

# Cosas de Laravel

Algunos tips de Laravel

- [Crear una Clase Helper para un proyecto Laravel](#)
- [Mailhog como mailtrap para desarrollos con Laravel](#)
- [Método en TestCase para facilitar los test de usuario logeado](#)
- [Métodos dump en el proceso de Testing con Laravel](#)
- [Asignar múltiples variables a la vez en PHP](#)
- [Configurar Carbon::now\(\) a una fecha para trabajar con tests](#)
- [Código de estado HTTP para llamadas API](#)
- [Testing error SQLSTATE\[HY000\]: General error: 1 near "ALTER": syntax error \(SQL: ALTER TABLE](#)
- [Laravel rescue\(\) helper](#)
- [Sail, Access can't connect to Mysql](#)
- [Sail y docker](#)
- [SAGE API 3.1, Laravel Socialite](#)
- [Laravel Filament Admin funciona en Sail, pero no en producción. error 404 en ficheros .js](#)
- [Traducciones no funcionan en Laravel](#)
- [Comprobando el uso del trait](#)
- [Pest, PHPStorm y Laravel Sail](#)
- [Laravel Herd y cosas que no estan documentadas.](#)
- [PHP enums un gran aliado.](#)
- [Manera elegante de obtener el phpinfo\(\) en tu proyecto](#)
- [Instalar ionCube en Laravel Herd en un mac Silicon](#)
- [El cast y su importancia en el modelo Eloquent de Laravel](#)
- [Phpredis en Laravel 10/11](#)
- [Livewire && Laravel Localization: The GET method is not supported for route livewire/update 404](#)

# Crear una Clase Helper para un proyecto Laravel

## Introducción

Muchas veces necesitamos ciertas funciones o métodos, para nuestros proyectos, que por su repercusión o repetición, puede ser interesante tenerlos agrupados. Solemos llamarlos Helpers.

Podemos hacerlo de tres formas Crear un fichero de funciones (el más común entre los bloggers de Laravel) con carga mediante autoload Creación de una Clase estática (no muy común pero más fina y menos propensa a choques) Creación de una paquete, que sería interesante si tuviéramos muchísimos helpers en distintas clases y que pudieran ser compartido por multitud de nuestros proyectos o de otros programadores Aquí vamos a mostrar los dos primeros.

## helpers.php

Crearemos un fichero cuyo lugar y nombre podría ser `app/helpers.php` con el contenido de abajo.

## Crear un archivo helpers.php

### Método incorrecto

No se la de veces que lo habré visto, y me parece horrible y alejado de las buenas prácticas.

```
<?php

function asString($data)
{
    $json = asJSON($data);

    return wordwrap($json, 76, "\n    ");
}

function asJSON($data)
```

```

{
    $json = json_encode($data);
    $json = preg_replace('/(["\\}]])([,:])(["\\{])/',' '$1$2 $3', $json);

    return $json;
}

```

## Método correcto

Ya que no usamos una clase sino un archivo tipo include que cargaremos mediante un autoload, para evitar problemas de duplicidad de nombres con las funciones de PHP, lo correcto es hacerlo como el código de abajo

```

<?php

if (!function_exists('asString'))
{
    function asString($data)
    {
        $json = asJSON($data);

        return wordwrap($json, 76, "\n    ");
    }
}

if (!function_exists('asJson'))
{
    function asJSON($data)
    {
        $json = json_encode($data);
        $json = preg_replace('/(["\\}]])([,:])(["\\{])/',' '$1$2 $3', $json);

        return $json;
    }
}

```

## Carga del archivo

Editamos nuestro composer.json en la sección `autoload` añadiendo o creando la sección `files`

```
"autoload": {
    "psr-4": {
        "App\\": "app/",
        "Database\\Factories\\": "database/factories/",
        "Database\\Seeders\\": "database/seeders/"
    },
    "files": [
        "app/helpers.php"
    ]
},
```

## Actualización del autoload de la app

```
composer dump-autoload
```

## Uso

El uso es sencillo, ya que se le llama como si fuera una función nativa de PHP.

```
$headerData = [
    'category' => 'develop',
    'unique_args' => [
        'var_1' => 'abc'
    ]
];

$header = asString($headerData);
```

## Crear una clase estática

La creación de una clase estática, nos permite más seguridad, algo más de estandarización y para trabajar en grupo, y el camino a la creación de nuestro propio paquete de Helpers.

