){  
93             $query .= "tipo = '$tipo_usuario', ";  
94         }  
95  
96         //Termino a concatenação da query  
97         $queryFinal = rtrim($query, ", ") . " where usuario = '$usuario'";  
98  
99         //Executo a Query de atualização dos dados  
100        $this->db->query($queryFinal);  
101  
102        //Verificar se a atualização ocorreu com sucesso  
103        if($this->db->affected_rows() > 0){  
104            $dados = array('codigo' => 1,  
105                           'msg' => 'Usuário atualizado corretamente');  
106  
107        }else{  
108            $dados = array('codigo' => 6,  
109                           'msg' => 'Houve algum problema na atualização na tabela de usuários');  
110        }  
111    }catch (Exception $e) {  
112        $dados = array('codigo' => 00,  
113                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',  
114                           $e->getMessage(), "\n");  
115    }  
116    //Envia o array $dados com as informações tratadas  
117    //acima pela estrutura de decisão if  
118    return $dados;  
119}
```

E submetendo novamente o **endpoint** de **alterar** no **Insomnia**, ficamos assim:

```

PATCH ▾ http://localhost/compras/usuario/alterar Send 200 OK 36 ms 98 B A Minute Ago ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "usuario": "admin",
3   "nome": "",
4   "senha": "",
5   "tipo_usuario": ""
6 }
    
```

Tentando passar somente o usuário, mas sem parâmetros seguintes, como relatamos no início.

```

PATCH ▾ http://localhost/compras/usuario/alterar Send 200 OK 30.3 ms 54 B Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "usuario": "",
3   "nome": "",
4   "senha": "",
5   "tipo_usuario": ""
6 }
    
```

Usuário não informado.

Agora vamos mostrar as alterações acontecendo de acordo com os dados abaixo representados em nosso banco de dados:

id_usuario	usuario	senha	dtcria	estatus	nome	tipo
1	admin	0192023a7bbd73250516f069df18b500	2024-08-06 11:39:21		Marcos Costa	ADMINISTRADOR
2	arthur	827ccb0eea8a706c4c34a16891f84e7b	2024-09-08 17:18:38		Arthur Prado Costa	COMUM

```

PATCH ▾ http://localhost/compras/usuario/alterar Send 200 OK 43.9 ms 59 B Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "usuario": "admin",
3   "nome": "Marcos Costa de Sousa", ←
4   "senha": "",
5   "tipo_usuario": ""
6 }
    
```

Alterando o nome do usuário.

id_usuario	usuario	senha	dtcria	estatus	nome	tipo
1	admin	0192023a7bbd73250516f069df18b500	2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
2	arthur	827ccb0eea8a706c4c34a16891f84e7b	2024-09-08 17:18:38		Arthur Prado Costa	COMUM

Resultado na base de dados.

```

PATCH ▾ http://localhost/compras/usuario/alterar Send 200 OK 54.4 ms 59 B A Minute Ago ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "usuario": "admin",
3   "nome": "",
4   "senha": "123admin", ←
5   "tipo_usuario": ""
6 }
    
```

Alterando a senha.

id_usuario	usuario	senha	dcria	estatus	nome	tipo
1	admin	d829b843a6550a947e82f2f38ed6b7a7	2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
2	arthur	827ccb0eea8a706c4c34a16891f84e7b	2024-09-08 17:18:38		Arthur Prado Costa	COMUM

Percebam que a senha está diferente da anterior, lembrando que estamos utilizando a criptografia neste caso.

```

PATCH http://localhost/compras/usuario/alterar
Send
200 OK 32.4 ms 59 B Just Now
JSON Auth Query Header 1 Docs Preview Header 6 Cookie Timeline
1 {
2   "usuario": "admin",
3   "nome": "",
4   "senha": "",
5   "tipo_usuario": "Comum"
6 }
1 {
2   "codigo": 1,
3   "msg": "Usuário atualizado corretamente"
4 }

```

Mudando o tipo de usuário.

	id_usuario	usuario	senha	dcria	estatus	nome	tipo
1	1	admin	d829b843a6550a947e82f2f38ed6b7a7	2024-08-06 11:39:21		Marcos Costa de Sousa	COMUM

Atualização aplicada na base de dados no usuário respectivo.

Assim finalizamos o segundo exercício.

2.7.1.3 Correção do 3º e 4º exercícios.

Apesar de não serem os exercícios mais complexos, a solução dos mesmos acaba nos fazendo em mexer nos mesmos objetos, por isso, farei a abordagem toda de uma vez.

Os dois exercícios pedem que trabalhemos com a parte de desativação de nosso usuário, uma parte para que validemos se o usuário já está desativado na base de dados, e outra parte, exigindo que somente administradores consigam fazer tal atividade.

Vamos primeiramente, tratar a Controller Usario, onde teremos que receber mais um parâmetro do **Front End** ou **Insomnia**, que é o Usuario Login de nosso sistema, e claro, realizar as tratativas onde os mesmo são passados, mas antes de qualquer método a realizar, precisamos encapsular o **UsuarioLogin** que será mais um atributo que receberemos.

```

4 class Usuario extends CI_Controller {
5   //Atributos privados da classe
6   private $usuario;
7   private $senha;
8   private $nome;
9   private $tipoUsuario;
10  private $usuarioLogin;
11

```

Agora fazendo o **getter** e o **setter** do mesmo.

```

33 public function getUsuarioLogin()
34 {
35   return $this->usuarioLogin;
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59 public function setUsuarioLogin($usuarioLoginFront)
60 {
61   $this->usuarioLogin = $usuarioLoginFront;
62 }

```

Agora sim, temos o atributo declarado em nossa classe de **Usuario**.

O próximo passo é modificar o método **desativar()** já existente conforme a seguir, verificaremos na *helper* o elemento adicional, trataremos se ele foi realmente passado e teremos que modificar na chamada do método **desativar()** da **Model**, passando mais este parâmetro, vejam a seguir:

```

252     public function desativar(){
253         //Usuário recebido via JSON e colocado em variável
254         //Retornos possíveis:
255         //1 - Usuário desativado corretamente (Banco)
256         //2 - Usuário em Branco
257         //3 - Usuário que pediu a desativação não é administrador
258         //4 - Usuário inexistente na base de dados
259         //5 - Usuário já desativado na base de dados
260         //6 - Dados não encontrados (Banco)
261     try{
262         $json = file_get_contents('php://input');
263         $resultado = json_decode($json);
264
265         //Array com os dados que deverão vir do Front
266         $lista = array(
267             "usuario" => '0',
268             "usuarioLogin" => '0' ←
269         );
270
271         if (verificarParam($resultado, $lista) == 1) {
272
273             $json = file_get_contents('php://input');
274             $resultado = json_decode($json);
275
276             //Fazendo os setters
277             $this->setUsuario($resultado->usuario);
278             $this->setUsuarioLogin($resultado->usuarioLogin);
279
280             //Validação para do usuário que não deverá ser branco
281             if (trim($this->getUsuario()) == ''){
282                 $retorno = array('codigo' => 2,
283                                 'msg' => 'Usuário não informado');
284             }else if (trim($this->getUsuarioLogin()) == ''){
285                 $retorno = array('codigo' => 2,
286                                 'msg' => 'Usuário Logado no sistema não informado');
287             }else{
288                 //Realizo a instância da Model
289                 $this->load->model('M_usuario');
290
291                 //Atributo $retorno recebe array com informações
292                 $retorno = $this->M_usuario->desativar($this->getUsuario(),
293                                                 $this->getUsuarioLogin());
294             }
295         }else {
296             $retorno = array(
297                 'codigo' => 99,
298                 'msg' => 'Os campos vindos do FrontEnd não representam
299                               o método de login. Verifique.'
300             );
301         }
302     } catch (Exception $e) {
303         $retorno = array('codigo' => 0,
304                         'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
305                         '$e->getMessage()');
306     }
307
308     //Retorno no formato JSON
309     echo json_encode($retorno);
310 }

```

Agora vamos para a model **M_usuario**, no método **desativar()** estamos recebendo dois parâmetros, sendo eles, o usuário a ser desativado e o usuário que estaria desativando, e este precisa ser um administrador de nosso sistema, para este caso, pensando em otimização de código, aplicando conceitos de orientação a objetos e boas práticas, vamos criar dois métodos **privados** da classe, um para validar se nosso usuário é um administrador e outro para verificar se o usuário está desativado ou não cadastrado no sistema, no final irei falar mais sobre esta última parte.

Vamos primeiramente criar estes dois métodos auxiliares, depois programamos o método principal, o **desativar()** em si.

```
162     private function verificaAdmin($usuario){
163         try{
164             //Verificação se o usuário passado é administrador
165             $retorno = $this->db->query("select * from usuarios
166                                         where usuario = '$usuario'
167                                         and tipo = 'ADMINISTRADOR'
168                                         and estatus = ''");
169
170             //Verifica se a quantidade de linhas trazidas na consulta é superior a 0
171             if($retorno->num_rows() == 0){
172                 //Usuário não é administrador
173                 $dados = 0;
174             }else{
175                 //Usuário é administrador
176                 $dados = 1;
177             }
178
179         } catch (Exception $e) {
180             $dados = 0;
181         }
182
183         return $dados;
184     }
185 }
```

Este método verifica na íntegra se o usuário que passamos como usuário de login, é um administrador do sistema, vejam que até o parâmetro tipo é “chumbado” na query, o retorno é simples, informando somente se ele foi encontrado ou não.

```
187     private function validaUsuario($usuario){
188         try{
189             //Atributo retorno recebe o resultado do SELECT
190             //realizado na tabela de usuários lembrando da função MD5()
191             //por causa da criptografia, e sem status pois teremos que validar
192             //para verificar se está deletado virtualmente ou não.
193             $retorno = $this->db->query("select * from usuarios
194                                         where usuario = '$usuario'");
195
196             //Verifica se a quantidade de linhas trazidas na consulta é superior a 0,
197             //Vinculamos o resultado da query para tratarmos o resultado do status
198             $linha = $retorno->row();
199
200             //Criado um array com dois elementos para retorno do resultado
201             //1 - Código da mensagem
202             //2 - Descrição da mensagem
203
204             if($retorno->num_rows() == 0){
205                 $dados = array('codigo' => 4,
206                               'msg' => 'Usuário não existe na base de dados.');
207             }else{
208                 if(trim($linha->estatus) == "D"){
209                     $dados = array('codigo' => 5,
210                                   'msg' => 'Usuário JÁ DESATIVADO NA BASE DE DADOS!');
211                 }else{
212                     $dados = array('codigo' => 1,
213                                   'msg' => 'Usuário correto');
214                 }
215             }
216
217         } catch (Exception $e) {
218             $dados = array('codigo' => 00,
219                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
220                           '$e->getMessage(), "\n");
221         }
222
223         return $dados;
224     }
```

Neste outro método, fazemos uma query simples, passando somente o usuário, muito parecido com o que fizemos no login, irei falar disso mais a frente, e retornamos o resultado de acordo com esta busca, vamos nos atentar aos códigos de retorno, utilizaremos no método principal, e vale ressaltar, os dois métodos são privados, ou seja, só serão utilizados dentro da classe **M_usuario**.

Agora vamos para o método principal, o **desativar()**:

```

121     public function desativar($usuario, $usuarioLogin){
122         try{
123             //Verificar o tipo de usuário do solicitante a desativação
124             $retornoAdmin = $this->verificaAdmin($usuarioLogin); 1
125
126             if($retornoAdmin == 1){
127                 //Verificar o status do usuário antes de fazer o update
128                 $retornoUsuario = $this->validaUsuario($usuario);
129
130                 if($retornoUsuario['codigo'] == 1){
131                     //Query de atualização dos dados
132                     $this->db->query("update usuarios set estatus = 'D'
133                                     where usuario = '$usuario'");
134
135                     //Verificar se a atualização ocorreu com sucesso
136                     if($this->db->affected_rows() > 0){
137                         $dados = array('codigo' => 1,
138                                     'msg' => 'Usuário DESATIVADO corretamente');
139
140                     }else{
141                         $dados = array('codigo' => 6,
142                                     'msg' => 'Houve algum problema na DESATIVAÇÃO do usuário');
143                     }
144                 }else{
145                     $dados = array('codigo' => $retornoUsuario['codigo'],
146                                     'msg' => $retornoUsuario['msg']); 3
147                 }
148             }else{
149                 $dados = array('codigo' => 3,
150                                     'msg' => 'Usuário NÃO É ADMINISTRADOR, não pode excluir.');
151             }
152         }catch (Exception $e) {
153             $dados = array('codigo' => 00,
154                         'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
155                         '$e->getMessage(), "\n");
156         }
157         //Envia o array $dados com as informações tratadas
158         //acima pela estrutura de decisão if
159         return $dados;
160     }

```

Percebam que fazemos uso dos dois métodos codificados por nós, o número **1**, valida se o usuário desativador é um administrador, o número **2**, valida se o usuário está ativo e existente em nossa base de dados, caso isso não aconteça, o número **3**, retorna exatamente o que acontece com este usuário, ou seja, se ele existe na base de dados, ou se já está desativado.

E submetendo novamente o **endpoint** de **desativar** no **Insomnia**, ficamos assim:

	id_usuario	usuario	senha	dtcria	estatus	nome	tipo
▶	1	admin	d829b843a6550a947e82f2f38ed6b7a7	2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
	2	arthur	827ccb0eea8a706c434a16891f84e7b	2024-09-08 17:18:38	D	Arthur Prado Costa	COMUM

Vamos levar em consideração estes dados para testar o **endpoint**.

DELETE ▾ http://localhost/compras/usuario/des/ Send 200 OK 79.9 ms 59 B Just Now ▾

JSON	Auth	Query	Header 1	Docs	Preview	Header 6	Cookie	Timeline
<pre> 1 ▶ { 2 "usuario": "arthur", 3 "usuarioLogin": "admin" 4 }</pre>					<pre> 1 ▶ { 2 "codigo": 1, 3 "msg": "Usuário DESATIVADO corretamente" 4 }</pre>			

Tudo certo, o usuário é um administrador e desativou outro usuário.

	id_usuario	usuario	senha	dtcria	estatus	nome	tipo
▶	1	admin	d829b843a6550a947e82f2f38ed6b7a7	2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
	2	arthur	827ccb0eea8a706c434a16891f84e7b	2024-09-08 17:18:38	D	Arthur Prado Costa	COMUM

Voltando ao cenário anterior, vamos fazer inverter as situações, ou seja, fazer um usuário comum tentar desativar alguém.

	id_usuario	usuario	senha	dtcria	estatus	nome	tipo
1	admin	d829b843a6550a947e82f2f38ed6b7a7		2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
2	arthur	827ccb0eea8a706c434a16891f84e7b		2024-09-08 17:18:38		Arthur Prado Costa	COMUM

```

DELETE ▾ http://localhost/compras/usuario/desativar Send 200 OK 66.1 ms 89 B Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ▯ {
2   "usuario": "admin",
3   "usuarioLogin": "arthur"
4 }
5
1 ▯ {
2   "codigo": 3,
3   "msg": "Usuário NÃO É ADMINISTRADOR, não pode
        excluir."
4 }

```

Usuário que não é administrador, não consegue prosseguir com a operação.

```

DELETE ▾ http://localhost/compras/usuario/desativar Send 200 OK 52 ms 72 B Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ▯ {
2   "usuario": "arthur",
3   "usuarioLogin": "admin"
4 }
5
1 ▯ {
2   "codigo": 5,
3   "msg": "Usuário JÁ DESATIVADO NA BASE DE
        DADOS!"
4 }

```

E caso tentemos desativar alguém que já esteja desativado.

Com isso, finalizamos os exercícios 3 e 4 de nossa atividade.

2.7.1.4 Correção do 5º exercício

O quinto e último exercício na verdade, já deveria ser montado de acordo com o andamento das atividades anteriores, que resultariam na montagem do CRUD completo, **endpoint** por **endpoint**, ficando assim no **Insomnia**:

```

 Usuario
  POST API_Inserir
  GET API_Consultar
  PUT API_Alterar
  DELETE API_Desativar

```

2.7.1.5 Ajustando as coisas ...

Pessoal, perceberam que ao montarmos esta última classe, a **Usuario**, vocês repararam que todas as atividades com relação ao usuário, estão nela? Ou não?

Sabia que podemos melhorar as coisas ainda? Vocês se recordam que no início de nossas aulas, criamos uma **controller Login**, e uma **model M_acesso**?

Então, como esta atividade de login, é uma ação do usuário, podemos eliminar esses dois itens (**controller Login** e **model M_acesso**), juntando dentro das classes, **Usuario** e **M_usuario**, pois, são atividades pertencentes a estas duas classes.

Como vamos fazer isso? É complicado?

É muito simples, pessoal, teremos que refatorar o código, mas olha só como é simples, primeiramente, vamos na **controller Login**, e pegaremos o método **logar()**, copiar o mesmo e colocar na **controller Usuario**.

O mesmo temos que fazer com a **model**, iremos copiar o método **validaLogin()** da **model M acesso** e colocar na model **M usuario**, conforme a seguir:

```
application > models > M_acesso.php > M_acesso
1 <?php
2 defined('BASEPATH') or exit('No direct script access allowed');
3
4 class M_acesso extends CI_Model {
5     public function validaLogin($usuario, $senha){
6         try{
7             //Atributo retorno recebe o resultado do SELECT
8             //realizado na Tabela de usuários lembrando da Funcção
9             //para causa da criptografia, e sem status pois terem
10            //para verificar se está deletado virtualmente ou não
11             $retorno = $this->db->query("select * from usuarios
12             where usuario = '$usuario'
13             and senha = md5('$senha')");
14
15             //Verifica se a quantidade de linhas trazidas na consulta
16             //é igual a 1, caso contrário retornamos null
17             $linha = $retorno->row();
18
19             //Criado um array com dois elementos para retorno do
20             //1 - Código da mensagem
21             //2 - Descrição da mensagem
22             if($retorno->num_rows() == 0){
23                 $dados = array('codigo' => 4,
24                               'msg' => 'Usuário ou senha inválida');
25             }else{
26                 if(trim($linha->estatus) == "0"){
27                     $dados = array('codigo' => 5,
28                                   'msg' => 'Usuário DESATIVADO');
29                 }else{
30                     $dados = array('codigo' => 1,
31                                   'msg' => 'Usuário correto');
32                 }
33             }
34         } catch (Exception $e) {
35             $dados = array('codigo' => 0,
36                           'msg' => 'ATENÇÃO: O seguinte erro ocorreu: ' . $e->getMessage(), "\n");
37         }
38
39         return $dados;
40     }
41 }
42
43
44 class M_usuario extends CI_Model {
45
46     public function validaLogin($usuario, $senha){
47         try{
48             //Atributo retorno recebe o resultado do SELECT
49             //realizado na Tabela de usuários lembrando da Funcção
50             //para causa da criptografia, e sem status pois terem
51             //para verificar se está deletado virtualmente ou não
52             $retorno = $this->db->query("select * from usuarios
53             where usuario = '$usuario'
54             and senha = md5('$senha')");
55
56             //Verifica se a quantidade de linhas trazidas na consulta
57             //é igual a 1, caso contrário retornamos null
58             $linha = $retorno->row();
59
60             //Criado um array com dois elementos para retorno do
61             //1 - Código da mensagem
62             //2 - Descrição da mensagem
63             if($retorno->num_rows() == 0){
64                 $dados = array('codigo' => 4,
65                               'msg' => 'Usuário ou senha inválida');
66             }else{
67                 if(trim($linha->estatus) == "0"){
68                     $dados = array('codigo' => 5,
69                                   'msg' => 'Usuário DESATIVADO');
70                 }else{
71                     $dados = array('codigo' => 1,
72                                   'msg' => 'Usuário correto');
73                 }
74             }
75         } catch (Exception $e) {
76             $dados = array('codigo' => 0,
77                           'msg' => 'ATENÇÃO: O seguinte erro ocorreu: ' . $e->getMessage(), "\n");
78         }
79
80         return $dados;
81     }
82 }
```

Mas ainda o serviço não acabou, temos que mudar o **endpoint** no **Insomnia**, pois o mesmo ainda está apontando para a **controller Login**, e temos que mudar somente isso, apontar para a **controller Usuario**. vejam:

The screenshot shows the Postman interface with the following details:

- Request URL:** http://localhost/compras/login/logar
- Method:** GET
- Header:** Content-Type: application/json
- Body (JSON):**

```
1 * {
2   "usuario": "admin",
3   "senha": "123admin"
4 }
```
- Response Status:** 200 OK
- Response Time:** 1.72 s
- Response Size:** 43 B
- Response Headers:** Content-Type: application/json; charset=UTF-8
- Response Body (JSON):**

