

# LFTS - Newsletter and APIs

---

## 58.Mailchimp API Tinkering

En este episodio nos enseñan como funciona la instalacion de API y tomamos los primeros pasos para usarla

```
36
37 <a href="#newsletter" class="bg-blue-500 ml-3 rounded-full text-xs font-semibold text-white u
38     Subscribe for Updates
39 </a>
40 </div>
41 </nav>
42
43 {{slot}}
44
45 <footer id="newsletter" class="bg-gray-100 border border-black border-opacity-5 rounded-xl text-cente
46 
```

Ejecutamos el siguiente codigo

```
composer require mailchimp/marketing
```

```
32
33 'mailchimp' => [
34     'key' => env('MAILCHIMP_KEY')
35 ]
36
37 ];
38
```

```
53
54
55 MAILCHIMP_KEY=db4110dcd15721f683acaa8123479dd-us21
```

```
8
9 Route::get('ping', function () {
10     $mailchimp = new \MailchimpMarketing\ApiClient();
11
12     $mailchimp->setConfig([
13         'apiKey' => config('services.mailchimp.key'),
14         'server' => 'us21'
15     ]);
16     $response = $mailchimp->lists->getAllLists();
17     // $response = $mailchimp->lists->getList('4623c97960');
18     // $response = $mailchimp->lists->getListMembersInfo('4623c97960');
19     $response = $mailchimp->lists->addListMember('4623c97960', [
20         'email_address' => 'cristiam@gmail.com',
21         'status' => 'subscribed'
22     ]);
23     ddd($response);
24 });
```

## 59. Make the Newsletter Form Work

Ahora vamos a hacer que con el API que instalamos y nuestro pequeño form para inscribirse en nuestro newsletter sea funcional

```
8
9 Route::post('newsletter', function () {
10
11     request()->validate(['email' => 'required|email']);
12     $mailchimp = new \MailchimpMarketing\ApiClient();
13
14     $mailchimp->setConfig([
15         'apiKey' => config('services.mailchimp.key'),
16         'server' => 'us21'
17     ]);
18     try {
19         $response = $mailchimp->lists->addListMember('4623c97960', [
20             'email_address' => request('email'),
21             'status' => 'subscribed'
22         ]);
23     } catch (\Exception $e) {
24         throw \Illuminate\Validation\ValidationException::withMessages([
25             'email' => 'This email could not be added to our newsletter list.'
26         ]);
27     }
28     return redirect('/')->with('success', 'You are now signed up for our newsletter');
29 });
30
```

```

52
53     <form method="POST" action="/newsletter" class="lg:flex text-sm">
54         @csrf
55         <div class="lg:py-3 lg:px-5 flex items-center">
56             <label for="email" class="hidden lg:inline-block">
57                 
58             </label>
59
60             <div>
61                 <input id="email" name="email" type="text" placeholder="Your email address"
62                     class="lg:bg-transparent py-2 lg:py-0 pl-4 focus-within:outline-none">
63                 @error('email')
64                 <span class="text-xs text-red-500">{{ $message }}</span>
65                 @enderror
66             </div>
67         </div>
68
69         <button type="submit"
70             class="transition-colors duration-300 bg-blue-500 hover:bg-blue-600 mt-4 lg:mt-0 lg:ml-3 ro
71             Subscribe

```

## 60. Extract a Newsletter Service

En este episodio limpiamos código para que sea más limpio y fácil de entender

```

5  use Exception;
6  use App\Services\Newsletter;
7  use Illuminate\Http\Request;
8  use Illuminate\Validation\ValidationException;
9
10 class NewsletterController extends Controller
11 {
12     /**
13      * Handle the incoming request.
14      *
15      * @param  \Illuminate\Http\Request  $request
16      * @return \Illuminate\Http\Response
17      */
18     public function __invoke(Request $request)
19     {
20         request()->validate(['email' => 'required|email']);
21
22         try {
23             $newsletter = new Newsletter();
24             $newsletter->subscribe(request('email'));
25         } catch (Exception $e) {
26             throw ValidationException::withMessages([
27                 'email' => 'This email could not be added to our newsletter list.'
28             ]);
29         }
30         return redirect('/')->with('success', 'You are now signed up for our newsletter');
31     }
32 }

```

```

5 use MailchimpMarketing\ApiClient;
6
7 class Newsletter
8 {
9     public function subscribe(string $email, string $list = null)
10     {
11
12         $list ??= config('services.mailchimp.lists.subscribers');
13
14         // $response = $mailchimp->lists->getAllLists();
15         // $response = $mailchimp->lists->getList('4623c97960');
16         // $response = $mailchimp->lists->getListMembersInfo('4623c97960');
17
18         return $this->client()->lists->addListMember($list, [
19             'email_address' => $email,
20             'status' => 'subscribed'
21         ]);
22     }
23
24     protected function client()
25     {
26         $mailchimp = new ApiClient();
27
28         return $mailchimp->setConfig([
29             'apiKey' => config('services.mailchimp.key'),
30             'server' => 'us21'
31         ]);
32     }
33 }

```

```

55 MAILCHIMP_KEY=db4110dcd15721f683acaa8123479dd-us21
56 MAILCHIMP_LIST_SUBSCRIBERS=4623c97960

```

```

33     'mailchimp' => [
34         'key' => env('MAILCHIMP_KEY'),
35         'lists' => [
36             'subscribers' => env('MAILCHIMP_LIST_SUBSCRIBERS')
37         ]
38     ]
39 ]

```

```

8 use App\Http\Controllers\NewsletterController;
9
10
11 Route::get('/', [PostController::class, 'index'])->name('home');
12 Route::get('/posts/{post:slug}', [PostController::class, 'show'])->name('post');
13 Route::post('/posts/{post:slug}/comments', [CommentController::class, 'store']);
14
15 Route::post('newsletter', NewsletterController::class);
16
17 Route::get('/register', [RegisterController::class, 'create'])->middleware('guest');

```

## 61. Toy Chests and Contracts

```
1  <?php
2
3  namespace App\Services;
4
5  use MailchimpMarketing\ApiClient;
6
7  class MailChimpNewsletter implements Newsletter
8  {
9
10     protected ApiClient $client;
11
12     public function __construct(ApiClient $client)
13     {
14         $this->client = $client;
15     }
16
17     public function subscribe(string $email, string $list = null)
18     {
19
20         $list ??= config('services.mailchimp.lists.subscribers');
21
22         return $this->client->lists->addListMember($list, [
23             'email_address' => $email,
24             'status' => 'subscribed'
25         ]);
26     }
27 }
```

```
4
5  interface Newsletter
6  {
7      public function subscribe(string $email, string $list = null);
8  }
```

```
10 class NewsletterController extends Controller
11 {
12     /**
13      * Handle the incoming request.
14      *
15      * @param \Illuminate\Http\Request $request
16      * @return \Illuminate\Http\Response
17      */
18     public function __invoke(Newsletter $newsletter)
19     {
20         request()->validate(['email' => 'required|email']);
21
22         try {
23             $newsletter->subscribe(request('email'));
24         } catch (Exception $e) {
25             throw ValidationException::withMessages([
26                 'email' => 'This email could not be added to our newsletter list.'
27             ]);
28         }
29         return redirect('/')->with('success', 'You are now signed up for our newsletter');
30     }
31 }
```

```
19     public function register()
20     {
21         app()->bind(Newsletter::class, function () {
22             $client = new ApiClient();
23
24             $client->setConfig([
25                 'apiKey' => config('services.mailchimp.key'),
26                 'server' => 'us21'
27             ]);
28
29             return new MailChimpNewsletter($client);
30         });
31     }
```