## Crear el fichero de clase Helper

En mi ejemplo uso el directorio Helpers dentro de App porque tengo más clases de helpers en un proyecto largo `app/Helpers/MailHelpers`

```

<?php

namespace App\Helpers;

class MailHelpers {
    public static function asString($data)
    {
        $json = self::asJson($data);

        return wordwrap($json, 76, "\n    ");
    }

    public static function asJson ($data)
    {
        $json = json_encode($data);
        $json = preg_replace('/(("[\]]))([,:])(("[\]))/', '$1$2 $3', $json);

        return $json;
    }
}

```

De esta forma no necesitamos realizar ninguna modificación en nuestro fichero composer.json, ya que la carga se produce conforme al PSR-4, y es simplemente una clase más de tipo estático.

## Uso

Este ejemplo es del uso de la clase MailHelpers en una clase Mail.

```

<?php

namespace App\Mail;

use App\Helpers\MailHelpers;
...

public function build()
{
    $headerData = [
        'category' => 'develop',

```

```
'unique_args' => [  
    'var_1' => 'abc'  
]  
];  
  
$header = MailHelpers::asString($headerData);  
  
...  
...
```

# Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Mailhog como mailtrap para desarrollos con Laravel

## Introducción

Generalmente uso mailtrap.io para comprobar los correos ya que entre otras herramientas tiene **HTML Check** pero cuando se trata de trabajos en su inicio que no requieren de esto, y prima la portabilidad, prefiero usar Mailhog, un capturador de correo desarrollado en **Go**.

## Instalación en Ubuntu

### Descarga y conversión a ejecutable

“ Revisar siempre la versión disponible, ya que una entrada de blog o wiki puede no estar actualizada

```
$ wget https://github.com/mailhog/MailHog/releases/download/v1.0.0/MailHog_linux_amd64
$ sudo cp MailHog_linux_amd64 /usr/local/bin/mailhog
$ sudo chmod +x /usr/local/bin/mailhog
```

### Crear un servicio para Mailhog (systemd)

“ Se hace uso del comando whoami para cargar el servicio para nuestro usuario. Debemos verificar que lo hizo bien.

```
$ sudo tee /etc/systemd/system/mailhog.service <<EOL
[Unit]
Description=Mailhog
After=network.target
[Service]
```

```
User=$(whoami)
ExecStart=/usr/bin/env /usr/local/bin/mailhog > /dev/null 2>&1 &
[Install]
WantedBy=multi-user.target
EOL
```

## Verificar

```
$ sudo cat /etc/systemd/system/mailhog.service
[Unit]
Description=Mailhog
After=network.target
[Service]
User=abkrim
ExecStart=/usr/bin/env /usr/local/bin/mailhog > /dev/null 2>&1 &
[Install]
WantedBy=multi-user.target
```

## Activar y habilitar para arranque con el sistema

Es aconsejable siempre que modifiquemos algo en el systemd hacer un reload del demonio.

```
$ sudo systemctl daemon-reload
```

### Distintas acciones

```
$ sudo systemctl start mailhog.service
$ sudo systemctl enable mailhog.service
$ sudo systemctl status mailhog.service
```