```
1 * {
2   "codigo": 1,
3   "msg": "Usuário correto"
4 }
```

Assim como está

```

1 < {
2   "codigo": 1,
3   "msg": "Usuário correto"
4 }
    
```

Alteramos a **controller** e tudo funciona normalmente.

Agora para finalizar, podemos deslocar o **endpoint** para a pasta **Usuario** no **Insomnia**, deixando assim a estrutura de **endpoints**, veja:

Assim terminamos a correção e readequação do **Login** de nosso sistema.

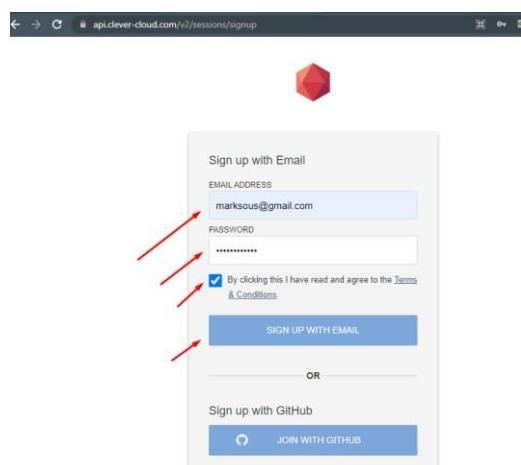
2.8. Criação de banco de dados externo para Log

Para nosso cenário, vamos implementar um banco de dados para LOG, nesse primeiro momento, vamos criá-lo para armazenar os logs de criação, alteração e “exclusão” dos dados de usuários nesse primeiro momento.

Um ponto interessante agora, é que esse banco estará em nuvem, para isso usei o <https://www.clever-cloud.com/en/> como repositório, ele é gratuito e restrito quanto a quantidade de bancos para serem criados e volume de dados, mas para nossa aula será suficiente.

2.8.1. Criando um banco de dados MySQL na nuvem

O cadastro é realizado conforme clicando no link a seguir: <https://api.clever-cloud.com/v2/sessions/signup>



Preencha os dados e prossiga.

Após o cadastro você terá o console da ferramenta e para criar um banco de dados MySQL, deverá seguir os passos abaixo:

The screenshot shows the Clever Cloud 'Personal space' interface. In the sidebar, under 'Create...', 'an add-on' is selected, indicated by a red arrow. Below the sidebar, there are cards for 'Elasticsearch with Kibana and APM server as options' and 'Persistent file system for your application'. On the right, there are cards for 'MongoDB' and 'MySQL'. The 'MySQL' card is highlighted with a red border. Both cards have a 'SELECT' button at the bottom.

What kind of MySQL do you need?

PLAN NAME	BACKUPS	LOGS	MAX CONNECTION LIMIT	MAX DB SIZE	MEMORY	METRICS	TYPE	VCPUS	PRICE
DEV	Daily - 7 Retained	No	5	10 MB	Shared	No	Shared	Shared	0.00 €

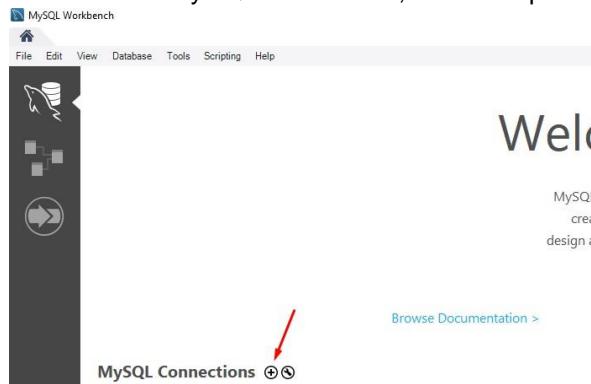
What is the name of your MySQL add-on? In which region should it be located?

The screenshot shows the configuration of a MySQL add-on. In the 'NAME:' field, 'Dblog' is entered. In the 'ZONE:' section, a world map shows the 'Paris France' location highlighted with a red box. To the right, there are icons for 'Montreal Canada' and 'Paris France' with their respective region names and Clever Cloud logos.

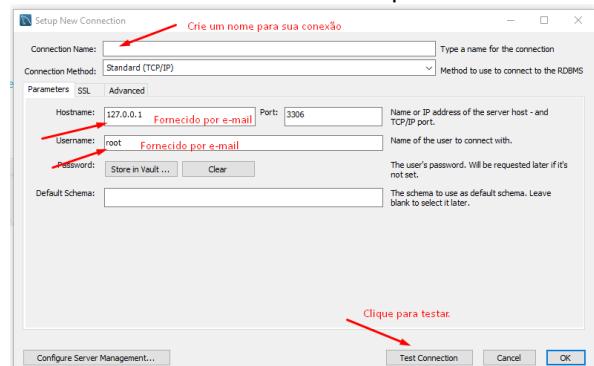
The screenshot shows the 'Addon dashboard' for the MySQL add-on. It displays 'Database Credentials' with fields for Host, Database Name, User, Password, and Port, all of which are highlighted with red boxes. Below these, a 'Connection URI' field contains a MySQL command-line connection string.

Nesse momento para conectar a esse banco temos duas formas, pelo MySQL Workbench ou pelo *phpMyAdmin* no próprio console você consegue acessar essa opção. Eu irei trabalhar com o MySQL Workbench, assim configuraremos dessa forma:

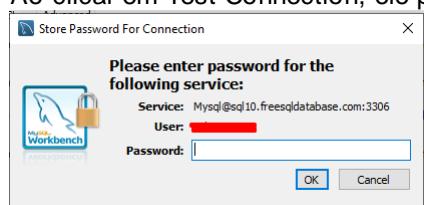
1º Ao abrir o MySQL Workbench, clicamos para adicionar uma nova conexão.



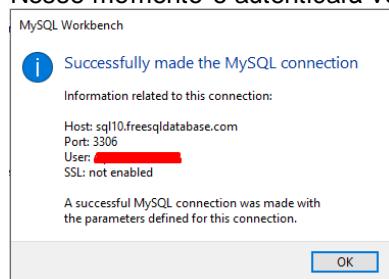
2º Passamos os dados para a configuração da conexão. *Hostname*, *Port*, *Username* e *Password* são fornecidos nesse e-mail que mostrei anteriormente.



Ao clicar em *Test Connection*, ele pedirá a senha fornecida.

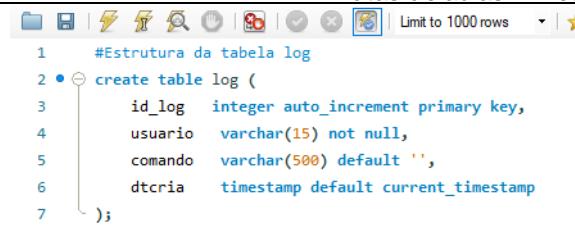


Nesse momento e autenticará você no servidor na nuvem, deverá retornar:



Realizado esses passos, você terá um banco de dados MySQL na nuvem.

Após carregar o MySQL Workbench, vamos criar a tabela LOG.



```
1  #Estrutura da tabela log
2  • create table log (
3      id_log    integer auto_increment primary key,
4      usuario   varchar(15) not null,
5      comando   varchar(500) default '',
6      dtcria    timestamp default current_timestamp
7  );
```

2.8.2. Implementar as alterações em nosso projeto

Com o banco criado na nuvem, teremos que configurar o arquivo `database.php` em nosso projeto no `CodeIgniter` para contemplar essa alteração. Lembrando que agora teremos **duas conexões** de banco de dados em nosso projeto.

```
76 //Conexão default, seria o banco de produção de nosso projeto
77 $db['default'] = array(
78     'dsn'      => 'BD_Compras', //Nome da conexão
79     'hostname' => 'localhost:3307', //Servidor onde está o banco de dados
80     'username' => 'root', //Usuário do banco de dados
81     'password' => '', //Caso possua, a senha do banco de dados
82     'database' => 'compras', //Nome do banco de dados criado
83     'dbdriver' => 'mysqli', //Driver do banco de dados, iremos utilizar esse por estarmos
84                           //trabalhando com o Banco MySQL
85     'dbprefix' => '',
86     'pconnect' => FALSE,
87     'db_debug' => ($ENVIRONMENT !== 'production'),
88     'cache_on' => FALSE,
89     'cachedir' => '',
90     'char_set' => 'utf8',
91     'dbcollat' => 'utf8_general_ci',
92     'swap_pre' => '',
93     'encrypt' => FALSE,
94     'compress' => FALSE,
95     'stricton' => FALSE,
96     'failover' => array(),
97     'save_queries' => TRUE
98 );
99
100 //Conexão do banco de dados de Log
101 $db['log'] = array(
102     'dsn'      => 'BD_Log', //Nome da conexão
103     'hostname' => 'bvsryyxueu6yo1j1r8g7c-mysql.services.clever-cloud.com', //Servidor onde está o banco de dados
104     'username' => '...', //Usuário fornecido pela clever cloud
105     'password' => '...', //Senha fornecida pela clever cloud
106     'database' => '...', //Nome do banco fornecido pela clever cloud
107     'dbdriver' => 'mysqli', //Driver do banco de dados, iremos utilizar esse por estarmos
108                           //trabalhando com o Banco MySQL
109     'dbprefix' => '',
110     'pconnect' => FALSE,
111     'db_debug' => ($ENVIRONMENT !== 'production'),
112     'cache_on' => FALSE,
113     'cachedir' => '',
114     'char_set' => 'utf8',
115     'dbcollat' => 'utf8_general_ci',
116     'swap_pre' => '',
117     'encrypt' => FALSE,
118     'compress' => FALSE,
119     'stricton' => FALSE,
120     'failover' => array(),
121     'save_queries' => TRUE
122 );
```

Temos o array **default** e também o array **log**

A partir de agora, vamos fazer a implementação da `model` para contemplar os logs de nosso projeto. Assim criaremos uma `m_log` com um método de inserção, percebam que fazemos a instância do banco de log no atributo `$dblog` abrindo essa conexão, temos a função `trocaCaractere()` que veremos a seguir e no final da operação fazemos o fechamento da conexão com o banco, vejam:

```

EXPLORER      ...
M_log.php x
Application > models > M_log.php > M_log

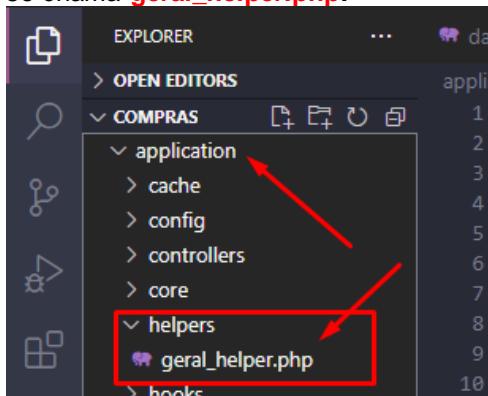
4 class M_log extends CI_Model {
5     public function inserirLog($usuario, $comando){
6         try{
7             //Instancia do banco de log
8             $dblog = $this->load->database('log', TRUE);
9
10            //Chamada da função na Helper para nos auxiliar
11            $comando = trocaCaractere($comando);
12
13            //Query de inserção dos dados
14            $dblog->query("insert into log (usuario, comando)
15                           values ('$usuario', '$comando')");
16
17            //Verificar se a inserção ocorreu com sucesso
18            if($dblog->affected_rows() > 0){
19                $dados = array('codigo' => 1,
20                               'msg' => 'Log cadastrado corretamente');
21
22            } else{
23                $dados = array('codigo' => 6,
24                               'msg' => 'Houve algum problema na inserção do log');
25            }
26
27            //Fecha a conexão com o banco de log
28            $dblog->close();
29        } catch (Exception $e) {
30            $dados = array('codigo' => 00,
31                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
32                           '$e->getMessage(), "\n");
33        }
34
35        //Envia o array $dados com as informações tratadas
36        //acima pela estrutura de decisão if
37        return $dados;
38    }
}

```

Acima no atributo \$comando recebe o resultado da função **troca_caractere()**, mas o que seria isso?

Faremos agora a utilização novamente da **HELPERS** no *Codeigniter*, como falamos no início da abordagem desse framework, não é nada mais que um ajudante, e implementamos já uma função e que chamamos no decorrer de nosso projeto, em nosso caso faremos essa função inicialmente, e caso precisemos, criaremos outras na sequência de nossa práxis.

Lembrando, que esta *helper* está localizada na sua pasta “*application*”. Nosso implemento se chama **geral_helper.php**.



A seguir vamos programar nossa função no **HELPER**, que será bem simples, somente fazer a troca do caractere (‘) aspas simples para (‘) crase no atributo \$comando, isso é necessário para que possamos salvar o comando *SQL* completo em nossa tabela de log, pois com aspas, temos problemas para fazer a inserção do mesmo, vejam:

```

EXPLORER      ...  geral_helper.php X
COMPRAS
  application
    > cache
    > config
    > controllers
    > core
  helpers
    geral_helper.php
  index.html
  hooks
  language
  libraries
  logs
  models
  third_party
  views
  .htaccess
  index.html
  system
  user_guide
  .editorconfig
  .gitignore
  .htaccess
  composer.json
  contributing.md
  index.php
  license.txt
  readme.rst

application > helpers > geral_helper.php > ...

1  <?php
2  defined('BASEPATH') or exit('No direct script access allowed');
3
4  /*
5   * Função para verificar os parâmetros vindos do FrontEnd
6   */
7  function verificarParam($atributos, $lista) {
8      //1º -Verificar se os elementos do FrontEnd estão nos atributos necessários
9      foreach($lista as $key => $value) {
10          if(array_key_exists($key, get_object_vars($atributos))){
11              $estatus = 1;
12          }else {
13              $estatus = 0;
14              break;
15          }
16      }
17
18      // 2º -Verificando a quantidade de elementos
19      if(count(get_object_vars($atributos)) != count($lista)) {
20          $estatus = 0;
21      }
22
23      return $estatus;
24  }
25
26  /*
27   * Função para trocar ' (aspas simples) por ` (acento agudo)
28   * para podermos montar a String para compor
29   * o campo comando
30   */
31  function trocaCaractere($value){
32      $retorno = str_replace("'", "`", $value);
33      return $retorno;
34  }
35 ?>

```

Fazemos abaixo as alterações no método de inserção na **controller USUARIO**, onde agora passaremos na chamada do método, o **usuário login** que está realizando a inclusão dos dados, na correção dos exercícios usamos essa ideia para validar o tipo de usuário na exclusão, e agora vamos alocar para armazenarmos o log, para isso temos que validar se o mesmo foi informado e alterar a chamada da *model*. E verificaremos o salvamento do *LOG*.

```
64     public function inserir(){
65         //Usuário, senha, nome, tipo (Administrador ou Comum)
66         //recebidos via JSON e colocados em variáveis
67         //retornos possíveis:
68         //1 - Usuário cadastrado corretamente (Banco)
69         //2 - Faltou informar o usuário (FrontEnd)
70         //3 - Faltou informar a senha (FrontEnd)
71         //4 - Faltou informar o nome (FrontEnd)
72         //5 - Faltou informar o tipo de usuário (FrontEnd)
73         //6 - Houve algum problema no insert da tabela (Banco)
74         //7 - Usuario do sistema não informado (Front)
75         //8 - Houve problema no salvamento do Log, mas o usuario foi inserido
76
77     try{
78         //Usuário e senha recebidos via JSON
79         //e colocados em atributos
80         $json = file_get_contents('php://input');
81         $resultado = json_decode($json);
82
83         //Array com os dados que deverão vir do Front
84         $lista = array(
85             "usuario" => '0',
86             "senha" => '0',
87             "nome" => '0',
88             "tipo_usuario" => '0',
89             "usuarioLogin" => '0' ←
90         );
91
92         if (verificarParam($resultado, $lista) == 1) {
93             //Fazendo os setters
94             $this->setUsuario($resultado->usuario);
95             $this->setSenha($resultado->senha);
96             $this->setNome($resultado->nome);
97             $this->setTipoUsuario(strtoupper($resultado->tipo_usuario));
98             $this->setUsuarioLogin($resultado->usuarioLogin); ←
99
100            //Faremos uma validação para sabermos se todos os dados
101            //foram enviados
102            if (trim($this->getUsuario()) == ''){
103                $retorno = array('codigo' => 2,
104                                'msg' => 'Usuário não informado');
105            elseif (trim($this->getSenha()) == ''){
106                $retorno = array('codigo' => 3,
107                                'msg' => 'Senha não informada');
108            elseif (trim($this->getNome()) == ''){
109                $retorno = array('codigo' => 4,
110                                'msg' => 'Nome não informado');
111            elseif ((trim($this->getTipoUsuario()) != 'ADMINISTRADOR' &&
112                    trim($this->getTipoUsuario()) != 'COMUM' ) ||
113                    trim($this->getTipoUsuario()) == ''){
114                $retorno = array('codigo' => 5,
115                                'msg' => 'Tipo de usuário inválido');
116            else if (trim($this->getUsuarioLogin()) == ''){
117                $retorno = array('codigo' => 7,
118                                'msg' => 'Usuário Logado no sistema não informado');
119            else{
120                //Realizo a instância da Model
121                $this->load->model('M_usuario');
```

```

122
123         //Atributo $retorno recebe array com informações
124         //da validação do acesso
125         $retorno = $this->M_usuario->inserir($this->getUsuario(), $this->getSenha(),
126                                         $this->getNome(), $this->getTipoUsuario(),
127                                         $this->getUsuarioLogin());
128     }
129 } else {
130     $retorno = array(
131         'codigo' => 99,
132         'msg' => 'Os campos vindos do FrontEnd não representam
133             o método de login. Verifique.'
134     );
135 }
136
137 } catch (Exception $e) {
138     $retorno = array('codigo' => 0,
139                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
140                     '$e->getMessage()');
141 }
142
143 //Retorno no formato JSON
144 echo json_encode($retorno);
145 }
```

Na **model m_usuario** também precisamos fazer alterações, necessidade teremos de chamar a **model m_log** e o método **inserir_log()**, também precisaremos colocar o comando de inserção em um atributo para que possamos passar o mesmo como parâmetro para o salvamento do log, e acrescentar os retornos que teremos de acordo com os acontecimentos, conforme explicado acima, vejam:

```

5     public function inserir($usuario, $senha, $nome, $tipo_usuario, $usuarioLogin){
6         try{
7             //Query de inserção dos dados
8             $sql = "insert into usuarios (usuario, senha, nome, tipo)
9                   values ('$usuario', md5('$senha'), '$nome', '$tipo_usuario')";
10
11            $this->db->query($sql);
12
13            //Verificar se a inserção ocorreu com sucesso
14            if($this->db->affected_rows() > 0){
15                //Usuario na base local inseriu corretamente
16                //Instância do banco.
17                $this->load->model('M_log'); ←
18
19                //Fazemos a chamada do método de inserção
20                $retornoLog = $this->M_log->inserirLog($usuarioLogin, $sql); ←
21
22                if($retornoLog['codigo'] == 1){ ←
23                    $dados = array('codigo' => 1,
24                               'msg' => 'Usuário cadastrado corretamente');
25                } else{ ←
26                    $dados = array('codigo' => 8, ←
27                               'msg' => 'Houve algum problema no salvamento do log, porém,
28                               usuário cadastrado corretamente.');
29                }
30            } else{
31                $dados = array('codigo' => 6,
32                               'msg' => 'Houve algum problema na inserção na tabela de usuários');
33            }
34        } catch (Exception $e) {
35            $dados = array('codigo' => 00,
36                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
37                           '$e->getMessage(), "\n");
38        }

```

Uma observação importante nessa lógica é percebermos que mesmo tendo um problema na inserção do *LOG*, o sistema não para, pois, o usuário é inserido, mas o alerta é dado.

Agora precisamos mudar o nosso *endpoint* no *Insomnia* na API_INSERIR, pois precisamos passar o usuário que está fazendo a inserção, vejam:

Method	Path	Response
POST	http://localhost/compras/usuario/inserir	200 OK 1.4 s 59 B

