

Monitoratge de servidors a temps real

Aplicacions i Serveis d'Internet

ERIK C.

ALEIX G.

TONI S.

OSCAR V.

Introducció

En aquest projecte final de l'assignatura hem creat una aplicació d'Internet on un usuari pot veure a temps real totes les dades dels seus servidors. Podrem veure percentatges d'espai lliure o utilitzat en els servidors, utilització de la xarxa, utilització de la cpu o també els processos que hi corren.

Per poder dur a terme aquest projecte, hem instal·lat els paquets llistats en el fitxer a les nostres màquines per: Instal·lar les llibreries necessàries per poder treballar amb la framework de Laravel i per poder obtenir mitjançant un script de python3 les dades del servidor.

Arxiu que ens permet instal·lar tot el necessari per dur a terme el monitoratge i treballar amb la frame:

```
1 #!/bin/sh
2 sudo apt-get update
3 sudo apt-get upgrade
4 sudo apt-get install composer
5 sudo apt-get install -y git curl wget zip unzip
6 sudo apt-get install apache2
7 sudo a2enmod rewrite
8 sudo systemctl restart apache2
9 sudo apt-get install mysql-server
10 mysql_secure_installation
11 sudo add-apt-repository -y ppa:ondrej/php
12 sudo apt-get update
13 sudo apt-get install -y php7.2 php7.2-fpm libapache2-mod-php7.2 php7.2-cli php7.2-curl php7.2-mysql php7.2-sqlite3 php7.2-gd php7.2-xml php7.1-mcrypt php7.2-mbstring php7.2-iconv
14 curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
15 sudo chown -R $USER $HOME/.composer
16 sudo add-apt-repository ppa:jonathonf/python-3.6
17 sudo apt update
18 sudo apt install python3.6
19 sudo apt install python3.6-dev
20 sudo apt install python3.6-venv
21 wget https://bootstrap.pypa.io/get-pip.py
22 sudo python3.6 get-pip.py
23 sudo ln -s /usr/bin/python3.6 /usr/local/bin/python3
24 sudo ln -s /usr/local/bin/pip /usr/local/bin/pip3
25 sudo pip3.6 install psutil
26 sudo apt-get update
27 sudo apt-get upgrade
```

Codi 1: "installer"

Aquest arxiu instal·larà a grans trets: Python3.6 en l'última versió disponible i la llibreria psutils per poder obtenir les dades del monitoratge; PHP a una versió suficient per poder usar Laravel, MySQL per poder gestionar la base de dades, el composer també per Laravel i finalment Apache2 per si es vol configurar aquest pel projecte.

Python

Per obtenir les dades del sistem que monitorarem a temps real hem utilitzat la llibreria psutils. Hem implementat un script de Python3.6 que recull aquestes dades.

Per afegir un nou servidor al sistema de monitoratge, cal executar aquest arxiu amb els següents paràmetres:

UBUNTU

```
./monitoring API_TOKEN SERVER_NAME INTERVAL(sec)
```

WINDOWS

```
./monitoring.exe API_TOKEN SERVER_NAME INTERVAL(sec)
```

API_TOKEN: Obligatori. Serveix per autenticar-te al servidor.

SERVER_NAME: Obligatori. Serveix per determinar el nom del servidor a utilitzar. En crea un nou si no existeix.

INTERVAL(sec): Opcional. Serveix per indicar el temps d'actualització de les dades. Per defecte són 60sec.

