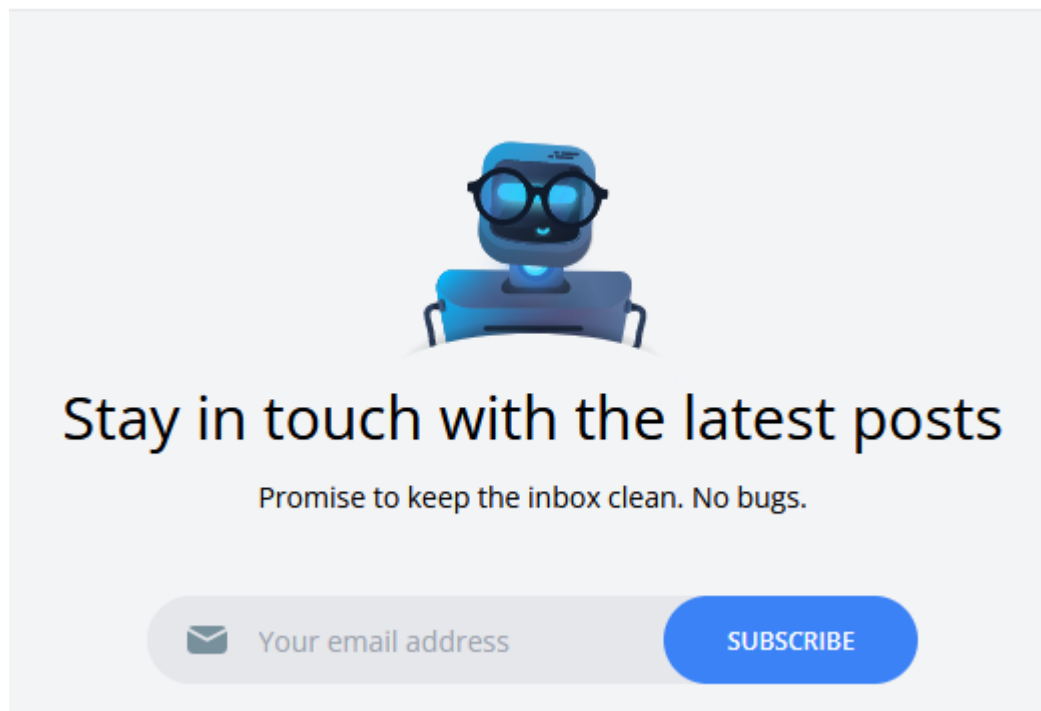


[<--- Volver](#)

Newspaper and APIs

Modificación de la API de Mailchimp / Mailchimp API Tinkering

Ahora vamos a agregar funcionalidad a la parte baja de nuestra pagina web, en concreto esta



En el componente `layout` vamos a editar la referencia

```
<a href="#newsletter" class="bg-blue-500 ml-3 rounded-full text-xs font-semibold
text-white uppercase py-3 px-5">
  Subscribe for Updates
</a>
```