## Configurar php.ini

En mi caso es para PHP-FPM y multi versión, por lo que necesito añadirlo a cada uno, para el fpm y para el cli.

```
$ sudo sed -i "s/;sendmail_path.*/sendmail_path='\usr\local\bin\mailhog sendmail
abkrim@nox.test'/" /etc/php/8.0/fpm/php.ini
$ sudo sed -i "s/;sendmail_path.*/sendmail_path='\usr\local\bin\mailhog sendmail
abkrim@nox.test'/" /etc/php/8.0/cli/php.ini
$ sudo sed -i "s/;sendmail_path.*/sendmail_path='\usr\local\bin\mailhog sendmail
abkrim@nox.test'/" /etc/php/7.4/fpm/php.ini
```



```
$ sudo sed -i "s/;sendmail_path.*/sendmail_path='\/usr\/local\/bin\/mailhog sendmail  
abkrim@nox.test'/" /etc/php/7.4/cli/php.ini
```

Necesito un reload de php-fpm

```
$ sudo systemctl restart php8.0-fpm.service  
$ sudo systemctl restart php7.4-fpm.service
```

## Ver Mailhog en el navegador

```
http://localhost:8025/
```

Mailhog Localhost

## Configuracion para Laravel

Editamos el fichero .env

El puerto por defecto es 1025 MAIL\_FROM\_ADDRESS es requerido

```
MAIL_MAILER=smtp  
MAIL_HOST=localhost  
MAIL_PORT=1025  
MAIL_USERNAME=null  
MAIL_PASSWORD=null  
MAIL_ENCRYPTION=null  
MAIL_FROM_ADDRESS=abkrim@nox.local  
MAIL_FROM_NAME="${APP_NAME}"
```

## Enlace

- [Install & Configure MailHog](#)

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Método en TestCase para facilitar los test de usuario logeado

## Desarrollo

En los tests mucha s veces necesitamos realziar pruebas como usuario logeado o usuario especifico. Un buen refactor para esta acción repitada es incluirla en la **clase TestCase** de la cual extendemos test en Laravel.

## TestCase

```
public function login(User $user = null): User
{
    $user ??= User::factory()->create();

    $this->actingAs($user);

    return $user;
}
```

Ahora será más fácil escribir nuestros tests usando simplemente `$this->login`

```
//$this->actingAs(User::factory()->create());

$this->login()
```

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Métodos dump en el proceso de Testing con Laravel

## Introducción

Una de las herramientas que más me gustan de **Laravel** es `dd()`. Una herramienta que permite un volcado con exit, que por lo general sale bien formateo, y que es muy útil en algunas corcustancias para localizar problemas o comprender mecanismo y estados en alguna parte del código.

En **PHPUnit** con Laravel tambien disponemos de herramientas para hacer algo parecido en el proceso de testing.

## Volcando datos en la construcción de un test

Tenemos tres elementos todo ellos formando parte de la clase **TestReponse** de **Illuminate/Response**

### dump()

Que vuelca el contenido de la respuesta (response)

### dumpHeaders()

Que vuelca solo el contenido de las headers muy útil cuando trabajamos con Api auqne tambien útil en otras areas

### dumpSession()

Que vuelca el contenido de la session de la respuesta

## Ejemplo

```
/** @test */  
function date_format_is_validate()  
{  
    $this->login();  
  
    $post = BlogPost::factory()->create();  
  
    $this  
        ->post(action([BlogPostAdminController::class, 'update'], $post->slug), [  
            'title' => $post->title,  
            'author' => $post->author,  
            'body' => $post->body,  
            'date' => '01/01/2021',  
        ])  
        ->dumpSession()  
        ->assertSessionHasErrors(['date']);  
}
```

dumpSession en Tetstin Laravel

# Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Asignar múltiples variables a la vez en PHP

## Introducción

La limpieza de código para su lectura es algo muy interesante. A veces tenemos métodos que devuelven un array numérico que queremos incorporar a una serie de variables. Podemos hacerlo al mismo tiempo.

### [] = array()

En las pruebas de test, por ejemplo queremos evaluar dos usuarios, uno con permisos y otro sin permisos.