Aquestes dades obtingudes per pas de paràmetre, juntament amb les dades obtingudes amb la llibreria psutils, s'envien amb post en JSON a una URL.

```
1 import psutil, requests, json, time, sys, socket, platform
2
3 # python3.6
4 # psutil
5 # pip3
6
7 def dsk():
8     prt = psutil.disk_partitions()
9     df = {}
10    for i in prt:
11        di = { i.device : {
12            'mount_point': i.mountpoint,
13            'file_system': i.fstype,
14            'total': psutil.disk_usage(i.mountpoint).total,
15            'used': psutil.disk_usage(i.mountpoint).used,
16            'free': psutil.disk_usage(i.mountpoint).free,
17            'percent': psutil.disk_usage(i.mountpoint).percent
18        } }
19        df = {**df, **di}
20    return df
21
22 def conStats():
23     con = psutil.net_connections()
24     stable = 0
25     listen = 0
26     for i in con:
27         cond = i.status
28         if cond == 'ESTABLISHED':
29             stable += 1
30         elif cond == 'LISTEN':
31             listen += 1
32     return { 'total': len(con), 'stable': stable, 'listen': listen }
33
34 def addrs():
35     ad = psutil.net_if_addrs()
36     fl = {}
37     for e in ad.keys():
38         for i in ad[e]:
39             if i.family == socket.AF_INET:
40                 sl = { e : i.address }
41                 fl = {**fl, **sl}
```

```

42         break
43     return fl
44
45 def data():
46
47     url = 'https://moni.erik.cat/api/data?api_token=' + sys.argv[1]
48     payload = {
49         'server': sys.argv[2],
50         'os': platform.system(),
51         'version': platform.release(),
52         'platform': platform.platform(),
53         'processor': platform.processor(),
54         'node': platform.node(),
55         'data': {
56             'cpu': {
57                 'cores': psutil.cpu_count(),
58                 'percent': psutil.cpu_percent(), # Total
59                 'frequency': psutil.cpu_freq().current,
60                 'min_frequency': psutil.cpu_freq().min,
61                 'max_frequency': psutil.cpu_freq().max
62             },
63             'mem': { # System
64                 'total': psutil.virtual_memory().total,
65                 'available': psutil.virtual_memory().available,
66                 'used': psutil.virtual_memory().used,
67                 'percent': psutil.virtual_memory().percent,
68                 'free': psutil.virtual_memory().free,
69             },
70             'disks': dsk(),
71             'net': conStats(),
72             'addresses': addrs(),
73             'pids': len(psutil.pids())
74         }
75     }
76
77     try:
78         r = requests.post(url, json = payload, headers = {'Accept': 'application/json'})
79     except:
80         if r.status_code == 200:
81             print('Data sent to the server')
82         else:
83             print(r.json())
84             print('There was an error with the request.')
85     except:
86         print('There was an error with the request.')
87
88 def main(interval):
89     while True:
90         d = data()
91         time.sleep(interval)
92
93 if __name__ == "__main__":
94     if len(sys.argv) == 1:
95         print('The api_token is missing, please enter the API token as the first parameter')
96         sys.exit(0)
97     if len(sys.argv) == 2:
98         print('The server_name is missing, please enter a server name as a second parameter. It will be created if it did not exist')
99         sys.exit(0)
100     if len(sys.argv) == 3:
101         interval = 60
102     elif len(sys.argv) > 3:
103         interval = sys.argv[3]
104         try:
105             interval = int(interval)
106         except:
107             print('Wrong interval')

```

```
107         sys.exit(0)
108     print('Interval set to ' + str(interval) + ' seconds')
109     main(interval)
```

Codi 2: "Monitoring.py"

PHP

Malgrat podríem fer aquest projecte únicament amb aquest llenguatge però, l'utilitzarem principalment per utilitzar la framework.

Principalment usarem la comanda: `php artisan comand` per poder fer moltes coses.

Laravel, PHP i MySQL

Un cop ens van arribant les dades des de l'arxiu de python, les hem de classificar segons el que vulguem veure. Hem creat diferents mòduls on a cada mòdul farem diferents relacions i classifcarem les dades:

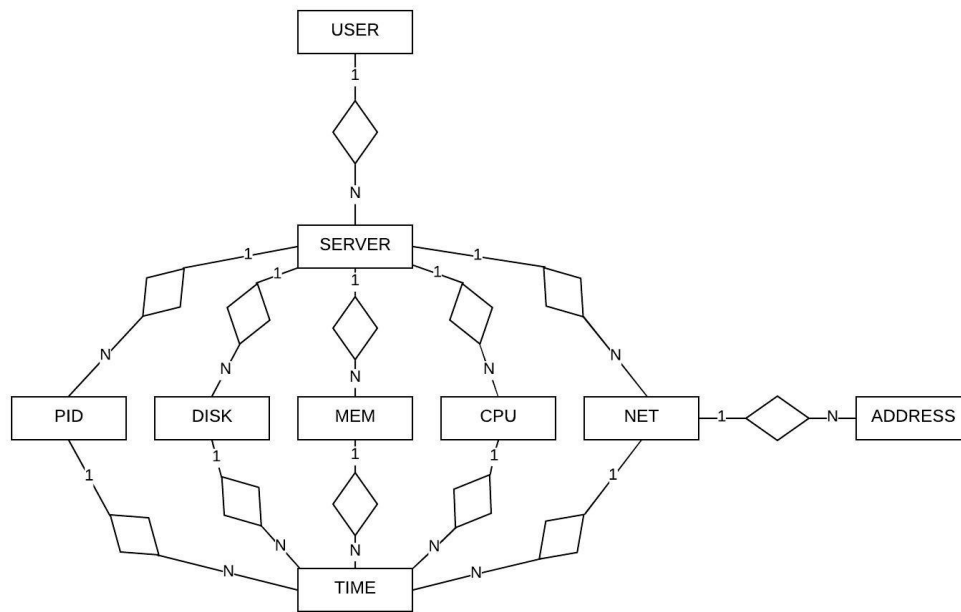


Figura 1: Esquema mòduls

Gracies a l'ORM de laravel, podem traduir els diferents mòduls a una base de dades amb MySQL i es podrà gestionar la base de dades.