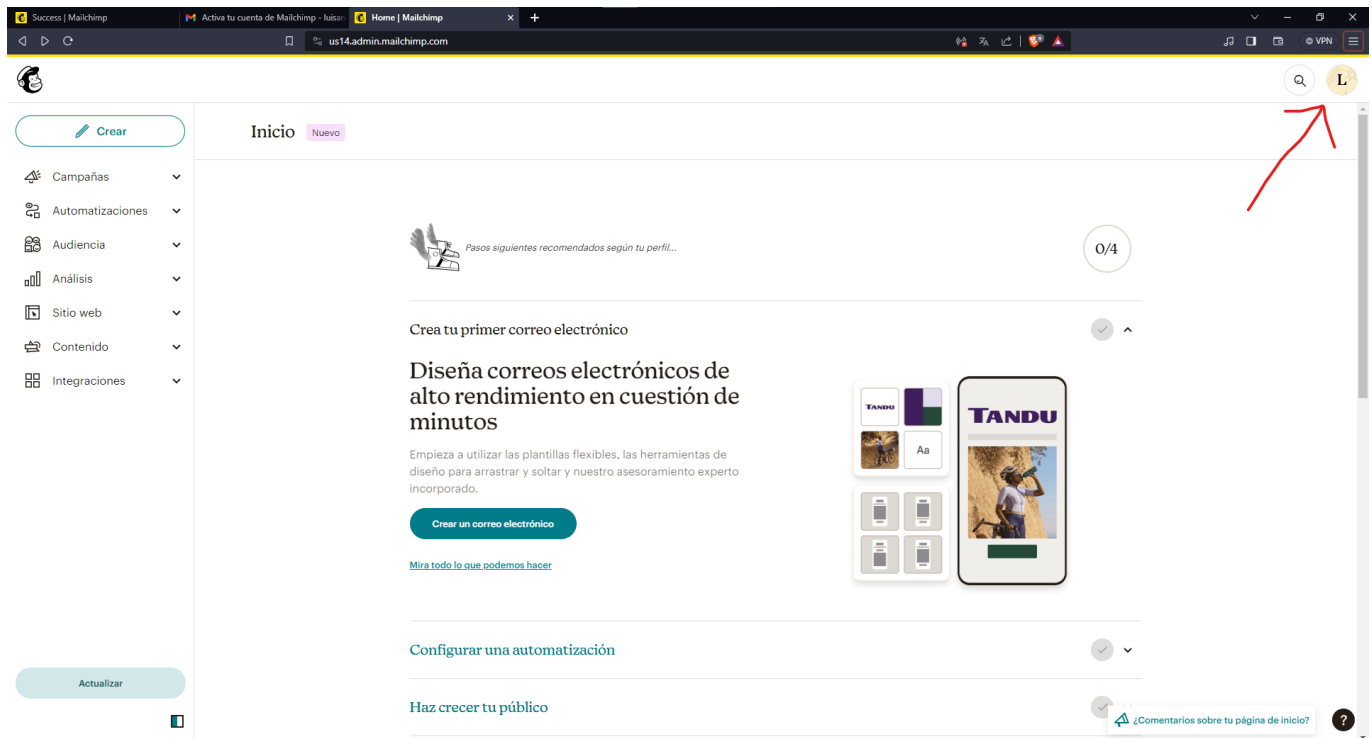
Posterior a esto lo que haremos sera ir a la pagina de mailchimp, para consumir la API que necesitamos para poder enviar los mensajes por medio del correo

- [Pagina oficial de Mailchip](#)

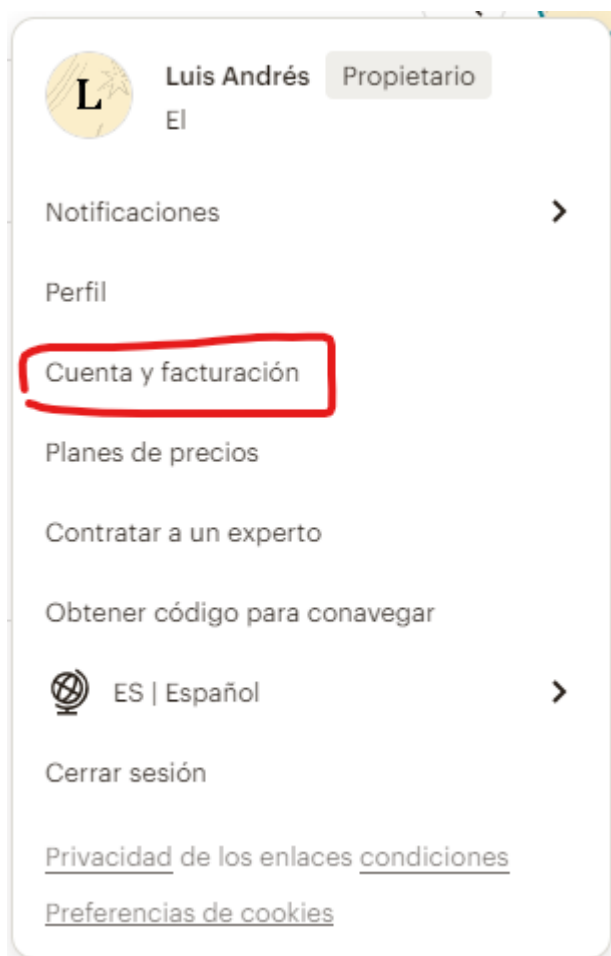
Aca lo que haremos sera darle al boton que dice `Sign In` o `Registrate` para registrarnos en la pagina y poder utilizar la API.