```

1 > {
2   "usuario": "marcos",
3   "senha": "abcd123",
4   "nome": "Marcos Costa de Sousa",
5   "tipo_usuario": "administrador",
6   "usuarioLogin": "admin"
7 }
  
```

Agora é só inserirmos, perceberão que esta operação irá demandar mais tempo, isso acontece porque teremos que submeter nossa aplicação a um segundo banco de dados, e ainda mais, o mesmo não é mais local, e sim remoto:

Verificar na tabela de usuário se o mesmo foi cadastrado, você perceberá uma diferença de três horas no campo **dtcria** das duas tabelas, isso acontece por causa do fuso horário do banco de dados de log, pois estamos usando um servidor que está fisicamente na França:

	id_usuario	usuario	senha	dtcria	estatus	nome	tipo
1	1	admin	542429e6ff38dec75594db476cffbf73	2024-08-06 11:39:21		Marcos Costa de Sousa	ADMINISTRADOR
7	7	marcos	79cfbe94595de33b3326c06ab1c7dbda	2024-09-17 10:59:50		Marcos Costa de Sousa	ADMINISTRADOR

Verificar se o log foi gravado corretamente na nuvem:

The screenshot shows a database interface with a sidebar labeled 'Navigator' containing 'SCHEMAS' and 'Tables'. A red arrow points from the 'Tables' section to the 'log' table under the 'bvsryyyxcu6yo1j1r8q7c' schema. The main window displays a SQL query 'SELECT * FROM log' and its results in a grid. The results show various log entries with columns: id_log, usuario, comando, and dtcria. The last entry is highlighted with a red box.

id_log	usuario	comando	dtcria
96	arthur	insert into pedido_det (num_pedido, usucria, cod_produto, qtdc) values (6, 'arthur', 2, 3);insert into pedido_det (num_pedido, usuc...	2021-12-06 20:00:59
87	admin	update pedido_cab set prazo = '', observacao = '' where num_pedido = 1	2021-12-16 22:52:13
88	admin	update pedido_cab set prazo = '2021-12-16', observacao = 'Teste' where num_pedido = 1	2021-12-16 22:53:01
89	admin	update pedido_det set qtdc = 10 where num_pedido = 1 and cod_produto = 1	2021-12-16 22:54:37
90	admin	update pedido_det set estatus = 'D' where num_pedido = 1 and cod_produto = 1	2021-12-16 22:56:47
91	admin	update pedido_cab set estatus = 'D' where num_pedido = 1	2021-12-16 22:57:37
92	admin	update pedido_cab set prazo = '2021-12-16', observacao = 'Aula final' where num_pedido = 2	2021-12-16 23:26:19
93	admin	update pedido_det set qtdc = 10 where num_pedido = 2 and cod_produto = 1	2021-12-16 23:34:58
94	admin	update pedido_det set estatus = 'D' where num_pedido = 2 and cod_produto = 1	2021-12-16 23:40:44
95	admin	insert into usuarios (usuario, senha, nome, tipo) values ('marcos', 'md5('abcd123')', 'Marcos Costa de Sousa', 'ADMINISTRADOR')	2024-09-17 13:59:42

Com isso, finalizamos essa etapa, vale lembrar que essa é uma das formas de fazer essa tarefa de log.

Agora é com vocês, implementem a explicação dada até agora e depois, alterem os métodos de **alteração** e **exclusão** de usuários, além dos *endpoints* envolvidos no *Insomnia*, para esta estrutura explicada em aula.

2.9. Criando o método de cadastro de unidade de medida

Vamos agora criar o método de cadastro da unidade de medida que aplicaremos em nossos produtos que serão cadastrados em nosso sistema, lembrando que faremos a parte de *backend*, para isso vamos criar uma nova *controller* para tratar estas unidades, assim teremos a *unidmedida.php*, iremos começar pelos métodos (*getters* e *setters*, inserir, consultar, alterar e desativar), e modificadores de acesso.

2.9.1 Tabelas de produtos e unidade de medida

Precisamos criar em nosso banco de dados mais duas tabelas, conforme citado anteriormente, mas agora vamos remover o *id_usuario* como chave primária da tabela de usuários e colocar o *usuario* como chave primaria para nos auxiliar no log vejam:

```

26      #mudando a estrutura da tabela de usuário
27 •  alter table usuarios drop column id_usuario;
28 •  alter table usuarios modify usuario varchar(15) not null primary key;
29
30      #Estrutura da tabela de unidade de medidas
31 •  create table unid_medida (
32          cod_unidade integer auto_increment primary key,
33          sigla      varchar(03) default '',
34          descricao  varchar(30) default '',
35          dtcria     datetime default now(),
36          usucria   varchar(15) default '',
37          estatus    char(01) default '',
38
39          constraint foreign key fk_unidmed_prod (usucria) references usuarios(usuario)
40      );

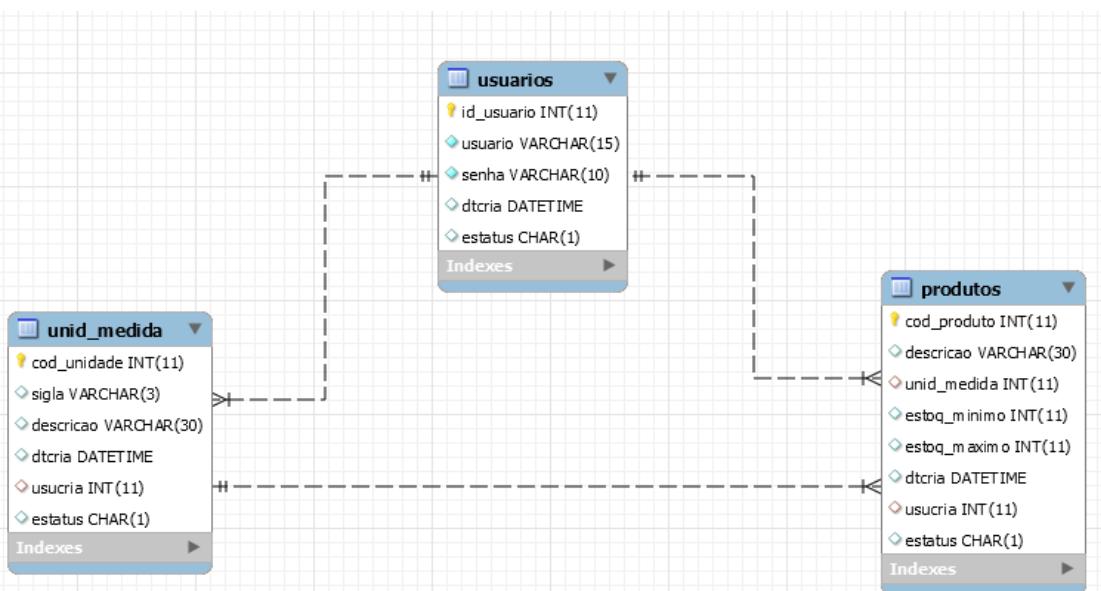
```

```

42      #Estrutura da tabela de produtos
43 •  create table produtos (
44      cod_produto integer auto_increment primary key,
45      descricao varchar(30) default '',
46      unid_medida integer default 0,
47      estoq_minimo integer default 0,
48      estoq_maximo integer default 0,
49      dtcria      datetime default now(),
50      usucria     varchar(15) default '',
51      estatus      char(01) default '',
52
53      constraint foreign key fk_prod_unidmed (unid_medida) references unid_medida(cod_unidade),
54      constraint foreign key fk_prod_usuarios (usucria) references usuarios(usuario)
55  );

```

Percebam que agora temos chaves estrangeiras na tabela de produto e unidade de medida, na última, fazendo a amarração do usuário, já a tabela de produtos, além de amarrar o usuário, teremos que amarrar o código da unidade de medida, abaixo temos um DER de nosso banco de dados até o momento:



Percebam que agora de acordo com nosso DER, para cadastrarmos um produto em nosso projeto, haverá a necessidade obrigatória da existência prévia da unidade de medida, e não menos importante, também precisaremos ter um usuário criado também, pois agora teremos que colocar na tabela de produtos e unidades de medida o usuário que está fazendo a implementação.

Sendo assim, agora podemos criar as *controllers* e *models* tanto de unidade de medida quanto produtos.

2.9.2 Unidade de medida

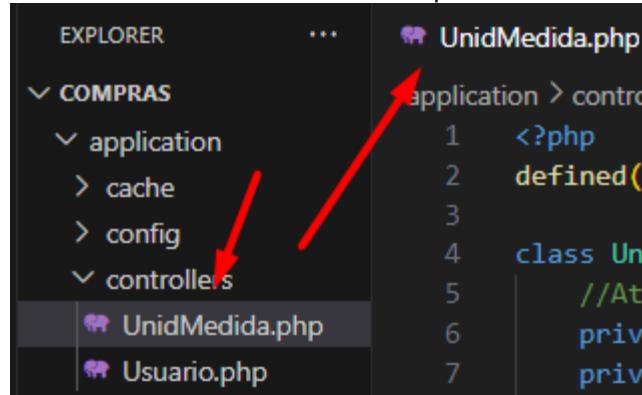
Por ser mais simples sua confecção, iremos começar pelo cadastro de unidades medidas, como fizemos em usuários, iremos fazer inclusão, alteração, consulta e “exclusão” da mesma. A ideia da criação dessa classe com seus métodos é entender que em nossa regra de negócio cada produto precisará ter uma unidade de medida “amarrada”.

2.9.3 Controller de Unidade de medida (*getters* e *setters*, métodos inserir, consultar, alterar e desativar)

Nesse primeiro momento, iremos trabalhar com dois métodos de inserção e consulta, onde devemos nos atentar as validações dos dados enviados pelo *FrontEnd* e o salvamento do LOG

em nosso banco de dados, lembrando que faremos essa operação nos métodos de inserção, alteração e “exclusão” dos registros.

Então criamos mais uma *controller* que chamaremos UnidMedida.php.



Para ganharmos tempo, vamos nessa fase montar toda a *Controller* com os métodos relatados acima, começamos pelo inserir e seguiremos com os demais métodos.

```
UnidMedida.php
application > controllers > UnidMedida.php > UnidMedida > desativar
1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3
4  class UnidMedida extends CI_Controller {
5      //Atributos privados da classe
6      private $codigo;
7      private $sigla;
8      private $descricao;
9      private $usuarioLogin;
10
11     //Getters dos atributos
12     public function getCodigo()
13     {
14         return $this->codigo;
15     }
16
17     public function getSigla()
18     {
19         return $this->sigla;
20     }
21
22     public function getDescricao()
23     {
24         return $this->descricao;
25     }
26
27     public function getUsuarioLogin()
28     {
29         return $this->usuarioLogin;
30     }
31
32     //Setters dos atributos
33     public function setCodigo($codigoFront)
34     {
35         $this->codigo = $codigoFront;
36     }
37
```

```
38     public function setSigla($siglaFront)
39     {
40         $this->sigla = $siglaFront;
41     }
42
43     public function setDescricao($descricaoFront)
44     {
45         $this->descricao = $descricaoFront;
46     }
47
48     public function setUsuarioLogin($usuarioLoginFront)
49     {
50         $this->usuarioLogin = $usuarioLoginFront;
51     }
52
53     public function inserir(){
54         //Sigla e Descrição
55         //recebidas via JSON e colocadas em variáveis
56         //Retornos possíveis:
57         //1 - Unidade cadastrada corretamente (Banco)
58         //2 - Faltou informar a sigla (FrontEnd)
59         //3 - Quantidade de caracteres da sigla é superior a 3 (FrontEnd)
60         //4 - Descrição não informada (FrontEnd)
61         //5 - Usuário não informado (FrontEnd)
62         //6 - Houve algum problema no insert da tabela (Banco)
63         //7 - Houve problema no salvamento do LOG, mas a unidade foi inclusa (LOG)
64     try{
65         $json = file_get_contents('php://input');
66         $resultado = json_decode($json);
67
68         //Array com os dados que deverão vir do Front
69         $lista = array(
70             "sigla" => '0',
71             "descricao" => '0',
72             "usuarioLogin" => '0'
73         );
74
75         if (verificarParam($resultado, $lista) == 1) {
76             //Fazendo os setters
77             $this->setSigla($resultado->sigla);
78             $this->setDescricao($resultado->descricao);
79             $this->setUsuarioLogin($resultado->usuarioLogin);
80
81             //Faremos uma validação para sabermos se todos os dados
82             //foram enviados corretamente
83             if (trim($this->getSigla()) == ''){
84                 $retorno = array('codigo' => 2,
85                                 'msg' => 'Sigla não informada.');
86             }elseif (strlen($this->getSigla()) > 3){
87                 $retorno = array('codigo' => 3,
88                                 'msg' => 'Sigla pode conter no máximo 3 caracteres.');
89             }elseif (trim($this->getDescricao()) == ''){
90                 $retorno = array('codigo' => 4,
91                                 'msg' => 'Descrição não informada.');
92             }elseif (trim($this->getUsuarioLogin()) == ''){
93                 $retorno = array('codigo' => 5,
94                                 'msg' => 'Usuário não informado');
95             }else{
96                 //Realizo a instância da Model
97                 $this->load->model('m_unidmedida');
```

```

100         //Atributo $retorno recebe array com informações
101         $retorno = $this->m_unidmedida->inserir($this->getSigla(), $this->getDescricao(),
102                                         $this->getUsuarioLogin());
103     }
104 } else {
105     $retorno = array(
106         'codigo' => 99,
107         'msg' => 'Os campos vindos do FrontEnd não representam
108             | o método de login. Verifique.'
109     );
110 }
111 } catch (Exception $e){
112     $retorno = array('codigo' => 0,
113                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
114                     '$e->getMessage()');
115 }
116 //Retorno no formato JSON
117 echo json_encode($retorno);
118 }

119
120 public function consultar(){
121     //Código, Sigla e Descrição
122     //recebidos via JSON e colocados em variáveis
123     //Retornos possíveis:
124     //1 - Dados consultados corretamente (Banco)
125     //2 - Quantidade de caracteres da sigla é superior a 3 (FrontEnd)
126     //6 - Dados não encontrados (Banco)
127     try{
128         $json = file_get_contents('php://input');
129         $resultado = json_decode($json);
130
131         //Array com os dados que deverão vir do Front
132         $lista = array(
133             "codigo" => '0',
134             "sigla" => '0',
135             "descricao" => '0'
136         );
137
138         if (verificarParam($resultado, $lista) == 1) {
139             //Fazendo os setters
140             $this->setCodigo($resultado->codigo);
141             $this->setSigla($resultado->sigla);
142             $this->setDescricao($resultado->descricao);
143
144             //Verifico somente a qtd de caracteres da sigla, poder ter até 3
145             //caracteres ou nenhum para trazer todas as siglas
146             if (strlen($this->getSigla()) > 3){
147                 $retorno = array('codigo' => 2,
148                                 'msg' => 'Sigla pode conter no máximo 3 caracteres ou nenhum para todas');
149             }else{
150                 //Realizo a instância da Model
151                 $this->load->model('m_unidmedida');
152
153                 //Atributo $retorno recebe array com informações
154                 //da consulta dos dados
155                 $retorno = $this->m_unidmedida->consultar($this->getCodigo(), $this->getSigla(),
156                                                 $this->getDescricao());
157             }
158         }else {
159             $retorno = array(
160                 'codigo' => 99,
161                 'msg' => 'Os campos vindos do FrontEnd não representam
162                     | o método de login. Verifique.'
163             );
164         }
165     }catch (Exception $e) {
166         $retorno = array('codigo' => 0,
167                         'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
168                         '$e->getMessage()');
169     }
}

```

```

170     //Retorno no formato JSON
171     echo json_encode($retorno);
172 }
173
174 public function alterar(){
175     //Código, Sigla e Descrição
176     //recebidos via JSON e colocadas em variáveis
177     //Retornos possíveis:
178     //1 - Dado(s) alterado(s) corretamente (Banco)
179     //2 - Faltou informar o código (FrontEnd)
180     //3 - Quantidade de caracteres da sigla é superior a 3 (FrontEnd)
181     //4 - Sigla ou Descrição não informadas, ai não tem o que alterar (FrontEnd)
182     //5 - Usuário não informado (FrontEnd)
183     //6 - Dados não encontrados (Banco)
184     //7 - Houve problema no salvamento do LOG, mas a unidade foi inclusa (LOG)
185     try{
186         $json = file_get_contents('php://input');
187         $resultado = json_decode($json);
188
189         //Array com os dados que deverão vir do Front
190         $lista = array(
191             "codigo" => '0',
192             "sigla" => '0',
193             "descricao" => '0',
194             "usuarioLogin" => '0'
195         );
196
197         if (verificarParam($resultado, $lista) == 1) {
198             //Fazendo os setters
199             $this->setCodigo($resultado->codigo);
200             $this->setSigla($resultado->sigla);
201             $this->setDescricao($resultado->descricao);
202             $this->setUsuarioLogin($resultado->usuarioLogin);
203
204             //Faremos uma validação para sabermos se os dados
205             //foram enviados corretamente
206
207             if (trim($this->getCodigo()) == '' || trim($this->getCodigo()) == 0){
208                 $retorno = array('codigo' => 2,
209                                 'msg' => 'Codigo não informado.');
210             }elseif (strlen(trim($this->getSigla())) > 3){
211                 $retorno = array('codigo' => 3,
212                                 'msg' => 'Sigla pode conter no máximo 3 caracteres.');
213             }elseif (trim($this->getDescricao()) == '' && trim($this->getDescricao()) == ''){
214                 $retorno = array('codigo' => 4,
215                                 'msg' => 'Sigla ou Descrição não foram informadas.');
216             }elseif (trim($this->getUsuarioLogin()) == ''){
217                 $retorno = array('codigo' => 5,
218                                 'msg' => 'Usuário não informado');
219             }else{
220                 //Realizo a instância da Model
221                 $this->load->model('m_unidmedida');
222
223                 //Atributo $retorno recebe array com informações
224                 //da validação do acesso
225                 $retorno = $this->m_unidmedida->alterar($this->getCodigo(), $this->getSigla(),
226                                                 $this->getDescricao(), $this->getUsuarioLogin());
227             }
228         }else{
229             $retorno = array(
230                 'codigo' => 99,
231                 'msg' => 'Os campos vindos do FrontEnd não representam
232                             o método de login. Verifique.'
233             );
234         }
235     }catch (Exception $e) {
236         $retorno = array('codigo' => 0,
237                         'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
238                         $e->getMessage());
239     }

```

```
240     //Retorno no formato JSON
241     echo json_encode($retorno);
242 }
243
244 public function desativar(){
245     //Código da unidade recebido via JSON e colocado em variável
246     //Retornos possíveis:
247     //1 - Unidade desativada corretamente (Banco)
248     //2 - Código não informado;
249     //3 - Existem produtos cadastrados com essa unidade de medida
250     //5 - Usuário não informado (FrontEnd)
251     //6 - Dados não encontrados (Banco)
252     //7 - Houve problema no salvamento do LOG, mas a unidade foi alterada (LOG)
253     try{
254         $json = file_get_contents('php://input');
255         $resultado = json_decode($json);
256
257         //Array com os dados que deverão vir do Front
258         $lista = array(
259             "codigo" => '0',
260             "usuarioLogin" => '0'
261         );
262
263         if (verificarParam($resultado, $lista) == 1) {
264             //Fazendo os setters
265             $this->setCodigo($resultado->codigo);
266             $this->setUsuarioLogin($resultado->usuarioLogin);
267
268             //Validação para tipo de usuário que deverá ser ADMINISTRADOR, COMUM ou VAZIO
269             if (trim($this->getCodigo()) == '' || trim($this->getCodigo()) == 0){
270                 $retorno = array('codigo' => 2,
271                                 'msg' => 'Código da unidade não informado');
272             }elseif (trim($this->getUsuarioLogin()) == ''){
273                 $retorno = array('codigo' => 5,
274                                 'msg' => 'Usuário não informado');
275             }else{
276                 //Realizo a instância da Model
277                 $this->load->model('m_unidmedida');
278
279                 //Atributo $retorno recebe array com informações
280                 $retorno = $this->m_unidmedida->desativar($this->getCodigo(),
281                                                 $this->getUsuarioLogin());
282             }
283         }else {
284             $retorno = array(
285                 'codigo' => 99,
286                 'msg' => 'Os campos vindos do FrontEnd não representam
287                           o método de login. Verifique.'
288             );
289         }
290     } catch (Exception $e) {
291         $retorno = array('codigo' => 0,
292                         'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
293                         $e->getMessage());
294     }
295     //Retorno no formato JSON
296     echo json_encode($retorno);
297 }
298 }
```

2.9.4 Model de Unidade de medida (métodos inserir, consultar, alterar e desativar)

De acordo com os métodos criados na *controller*, iremos montar a *model* que se chamará *M_unidmedida.php*, iremos também implementar o log em nosso banco na nuvem.

```

EXPLORER ... M_unidmedida.php X
COMPRAS application > models > M_unidmedida.php
    > cache
    > config
    > controllers
    > core
    > helpers
    > hooks
    > language
    > libraries
    > logs
    > models
        M_log.php
        M_unidmedida.php
        M_usuario.php

```

Vamos nessa fase montar toda a *Controller* com os métodos relatados acima, começamos pelo inserir e seguiremos com os demais métodos, no método de consulta, foi construído para permitir 4 situações possíveis de parametrização, conforme relatado no comentário do próprio método, isso quer dizer que teremos uma *query* dinâmica, ou seja, ela será construída de acordo com os parâmetros passados, e não faremos log nesse caso, quanto aos métodos alteração e “exclusão”, devemos nos atentar as validações dos dados enviados pelo *FrontEnd* e o salvamento do LOG em nosso banco de dados na nuvem, lembrando que faremos essa operação nos métodos de inserção, alteração e desativação.