Nuestra factoría (Laravel), nos permite obtener un array con dos colecciones en un array y podemos asignarlas a dos variables `$guest` y `$admin`

```
/** @test */
function only_admin_users_are_allowed()
{
    [$guest, $admin] = User::factory()
        ->count(2)
        ->sequence(
            ['is_admin' => false],
            ['is_admin' => true],
        )
        ->create();
}
```

“ Similar trabajo tiene la [función list\(\)](#) aunque en este caso no me parece tan limpia.

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).



Configurar Carbon::now() a una fecha para trabajar con tests

# Código de estado HTTP para llamadas API

## Códigos de respuesta HTTP

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases:

1. Respuestas informativas (100-199),
2. Respuestas satisfactorias (200-299),
3. Redirecciones (300-399),
4. Errores de los clientes (400-499),
5. Errores de los servidores (500-599).

Los códigos de estado se definen en la sección 10 de [RFC 2616](#). Puedes obtener las especificaciones actualizadas en [RFC 7231](#).

## Tabla de uso más cotidiano

No estan todos, pero si los que uso yo, que muchas veces no lo hago por que me quede claro, sino porque Laravel lo hace así, y pese a que en algunos casos no estoy de acuerdo, creo que Taylor sabe más.

Código	Respuesta	Apreciaciones
100	Continue	
200	Ok	No todo es 200 y es una manía extendida entre programadores no actualizados
201	Created	Típica respuesta de un PUT con resultado correcto
202	Accepted	Solicitud sin compromiso, es decir no hay respuesta asincrona
301	Moved Permanently	La URI se modifiko
400	Bad request	Posiblemente una mala sintaxis en la llamada a la api

Código	Respuesta	Apreciaciones
401	Unauthorized	Es necesario autenticarse. Similar a 403 pero indicando que si se puede logear haciendolo debdamente
403	Forbidden	El login no es valido para acceder al recurso solicitado
404	Not found	El servidor no pudo encontrar el contenido solicitado. El más famoso
422	Unprocessable Entity	La petición estaba bien formada pero no se pudo seguir debido a errores de semántica. Usado por laravel para muchas cosas.
429	Too many requests	Exceso de peticiones en un periodo de tiempo. (Throttling)
500	Internal Server Error	El servidor ha encontrado una situación que no sabe cómo manejarla
502	Bad Gateway	El servidor anda raro
503	Service unavailable	El servidor no está listo para manejar la petición. Causas comunes puede ser que el servidor está caído por mantenimiento o está sobrecargado.

## Fuente

[Códigos de estado de respuesta HTTP](#)

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Testing error SQLSTATE[HY000]: General error: 1 near "ALTER": syntax error (SQL: ALTER TABLE

## Introducción

Algunas veces hay que modificar columnas en nuestros desarrollos. Laravel nos permite la creación de migraciones especializadas en este tipo de acciones, pero supeditadas a **Doctrine/dbal** el cual muchas cosas no las hace

“ Presta atención a esa peculiaridad sobre todo en el uso de cosas como las columnas enum y cosas parecidas que son quebraderos de cabeza a demás de poco efectivas en su uso.

Me lleve una sorpresa cuando quise cambiar un unsignedTinyInteger.

```
public function up()
{
    // Not work because doctrine not work with tinyInteger and others
    Schema::table('subscribers', function (Blueprint $table) {
        $table
            ->tinyInteger('status')->unsigned()
            ->default(array_search('Pendiente', Subscriber::STATUS_SELECT))
            ->change();
    });
}
```

## Error

```
> a migrate
```

```
Migrating: 2022_02_02_124904_modify_status_default_value_to_subscribers_table
```

```
Doctrine\DBAL\Exception
```

Unknown column type "tinyinteger" requested. Any Doctrine type that you use has to be registered with `\Doctrine\DBAL\Types\Type::addType()`. You can get a list of all the known types with `\Doctrine\DBAL\Types\Type::getTypesMap()`. If this error occurs during database introspection then you might have forgotten to register all database types for a Doctrine Type. Use `AbstractPlatform#registerDoctrineTypeMapping()` or have your custom types implement `Type#getMappedDatabaseTypes()`. If the type name is empty you might have a problem with the cache or forgot some mapping information.