Al darle al boton nos saldra la opcion de suscribirnos en varios planes, en este caso utilizamos el plan gratis.

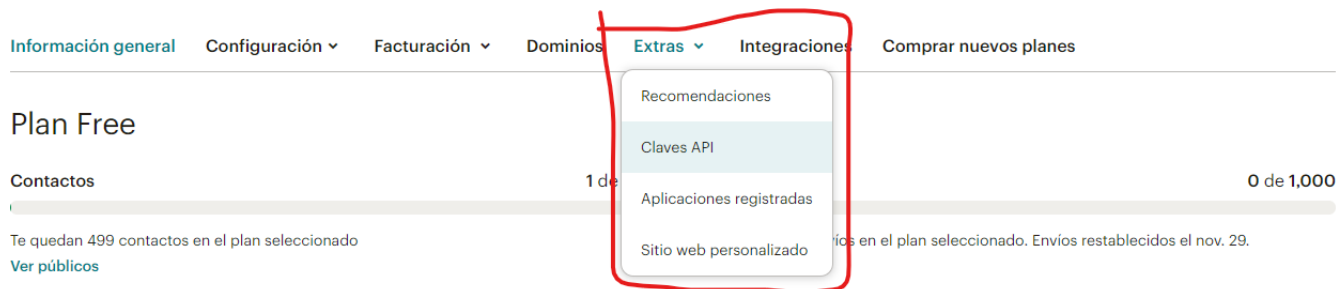
Una vez dentro de la pagina luego de habernos registrado y completado el correspondiente formulario, vamos a nuestra cuenta para crear una API key



Nos vamos a cuenta y facturación



Y una vez aca nos dirigimos hacia la parte de extras, y al apartado de claves API



Escolemos un poco hasta toparnos con esta parte y damos click deonde dice **crear una clave**

Tus claves API

Puedes revisar, revocar o generar nuevas claves API a continuación. [Aquí puedes obtener más información sobre cómo generar, revocar y acceder a las claves API.](#)



No tienes ninguna clave de API activa

Crear Una Clave

Una vez creada la clave, la copiamos y la llevamos a nuestro .env para guardarla

```
MAILCHIMP_KEY=YOURAPIKEYHERE
```

Luego nos vamos a la ruta *config* y nos dirigimos al archivo **services.php**, aca lo que haremos sera crear el nuevo servicio que vamos a annadir a nuestro proyecto.

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

Ahora vamos a la terminal de nuestra VM webserver para verificar que el programa reconozca el servicio

```
php artisan tinker  
config('services.mailchimp')
```

Nos dirigimos a la pagina de documentacion de la API para que sea mucho mas facil seguir los pasos para contruir la llamada de la API