```

M_unidmedida.php X
application > models > M_unidmedida.php > M_unidmedida > inserir
1 <?php
2 defined('BASEPATH') or exit('No direct script access allowed');
3
4 class M_unidmedida extends CI_Model {
5
6     public function inserir($sigla, $descricao, $usuarioLogin){
7         try{
8             //Query de inserção dos dados
9             $sql = "insert into unid_medida (sigla, descricao, usucria)
10                 values ('$sigla', '$descricao', '$usuarioLogin')";
11
12             $this->db->query($sql);
13
14             //Verificar se a inserção ocorreu com sucesso
15             if($this->db->affected_rows() > 0){
16                 //Fazemos a inserção no Log na nuvem
17                 //Fazemos a instância da model M_log
18                 $this->load->model('m_log');
19
20                 //Fazemos a chamada do método de inserção do Log
21                 $retorno_log = $this->m_log->inserirLog($usuarioLogin, $sql);
22
23                 if ($retorno_log['codigo'] == 1){
24                     $dados = array('codigo' => 1,
25                               'msg' => 'Unidade de medida cadastrada corretamente');
26                 }else{
27                     $dados = array('codigo' => 7,
28                               'msg' => 'Houve algum problema no salvamento do Log, porém,
29                                         Unidade de Medida cadastrada corretamente');
30                 }
31             }
32         }
33     }
34 }

```

```
31
32         }else{
33             $dados = array('codigo' => 6,
34                         | | | 'msg' => 'Houve algum problema na inserção na tabela de unidade de medida');
35         }
36     } catch (Exception $e) {
37         $dados = array('codigo' => 00,
38                         | | | 'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
39                         | | | $e->getMessage(), "\n");
40     }
41     //Envia o array $dados com as informações tratadas
42     //acima pela estrutura de decisão if
43     return $dados;
44 }
45
46 public function consultar($codigo, $sigla, $descricao){
47     //-----
48     //Função que servirá para quatro tipos de consulta:
49     // * Para todos as unidades de medida;
50     // * Para uma determinada sigla de unidade;
51     // * Para um código de unidade de medida;
52     // * Para descrição da unidade de medida;
53     //-----
54
55     try{
56         //Query para consultar dados de acordo com parâmetros passados
57         $sql = "select * from unid_medida where estatus = '' ";
58
59         if($codigo != '' && $codigo != 0) {
60             $sql = $sql . "and cod_unidade = '$codigo' ";
61         }
62
63         if($sigla != ''){
64             $sql = $sql . "and sigla = '$sigla' ";
65         }
66
67         if($descricao != ''){
68             $sql = $sql . "and descricao like '%$descricao%' ";
69         }
70
71         $retorno = $this->db->query($sql);
72
73         //Verificar se a consulta ocorreu com sucesso
74         if($retorno->num_rows() > 0){
75             $dados = array('codigo' => 1,
76                           | | | 'msg' => 'Consulta efetuada com sucesso',
77                           | | | 'dados' => $retorno->result());
78         }else{
79             $dados = array('codigo' => 6,
80                           | | | 'msg' => 'Dados não encontrados');
81         }
82     }catch (Exception $e) {
83         $dados = array('codigo' => 00,
84                         | | | 'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
85                         | | | $e->getMessage(), "\n");
86     }
87     //Envia o array $dados com as informações tratadas
88     //acima pela estrutura de decisão if
89     return $dados;
90 }
91
92 public function alterar($codigo, $sigla, $descricao, $usuario){
93     try{
94         //Query de atualização dos dados
95         if (trim($sigla) != '' && trim($descricao) != '') {
96             $sql = "update unid_medida set sigla = '$sigla', descricao = '$descricao'
97                         | | | where cod_unidade = '$codigo'";
98         }elseif (trim($sigla) != '') {
99             $sql = "update unid_medida set sigla = '$sigla'
100                         | | | where cod_unidade = '$codigo'";
101         }elseif (trim($descricao) != '') {
102             $sql = "update unid_medida set descricao = '$descricao'
103                         | | | where cod_unidade = '$codigo'";
104         }
105     }catch (Exception $e) {
106         $dados = array('codigo' => 00,
107                         | | | 'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
108                         | | | $e->getMessage(), "\n");
109     }
110 }
```

```

99         $sql = "update unid_medida set sigla = '$sigla' where cod_unidade = $codigo";
100    }else{
101        $sql = "update unid_medida set descricao = '$descricao' where cod_unidade = $codigo";
102    }
103
104    $this->db->query($sql);
105
106    //Verificar se a atualização ocorreu com sucesso
107    if($this->db->affected_rows() > 0){
108        //Fazemos a inserção no Log na nuvem
109        //Fazemos a instância da model M_log
110        $this->load->model('m_log');
111
112        //Fazemos a chamada do método de inserção do Log
113        $retorno_log = $this->m_log->inserirLog($usuario, $sql);
114
115        if ($retorno_log['codigo'] == 1){
116            $dados = array('codigo' => 1,
117                           'msg' => 'Unidade de medida atualizada corretamente');
118        }else{
119            $dados = array('codigo' => 7,
120                           'msg' => 'Houve algum problema no salvamento do Log, porém,
121                           |unidade de medida cadastrada corretamente');
122        }
123    }else{
124        $dados = array('codigo' => 6,
125                           'msg' => 'Houve algum problema na atualização na tabela de unidade de medida');
126    }
127 }catch (Exception $e) {
128     $dados = array('codigo' => 00,
129                    'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
130                    $e->getMessage(), "\n");
131 }
132 //Envia o array $dados com as informações tratadas
133 //acima pela estrutura de decisão if
134 return $dados;
135 }
136
137 public function desativar($codigo, $usuario){
138     try{
139         //Há necessidade de verificar se existe algum produto com
140         //essa unidade de medida já cadastrado, se tiver não podemos
141         //desativar essa unidade
142         $sql = "select * from produtos where unid_medida = $codigo and estatus = '' ";
143
144         $retorno = $this->db->query($sql);
145
146         //Verificar se a consulta trouxe algum produto
147         if($retorno->num_rows() > 0){
148             //Não posso fazer a desativação
149             $dados = array('codigo' => 3,
150                           'msg' => 'Não podemos desativar, existem produtos com essa unidade de
151                           |medida cadastrado(s).');
152         }else{
153             //Query de atualização dos dados
154             $sql2 = "update unid_medida set estatus = 'D' where cod_unidade = '$codigo'";
155
156             $this->db->query($sql2);
157
158             //Verificar se a atualização ocorreu com sucesso
159             if($this->db->affected_rows() > 0){
160                 //Fazemos a inserção no Log na nuvem
161                 //Fazemos a instância da model M_log
162                 $this->load->model('m_log');
163
164                 //Fazemos a chamada do método de inserção do Log
165                 $retorno_log = $this->m_log->inserirLog($usuario, $sql2);
166             }
167         }
168     }
169 }
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1127
1128
1129
1130
1131
1132
1133
1134
1135
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1143
1144
1145
1146
1146
1147
1148
1149
1149
1150
1151
1152
1153
1153
1154
1155
1156
1156
1157
1158
1159
1159
1160
1161
1162
1162
1163
1164
1165
1165
1166
1167
1168
1168
1169
1170
1171
1171
1172
1173
1174
1174
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1182
1183
1184
1185
1185
1186
1187
1188
1188
1189
1190
1191
1191
1192
1193
1194
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1202
1203
1204
1205
1205
1206
1207
1208
1208
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1891
1892
1893
1893
1894
1895
1895
1896
1897

```

```

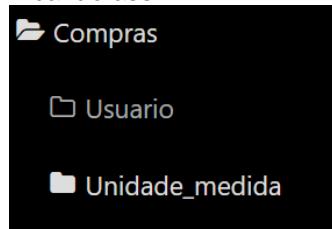
167     if ($retorno_log['codigo'] == 1){
168         $dados = array('codigo' => 1,
169                     'msg' => 'Unidade de medida DESATIVADA corretamente');
170     }else{
171         $dados = array('codigo' => 8,
172                     'msg' => 'Houve algum problema no salvamento do Log, porém,
173                     | usuário desativado corretamente');
174     }
175
176     }else{
177         $dados = array('codigo' => 7,
178                     'msg' => 'Houve algum problema na DESATIVAÇÃO da unidade de medida');
179     }
180 }
181 }catch (Exception $e) {
182     $dados = array('codigo' => 00,
183                     'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
184                     | $e->getMessage(), "\n");
185 }
186 //Envia o array $dados com as informações tratadas
187 //acima pela estrutura de decisão if
188 return $dados;
189 }
190 }
191

```

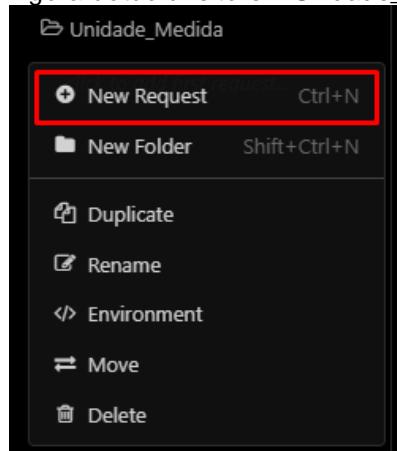
Terminado a **controller** e a **model**, configuraremos o *Insomnia* para executar os **endpoints** que fizemos, vamos criar uma pasta conforme imagem a seguir:



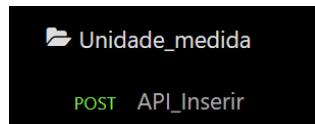
Ficando assim:



Agora botão direito em Unidade_Medida e *New Request*



Criaremos a API, chamaremos de API_Inserir, como POST e Form

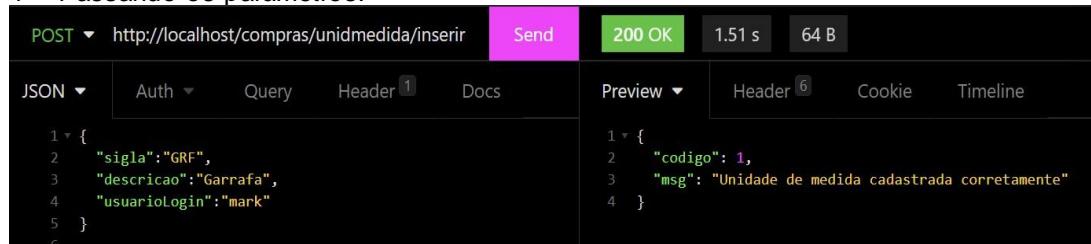


Agora informamos o caminho da controller que contém o método de inserção e atribuímos as variáveis.



Agora podemos ver as inclusões sendo:

1º - Passando os parâmetros:

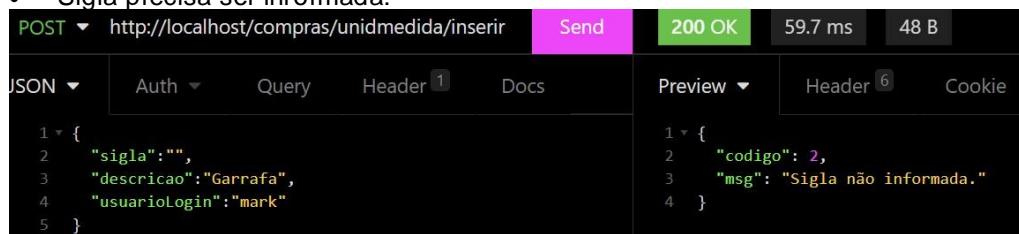


No banco de dados:

Result Grid Filter Rows: Edit: Export					
cod_unidade	sigla	descricao	dtcria	usucria	estatus
1	GRF	Garrafa	2024-09-24 11:37:45	mark	

2º - Críticas para que a inclusão e alteração aconteçam:

- Sigla precisa ser informada:



- Sigla maior que 3 caracteres:

```
POST ▼ http://localhost/compras/unidmedida/inserir Send 200 OK 82.9 ms 69 B
JSON ▼ Auth ▼ Query Header 1 Docs Preview ▼ Header 6 Cookie Timeline
1 ▶ {
2   "sigla": "GRFAS",
3   "descricao": "Garrafa",
4   "usuarioLogin": "mark"
5 }
```

```
1 ▶ {
2   "codigo": 3,
3   "msg": "Sigla pode conter no máximo 3 caracteres."
4 }
```

- Descrição vazia:

```
POST ▼ http://localhost/compras/unidmedida/inserir Send 200 OK 96.3 ms 62 B
JSON ▼ Auth ▼ Query Header 1 Docs Preview ▼ Header 6 Cookie
1 ▶ {
2   "sigla": "GRF",
3   "descricao": "",
4   "usuarioLogin": "mark"
5 }
```

```
1 ▶ {
2   "codigo": 4,
3   "msg": "Descrição não informada."
4 }
```

- Usuário vazio ou zerado:

```
POST ▼ http://localhost/compras/unidmedida/inserir Send 200 OK 73.4 ms 54 B
JSON ▼ Auth ▼ Query Header 1 Docs Preview ▼ Header 6 Cookie
1 ▶ {
2   "sigla": "GRF",
3   "descricao": "Garrafa",
4   "usuarioLogin": ""
5 }
```

```
1 ▶ {
2   "codigo": 5,
3   "msg": "Usuário não informado"
4 }
```

Método consultar agora, criaremos a API, chamaremos de API_Consultar, como GET::

New Request X

Name (defaults to your request URL if left empty)
API_Consultar

Method **GET** ▼

* Tip: paste Curl command into URL afterwards to import it Create

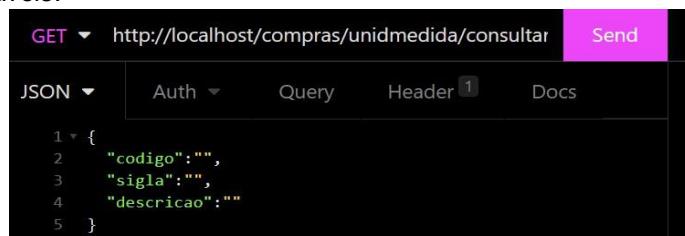
GET ▼ http://localhost/compras/unidmedida/consultar Send

JSON ▼ Auth ▼ Query Header 1 Docs

Content Type ▼

- STRUCTURED
- Multipart Form
- Form URL Encoded
- GraphQL Query
- TEXT
- JSON** ✓
- XML

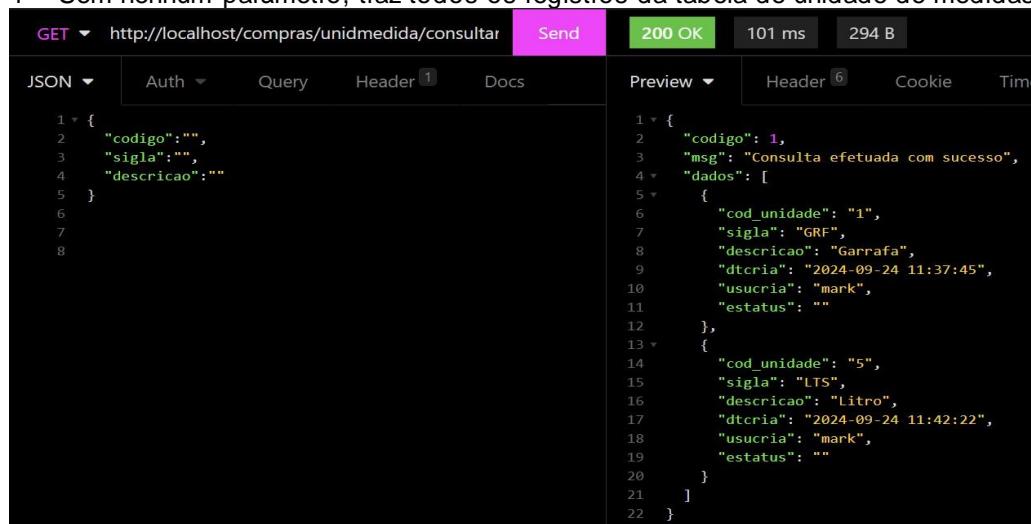
Agora informamos o caminho da *controller* que contém o método de consulta e atribuímos as variáveis.



```
1 ↴ {
2   "codigo": "",
3   "sigla": "",
4   "descricao": ""
5 }
```

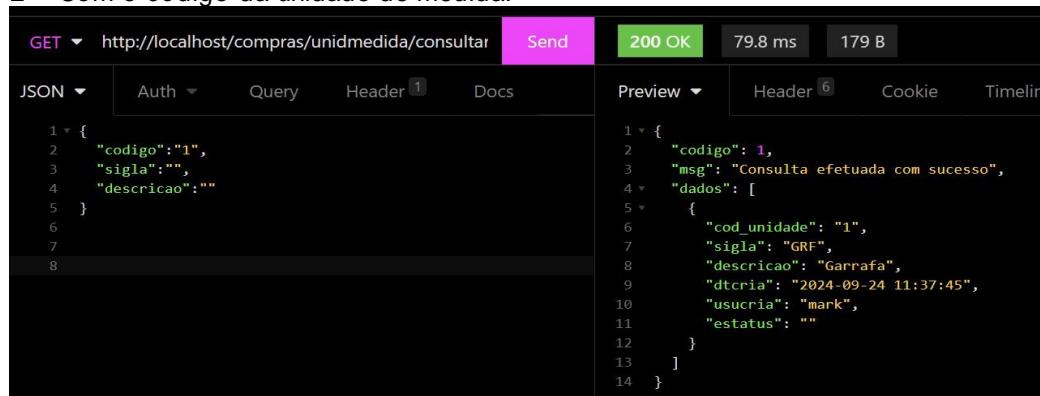
Agora podemos ver as consultas sendo:

1º - Sem nenhum parâmetro, traz todos os registros da tabela de unidade de medidas



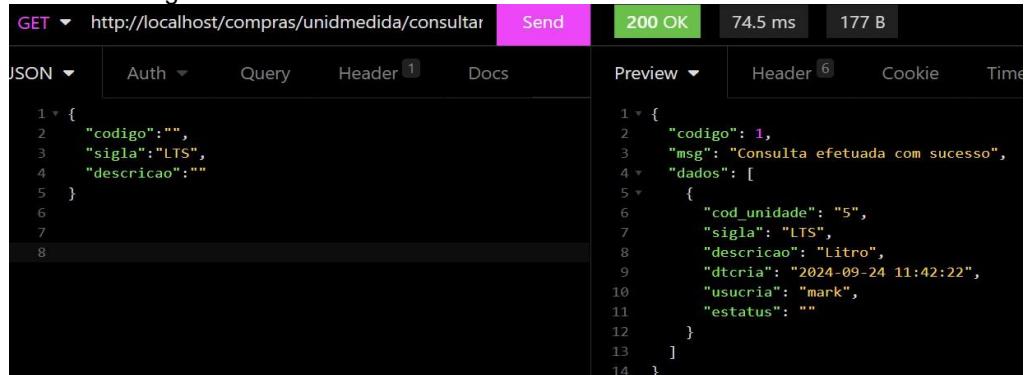
```
1 ↴ {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_unidade": "1",
7       "sigla": "GRF",
8       "descricao": "Garrafa",
9       "dtcria": "2024-09-24 11:37:45",
10      "usucria": "mark",
11      "estatus": ""
12    },
13    {
14      "cod_unidade": "5",
15      "sigla": "LTS",
16      "descricao": "Litro",
17      "dtcria": "2024-09-24 11:42:22",
18      "usucria": "mark",
19      "estatus": ""
20    }
21  ]
22 }
```

2º - Com o código da unidade de medida:



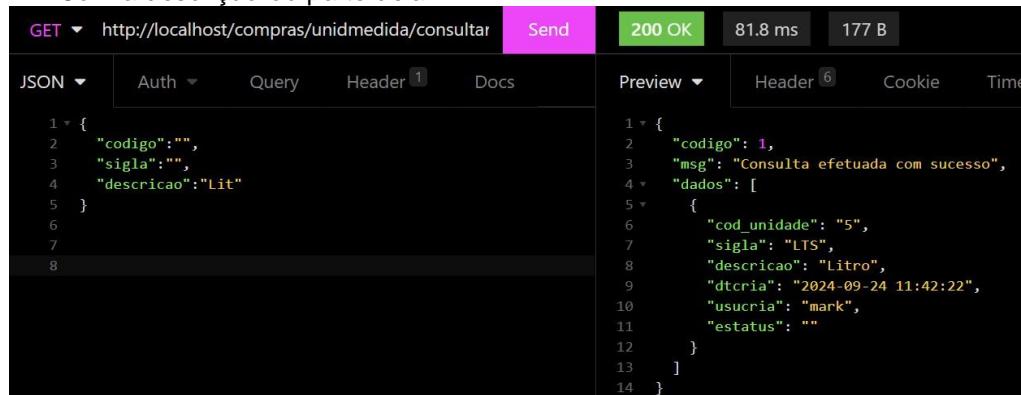
```
1 ↴ {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_unidade": "1",
7       "sigla": "GRF",
8       "descricao": "Garrafa",
9       "dtcria": "2024-09-24 11:37:45",
10      "usucria": "mark",
11      "estatus": ""
12    }
13  ]
14 }
```

3º - Com a sigla:



```
1 ↴ {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_unidade": "5",
7       "sigla": "LTS",
8       "descricao": "Litro",
9       "dtcria": "2024-09-24 11:42:22",
10      "usucria": "mark",
11      "estatus": ""
12    }
13  ]
14 }
```

4º - Com a descrição ou parte dela:

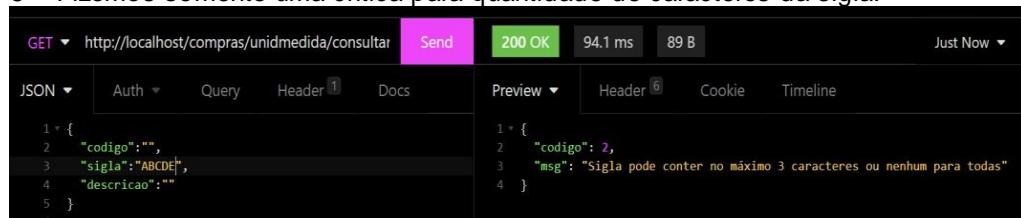


```

GET ▾ http://localhost/compras/unidmedida/consultar Send 200 OK 81.8 ms 177 B
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "codigo": "",
3   "sigla": "",
4   "descricao": "Lit"
5 }
6
7
8
1 ✎ {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4 ✎ "dados": [
5   {
6     "cod_unidade": "5",
7     "sigla": "LTS",
8     "descricao": "Litro",
9     "dtcria": "2024-09-24 11:42:22",
10    "usucria": "mark",
11    "estatus": ""
12  }
13 ]
14 }

```

5º - Fizemos somente uma crítica para quantidade de caracteres da sigla:

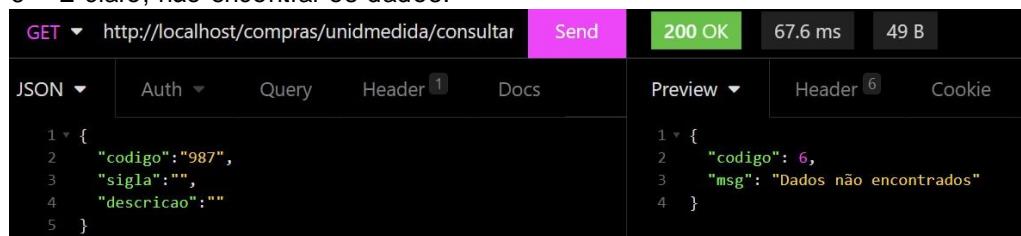


```

GET ▾ http://localhost/compras/unidmedida/consultar Send 200 OK 94.1 ms 89 B Just Now ▾
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie Timeline
1 ✎ {
2   "codigo": "",
3   "sigla": "ABCDE",
4   "descricao": ""
5 }
6
1 ✎ {
2   "codigo": 2,
3   "msg": "Sigla pode conter no máximo 3 caracteres ou nenhum para todas"
4 }

```

6º - E claro, não encontrar os dados:

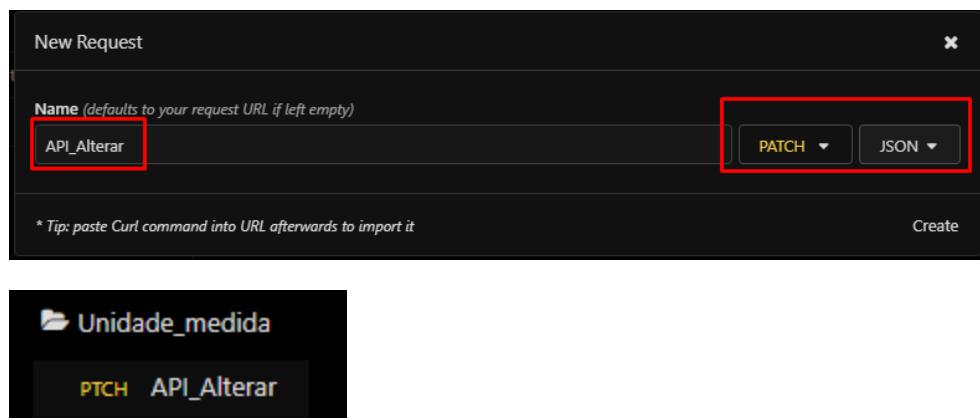


```

GET ▾ http://localhost/compras/unidmedida/consultar Send 200 OK 67.6 ms 49 B
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 6 Cookie
1 ✎ {
2   "codigo": "987",
3   "sigla": "",
4   "descricao": ""
5 }
6
1 ✎ {
2   "codigo": 6,
3   "msg": "Dados não encontrados"
4 }

```

Método alterar agora, criaremos a *API*, chamaremos de *API_Alterar*, como *PATCH* e JSON:



New Request

Name (defaults to your request URL if left empty)

API_Alterar

PATCH ▾ JSON ▾

* Tip: paste Curl command into URL afterwards to import it

Create

Unidade_medida

PTCH API_Alterar

Agora informamos o caminho da *controller* que contém o método de alteração e atribuímos as variáveis.

PATCH ▾ http://localhost/compras/unidmedida/alterar Send

JSON ▾	Auth ▾	Query	Header 1	Docs
--------	--------	-------	----------	------

```

1 v {
2   "codigo":"",
3   "sigla":"",
4   "descricao":"",
5   "usuarioLogin":""
6 }

```

Agora podemos ver as alterações sendo:

1º - Passando os parâmetros, ou somente um dos parâmetros, usarei como base os dados da tabela conforme abaixo:

5 • select * from unid_medida;

	cod_unidade	sigla	descricao	dtria	usucria	estatus
1	GRF	Garrafa	2024-09-24 11:37:45	mark		
5	LTS	Litro	2024-09-24 11:42:22	mark		

PATCH ▾ http://localhost/compras/unidmedida/alterar Send 200 OK 1.71 s 64 B

JSON ▾	Auth ▾	Query	Header 1	Docs	Preview ▾	Header 6	Cookie	Timeline
--------	--------	-------	----------	------	-----------	----------	--------	----------

```

1 v {
2   "codigo":"5",
3   "sigla":"",
4   "descricao":"Litros",
5   "usuarioLogin":"mark"
6 }

```

```

1 v {
2   "codigo": 1,
3   "msg": "Unidade de medida atualizada corretamente"
4 }

```

Ficando assim:

5 • select * from unid_medida;

	cod_unidade	sigla	descricao	dtria	usucria	estatus
1	GRF	Garrafa	2024-09-24 11:37:45	mark		
5	LTS	Litros	2024-09-24 11:42:22	mark		

Há exigência nesse método de informar o código da unidade de medida que se deseja alterar.

PATCH ▾ http://localhost/compras/unidmedida/alterar Send 200 OK 81.7 ms 49 B

JSON ▾	Auth ▾	Query	Header 1	Docs	Preview ▾	Header 6	Cookie	Timeline
--------	--------	-------	----------	------	-----------	----------	--------	----------

```

1 v {
2   "codigo":"",
3   "sigla":"",
4   "descricao":"Litros",
5   "usuarioLogin":"mark"
6 }

```

```

1 v {
2   "codigo": 2,
3   "msg": "Codigo não informado."
4 }

```

Há também a necessidade de informar o usuário que está fazendo a atualização:

PATCH ▾ http://localhost/compras/unidmedida/alterar Send 200 OK 55.4 ms 54 B

JSON ▾	Auth ▾	Query	Header 1	Docs	Preview ▾	Header 6	Cookie
--------	--------	-------	----------	------	-----------	----------	--------

```

1 v {
2   "codigo":"5",
3   "sigla":"",
4   "descricao":"Litros",
5   "usuarioLogin":""
6 }

```

```

1 v {
2   "codigo": 5,
3   "msg": "Usuário não informado"
4 }

```

E para finalizar, o método desativar, criaremos a API, chamaremos de API_Desativar, como POST e Delete:

The screenshot shows the Postman interface with a 'New Request' dialog open. The 'Name' field contains 'API_Desativar' and the 'DELETE' button is highlighted with a red box. Below the dialog, a collection named 'Unidade_medida' is shown, containing a 'DEL API_Desativar' endpoint.

The screenshot shows the Postman interface with a DELETE request configured for the URL 'http://localhost/compras/unidmedida/desativar'. The 'JSON' dropdown is selected. On the left, a sidebar shows various data structures: STRUCTURED (Multipart Form, Form URL Encoded, GraphQL Query), TEXT (JSON, XML, YAML), and a collapsed section for AUTHENTICATION.

Agora informamos o caminho da controller que contém o método de desativação e atribuímos as variáveis.

The screenshot shows the Postman interface with a DELETE request for 'http://localhost/compras/unidmedida/desativar'. The 'JSON' dropdown is selected. The JSON body is defined as follows:

```

1 {
2   "codigo": "2",
3   "usuarioLogin": "mark"
4 }

```

Agora podemos ver a desativação sendo:

1º - Passando o código da unidade de medida, usarei a tabela abaixo:

The screenshot shows the MySQL Workbench Result Grid with the following data:

	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	1	GRF	Garrafa	2024-09-24 11:37:45	mark	
	5	LTS	Litros	2024-09-24 11:42:22	mark	

The screenshot shows the Postman interface with a successful response for the DELETE request. The status is '200 OK' and the response body is:

```

1 {
2   "codigo": 1,
3   "msg": "Unidade de medida DESATIVADA corretamente"
4 }

```

Ficando assim:

	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	1	GRF	Garrafa	2024-09-24 11:37:45	mark	
	5	LTS	Litros	2024-09-24 11:42:22	mark	D



2º - Necessidade de informar o código da unidade de medida para desativação.

DELETE ▾ http://localhost/compras/unidmedida/desativar Send 200 OK 89.8 ms 64 B

JSON	Auth	Query	Header [1]	Docs	Preview	Header [6]	Cookie	Time
1 v { 2 "codigo": "", 3 "usuarioLogin": "mark" 4 }					1 v { 2 "codigo": 2, 3 "msg": "Código da unidade não informado" 4 }			

3º - O usuário também precisa ser informado.

DELETE ▾ http://localhost/compras/unidmedida/desativar Send 200 OK 72.8 ms 54 B

JSON	Auth	Query	Header [1]	Docs	Preview	Header [6]	Cookie	
1 v { 2 "codigo": "5", 3 "usuarioLogin": "" 4 }					1 v { 2 "codigo": 5, 3 "msg": "Usuário não informado" 4 }			

Após isso finalizamos a ideia de nosso *CRUD* para unidade de medida, lembrando que foi pensado um cenário para a aula, podemos fazer de outras formas de acordo com o desenvolvedor ou até mesmo as regras de negócio.

Observando nosso *Insomnia* até o momento temos:

- 📁 Compras
 - ➡ Usuario
 - POST API_Inserir
 - GET API_Login
 - GET API_Consultar
 - PTCH API_Alterar
 - DEL API_Desativar
 - 📁 Unidade_medida
 - POST API_Inserir
 - GET API_Consultar
 - PTCH API_Alterar
 - DEL API_Desativar

Lembrando aqui, que todos os métodos de inserção, alteração e desativação, tiveram os dados salvos na tabela de log.

5 • SELECT * FROM log;

Result Grid			Filter Rows:	Edit:	Export/Import:	Wrap Cell Content
id_log	usuario	comando				
100	admin	insert into usuarios (usuario, senha, nome, tipo) values ('mark', md5('12345'), 'Marcos Costa de Sousa', 'ADMINISTRADOR')				
101	mark	insert into unid_medida (sigla, descricao, usucria) values ('GRF', 'Garrafa', 'mark')				
102	mark	update unid_medida set sigla = 'LTS', descricao = 'Litro' where cod_unidade = 3				
103	mark	update unid_medida set sigla = 'LTS', descricao = 'Litross' where cod_unidade = 3				
104	mark	update unid_medida set estatus = 'D' where cod_unidade = '2'				
105	mark	insert into unid_medida (sigla, descricao, usucria) values ('GRF', 'Garrafa', 'mark')				
106	mark	insert into unid_medida (sigla, descricao, usucria) values ('LTS', 'Litro', 'mark')				
107	mark	update unid_medida set sigla = 'LTS', descricao = 'Litross' where cod_unidade = 5				
108	mark	update unid_medida set descricao = 'Litros' where cod_unidade = 5				
109	mark	update unid_medida set estatus = 'D' where cod_unidade = '5'				

Agora é com vocês, implementem essa parte em vossa projeto, testem tudo o que fizemos, lembrando que a continuidade das aulas, como as demais avaliações, depende de tudo o que fizemos até o momento.

2.9.5 Desafio e Melhorias em métodos já criados

Até o momento, temos feitos dois *CRUDs* completos, de usuário e unidade de medida, porém temos alguns pontos que podemos melhorar em nosso código, quer dizer, vocês. Como um gestor dos sistemas, abaixo irei pedir algumas implementações que deverão ser feitas em nosso sistema, para tal, gostaria que vocês fizessem em duplas (não é obrigatório), mas para que vocês possam aplicar uma técnica chamada *Pair Programming*, ou programação em par em tradução literal, trata-se de uma técnica de desenvolvimento de software ágil. O objetivo é aumentar a qualidade do software desenvolvido. Na prática, todo o trabalho é feito por duas pessoas, em um único computador. Um fica responsável por controlar o processo de criação e escrever os códigos. Enquanto isso, o outro faz um trabalho mais analítico, observando cada linha de código, a fim de identificar erros.

Para tornar o processo ainda mais dinâmico e eficaz, é comum que vocês invertam os papéis ao longo do desenvolvimento do trabalho. Isso contribui para tornar o processo ainda mais produtivo e menos suscetível a erros, uma vez que o observador sempre estará atento às falhas e o controlador pode se concentrar na digitação dos códigos. Vejam esse vídeo, é do canal “Código Fonte TV”, gosto muito desse canal porque eles explicam tudo de uma forma bem didática, <https://www.youtube.com/watch?v=5M8yNQSFBPg> curtam lá.

De acordo com essa prática, vou passar o **DESAFIO DO MARCÃO** 😊 para vocês:

Existe uma situação na API_Inserir da unidade de medida que não tratamos, observe as imagens abaixo:

Irei levar em consideração os dados da tabela de usuários abaixo para demonstrar a ideia do desafio:

79 • select * from usuarios;

Result Grid			Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
usuario	senha	dtria	estatus	nome	tipo	
admin	0192023a7bbd73250516f069df18b500	2021-09-23 12:41:35		Marcos Costa	ADMINISTRADOR	
arthur	a738350acd3bd842bfdddf857914d9c9	2021-10-04 14:36:07		Arthur Prado Costa	COMUM	
eliane	1db06ed346f069d21ec8022739575e55	2021-10-04 14:35:16		Eliane Prado	COMUM	

De acordo com esses dados, irei utilizar a API_Inserir: da unidade de medida levando em consideração a tabela de unidade de medida abaixo:

79 • select * from unid_medida;

	Result Grid	Filter Rows:	Edit:	Export/In		
	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	1	PC	Peça	2021-09-23 13:56:38	admin	
▶	2	KG	Quilograma	2021-09-23 13:57:03	admin	D
▶	3	LT	Litro	2021-09-23 13:57:13	admin	
▶	4	UN	Unidade	2021-09-23 13:57:23	admin	

Em meu *Insomnia* usando a API parametrizando de acordo com a imagem abaixo:

POST ▼ http://localhost/compras/unidmedida/inserir Send

JSON ▼ Auth □ Query Header 1 Docs

```

1 + {
2     "sigla": "LTS",
3     "descricao": "Litro",
4     "usuarioLogin": "mark"
5 }
```

Ao executar essa API, é apresentada esta situação:

POST ▼ http://localhost/compras/unidmedida/inserir Send 500 Internal Server Error 740 ms 1490 B A Minute Ago

JSON ▼ Auth □ Query Header 1 Docs Preview ▼ Header 6 Cookie Timeline

```

1 + {
2     "sigla": "LTS",
3     "descricao": "Litro",
4     "usuarioLogin": "mark"
5 }
```

A Database Error Occurred

Error Number: 1452

Cannot add or update a child row: a foreign key constraint fails ('compras'.`unid_medida`, CONSTRAINT 'fk_unidmed_prod' FOREIGN KEY ('usucria') REFERENCES `usuarios` ('usuario'))

insert into unid_medida (sigla, descricao, usucria) values ('LTS', 'Litro', 'mark')

Filename: C:/xampp/htdocs/Compras/system/database/DB_driver.php

Line Number: 665

Com base em tudo que expliquei, analisem e responda:

1º - Por que aconteceu esse erro? Justifique sua resposta.

2º - Como você resolveria esse problema? Descreva com suas palavras.

3º - Implemente no PHP a solução explicada por você na questão anterior.



Essa é a primeira etapa, a segunda é direcionada a um equívoco diagnosticado no sistema na hora de realizar alguns testes que nosso gestor de projeto, ou *Product Owner* (dono do produto) diagnosticou, e vocês deverão prontamente providenciar uma solução:

Na Unidade de Medida, quando inserimos uma unidade de medida nova, a mesma informa o cadastramento correto, foi o que implementamos em aula, porém, se clicarmos para fazer um

novo *insert*, o mesmo acontece novamente, sem dar erros, isto não está correto, pois se ficarmos inserirmos a mesma unidade de medida a mesma será cadastrada diversas vezes, como abaixo:

The screenshot shows a Postman request to `http://localhost/compras/unidmedida/inserir`. The JSON payload is:

```

1 v {
2   "sigla": "LTS",
3   "descricao": "Litro",
4   "usuarioLogin": "admin"
5 }

```

The 'Send' button is highlighted with a red arrow. The response is a 200 OK status with a message: "Unidade de medida cadastrada corretamente".

Teremos o seguinte resultado:

13 • `select * from unid_medida;`

	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	6	LTS	Litro	2024-10-01 12:27:13	admin	
▶	7	LTS	Litro	2024-10-01 12:27:19	admin	
▶	8	LTS	Litro	2024-10-01 12:27:23	admin	
*	NULL	NULL	NULL	NULL	NULL	NULL

Three rows are highlighted with red arrows pointing to the 'cod_unidade' column.

Isto não pode ocorrer, pois, a integridade dos dados seria comprometida, portanto, gostaria que:

- 1º - Por que aconteceu esta situação? Justifique sua resposta.
- 2º - Como você resolveria esse problema? Descreva com suas palavras.
- 3º - Implemente no PHP a solução explicada por você na questão anterior.



Para finalizar esta atividade, foi diagnosticado mais uma situação que devemos nos atentar em nosso sistema, levando em consideração a mesma tabela do exercício anterior:

13 • `select * from unid_medida;`

	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	6	LTS	Litro	2024-10-01 12:27:13	admin	
▶	7	LTS	Litro	2024-10-01 12:27:19	admin	
▶	8	LTS	Litro	2024-10-01 12:27:23	admin	
*	NULL	NULL	NULL	NULL	NULL	NULL

Three rows are highlighted with red arrows pointing to the 'cod_unidade' column.

Fazendo a submissão do endpoint de alteração, aparece assim:

The PATCH request to `http://localhost/compras/unidmedida/alterar` has a JSON payload:

```

1 v {
2   "codigo": "5",
3   "sigla": "",
4   "descricao": "Litros",
5   "usuarioLogin": "mark"
6 }

```

The response is a 200 OK status with a message: "Houve algum problema na atualização na tabela de unidade de medida".

- 1º - Por que aconteceu esta situação? Justifique sua resposta.
- 2º - Como você resolveria esse problema? Poderíamos mostrar ao nosso usuário, uma mensagem mais concreta do que realmente aconteceu? Descreva com suas palavras.
- 3º - Implemente no PHP a solução explicada por você na questão anterior.

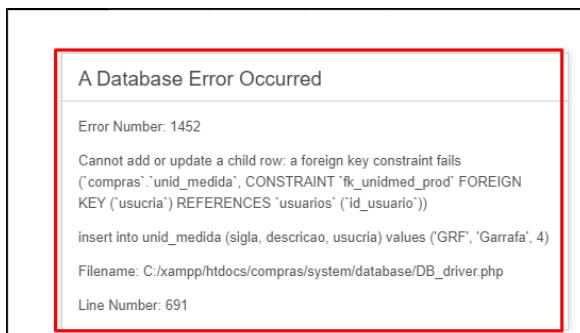
O desafio e as melhorias devem ser realizados para o seguimento de nossas aulas, iremos fazer as implementações (correção) juntos no dia 22/10, a nossa ideia é que vocês pratiquem nova implementações feitas por vocês, além de exercitar o *Pair Programming*. Portanto, vocês precisam apresentar **em duplas, o desafio solucionado até o dia 15/10.**

2.9.6 Correção do desafio do Marcão

Bom pessoal, vamos agora ver o que aconteceu no desafio que passei para vocês, para isso vamos responder a cada pergunta que passei:

2.9.6.1 Por que aconteceu esse erro? Justifique sua resposta.

Para isso precisamos entender o que erro nos relata, o erro foi esse:



Se traduzirmos ao “pé da letra”, veremos que foi dito que não podemos inserir ou atualizar um registro filho, pois houve uma falha na regra da chave estrangeira da tabela de unidade de medida, se verificarmos o *insert* que foi realizado, percebemos que o código de usuário passado no campo usucria, não existe na tabela de usuários, vejam:

The screenshot shows the Insomnia API client interface. A query is being run against a database, specifically selecting all columns from the 'usuarios' table. The results grid shows three rows of data with columns: id_usuario, usuario, senha, and dtcria. The data is as follows:

	id_usuario	usuario	senha	dtcria
▶	1	admin	admin123	2020-10-20 09
▶	2	eliane	etec2020	2020-10-20 15
▶	3	arthur	papaimarco	2020-10-20 15

A callout box points to the 'id_usuario' column of the third row, stating: "Percebiam que temos até o id_usuario de número 3".

Percebiam agora que no *Insomnia* passo propositalmente o id_usuario número 4, para ser armazenado.

The screenshot shows the Insomnia API client interface with a JSON request body. The body contains the following data:

```

1 ▶ {
2   "sigla": "GRF",
3   "descricao": "Garrafa",
4   "usuario": "4"
5 }

```

A callout box points to the value '4' in the 'usuario' field, stating: "Id desse usuário não existe em nosso banco de dados".

Então por isso, há uma quebra de integridade dos dados, como o nosso sistema não tratou esse evento, o próprio banco retorna esse erro, ele é mais comum do que pensamos, no dia a dia isso precisa ser verificado nos sistemas, erro de integridade acontecem com frequência, isso deriva principalmente da falta de testes antes de serem colocados em produção.

2.9.6.2 Como você resolveria esse problema?

A solução que vem à cabeça a recebermos esse erro, seria colocar uma crítica na *model*, verificando a existência desse usuário no banco de dado antes de efetuar o *insert*, lembrando que verificamos na *controller* se o usuário é informado, e não se é válido.

Caso não exista, retornamos um erro informando ao *frontend* que esse usuário não existe na base de dados.

Primeiramente vamos criar uma **function privada** para validar esse usuário. Função privada é executada somente dentro da classe, em nosso caso na *m_unidmedida*.