User

Principalment tenim el modul **user**, on demanem a l'usuari que ens doni un nom, el seu email i la contrasenya, i crearem un token (anomenat `api_token`).

Per fer-ho segur tenim la funcio `setPasswordAttribute`, que aquí creem el hash de la contrasenya. Per últim veiem l'última funció `servers`, on fem la relació amb l'esquema anterior, és a dir, un usuari pot tenir diversos servidors.

```

1 <?php
2
3 namespace App;
4
5 use Illuminate\Notifications\Notifiable;
6 use Illuminate\Foundation\Auth\User as Authenticatable;

```

```

7
8 class User extends Authenticatable
9 {
10     use Notifiable;
11
12     /**
13      * The attributes that are mass assignable.
14      *
15      * @var array
16      */
17     protected $fillable = [
18         'name', 'email', 'password', 'api_token'
19     ];
20
21     /**
22      * The attributes that should be hidden for arrays.
23      *
24      * @var array
25      */
26     protected $hidden = [
27         'password', 'remember_token',
28     ];
29
30     /**
31      * Mutator to automatically hash the user password.
32      *
33      * @param string $value
34      * @return void
35      */
36     public function setPasswordAttribute($value)
37     {
38         $this->attributes['password'] = bcrypt($value);
39     }
40
41     /**
42      * Return the user servers.
43      *
44      * @return Collection
45      */
46     public function servers()
47     {
48         return $this->hasMany(Server::class);
49     }
50 }

```

Codi 3: "User.php"

Server

En el modul **server** primerament rebrem el món, el sistema operatiu actiu, la seva versió, el processador, node i la plataforma utilitzada, i tot un seguit de funcions on podem veure la relació amb la resta de mòduls. Primer tenim la relació amb user, on veiem que el servidor només correspon a un únic usuari, però les següents relacions podem comprobar com per exemple un servidor pot tenir diversos processos (`return $this->hasMany(Pid::class);`), i així amb els moduls **net**, **Disk**, **Cpu**, **Mem** i **Net**.

```

1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Server extends Model
8 {
9     /**

```

```
10     * Mass assignable attributes.
11     *
12     * @var array
13     */
14     public $fillable = [
15         'name', 'os', 'version', 'processor', 'node', 'platform'
16     ];
17
18     /**
19     * Returns the server user.
20     *
21     * @return App\User
22     */
23     public function user()
24     {
25         return $this->belongsTo(User::class);
26     }
27
28     /**
29     * Returns the server PIDs.
30     *
31     * @return App\Pid
32     */
33     public function pids()
34     {
35         return $this->hasMany(Pid::class);
36     }
37
38     /**
39     * Returns the server NET.
40     *
41     * @return App\Net
42     */
43     public function nets()
44     {
45         return $this->hasMany(Net::class);
46     }
47
48     /**
49     * Returns the server memory.
50     *
51     * @return App\Mem
52     */
53     public function mems()
54     {
55         return $this->hasMany(Mem::class);
56     }
57
58     /**
59     * Returns the server CPU.
60     *
61     * @return App\Cpu
62     */
63     public function cpus()
64     {
65         return $this->hasMany(Cpu::class);
66     }
67
68     /**
69     * Returns the server DISK.
70     *
71     * @return App\Disk
72     */
73     public function disks()
74     {
75         return $this->hasMany(Disk::class);
76     }
77 }
```


Codi 4: "Server.php"

Net, Pid, Disk, Cpu, Mem, Net

En aquests mòduls pràcticament farem el mateix en tots. De tota la informació que ens arriba agafem la que ens sigui necessària segons el modul, i fem la seva relació amb la resta de mòduls. Per exemple en el mòdul **Disk** ens interès el seu espai lliure, total, ocupat i el percentatge corresponent, i després fem la relació amb el mòdul **Server** i **Time**

```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Disk extends Model
8 {
9     /**
10      * Mass assignable attributes.
11      *
12      * @var array
13      */
14     public $fillable = [
15         'device', 'mount_point', 'file_system', 'total', 'used', 'free', 'percent'
16     ];
17
18     /**
19      * Returns the disk server.
20      *
21      * @return App\Server
22      */
23     public function server()
24     {
25         return $this->belongsTo(Server::class);
26     }
27
28     /**
29      * Return the disk time.
30      *
31      * @return App\Time
32      */
33     public function time()
34     {
35         return $this->belongsTo(Time::class);
36     }
37 }
```

Codi 5: "Disk.php"