-Documentacion de la API Mailchimp

Igualmente dentro de nuestra terminal vamos a instalar la libreria para poder utilizar la API, esto con el siguiente comando

```
composer require mailchimp/marketing
```

```
vagrant@webserver:/vagrant/sites/lfts.isw811.xyz$ composer require mailchimp/marketing
./composer.json has been updated
Running composer update mailchimp/marketing
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking mailchimp/marketing (3.0.80)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading mailchimp/marketing (3.0.80)
  - Installing mailchimp/marketing (3.0.80): Extracting archive
Package fruitcake/laravel-cors is abandoned, you should avoid using it. No replacement was suggested.
Package swiftmailer/swiftmailer is abandoned, you should avoid using it. Use symfony/mailer instead.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fruitcake/laravel-cors
Discovered Package: itsgoingd/clockwork
Discovered Package: laravel/breeze
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
79 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
No publishable resources for tag [laravel-assets].
Publishing complete.
No security vulnerability advisories found.
Using version ^3.0 for mailchimp/marketing
```

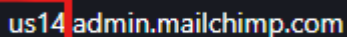
Ahora copiamos el codigo de la API call en nuestro archivo de rutas esto sera temporal

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

    $mailchimp->setConfig([
        'apiKey' => 'YOUR_API_KEY',
        'server' => 'YOUR_SERVER_PREFIX'
    ]);

    $response = $mailchimp->ping->get();
    ddd($response);
});
```

Aca colocamos la API key y ademas de eso nos pide el server Prefix, ese lo conseguimos viendo la URL que tenemos al entrar en nuestra cuenta en mailchimp, en mi caso es `us14`



Si al entrar al endpoint de /ping y nos sale esto, significa que todo esta perfectamente configurado y ya podemos hacer uso de la API

Dump

Content

```
{#887 ▼  
  + "health_status": "Everything's Chimp!"  
}
```

Location

routes/web.php:22 

Ahora cambiamos esta linea en el response para obtener un array con los datos

```
$response = $mailchimp->lists->getAllLists();
```

Y recibiremos esto

```
{#457 ▼  
  + "lists": array:1 [▼  
    0 => {#887 ▼  
      + "id": "9b3f6bdc7d"  
      + "web_id": 551743  
      + "name": "E1"  
      + "contact": {#890 ►}  
      + "permission_reminder": "Has recibido este correo electrónico porque lo has aceptado en nuestro sitio web."  
      + "use_archive_bar": true  
      + "campaign_defaults": {#888 ►}  
      + "notify_on_subscribe": ""  
      + "notify_on_unsubscribe": ""  
      + "date_created": "2023-10-30T20:32:41+00:00"  
      + "list_rating": 0  
      + "email_type_option": false  
      + "subscribe_url_short": "http://eepurl.com/iCUPMU"  
      + "subscribe_url_long": "https://gmail.us14.list-manage.com/subscribe?u=750461bf932e1d6133455b0fd&id=9b3f6bdc7d"  
      + "beamer_address": "us14-4c47c54686-ccalcef38b@inbound.mailchimp.com"  
      + "visibility": "prv"  
      + "double_optin": false  
      + "has_welcome": false  
      + "marketing_permissions": false  
      + "modules": []  
      + "stats": {#895 ►}  
      + "_links": array:16 [►]  
    ]  
  + "total_items": 1  
  + "constraints": {#456 ►}  
  + "_links": array:2 [►]  
}
```

Ahora al realizar un hardcoding con los datos de mi usuario, nos estariamos subscribiendo para recibir correos

```
$response = $mailchimp->lists->addListMember('9b3f6bdc7d', [  
    'email_address' => 'luis@gmail.com',  
    'status' => 'suscribed'  
]);
```

Hacer que el formulario del periodico funcione / Make the Newsletter Form Work

Ahora vamos a hacer que el formulario de envio de correos funcione dinamicamente, para esto modificamos el endpoint en `web.php`

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

    $mailchimp->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us14'
    ]);

    try {
        $response = $mailchimp->lists->addListMember('9b3f6bdc7d', [
            'email_address' => request('email'),
            'status' => 'suscribed'
        ]);
    }
    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 op for our newsletter');
});
```

Luego editamos en layout para que el post sea redigido a la ruta que necesitamos, aproximadamente esta en la linea 64 del codigo y la linea seria esta

```
<form method="POST" action="/newsletter" class="lg:flex text-sm">
```

