# LFTS - Newsletter and APIs

## 58.Mailchimp API Tinkering

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

```
            <a href="#newsletter" class="bg-blue-500 ml-3 rounded-full text-xs font-semibold text-white u
                Subscribe for Updates
            </a>
        </div>
    </nav>

    {{$slot}}

    <footer id="newsletter" class="bg-gray-100 border border-black border-opacity-5 rounded-xl text-cente
        <img src="./images/lary-newsletter-icon.svg" alt="" class="mx-auto -mb-6" style="width: 145px;">
```

**Ejecutamos el siguiente codigo**

```
composer require mailchimp/marketing
```

```
    'mailchimp' => [
        'key' => env('MAILCHIMP_KEY')

    ]
];
```

```
MAILCHIMP_KEY=dbe4110dcd15721f683acaa8123479dd-us21
```

```php
Route::get('ping', function () {
    $mailchimp = new \MailchimpMarketing\ApiClient();

    $mailchimp->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us21'
    ]);
    $response = $mailchimp->lists->getAllLists();
    // $response = $mailchimp->lists->getList('4623c97960');
    // $response = $mailchimp->lists->getListMembersInfo('4623c97960');
    $response = $mailchimp->lists->addListMember('4623c97960', [
        'email_address' => 'cristiam@gmail.com',
        'status' => 'subscribed'
    ]);
    ddd($response);
});
```

## 59. Make the Newsletter Form Work

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

```php
Route::post('newsletter', function () {

    request()->validate(['email' => 'required|email']);
    $mailchimp = new \MailchimpMarketing\ApiClient();

    $mailchimp->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us21'
    ]);
    try {
        $response = $mailchimp->lists->addListMember('4623c97960', [
            'email_address' => request('email'),
            'status' => 'subscribed'
        ]);
    } catch (\Exception $e) {
        throw \Illuminate\Validation\ValidationException::withMessages([
            'email' => 'This email could not be added to our newsletter list.'
        ]);
    }
    return redirect('/')->with('success', 'You are now signed up for our newsletter');
});
```

```blade
<form method="POST" action="/newsletter" class="lg:flex text-sm">
    @csrf
    <div class="lg:py-3 lg:px-5 flex items-center">
        <label for="email" class="hidden lg:inline-block">
            <img src="/images/mailbox-icon.svg" alt="mailbox letter">
        </label>

        <div>
            <input id="email" name="email" type="text" placeholder="Your email address"
                class="lg:bg-transparent py-2 lg:py-0 pl-4 focus-within:outline-none">
            @error('email')
            <span class="text-xs text-red-500">{{$message}}</span>
            @enderror
        </div>
    </div>

    <button type="submit"
    class="transition-colors duration-300 bg-blue-500 hover:bg-blue-600 mt-4 lg:mt-0 lg:ml-3 ro
        Subscribe
```

## 60. Extract a Newsletter Service

En este episodio limpiamos codigo para que sea mas limpio y facil de entender

```php
use Exception;
use App\Services\Newsletter;
use Illuminate\Http\Request;
use Illuminate\Validation\ValidationException;

class NewsletterController extends Controller
{
    /**
     * Handle the incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function __invoke(Request $request)
    {
        request()->validate(['email' => 'required|email']);

        try {
            $newsletter = new Newsletter();
            $newsletter->subscribe(request('email'));
        } catch (Exception $e) {
            throw ValidationException::withMessages([
                'email' => 'This email could not be added to our newsletter list.'
            ]);
        }
        return redirect('/')->with('success', 'You are now signed up for our newsletter');
    }
}
```

```php
use MailchimpMarketing\ApiClient;

class Newsletter
{
    public function subscribe(string $email, string $list = null)
    {

        $list ??= config('services.mailchimp.lists.subscribers');

        //$response = $mailchimp->lists->getAllLists();
        // $response = $mailchimp->lists->getList('4623c97960');
        // $response = $mailchimp->lists->getListMembersInfo('4623c97960');

        return $this->client()->lists->addListMember($list, [
            'email_address' => $email,
            'status' => 'subscribed'
        ]);
    }

    protected function client()
    {
        $mailchimp = new ApiClient();

        return $mailchimp->setConfig([
            'apiKey' => config('services.mailchimp.key'),
            'server' => 'us21'
        ]);
    }
}
```

```
MAILCHIMP_KEY=dbe4110dcd15721f683acaa8123479dd-us21
MAILCHIMP_LIST_SUBSCRIBERS=4623c97960
```

```php
    'mailchimp' => [
        'key' => env('MAILCHIMP_KEY'),
        'lists' => [
            'subscribers' => env('MAILCHIMP_LIST_SUBSCRIBERS')
        ]

    ]
```

```php
use App\Http\Controllers\NewsletterController;


Route::get('/', [PostController::class, 'index'])->name('home');
Route::get('/posts/{post:slug}', [PostController::class, 'show'])->name('post');
Route::post('/posts/{post:slug}/comments', [CommentController::class, 'store']);

Route::post('newsletter', NewsletterController::class);

Route::get('/register', [RegisterController::class, 'create'])->middleware('guest');
```

## 61. Toy Chests and Contracts

```php
<?php

namespace App\Services;

use MailchimpMarketing\ApiClient;

class MailChimpNewsletter implements Newsletter
{

    protected ApiClient $client;

    public function __construct(ApiClient $client)
    {
        $this->client = $client;
    }

    public function subscribe(string $email, string $list = null)
    {

        $list ??= config('services.mailchimp.lists.subscribers');

        return $this->client->lists->addListMember($list, [
            'email_address' => $email,
            'status' => 'subscribed'
        ]);
    }
```

```php

interface Newsletter
{
    public function subscribe(string $email, string $list = null);
}
```

```php
class NewsletterController extends Controller
{
    /**
     * Handle the incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function __invoke(Newsletter $newsletter)
    {
        request()->validate(['email' => 'required|email']);

        try {
            $newsletter->subscribe(request('email'));
        } catch (Exception $e) {
            throw ValidationException::withMessages([
                'email' => 'This email could not be added to our newsletter list.'
            ]);
        }
        return redirect('/')->with('success', 'You are now signed up for our newsletter');
    }
}
```

```php
    public function register()
    {
        app()->bind(Newsletter::class, function () {
            $client = new ApiClient();

            $client->setConfig([
                'apiKey' => config('services.mailchimp.key'),
                'server' => 'us21'
            ]);

            return new MailChimpNewsletter($client);
        });
    }
```