```

199     private function verificaUsuario($usuario) {
200         try{
201             //Função PRIVADA só para verificar se o usuário existe
202             //em nosso banco de dados na tabela de usuários
203             //Retornos:
204             //1 - Usuário cadastrado na base de dados
205             //8 - Usuário desativado no banco de dados
206             //9 - Usuário não encontrado
207
208             $sql = "select * from usuarios where usuario = '$usuario' ";
209
210             $retorno = $this->db->query($sql);
211
212             //Verificar se a consulta trouxe algum usuário
213             if($retorno->num_rows() > 0){
214                 //Verifico o status do usuário
215                 if($retorno->row()->estatus == 'D'){
216                     //Não se pode cadastrar o usuário
217                     $dados = array('codigo' => 8,
218                               'msg' => 'Não pode cadastrar, usuário informado está DESATIVADO');
219                 }else{
220                     $dados = array('codigo' => 1,
221                               'msg' => 'Usuário ativo na base de dados');
222                 }
223             }else{
224                 $dados = array('codigo' => 9,
225                               'msg' => 'Usuário não encontrado na base de dados');
226             }
227         }catch (Exception $e) {
228             $dados = array('codigo' => 00,
229                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
230                           $e->getMessage(), "\n");
231         }
232         //Envia o array $dados com as informações tratadas
233         //acima pela estrutura de decisão if
234         return $dados;
235     }

```

Após a criação da mesma, precisamos modificar nossa *function* inserir na *model* m_unidmedida, para instanciar esse método privado e tratarmos seu retorno, vejam:

```

6     public function inserir($sigla, $descricao, $usuario){
7         try{
8             //Atributo $retorno recebe array com informações (Correção Desafio)
9             $ret_usuario = $this->verificaUsuario($usuario);
10
11             //Validação se o usuário está com estatus valido para
12             //inserir o registro
13             if ($ret_usuario['codigo'] == 8 || $ret_usuario['codigo'] == 9 ){
14                 //retorno para controller usuário com problema
15                 $dados = $ret_usuario;
16             }else{
17                 //Query de inserção dos dados
18                 $sql = "insert into unid_medida (sigla, descricao, usucria)
19                         values ('$sigla', '$descricao', '$usuario')";
20
21                 $this->db->query($sql);
22
23                 //Verificar se a inserção ocorreu com sucesso
24                 if($this->db->affected_rows() > 0){
25                     //Fazemos a inserção no Log na nuvem
26                     //Fazemos a instância da model M_log
27                     $this->load->model('m_log');
28
29                     //Fazemos a chamada do método de inserção do Log
30                     $retorno_log = $this->m_log->inserir_log($usuario, $sql);
31
32                     if ($retorno_log['codigo'] == 1){
33                         $dados = array('codigo' => 1,
34                                       'msg' => 'Unidade de medida cadastrada corretamente');
35                     }else{
36                         $dados = array('codigo' => 7,
37                                       'msg' => 'Houve algum problema no salvamento do Log, porém,
38                                           Unidade de Medida cadastrada corretamente');
39                     }
40
41                 }else{
42                     $dados = array('codigo' => 6,
43                                   'msg' => 'Houve algum problema na inserção na tabela de unidade de medida');
44                 }
45             }
46         }catch (Exception $e) {
47             $dados = array('codigo' => 00,
48                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
49                           $e->getMessage(), "\n");
50         }
51         //Envia o array $dados com as informações tratadas
52         //acima pela estrutura de decisão if
53         return $dados;
54     }

```

Agora vamos testar no *Insomnia* para vermos as validações, levando em consideração a tabela abaixo.

	usuario	senha	dtcria	estatus	nome	tipo
▶	admin	0192023a7bbd73250516f069df18b500	2021-09-23 12:41:35		Marcos Costa	ADMINISTRADOR
	arthur	a738350acd3bd842bfdddf857914d9c9	2021-10-04 14:36:07	D	Arthur Prado Costa	COMUM
	eliane	1db06ed346f069d21ec8022739575e55	2021-10-04 14:35:16		Eliane Prado	COMUM

1º Usuário não encontrado.

```
POST ▾ http://localhost/compras/unidmedida/inserir
Send 200 OK 170 ms 72 B
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1+ {
2 "sigla": "GRF",
3 "descricao": "Garrafa",
4 "usuario": "danilo"
5 }
```

```
1 {
2 "codigo": 9,
3 "msg": "Usuário não encontrado na base de dados"
4 }
```

2º Usuário desativado.

```
POST ▾ http://localhost/compras/unidmedida/inserir
Send 200 OK 43.5 ms 91 B
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1+ {
2 "sigla": "GRF",
3 "descricao": "Garrafa",
4 "usuario": "arthur"
5 }
```

```
1 {
2 "codigo": 8,
3 "msg": "Não pode cadastrar, usuário informado está DESATIVADO"
4 }
```

3º Cadastro efetuado.

```
POST ▾ http://localhost/compras/unidmedida/inserir
Send 200 OK 1.8 s 64 B
JSON ▾ Auth ▾ Query Header 1 Docs Preview ▾ Header 5 Cookie Timeline
1+ {
2 "sigla": "GRF",
3 "descricao": "Garrafa",
4 "usuario": "eliane"
5 }
```

```
1 {
2 "codigo": 1,
3 "msg": "Unidade de medida cadastrada corretamente"
4 }
```

2.9.6.3 Por que aconteceu esse segundo erro? Justifique sua resposta.

Para isso precisamos entender o que erro nos relata, o erro foi esse:

13 • `select * from unid_medida;`

	cod_unidade	sigla	descricao	dtcria	usucria	estatus
▶	6	LTS	Litro	2024-10-01 12:27:13	admin	
	7	LTS	Litro	2024-10-01 12:27:19	admin	
	8	LTS	Litro	2024-10-01 12:27:23	admin	
*	HULL	NULL	NULL	NULL	NULL	NULL

Isso não é um erro, mas do ponto de vista de sistema, não é uma boa prática, pois, posso cadastrar diversas siglas iguais, pois a chave primária da tabela é o código da unidade e ele é auto incrementável, assim, nunca teremos a quebra desta regra.

2.9.6.4 Como você resolveria esse problema?

Uma forma seria, na tabela de unidade de medida, informar que o campo sigla, é **UNIQUE**, ou seja, não poderá se repetir, porém, quando formos fazer esta alteração, a nossa tabela não poderá ter as siglas repetidas na base, teremos que remover esta parte, senão teremos um erro.

Mas isso, do ponto de vista de sistema, com retorno ao usuário, não seria o suficiente, teremos que fazer uma consulta prévia, e verificar se a sigla, já se encontra na base de dados, ficando assim, uma verificação mais eficiente e profissional.

Então vamos começar alterando a **constraint** do campo sigla para **UNIQUE**:

1 • `alter table unid_medida`
2 `add constraint unique (sigla);`

Agora vamos implementar um método na M_unidmedida que faça a verificação se esta sigla, já não está cadastrada em nosso banco:

```

245     public function verificaUM($unidadeMedida){
246         try{
247             //Query para consultar a unidade de medida
248             $sql = "select * from unid_medida
249                 where sigla = '$unidadeMedida'
250                 and estatus = '' ";
251
252             $retorno = $this->db->query($sql);
253
254             if($retorno->num_rows() > 0){
255                 $dados = array('codigo' => 10,
256                               'msg' => 'Unidade de medida já cadastrada na base de dados.');
257             }else{
258                 $dados = array('codigo' => 2,
259                               'msg' => 'Unidade de medida não cadastrada.');
260             }
261         }catch (Exception $e) {
262             $dados = array('codigo' => 00,
263                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
264                           $e->getMessage(), "\n");
265         }
266         //Envia o array $dados com as informações tratadas
267         //acima pela estrutura de decisão if
268         return $dados;
269     }

```

Após isso, temos que modificar o método de inserção para realizar esta verificação antes da inserção.

```

6     public function inserir($sigla, $descricao, $usuario){
7         try{
8             //Atributo $retorno recebe array com informações (Correção Desafio)
9             $ret_usuario = $this->verificaUsuario($usuario);
10
11            //Validação se o usuário está com estatus valido para
12            //inserir o registro
13            if ($ret_usuario['codigo'] == 8 || $ret_usuario['codigo'] == 9 ){
14                //retorno para controller usuário com problema
15                $dados = $ret_usuario;
16            }else{
17                //Verificar se a unidade de medida já não está cadastrada
18                $ret_unidMedida = $this->verificaUM($sigla);
19
20                if ($ret_unidMedida['codigo'] == 2){
21
22                    //Query de inserção dos dados
23                    $sql = "insert into unid_medida (sigla, descricao, usucria)
24                        values ('$sigla', '$descricao', '$usuario')";
25
26                    $this->db->query($sql);
27
28                    //Verificar se a inserção ocorreu com sucesso
29                    if($this->db->affected_rows() > 0){
30                        //Fazemos a inserção no Log na nuvem
31                        //Fazemos a instância da model M_log
32                        $this->load->model('m_log');
33
34                        //Fazemos a chamada do método de inserção do Log
35                        $retorno_log = $this->m_log->inserir_log($usuario, $sql);
36
37                        if ($retorno_log['codigo'] == 1){
38                            $dados = array('codigo' => 1,
39                                          'msg' => 'Unidade de medida cadastrada corretamente');
40                        }else{
41                            $dados = array('codigo' => 7,
42                                          'msg' => 'Houve algum problema no salvamento do Log, porém,
43                                          Unidade de Medida cadastrada corretamente');
44                        }
45
46                        }else{
47                            $dados = array('codigo' => 6,
48                                          'msg' => 'Houve algum problema na inserção na tabela de unidade de medida');
49                        }
50                    }else{
51                        $dados = $ret_unidMedida;
52                    }
53                }
54            }catch (Exception $e) {
55                $dados = array('codigo' => 00,
56                              'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
57                              $e->getMessage(), "\n");
58            }
59            //Envia o array $dados com as informações tratadas
60            //acima pela estrutura de decisão if
61            return $dados;
62        }

```

No método de alteração, também precisamos fazer a verificação, senão teremos o mesmo problema, vejam:

```

109     public function alterar($codigo, $sigla, $descricao, $usuario){
110         try{
111             if (trim($sigla) != ''){
112                 //Verificar se a unidade de medida já não está cadastrada
113                 $ret_unidMedida = $this->verificaUM($sigla);
114
115                 if ($ret_unidMedida['codigo'] == 2){
116                     $dados = $ret_unidMedida;
117                     return $dados;
118                 }
119
120             }
121
122             //Query de atualização dos dados
123             if (trim($sigla) != '' && trim($descricao) != ''){
124                 $sql = "update unid_medida set sigla = '$sigla', descricao = '$descricao'
125                         where sigla = '$sigla'";
126             }elseif (trim($sigla) != ''){
127                 $sql = "update unid_medida set sigla = '$sigla' where sigla = '$sigla'";
128             }else{
129                 $sql = "update unid_medida set descricao = '$descricao' where sigla = '$sigla'";
130             }
131
132             $this->db->query($sql);
133
134             //Verificar se a atualização ocorreu com sucesso
135             if($this->db->affected_rows() > 0){
136                 //Fazemos a inserção no Log na nuvem
137                 //Fazemos a instância da model M_log
138                 $this->load->model('m_log');
139
140                 //Fazemos a chamada do método de inserção do Log
141                 $retorno_log = $this->m_log->inserir_log($usuario, $sql);
142
143                 if ($retorno_log['codigo'] == 1){
144                     $dados = array('codigo' => 1,
145                               'msg' => 'Unidade de medida atualizada corretamente');
146                 }else{
147                     $dados = array('codigo' => 7,
148                               'msg' => 'Houve algum problema no salvamento do Log, porém,
149                                         unidade de medida cadastrada corretamente');
150                 }
151             }else{
152                 $dados = array('codigo' => 6,
153                               'msg' => 'Houve algum problema na atualização na tabela de unidade de medida');
154             }
155         }catch (Exception $e) {
156             $dados = array('codigo' => 00,
157                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
158                           $e->getMessage(), "\n");
159         }
160         //Envia o array $dados com as informações tratadas
161         //acima pela estrutura de decisão if
162         return $dados;
163     }

```

2.9.6.5 Por que aconteceu esse erro? Justifique sua resposta.

Isso aconteceu, porque ao fazer a alteração, não houve mudanças no registro, pois, os dados são os mesmos que estão na base de dados, não acontece mudanças na tabela.

2.9.6.6 Como você resolveria esse problema?

Uma forma seria, modificar a mensagem de retorno somente, não teria necessidade de criação de novos métodos, ou algo significativo no sistema, veja:

```

109     public function alterar($codigo, $sigla, $descricao, $usuario){
110         try{
111             if (trim($sigla) != ''){
112                 //Verificar se a unidade de medida já não está cadastrada
113                 $ret_unidMedida = $this->verificaUM($sigla);
114
115                 if ($ret_unidMedida['codigo'] == 2){
116                     $dados = $ret_unidMedida;
117                     return $dados;
118                 }
119
120             }
121
122             //Query de atualização dos dados
123             if (trim($sigla) != '' && trim($descricao) != ''){
124                 $sql = "update unid_medida set sigla = '$sigla', descricao = '$descricao'
125                         where sigla = '$sigla'";
126             }

```

```
125     }elseif(trim($sigla) != ''){
126         $sql = "update unid_medida set sigla = '$sigla' where sigla = '$sigla'";
127     }else{
128         $sql = "update unid_medida set descricao = '$descricao' where sigla = '$sigla'";
129     }
130
131     $this->db->query($sql);
132
133     //Verificar se a atualização ocorreu com sucesso
134     if($this->db->affected_rows() > 0){
135         //Fazemos a inserção no Log na nuvem
136         //Fazemos a instância da model M_log
137         $this->load->model('m_log');
138
139         //Fazemos a chamada do método de inserção do Log
140         $retorno_log = $this->m_log->inserir_log($usuario, $sql);
141
142         if ($retorno_log['codigo'] == 1){
143             $dados = array('codigo' => 1,
144                           'msg' => 'Unidade de medida atualizada corretamente');
145         }else{
146             $dados = array('codigo' => 7,
147                           'msg' => 'Houve algum problema no salvamento do Log, porém,
148                           |unidade de medida cadastrada corretamente');
149         }
150     }else{
151         $dados = array('codigo' => 6,
152                           'msg' => 'Não houve alteração no registro especificado.');
153     }
154
155 }catch (Exception $e) {
156     $dados = array('codigo' => 00,
157                   'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
158                   '$e->getMessage(), "\n");
159 }
160 //Envia o array $dados com as informações tratadas
161 //acima pela estrutura de decisão if
162 return $dados;
163 }
```

2.10. Atividade final de semestre

A missão agora é a implementação do objeto PRODUTO, onde é para criar o CRUD do mesmo, contemplando a *Controller* e a *Model*, devendo atender os campos criados na tabela de produtos idealizada em nossa práxis, se atente a mesma para realizar as implementações, após suas implementações, vocês devem subir o projeto no GitHub e juntamente com o relatório, contendo as explicações sobre as implementações realizadas por vocês.

Irei criar uma atividade no Microsoft Teams que vocês deverão enviar o link do GitHub do projeto de vocês, juntamente com o relatório. Este prazo será até o dia 09/12/2024 23:59.

2.10.1 Controller Produtos

A partir do modelo para as unidades de medida, foi criada a seguinte controller para produtos:

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

Class Produtos extends CI_Controller {
    //Atributos privados da classe
    private $codigo;
    private $descricao;
    private $unidMedida;
    private $estoqMinimo;
    private $estoqMaximo;
    private $dtCria;
    private $usuarioLogin;
    private $estatus;

    //Getters dos atributos
    public function getCodigo()
    {
        return $this->codigo;
    }

    public function getDescricao()
    {
        return $this->descricao;
    }

    public function getUnidMedida()
    {
        return $this->unidMedida;
    }

    public function getEstoqMinimo()
    {
        return $this->estoqMinimo;
    }

    public function getEstoqMaximo()
    {
        return $this->estoqMaximo;
    }

    public function getDtCria()
    {
        return $this->dtCria;
    }

    public function getUsuarioLogin()
    {
        return $this->usuarioLogin;
    }

    public function getEstatus()
    {
        return $this->estatus;
    }

    //Setters dos atributos
    public function setCodigo($codigoFront)
    {
        $this->codigo = $codigoFront;
    }

    public function setDescricao($descricaoFront)
    {
        $this->descricao = $descricaoFront;
    }

    public function setUnidMedida($unidMedidaFront)
    {
        $this->unidMedida = $unidMedidaFront;
    }

    public function setEstoqMinimo($estoqMinimoFront)
    {
        $this->estoqMinimo = $estoqMinimoFront;
    }

    public function setEstoqMaximo($estoqMaximoFront)
    {
        $this->estoqMaximo = $estoqMaximoFront;
    }

    public function setDtCria($dtCriaFront)
    {
        $this->dtCria = $dtCriaFront;
    }

    public function setUsuarioLogin($usuarioLoginFront)
    {
        $this->usuarioLogin = $usuarioLoginFront;
    }

    public function setEstatus($estatusFront)
    {
        $this->estatus = $estatusFront;
    }
}
```

Na mesma controller, foram criadas as funções de onde viriam as informações do frontend (inserir, consultar, alterar e desativar).