El formulario no acepta cualquier email, por lo que agregamos al codigo de arriba un try catch para enviar un mensaje en caso que el email no funcione, en este caso lo voy a probar con un email invalido para verificar si funciona



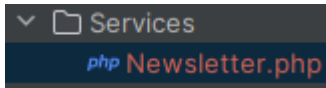
Your email address

This email could not be added to our newsletter list

SUBSCRIBE

Extraer el servicio de periodico / Extract a Newsletter Service

Creamos un nuevo folder que llamaremos services, en el que creamos una clase llamada Newsletter, en la cual pondremos todo el servicio del Newsletter



Y añadimos la siguiente función

```
public function subscribe(string $email) {
    $mailchimp = new \MailchimpMarketing\ApiClient();

    $mailchimp->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us14'
    ]);

    return @$mailchimp->lists->addListMember('9b3f6bdc7d', [
        'email_address' => request('email'),
        'status' => 'suscribed'
    ]);
}
```

El endpoint quedaria de esta manera

```
Route::post('newsletter', function () {
    request()->validate(['email'=>'required|email']);

    try {
        (new 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 op for our newsletter');
});
```

Nos vamos al archivo services.php que esta en la ruta config y ahí editamos

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

Asi quedaria la clase de Newsletter

```
public function subscribe(string $email, string $list = null) {
    $list ??= config('services.mailchimp.lists.subscribers');
    return $this->client()->lists->addListMember($list, [
        'email_address' => request('email'),
        'status' => 'suscribed'
    ]);
}

public function client() {
    $mailchimp = new \MailchimpMarketing\ApiClient();
    $mailchimp->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us14'
    ]);
}
```

Y asi quedaria el endpoint

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

Nos vamos a la consola de la VM webserver a crar el nuevo controller

```
php artisan make:controller NewsletterController --invokable
```

Asi quedaria el codigo en el nuevo controller

```
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!');
}
```

Cofres de juguetes y contratos / Toy Chests and Contracts

Nos vamos a la clase Newsletter y editamos

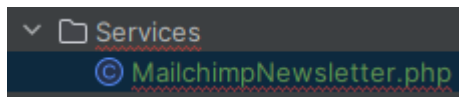
```
public function __constructor(protected ApiClient $client, protected string $foo)
{
    //
}
public function subscribe(string $email, string $list = null) {
    $list ??= config('services.mailchimp.lists.subscribers');

    return $this->client()->lists->addListMember($list, [
        'email_address' => $email,
        'status' => 'suscribed'
    ]);
}
protected function client() {
    return $this->client()->setConfig([
        'apiKey' => config('services.mailchimp.key'),
        'server' => 'us14'
    ]);
}
```

Ahora nos vamos al archivo `AppServiceProvider`, y pegamos dentro de la funcion register el siguiente codigo

```
public function register()
{
    app()->bind(Newsletter::class, function () {
        $client = (new ApiClient)->setConfig([
            'apiKey' => config('services.mailchimp.key'),
            'server' => 'us14'
        ]);
        return new Newsletter($client);
    });
}
```

Cambiamos el nombre de la clase Newsletter a MailchimpNewsletter



Creamos otra nueva clase llamada ConvertKitNewsletter y una interfaz llamada Newsletter

Copiamos esto en la nueva interfaz

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

Agregamos tanto a MailchimpNewsletter como a ConvertKitNewsletter lo siguiente luego del nombre de la clase

```
implements Newsletter
```

El código en AppServiceProvider quedaria así

```
public function register()
{
    app()->bind(Newsletter::class, function () {
        $client = (new ApiClient)->setConfig([
            'apiKey' => config('services.mailchimp.key'),
            'server' => 'us14'
        ]);
        return new MailchimpNewsletter($client);
    });
}
```

Además de esto debemos importar tanto la interfaz Newsletter como la clase MailchimpNewsletter donde sea necesario