Address

Mostra totes les adreces IP del teu servidor amb les interfícies corresponent. Aquest és relaciona amb el modul **Net**, ja que una xarxa pot tenir diferents IP.

```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
```

```

7 class Address extends Model
8 {
9     /**
10      * Mass assignable attributes.
11      *
12      * @var array
13      */
14     public $fillable = [
15         'name', 'ip'
16     ];
17
18     /**
19      * Returns the user.
20      *
21      * @return void
22      */
23     public function user()
24     {
25         return $this->belongsTo(Net::class);
26     }
27 }

```

Codi 6: "Address.php"

Controller

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\User;
6 use Illuminate\Http\Request;
7
8 class MoniController extends Controller
9 {
10     /**
11      * Stores the server information.
12      *
13      * @param Request $request
14      * @return array
15      */
16     public function data(Request $request)
17     {
18         $user = auth('api')->user();
19
20         $server = $user->servers()->where('name', $request->get('server'))->
firstOrCreate([
21             'name' => $request->get('server'),
22             'os' => $request->get('os'),
23             'version' => $request->get('version'),
24             'node' => $request->get('node'),
25             'processor' => $request->get('processor'),
26             'platform' => $request->get('platform'),
27         ]);
28
29         $server->update([
30             'os' => $request->get('os'),
31             'version' => $request->get('version'),
32             'node' => $request->get('node'),
33             'processor' => $request->get('processor'),
34             'platform' => $request->get('platform'),
35         ]);
36
37         $data = $request->get('data');
38

```

```

39 // CPU
40 $server->cpus()->updateOrcreate([
41     'time_id' => null
42 ], [
43     'cores' => $data['cpu']['cores'],
44     'percent' => $data['cpu']['percent'],
45     'frequency' => $data['cpu']['frequency'],
46     'min_frequency' => $data['cpu']['min_frequency'],
47     'max_frequency' => $data['cpu']['max_frequency']
48 ]);
49
50 // Disks
51 $server->disks()->where('time_id', null)->delete();
52
53 foreach ($data['disks'] as $key => $disk) {
54     $server->disks()->create([
55         'device' => $key,
56         'mount_point' => $disk['mount_point'],
57         'file_system' => $disk['file_system'],
58         'total' => round($disk['total'] * 9.31e-10),
59         'used' => round($disk['used'] * 9.31e-10),
60         'free' => round($disk['free'] * 9.31e-10),
61         'percent' => $disk['percent']
62     ]);
63 }
64
65 // Mem
66 $server->mems()->updateOrcreate([
67     'time_id' => null
68 ], [
69     'total' => round($data['mem']['total'] * 9.31e-10),
70     'available' => round($data['mem']['available'] * 9.31e-10),
71     'used' => round($data['mem']['used'] * 9.31e-10),
72     'percent' => $data['mem']['percent'],
73     'free' => round($data['mem']['free'] * 9.31e-10)
74 ]);
75
76 // Net
77 $net = $server->nets()->updateOrcreate([
78     'time_id' => null
79 ], [
80     'total' => $data['net']['total'],
81     'stable' => $data['net']['stable'],
82     'listen' => $data['net']['listen']
83 ]);
84
85 $net->addresses()->delete();
86 foreach ($data['addresses'] as $key => $address) {
87     $net->addresses()->create([
88         'name' => $key,
89         'ip' => $address
90     ]);
91 }
92
93 // Pid
94 $server->pids()->updateOrcreate([
95     'time_id' => null
96 ], [
97     'number' => $data['pids']
98 ]);
99
100 return [
101     'response' => 'ok'
102 ];
103 }
104 }

```

Codi 7: "Time.php"

Routes:

En el projecte de Laravel usar 2 routes una per obtenir la interfície i una altre pel backend.

Route de backend:

```
1 <?php
2
3 use Illuminate\Http\Request;
4
5 /*
6 |
7 | API Routes
8 |
9 |
10 | Here is where you can register API routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | is assigned the "api" middleware group. Enjoy building your API!
13 |
14 */
15
16 Route::post('/login', 'AppController@login');
17 Route::post('/register', 'AppController@register');
18
19 Route::middleware('auth:api')->group(function () {
20     Route::post('/servers', 'AppController@servers');
21     Route::post('/servers/{server}', 'AppController@server');
22     Route::post('/data', 'MoniController@data');
23 });
```

Codi 8: "api.php"

Les línies de Route::post, envien l'execució amb un metode post a MoniController@lacommanda

El Route::middleware('auth:api') protegeix l'execució de que sigui el usuari correcte mitjançant el token.

Route de la interfície:

```
1 <?php
2
3 /*
4 |
5 | Web Routes
6 |
7 |
8 | Here is where you can register web routes for your application. These
9 | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/{any}', 'AppController@index')->where('any', '.*');
```

Codi 9: "web.php"

Mitjançant la comanda post el que fa es obtenir la interfície.

Web socket

Mitjançant el servei pusher, avisa que hi ha les dades noves mitjançant web sockets.