Inserir:

```

Public function inserir(){
    //Descrição, Unidade de medida, Estoque máximo, Estoque mínimo e Descrição
    //recebidas via JSON e colocadas em variáveis
    //Retornos possíveis:
    //1 - Produto cadastrado corretamente (Banco)
    //2 - Faltou informar a Unidade de medida (FrontEnd)
    //3 - Unidade de medida não encontrada no banco de dados (Banco)
    //4 - Descrição não informada (FrontEnd)
    //5 - Usuário não informado (FrontEnd)
    //6 - Houve algum problema no insert da tabela (Banco)
    //7 - Houve problema no salvamento do LOG, mas a unidade foi inclusa (LOG)
    //8 - Usuário desativado no banco de dados
    //9 - Usuário não encontrado
    //10 - Unidade de medida já cadastrada na base de dados (Model)
    //11 - Estoque mínimo não informado (FrontEnd)
    //12 - Estoque maximo não informado (FrontEnd)
    //13 - Estoque maximo e minimo iguais (FrontEnd)

    try{
        $json = file_get_contents('php://input');
        $resultado = json_decode($json);

        //Array com os dados que deverão vir do Front
        $lista = array(
            "descricao" => '0',
            "unidMedida" => '0',
            "estogMinimo" => '0',
            "estogMaximo" => '0',
            "usuarioLogin" => '0'
        );

        if(verificarParam($resultado, $lista) == 1){
            //Fazendo os seters
            $this->setDescricao($resultado->descricao);
            $this->setUnidMedida($resultado->unidMedida);
            $this->setEstoqMinimo($resultado->estogMinimo);
            $this->setEstoqMaximo($resultado->estogMaximo);
            $this->setUsuarioLogin($resultado->usuarioLogin);

            //Faremos uma validação para sabermos se todos os dados
            //foram enviados corretamente
            if(trim($this->getUnidMedida()) == ''){
                $retorno = array('codigo' => 2,
                    'msg' => 'Unidade de medida não informada.');
            }elseif(trim($this->getDescricao()) == ''){
                $retorno = array('codigo' => 4,
                    'msg' => 'Descrição não informada.');
            }elseif(trim($this->getUsuarioLogin()) == ''){
                $retorno = array('codigo' => 5,
                    'msg' => 'Usuário não informado');
            }elseif(trim($this->getEstoqMinimo()) == ''){
                $retorno = array('codigo' => 11,
                    'msg' => 'Estoque mínimo não informado.');
            }elseif(trim($this->getEstoqMaximo()) == ''){
                $retorno = array('codigo' => 12,
                    'msg' => 'Estoque maximo não informado.');
            }elseif(trim($this->getEstoqMinimo()) == trim($this->getEstoqMaximo())){
                $retorno = array('codigo' => 13,
                    'msg' => 'Estoque mínimo e maximo são iguais.');
            }elseif($this->getEstoqMinimo() > $this->getEstoqMaximo()){
                $retorno = array('codigo' => 14,
                    'msg' => 'Estoque mínimo maior que o maximo.');
            }else{
                //Realizo a instância da Model
                $this->load->model('m_produtos');

                //Atributo $retorno recebe array com informações
                $retorno = $this->m_produtos->inserir($this->getDescricao(), $this->getUnidMedida(), $this->getUsuarioLogin(), $this->getEstoqMinimo(), $this->getEstoqMaximo());
            }
        }
    }catch (Exception $e){
        $retorno = array('codigo' => 0,
            'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
            $e->getMessage());
    }
}

```

Consultar:

```

public function consultar(){
    //Código, Descrição, Unidade de medida e Usuário que criou
    //recebidos via JSON e colocados em variáveis
    //Retornos possíveis:
    //1 - Dados consultados corretamente (Banco)
    //2 - Quantidade de caracteres da Unidade de medida é superior a 3 (FrontEnd)
    //6 - Dados não encontrados (Banco)
    try{
        $json = file_get_contents('php://input');
        $resultado = json_decode($json);

        //Array com os dados que deverão vir do Front
        $lista = array(
            "codigo" => '0',
            "descricao" => '0',
            "unidMedida" => '0',
            "usuarioLogin" => '0'
        );

        if (verificarParam($resultado, $lista) == 1) {
            //Fazendo os seters
            $this->setCodigo($resultado->codigo);
            $this->setDescricao($resultado->descricao);
            $this->setUnidMedida($resultado->unidMedida);
            $this->setUsuarioLogin($resultado->usuarioLogin);

            //Realizo a instância da Model
            $this->load->model('_produtos');
            //Atributo $retorno recebe array com informações
            //da consulta dos dados
            $retorno = $this->m_produtos->consultar($this->getCodigo(),$this->getDescricao(), $this->getUnidMedida(),
                $this->getUsuarioLogin());

        }else {
            $retorno = array(
                'codigo' => 99,
                'msg' => 'Os campos vindos do FrontEnd não representam
                    o método de consulta. Verifique.'
            );
        }
    }catch (Exception $e){
        $retorno = array('codigo' => 0,
            'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
            $e->getMessage());
    }
    //Retorno no formato JSON
    echo json_encode($retorno);
}

```

```

public function alterar(){
    //Código, Unidade de medida e Descrição
    //recebidos via JSON e colocadas em variáveis
    //Retornos possíveis:
    //1 - Dado(s) alterado(s) corretamente (Banco)
    //2 - Faltou informar o código (FrontEnd)
    //3 - Quantidade de caracteres da unidade de medida é superior a 3 (FrontEnd)
    //4 - Unidade de medida ou Descrição não informadas, ai não tem o que alterar (FrontEnd)
    //5 - Usuário não informado (FrontEnd)
    //6 - Dados não encontrados (Banco)
    //7 - Houve problema no salvamento do LOG, mas a unidade foi inclusa (LOG)
    try{
        $json = file_get_contents('php://input');
        $resultado = json_decode($json);

        //Array com os dados que deverão vir do Front
        $lista = array(
            "codigo" => '0',
            "descricao" => '0',
            "unidMedida" => '0',
            "usuarioLogin" => '0'
        );

        if (verificarParam($resultado, $lista) == 1) {
            //Fazendo os seters
            $this->setCodigo($resultado->codigo);
            $this->setUnidMedida($resultado->unidMedida);
            $this->setDescricao($resultado->descricao);
            $this->setUsuarioLogin($resultado->usuarioLogin);

            //Faremos uma validação para sabermos se os dados
            //foram enviados corretamente
            if (trim($this->getCodigo()) == '' || trim($this->getCodigo()) == 0){
                $retorno = array('codigo' => 2,
                    'msg' => 'Codigo não informado.');
            }elseif (strlen(trim($this->getUnidMedida())) > 3){
                $retorno = array('codigo' => 3,
                    'msg' => 'Unidade de medida pode conter no máximo 3 caracteres.');
            }elseif (trim($this->getDescricao()) == '' && trim($this->getUnidMedida()) == ''){
                $retorno = array('codigo' => 4,
                    'msg' => 'Unidade de medida ou Descrição não foram informadas.');
            }elseif (trim($this->getUsuarioLogin()) == ''){
                $retorno = array('codigo' => 5,
                    'msg' => 'Usuário não informado');
            }else{
                //Realizo a instância da Model
                $this->load->model('m_unidmedida');

                //Atributo $retorno recebe array com informações
                //da validação do acesso
                $retorno = $this->m_unidmedida->alterar($this->getCodigo(), $this->getUnidMedida(),
                    $this->getDescricao(), $this->getUsuarioLogin());
            }
        }else {
            $retorno = array(
                'codigo' => 99,
                'msg' => 'Os campos vindos do FrontEnd não representam
                o método de alteração. Verifique.'
            );
        }
    }catch (Exception $e){
        $retorno = array('codigo' => 0,
            'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
            $e->getMessage());
    }
    //Retorno no formato JSON
    echo json_encode($retorno);
}

```

Desativar:

```

public function desativar(){
    //Código da unidade recebido via JSON e colocado em variável
    //Retornos possíveis:
    //1 - Unidade desativada corretamente (Banco)
    //2 - Código não informado;
    //3 - Existem produtos cadastrados com essa unidade de medida
    //5 - Usuário não informado (FrontEnd)
    //6 - Dados não encontrados (Banco)
    //7 - Houve problema no salvamento do LOG, mas a unidade foi alterada (LOG)
    try{
        $json = file_get_contents('php://input');
        $resultado = json_decode($json);

        //Array com os dados que deverão vir do Front
        $lista = array(
            "codigo" => '0',
            "usuarioLogin" => '0'
        );

        if (verificarParam($resultado, $lista) == 1) {
            //Fazendo os seters
            $this->setCodigo($resultado->codigo);
            $this->setUsuarioLogin($resultado->usuarioLogin);

            //Validação para tipo de usuário que deverá ser ADMINISTRADOR, COMUM ou VAZIO
            if (trim($this->getCodigo()) == '') || trim($this->getCodigo()) == 0 ){
                $retorno = array('codigo' => 2,
                    'msg' => 'Código da unidade não informado');
            }elseif (trim($this->getUsuarioLogin()) == ''){
                $retorno = array('codigo' => 5,
                    'msg' => 'Usuário não informado');
            }else{
                //Realizo a instância da Model
                $this->load->model('m_produtos');

                //Atributo $retorno recebe array com informações
                $retorno = $this->m_produtos->desativar($this->getCodigo(), $this->
getUsuarioLogin());
            }
        }else {
            $retorno = array(
                'codigo' => 99,
                'msg' => 'Os campos vindos do FrontEnd não representam
o método de desativação. Verifique.'
            );
        }
    }catch (Exception $e){
        $retorno = array('codigo' => 0,
            'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
            $e->getMessage());
    }
    //Retorno no formato JSON
    echo json_encode($retorno);
}

```

2.10.2 Model Produtos

Após a criação da Controller, é necessário a criação da model onde ela foi chamada no código, apesar de testes, infelizmente a function de Inserir ocorreu um erro do qual não soube como resolver:

```
<?php
defined('BASEPATH') or exit('No direct script access allowed');

class m_produtos extends CI_Model {
    public function inserir($descricao, $unidMedida, $usuario, $estoqMinimo, $estoqMaximo){
        // Inicializar $dados com valor padrão
        $dados = array('codigo' => -1, 'msg' => 'Nenhuma operação realizada');
        try {
            // Atributo $retorno recebe array com informações
            $ret_usuario = $this->verificaUsuario($usuario);

            // Validação se o usuário está com status válido
            if ($ret_usuario['codigo'] == 8 || $ret_usuario['codigo'] == 9) {
                $dados = $ret_usuario; // Retorno para controller com problema do usuário
            } else {
                // Verificar se a unidade de medida já está cadastrada
                $ret_produtos = $this->verificaUM($unidMedida);

                if ($ret_produtos['codigo'] == 2) {
                    // Query de inserção dos dados
                    $sql = "insert into produtos (descricao, unid_medida, usucria, estoqMaximo, estoqMinimo)
                            values ('$descricao', '$unidMedida', '$usuario', '$estoqMaximo', '$estoqMinimo')";
                    $this->db->query($sql);

                    // Verificar se a inserção ocorreu com sucesso
                    if ($this->db->affected_rows() > 0) {
                        // Inserção no log
                        $this->load->model('m_log');
                        $retorno_log = $this->m_log->inserir_log($usuario, $sql);

                        if ($retorno_log['codigo'] == 1) {
                            $dados = array('codigo' => 1, 'msg' => 'Produto cadastrado corretamente');
                        } else {
                            $dados = array(
                                'codigo' => 7,
                                'msg' => 'Houve algum problema no salvamento do Log, porém, Produto adicionado corretamente'
                            );
                        }
                    } else {
                        $dados = array(
                            'codigo' => 6,
                            'msg' => 'Houve algum problema na inserção na tabela de produtos $atributos'
                        );
                    }
                } else {
                    $dados = $ret_produtos; // Retorno de erro na verificação de unidade de medida
                }
            }
        } catch (Exception $e) {
            $dados = array(
                'codigo' => 00,
                'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ' . $e->getMessage()
            );
        }
    }

    // Retorna o array $dados com informações
    return $dados;
}
```

```

public function consultar($codigo, $descricao, $unidMedida, $usuarioLogin){
    //-----
    //Função que servirá para quatro tipos de consulta:
    // * Para todos as unidades de medida;
    // * Para uma determinada sigla de unidade;
    // * Para um código de unidade de medida;
    // * Para descrição da unidade de medida;
    //-----
    try{
        //Query para consultar dados de acordo com parâmetros passados
        $sql = "select * from produtos where estatus = ' ' ";

        if($codigo != '' && $codigo != 0) {
            $sql = $sql . "and cod_produto = '$codigo' ";
        }

        if($descricao != ''){
            $sql = $sql . "and descricao like '%$descricao%' ";
        }

        if($unidMedida != ''){
            $sql = $sql . "and unid_medida = '$unidMedida' ";
        }

        if($usuarioLogin != ''){
            $sql = $sql . "and usucria = '$usuarioLogin' ";
        }

        $retorno = $this->db->query($sql);

        //Verificar se a consulta ocorreu com sucesso
        if($retorno->num_rows() > 0){
            $dados = array('codigo' => 1,
                           'msg' => 'Consulta efetuada com sucesso',
                           'dados' => $retorno->result());
        }else{
            $dados = array('codigo' => 6,
                           'msg' => 'Dados não encontrados');
        }
    }catch (Exception $e) {
        $dados = array('codigo' => 00,
                           'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
                           $e->getMessage(), "\n");
    }
    //Envia o array $dados com as informações tratadas
    //acima pela estrutura de decisão if
    return $dados;
}

```

Função Alterar:

```

public function alterar($codigo, $descricao, $unidMedida, $usuario){
    try{
        if (trim($descricao) != ''){
            //Verificar se a unidade de medida já não está cadastrada
            $ret_produtos = $this->verificaUM($unidMedida);

            if ($ret_produtos['codigo'] == 2){
                $dados = $ret_produtos;
                return $dados;
            }
        }

        //Query de atualização dos dados
        if (trim($unidMedida) != '' && trim($descricao) != ''){
            $sql = "update produtos set unid_medida = '$unidMedida', descricao = '$descricao'
                    where cod_produto = '$codigo'";
        }elseif (trim($unidMedida) != ''){
            $sql = "update produtos set unid_medida = '$unidMedida' where cod_produto = '$codigo'";
        }else{
            $sql = "update produtos set descricao = '$descricao' where cod_produto = '$codigo'";
        }

        $this->db->query($sql);

        //Verificar se a atualização ocorreu com sucesso
        if($this->db->affected_rows() > 0){
            //Fazemos a inserção no Log na nuvem
            //Fazemos a instância da model M_log
            $this->load->model('m_log');

            //Fazemos a chamada do método de inserção do Log
            $retorno_log = $this->m_log->inserir_log($usuario, $sql);

            if ($retorno_log['codigo'] == 1){
                $dados = array('codigo' => 1,
                              'msg' => 'Produto atualizado corretamente');
            }else{
                $dados = array('codigo' => 7,
                              'msg' => 'Houve algum problema no salvamento do Log, porém,
                                         produto cadastrado corretamente');
            }
        }else{
            $dados = array('codigo' => 6,
                          'msg' => 'Não houve alteração no registro especificado.');
        }

    }catch (Exception $e) {
        $dados = array('codigo' => 00,
                      'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
                      $e->getMessage(), "\n");
    }
    //Envia o array $dados com as informações tratadas
    //acima pela estrutura de decisão if
    return $dados;
}

```

```

public function desativar($codigo, $usuario){
    try{

        //Query de atualização dos dados
        $sql2 = "update produtos set estatus = 'D' where cod_produto = '$codigo';

        $this->db->query($sql2);

        //Verificar se a atualização ocorreu com sucesso
        if($this->db->affected_rows() > 0){
            //Fazemos a inserção no Log na nuvem
            //Fazemos a instância da model M_log
            $this->load->model('m_log');

            //Fazemos a chamada do método de inserção do Log
            $retorno_log = $this->m_log->inserir_log($usuario, $sql2);

            if ($retorno_log['codigo'] == 1){
                $dados = array('codigo' => 1,
                              'msg' => 'Produto DESATIVADO corretamente');
            }else{
                $dados = array('codigo' => 8,
                              'msg' => 'Houve algum problema no salvamento do Log, porém,
                                         produto desativado corretamente');
            }

        }else{
            $dados = array('codigo' => 7,
                          'msg' => 'Houve algum problema na DESATIVAÇÃO do produto');
        }

    }catch (Exception $e) {
        $dados = array('codigo' => 00,
                      'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
                      $e->getMessage(), "\n");
    }
    //Envia o array $dados com as informações tratadas
    //acima pela estrutura de decisão if
    return $dados;
}

```

Funções de suporte:

Assim como foi feito com as unidades de medida, as funções de suporte ajudam com a integridade de dados evitando erro:

```

private function verificaUsuario($usuario) {
    try{
        //Função PRIVADA só para verificar se o usuário existe
        //em nosso banco de dados na tabela de usuários
        //Retornos:
        //1 - Usuário cadastrado na base de dados
        //8 - Usuário desativado no banco de dados
        //9 - Usuário não encontrado

        $sql = "select * from usuarios where usuario = '$usuario'  ";

        $retorno = $this->db->query($sql);

        //Verificar se a consulta trouxe algum usuário
        if($retorno->num_rows() > 0){
            //Verifico o status do usuário
            if($retorno->row()->estatus == 'D'){
                //Não se pode cadastrar o usuário
                $dados = array('codigo' => 8,
                               'msg' => 'Não pode cadastrar, usuário informado está DESATIVADO');
            }else{
                $dados = array('codigo' => 1,
                               'msg' => 'Usuário ativo na base de dados');
            }
        }else{
            $dados = array('codigo' => 9,
                           'msg' => 'Usuário não encontrado na base de dados');
        }
    }catch (Exception $e) {
        $dados = array('codigo' => 00,
                       'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ',
                       $e->getMessage(), "\n");
    }
    //Envia o array $dados com as informações tratadas
    //acima pela estrutura de decisão if
    return $dados;
}

```

Função verifica unidade de medida:

```

public function verificaUM($unidMedida) {
    try {
        // Inicializa a variável $dados com um valor padrão
        $dados = array('codigo' => -1, 'msg' => 'Nenhuma operação realizada.');

        // Query para consultar a unidade de medida
        $sql = "select * from unid_medida
                where cod_unidade = '$unidMedida'
                and estatus = '' ";

        $retorno = $this->db->query($sql);

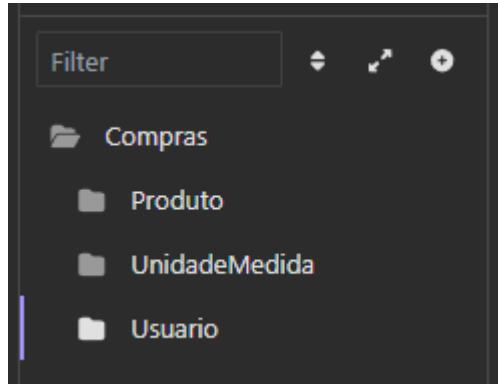
        if ($retorno->num_rows() == 0) {
            $dados = array(
                'codigo' => 2,
                'msg' => 'Unidade de medida não cadastrada.'
            );
        } else {
            $dados = array(
                'codigo' => 1,
                'msg' => 'Unidade de medida encontrada.'
            );
        }
    } catch (Exception $e) {
        $dados = array(
            'codigo' => 0,
            'msg' => 'ATENÇÃO: O seguinte erro aconteceu -> ' . $e->getMessage()
        );
    }

    // Retorna o array $dados
    return $dados;
}

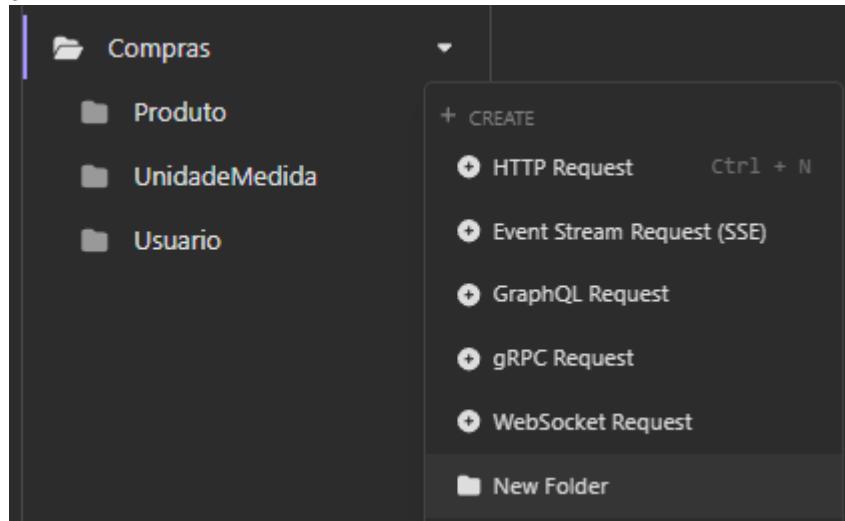
```

2.10.3 Criação no Insomnia

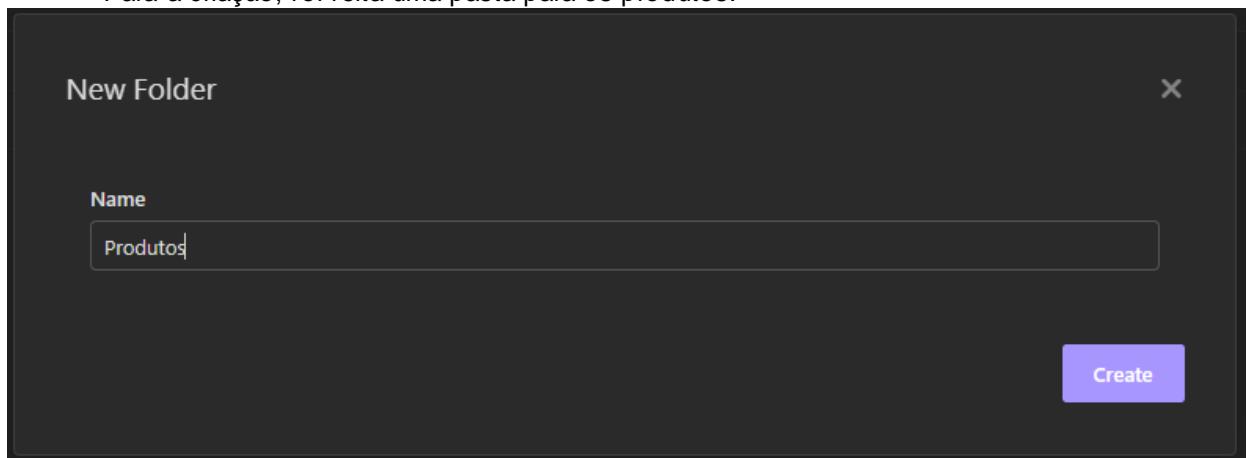
Como os testes são feitos a partir do Insomnia, o teste de criação das APIs será realizada por lá para verificar como está funcionando o CRUDE dos dados.