# Solución

El uso de raw, pero teniendo una atención relativa al entorno de testing, ya que si no lo hacemos así, cuando ejecutemos nuestras pruebas (test) obtendríamos un error.

```
There was 1 error:
```

```
1)
Tests\Http\Controllers\Api\Admin\Subscriber\SubscriberControllerStoreWithCampaignTest::call_with_correct_params_create_new_subscriber_associate_a_one_campaign
Illuminate\Database\QueryException: SQLSTATE[HY000]: General error: 1 near "ALTER": syntax error (SQL: ALTER TABLE mailer.subscribers ALTER status SET DEFAULT 1)
```

Para ello editaremos la migración de nuestra tabla

```
public function up()
{
    if (App::environment() !== 'testing') {
        DB::statement(
            'ALTER TABLE mailer.subscribers ALTER status SET DEFAULT '
            . array_search('Pendiente', Subscriber::STATUS_SELECT)
        );
    }
}
```

# Editado 09/02/2022

Tras actualizar en producción me di cuenta de un error. Estoy definiendo en el código el nombre exacto de la base de datos, y no es correcto.

“ SQLSTATE[42000]: Syntax error or access violation: 1142 ALTER command denied to user 'cwcl\_user'@'localhost' for table 'subscribers' (SQL: ALTER TABLE mailer.subscribers ALTER status SET DEFAULT 1)

El código de abajo lo arregla.

```
public function up()
{
    if (App::environment() !== 'testing') {
        DB::statement(
            'ALTER TABLE '
            . config('database.connections.mysql.database')
            . '.subscribers ALTER status SET DEFAULT '
            . array_search('Pendiente', Subscriber::STATUS_SELECT)
        );
    }
}
```

Esto también nos obliga a editar la migración inicial con el valor adecuado ya que de lo contrario, en nuestros tests, no tendríamos el valor por defecto deseado para esa columna.

De esta forma, podemos seguir trabajando tanto en local como en remoto, si tenemos que actualizar allí.

## Otras opciones

Seguro que puede haber otras opciones. Pero yo use esta y me funciona. Si tienes otra, no dudes en contactarme. [abdelkarim.mateos@laarroba.castris.com](mailto:abdelkarim.mateos@laarroba.castris.com)

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido se entrega, tal y como está, sin que ello implique ninguna obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).



# Laravel rescue() helper

## rescue()

[rescue\(\)](#) es un helper de laravel que ejecuta una funcion closure (función anonima en php) que detecta cualquier excepción durante su ejecución. Las excepciones se enviarán a su controlador de excepciones, pero la solicitud continuara siendo procesada.

```
// Viejo método - try / catch ignorando la excepción. Un poco feo
private static function existsOnCDN(string $path): bool
{
    $cdn = $false;

    try {
        $cdn = Storage::disk('cdn')->exists($path);
    } catch (\Exception $e) {
        // CDN no esta disponible por problemas de red.
    }

    return $cdn;
}

// Mas claro, rescue() ignora la excepcion y permite al código continuar. Opcionalmente
// podemos pasar un valor de retorno
private static function existsOnCDN(string $path): bool
{
    return rescue(fn () => Storage::disk('cdn')->exists($path), false);
}
```

## Agradecimientos

A @shawnlindstrom por su [tuit](#)

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como esta, sin que ello implique ningún



obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Sail, Access can't connect to Mysql

## Introducción

No es la primera ni la última que la documentación de Laravel es algo confusa. En este caso, siguiendo escrupulosamente las instrucciones de [Laravel Sail](#) pero al configurar mi TablePlus me da error de conexión.

## Solución

En primer lugar añadir al fichero .env de nuestro proyecto,

```
FORWARD_DB_PORT=3306
```

Apagar si está encendido, sail.

Ejecutar en nuestra máquina

```
> sail up -d
sail up -d
example-app-laravel.test-1  "start-container"  laravel.test      exited (0)
Shutting down old Sail processes...
[+] Running 5/5
   ⬢ Network example-app_sail
Created
                                0.0s
   ⬢ Container example-app-mysql-1
Started
                                0.6s
   ⬢ Container example-app-redis-1
Started
                                0.4s
   ⬢ Container example-app-mailhog-1
Started
                                0.6s
```

```

❏ Container example-app-laravel.test-1
Started

                                0.9s

> sail artisan config:cache
    INFO  Configuration cached successfully.
> sail artisan migrate

    INFO  Preparing database.

    Creating migration table
    .....
    ..... 29ms DONE

    INFO  Running migrations.

    2014_10_12_000000_create_users_table
    .....
    ..... 40ms DONE
    2014_10_12_100000_create_password_resets_table
    ..... 26ms
DONE
    2019_08_19_000000_create_failed_jobs_table
    .....
27ms DONE
    2019_12_14_000001_create_personal_access_tokens_table
    ..... 41ms DONE

```

Y ahora sí, que podremos conectarnos.

Table Plus

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Sail y docker

## Introducción

Aquí dejaré algunos tips que me han sido imprescindibles en el traspaso de mi máquina a mi mac, en el que ya solo uso [Laravel Sail](#) y Docker.

## Restaurar una copia de seguridad mysql

Algunos proyectos, uso alguna base reducida con el fin de poder trabajar ciertos aspectos casi reales, al margen de las pruebas de testing.

En una configuración básica de Sail yo uso este comando

```
docker-compose exec -T [mysql] mysql -uroot -p[password] < database/dump.sql
```

“ [mysql] es el nombre de host mysql que hallamos definido en el docker-compose.yml

### Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido se entrega, tal y como está, sin que ello implique ninguna obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# SAGE API 3.1, Laravel Socialite

## Introducción

Hacía ya dos años y más que realice un trabajo para mi contabilidad que combinaba, **WHMCS + SAGE + OVH + RedSys** para así, ahorrarme más de 10000 apuntes al año.

Quería implementar **Stripe**, y de paso actualizar. Ay!!! Aquí vino el dolor. Desarrollos olvidados, tips y cosas que se quedan en el tintero. En su día, no había módulo de Laravel **Socialite** y todo lo que probé así que hice mi propia adaptación y todo iba bien en la actualización hasta que llegué aquí.

## SAGE y OAuth 2

### Instalar Socialite

[Socialite](#) es fácil de instalar.

```
composer require laravel/socialite
```

Después necesitamos el paquete de [SAGE para Socialite](#) (es muy simple y en caso de discontinuarse se puede continuar por uno mismo)

```
composer require socialiteproviders/sage
```

Importante leer el How to de ese módulo para entender que debemos configurar el listener que hay en `EventServiceProvider` con el fin de que `Socialite` escuche al módulo.

## Comprobar las credenciales en SAGE

Esto me hizo perder el tiempo. Inexplicablemente pese a tener unas pocas modificaciones seguí manteniendo lo primordial, `SAGE_CLIENT_ID`, `SAGE_CLIENT_SECRET`, `SAGE_REDIRECT_URL`, `SAGE_STATE_CSRF`

Sin embargo tras llegar a la página de autenticación de **SAGE** (no encuentro en su doc que permita una autenticación stateless o sin servidor web) el retorno fallaba.

Primero lo achaque a que como he dockerizado mi contabilidad, uso localhost, en lugar de un FQDN como en mi vieja raspberry, donde usaba un dominio falso midominio.test.

Pero, revisando se me olvido (siempre se olvida algo) en la página de [Sage Development Portal](#) hay que configurar la app y entre otras cosas están los callbacks autorizados.

Asi que hay que añadirlo `http://localhost/login/sage/callback`

Pero volvió a fallar.