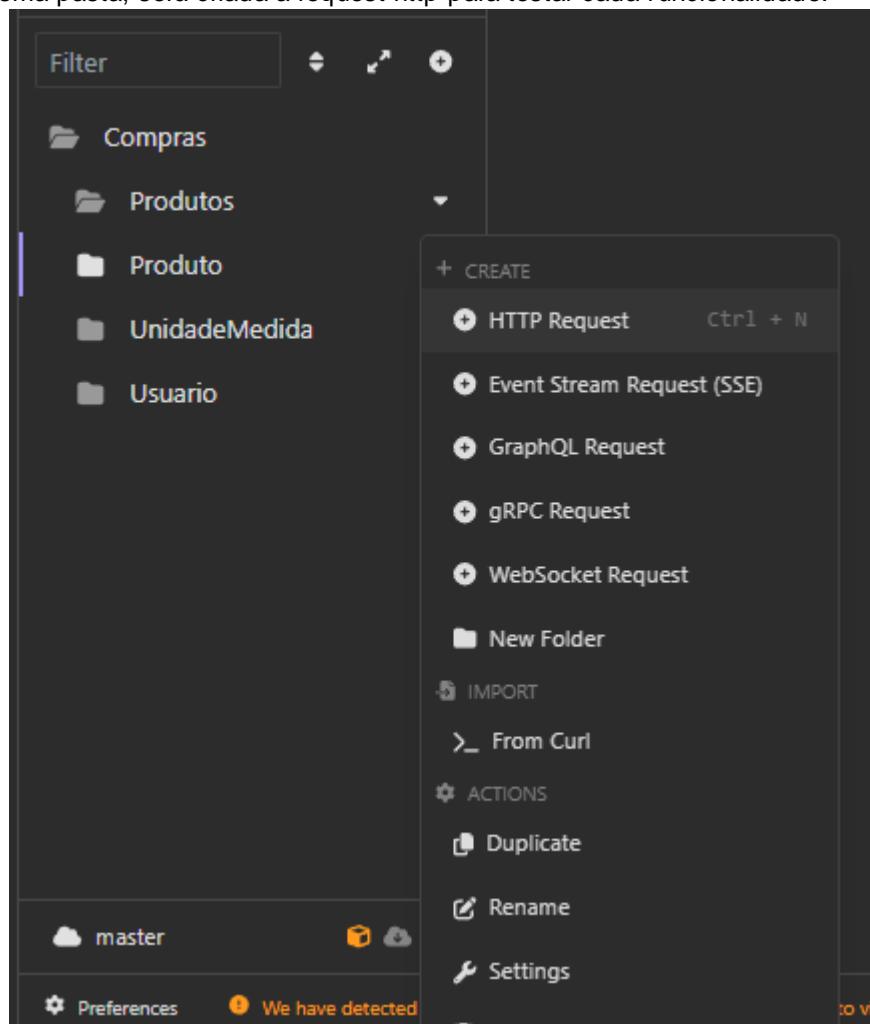
Nos prints para esta atividade já existirá uma pasta para produto no qual já foi feita para a criação da atividade.



Para a criação, foi feita uma pasta para os produtos.

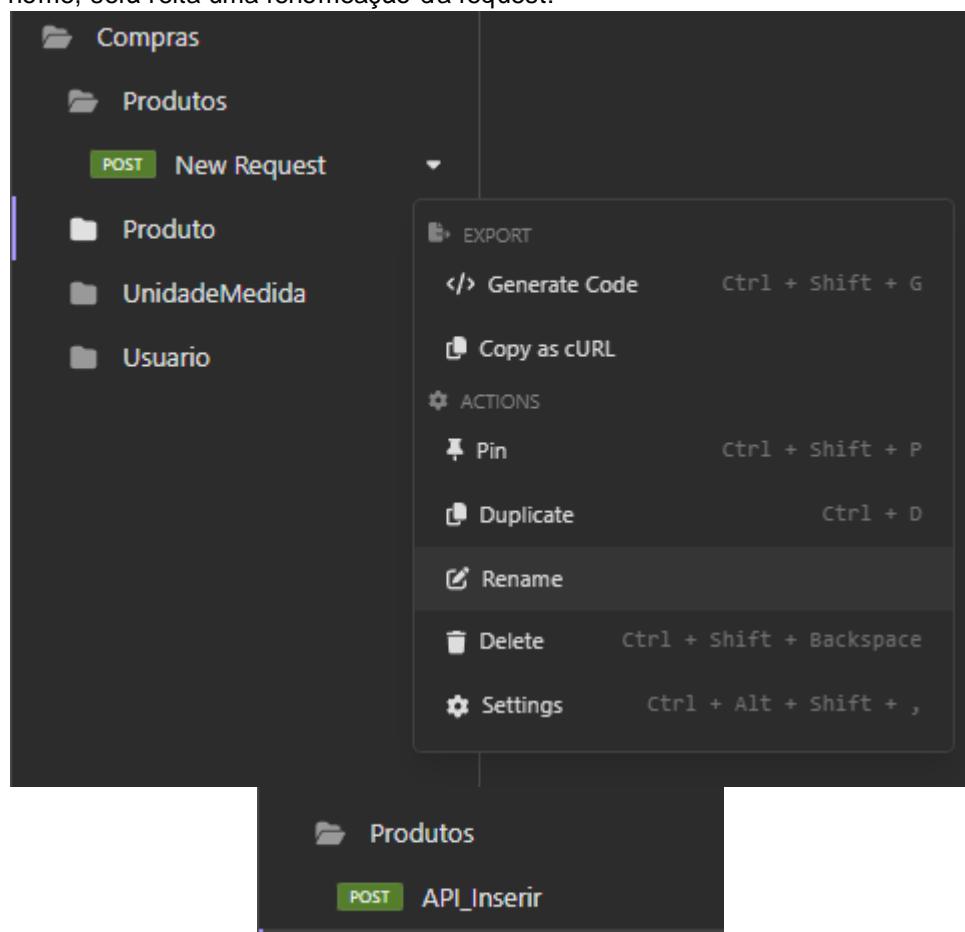


Na mesma pasta, será criada a request http para testar cada funcionalidade:



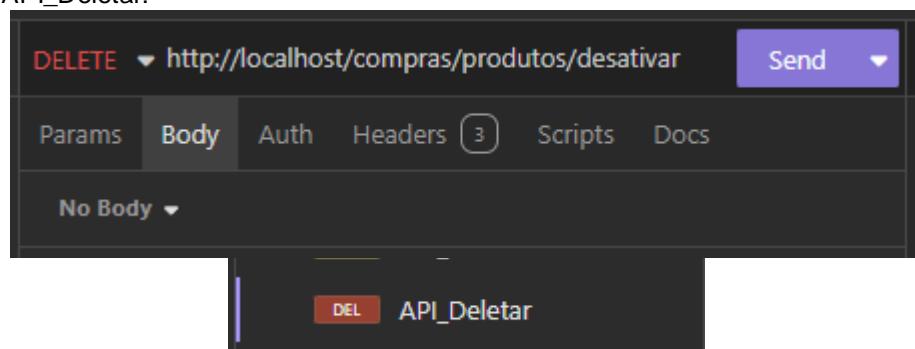
Para a API_Inserir, foram feitas as seguintes configurações:

Para o nome, será feita uma renomeação da request:

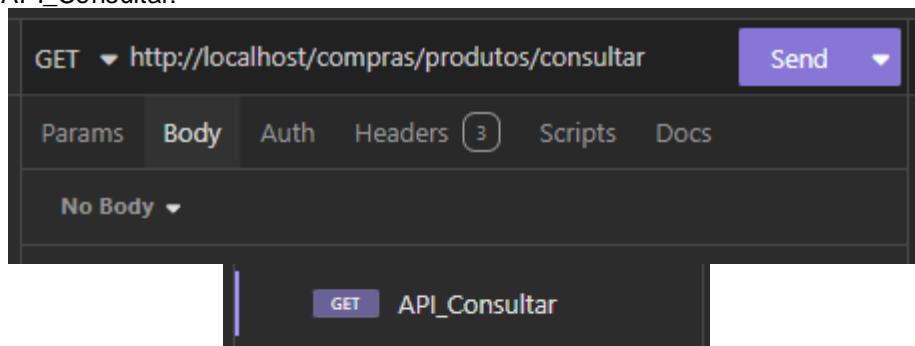


Seguindo o mesmo padrão, as novas requests foram criadas:

Request API_Deletar:



Request API_Consultar:



Request API_Alterar:

PATCH ▾ http://localhost/compras/produtos/alterar Send ▾

Params Body Auth Headers (3) Scripts Docs

No Body ▾

PTCH API_Alterar

Com os bodys de cada request, foram feitos cada JSON baseando nas controllers e models:

Body de API_Consultar:

GET ▾ http://localhost/compras/produtos/consultar Send ▾

Params Body (1) Auth Headers (4) Scripts Docs

JSON ▾

```

1 {
2   "codigo":"",
3   "descricao":"",
4   "unidMedida":"",
5   "usuarioLogin":""
6 }

```

Body de API_Alterar:

PATCH ▾ http://localhost/compras/produtos/alterar Send ▾

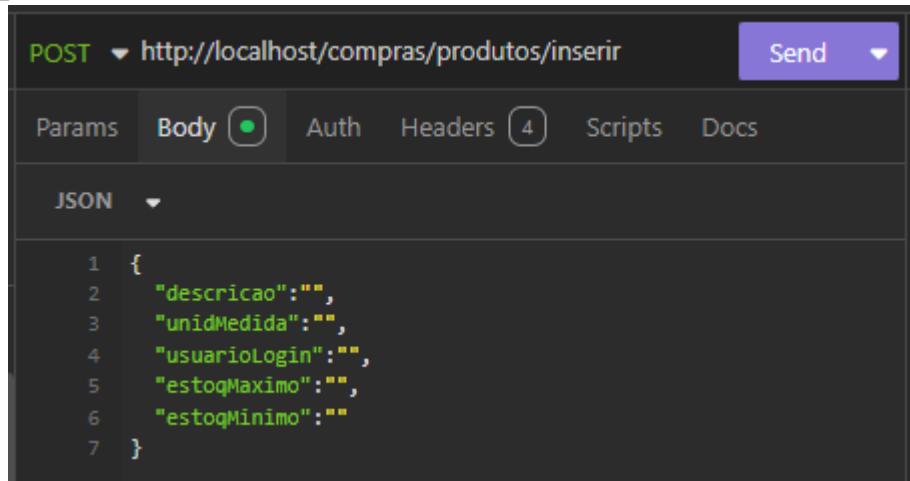
Params Body (1) Auth Headers (4) Scripts Docs

JSON ▾

```

1 {
2   "codigo":"",
3   "descricao":"",
4   "unidMedida":"",
5   "usuarioLogin":""
6 }

```



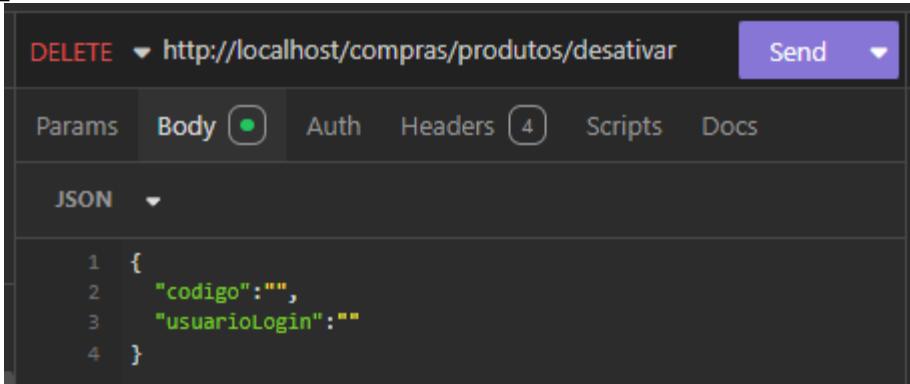
POST ▼ http://localhost/compras/produtos/inserir Send ▼

Params	Body	Auth	Headers 4	Scripts	Docs
--------	------	------	-----------	---------	------

JSON ▼

```
1 {
2   "descricao":"",
3   "unidMedida":"",
4   "usuarioLogin":"",
5   "estoqMaximo":"",
6   "estoqMinimo""
7 }
```

Body de API_Deletar:



DELETE ▼ http://localhost/compras/produtos/desativar Send ▼

Params	Body	Auth	Headers 4	Scripts	Docs
--------	------	------	-----------	---------	------

JSON ▼

```
1 {
2   "codigo":"",
3   "usuarioLogin""
4 }
```

2.10.4 Testes no Insomnia

Para iniciar o teste, será feito a ação de inserir um produto:

```
POST ▼ http://localhost/compras/produtos/inserir Send ▼
Params Body Auth Headers 4 Scripts Docs
JSON ▼
1 ▷ {
2   "descricao": "Vinho Azul",
3   "unidMedida": "2",
4   "usuarioLogin": "admin",
5   "estoqMaximo": "100",
6   "estoqMinimo": "10"
7 }
```

Porém, a única função da qual não consegui resolver foi a de inserir, onde a resposta para ela resulta em uma tela branca no insomnia:

```
POST ▼ http://localhost/compras/produtos/inserir Send ▼ 200 OK 136 ms 1 B Just Now ▼
Params Body Auth Headers 4 Scripts Docs Preview Headers 5 Cookies Tests 0/0 → Mock Console
JSON ▼ Preview ▼
1 ▷ {
2   "descricao": "Vinho Azul",
3   "unidMedida": "2",
4   "usuarioLogin": "admin",
5   "estoqMaximo": "100",
6   "estoqMinimo": "10"
7 }
```

Para a continuação dos testes, foi adicionado diretamente um item no banco de dados:

```
7 •      insert into produtos (descricao, unid_medida, usucria, estoq_Maximo, estoq_mininmo)
8           values ("Vinho azul", 2, "admin", 100, 10);
<
```

Result Grid								
	cod_produto	descricao	unid_medida	estoa_minimo	estoa_maximo	dtria	usucria	estatus
▶	1	Vinho azul	2	10	100	2024-12-07 22:36:18	admin	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Agora para o segundo teste, será realizada a consulta do produto:

Sem valores no body:

```
GET ▼ http://localhost/compras/produtos/consultar Send ▼ 200 OK 128 ms 228 B Just Now ▼
Params Body Auth Headers 4 Scripts Docs Preview Headers 5 Cookies Tests 0/0 → Mock Console
JSON ▼ Preview ▼
1 ▷ {
2   "codigo": "",
3   "descricao": "",
4   "unidMedida": "",
5   "usuarioLogin": ""
6 }
1 ▷ {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_produto": "1",
7       "descricao": "Vinho azul",
8       "unid_medida": "2",
9       "estoa_minimo": "10",
10      "estoa_maximo": "100",
11      "dtria": "2024-12-07 22:36:18",
12      "usucria": "admin",
13      "estatus": ""
14    }
15  ]
16 }
```

Somente com valor em código:

GET ▼ http://localhost/compras/produtos/consultar	Send ▾	200 OK	76 ms	228 B	Just Now ▾
Params	Body <input checked="" type="radio"/>	Auth	Headers 4	Scripts	Docs
JSON ▾			Preview ▾		
<pre> 1 v { 2 "codigo": "1", 3 "descricao": "", 4 "unidMedida": "", 5 "usuarioLogin": "" 6 }</pre>			<pre> 1 v { 2 "codigo": 1, 3 "msg": "Consulta efetuada com sucesso", 4 "dados": [5 { 6 "cod_produto": "1", 7 "descricao": "Vinho azul", 8 "unid_medida": "2", 9 "estq_minimo": "10", 10 "estq_maximo": "100", 11 "dtcria": "2024-12-07 22:36:18", 12 "usucria": "admin", 13 "estatus": "" 14 } 15] 16 }</pre>		

Somente com valor em descrição:

GET ▼ http://localhost/compras/produtos/consultar	Send ▾	200 OK	62 ms	228 B	Just Now ▾
Params	Body <input checked="" type="radio"/>	Auth	Headers 4	Scripts	Docs
JSON ▾			Preview ▾		
<pre> 1 { 2 "codigo": "", 3 "descricao": "Vinho Azul", 4 "unidMedida": "", 5 "usuarioLogin": "" 6 }</pre>			<pre> 1 { 2 "codigo": 1, 3 "msg": "Consulta efetuada com sucesso", 4 "dados": [5 { 6 "cod_produto": "1", 7 "descricao": "Vinho azul", 8 "unid_medida": "2", 9 "estq_minimo": "10", 10 "estq_maximo": "100", 11 "dtcria": "2024-12-07 22:36:18", 12 "usucria": "admin", 13 "estatus": "" 14 } 15] 16 }</pre>		

Somente com valor unidMedida:

GET ▼ http://localhost/compras/produtos/consultar	Send ▾	200 OK	59 ms	228 B	Just Now ▾
Params	Body <input checked="" type="radio"/>	Auth	Headers 4	Scripts	Docs
JSON ▾			Preview ▾		
<pre> 1 v { 2 "codigo": "", 3 "descricao": "", 4 "unidMedida": "2", 5 "usuarioLogin": "" 6 }</pre>			<pre> 1 v { 2 "codigo": 1, 3 "msg": "Consulta efetuada com sucesso", 4 "dados": [5 { 6 "cod_produto": "1", 7 "descricao": "Vinho azul", 8 "unid_medida": "2", 9 "estq_minimo": "10", 10 "estq_maximo": "100", 11 "dtcria": "2024-12-07 22:36:18", 12 "usucria": "admin", 13 "estatus": "" 14 } 15] 16 }</pre>		

Somente com valor usuárioLogin:

```

GET ▼ http://localhost/compras/produtos/consultar Send ▼ 200 OK 61 ms 228 B Just Now ▾
Params Body (1) Auth Headers (4) Scripts Docs Preview Headers (5) Cookies Tests 0 / 0 → Mock Console
JSON ▾ Preview ▾
1 + {
2   "codigo": "",
3   "descricao": "",
4   "unidMedida": "",
5   "usuarioLogin": "admin"
6 }
1 - {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_produto": "1",
7       "descricao": "Vinho azul",
8       "unid_medida": "2",
9       "estoq_mininmo": "10",
10      "estoq_maximo": "100",
11      "dtcria": "2024-12-07 22:36:18",
12      "usucria": "admin",
13      "estatus": ""
14    }
15  ]
16 }

```

Com todos os valores:

```

GET ▼ http://localhost/compras/produtos/consultar Send ▼ 200 OK 68 ms 228 B Just Now ▾
Params Body (1) Auth Headers (4) Scripts Docs Preview Headers (5) Cookies Tests 0 / 0 → Mock Console
JSON ▾ Preview ▾
1 + {
2   "codigo": "1",
3   "descricao": "Vinho Azul",
4   "unidMedida": "2",
5   "usuarioLogin": "admin"
6 }
1 - {
2   "codigo": 1,
3   "msg": "Consulta efetuada com sucesso",
4   "dados": [
5     {
6       "cod_produto": "1",
7       "descricao": "Vinho azul",
8       "unid_medida": "2",
9       "estoq_mininmo": "10",
10      "estoq_maximo": "100",
11      "dtcria": "2024-12-07 22:36:18",
12      "usucria": "admin",
13      "estatus": ""
14    }
15  ]
16 }

```

Teste em alterar:

(Em alterar ocorreu um erro do qual não poderá ser corrigido a tempo)
e uma visão de como foi feita a mudança:

(Em alterar ocorreu um erro do qual não poderá ser corrigido a tempo)

Teste em Deletar:

```

DELETE ▼ http://localhost/compras/produtos/desativar Send ▼ 200 OK 894 ms 63 B Just Now ▾
Params Body (1) Auth Headers (4) Scripts Docs Preview Headers (5) Cookies Tests 0 / 0 → Mock Console
JSON ▾ Preview ▾
1 + {
2   "codigo": "1",
3   "usuarioLogin": "admin"
4 }
1 - [
2   "codigo": 1,
3   "msg": "Unidade de medida DESATIVADA corretamente"
4 ]

```

E a visão de como foi feita a mudança:

```

5 • select * from produtos;
| Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | | | |
| cod_produto | descricao | unid_medida | estoq_mininmo | estoq_maximo | dtcria | usucria | estatus |
| 3 | Vinho azul | 2 | 10 | 100 | 2024-12-07 23:33:49 | admin | A |
| * | NULL |

5 • select * from produtos;
| Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | | | |
| cod_produto | descricao | unid_medida | estoq_mininmo | estoq_maximo | dtcria | usucria | estatus |
| 3 | Vinho azul | 2 | 10 | 100 | 2024-12-07 23:33:49 | admin | D |
| * | NULL |

```