¿Uhhh? Raro se me hace. Revisé las variables, y sorpresa... el SAGE\_ID\_CLIENT Y EL SAGE\_CLIENT\_SECRET no corresponden a las que me funcionan en la vieja raspberry. Vamos que mi vieja contabilidad está trabajando con unos datos obsoletos o que pertenecen a otra cuenta.

En fin, ahora sí.

# Cómo usarlo en pocos pasos

## .env

Los datos de cliente son los de la página del portal de desarrolladores

```
SAGE_CLIENT_ID=Client ID
SAGE_CLIENT_SECRET='Client Secret'
SAGE_REDIRECT_URL=http://localhost/login/sage/callback
SAGE_STATE_CSRF=Token de al menos 32 caracteres aleatorio
```

## Rutas

Es un ejemplo...

```
Route::get('/login/sage', [LoginController::class, 'redirectToSageProvider']);
Route::get('/login/sage/callback', [LoginController::class, 'handleProviderSageCallback']);
```

## Controlador

En mi caso como hice mi propio paquete tengo un modelo en el que almaceno los tokens de proveedores externos de API que usan OAuth 2.0.

```

<?php

namespace App\Http\Controllers;

use Abkrim\ApiSage\Models\ExtToken;
use Illuminate\Http\Request;
use Illuminate\Support\Carbon;
use Laravel\Socialite\Facades\Socialite;

class LoginController extends Controller
{
    public function handleProviderSageCallback()
    {
        $auth_token = Socialite::driver('sage')->user(); // Fetch authenticated user

        ExtToken::updateOrCreate(
            [ 'driver' => 'sage' ],
            [
                'type' => 'bearer',
                'scope' => 'full_access',
                'access' => $auth_token->token,
                'refresh' => $auth_token->refreshToken,
                'access_expires' => Carbon::now()->addSeconds($auth_token->expiresIn),
                'refresh_expires' => Carbon::now()->addSeconds($auth_token->accessTokenResponseBody['refresh_token_expires_in'])
            ]
        );

        return redirect()->to('/dondequeira');
    }

    public function redirectToSageProvider()
    {
        return Socialite::driver('sage')->redirect();
    }
}

```

Es curioso que el retorno me devuelva un objeto en el que puedo consultar todo menos el token de refresco que tengo que ir a por él en un objeto que es una array en el que establos mismo datos más ese token.

Espero que te sirva, si llegaste aquí, porque estas cosas no suelen estar escritas por ahí.

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).



# Laravel Filament Admin funciona en Sail, pero no en producción. error 404 en ficheros .js

## Introducción

Terrorífico error de documentación de [FilamentaAdmin](#), que tras unas cuantas horas encontré respuesta.

Cierto que mi fracaso vino de hacer las cosas como no se deben. **Laravel Sail**, no es del todo confiable porque no usa servidor web (ni el que usas en producción) y yo en ciertos proyectos como este, no uso testing en mi Gitlab sino en local. Entono el *mea culpa*

“ Una razón más de que el proceso **desarrollo** -> **producción** tenga en algún momento una fase de testing con el mismo escenario de producción.

## Error

El error es claro. Se produce un error en la llamada a los ficheros `*.js` de la aplicación laravel (los que le afectan a Filament).

```
/filament/assets/app.js?id=942414d090ce297f343eb13f12bc7 error 404  
livewire/livewire.js?id=de3fca26689cb5a39af4 error 404
```

Developers Tools

En su documentación no habla nada del tema.

En Google hay tropecientos post pero nada.

## Solución

Prestada de del [comentario de @webboty](#) está claro que para usuario que desplegamos nuestro trabajo en un servidor Nginx.

Añadir la directiva `try_files $uri /index.php?$query_string;` al fichero del sitio virtual, en la sección `location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js|ico|xml)$ {`

```
location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js|ico|xml)$ {  
    try_files $uri /index.php?$query_string;  
    access_log      off;  
    log_not_found   off;  
    expires         14d;  
}
```

“ Importante no confundir esta sección con la sección `location / {`

Con eso y está solventado.

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Traducciones no funcionan en Laravel

## Introducción

A veces en las actualizaciones de versión [Laravel](#) se nos escapan cosas que no vienen en la documentación por son cientos las variantes en las que están implicados terceros.

En mi caso, en una aplicación Laravel 9, con FilamentAdmin, está usando las traducciones de un plugin de este, que al publicarse puso las traducciones en `resources/lang/`

Ahora en Laravel 10, si existe ese directorio, las traducciones de otros `vendors` que estén en el nuevo directorio `lang/` no serán traducidas como por ejemplo las traducciones de las validaciones.

## Solución

Mover el contenido de `resources/lang/` a `lang/` y eliminar la carpeta `resources/lang/`

### Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Comprobando el uso del trait

## Introducción

Uso un trait en algunos proyectos con dos métodos. Uno individual, y otro de Modelo completo.

Siempre me pregunte como verificar si realmente estaba usando los datos de cache o los datos de Mysql

Aqui, la respuesta

## CacheTrait

```
<?php

namespace App\Traits;

use Illuminate\Support\Str;
use App\Exceptions\DatabaseException;

trait CacheTrait
{
    /**
     * Get or cache a row from the model
     * @param string $modelName The name of the model (base name)
     * @throws DatabaseException
     */
    private static function findModelItemId(string $modelName, int $id, int $ttl = null):
mixed
    {
        $ttl = $ttl ?? config('sitelight.cache.general', 600);
        $baseNamespace = 'App\Models\\';
        $fullModelName = $baseNamespace . $modelName;

        if (!class_exists($fullModelName)) {
            throw new \InvalidArgumentException("The model {$fullModelName} does not exist.");
        }
    }
}
```

```

$model = Str::camel($modelName);

try {
    return cache()->remember(
        $model . $id,
        $ttl,
        function () use ($fullModelName, $id) {
            return resolve($fullModelName)::find($id);
        }
    );
} catch (\Exception $e) {
    throw new DatabaseException("Error accessing the database for model
{$fullModelName} with id {$id}.", 0, $e);
}
}

/**
 * Gets or caches a complete model. use wisely
 */
private static function cacheModel(string $modelName, int $ttl = null)
{
    $ttl = $ttl ?? config('sitelight.cache.general', 600);
    $baseNamespace = 'App\Models\\';
    $fullModelName = $baseNamespace . $modelName;

    if (!class_exists($fullModelName)) {
        throw new \InvalidArgumentException("The model {$fullModelName} does not exist.");
    }

    $model = Str::camel($modelName);

    try {
        return cache()->remember(
            $model,
            $ttl,
            function () use ($fullModelName) {
                return resolve($fullModelName)::all();
            }
        );
    }
}

```

```

        );
    } catch (\Exception $e) {
        throw new DatabaseException("Error accessing the database for model
{$fullModelName}.", 0, $e);
    }
}
}
}

```

## Prueba

Uso siempre un pequeño truco en mis proyectos de Laravel, usando un fichero para una clase de comando artisan.

Con el hago pruebas rápidas como esta, ya que al ser un tarit es necesario ejecutarlo desde una clase que lo use o en un test.

“ Uso [ray\(\)](#) para debug y desarrollo rápido, pero se puede cambiar por `Log::message();`

```
<?php
```

```

namespace App\Console\Commands\Develop;

use App\Traits\CacheTrait;
use Illuminate\Console\Command;
use Illuminate\Support\Facades\Redis;

class PandoraBoxCommand extends Command
{
    use CacheTrait;

    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'test:pandora';

```

```
/**
 * The console command description.
 *
 * @var string
 */
protected $description = 'Command description';

/**
 * Execute the console command.
 */
public function handle()
{
    ray('Test pandora');

    ray($this->findModelItemId('User', 2));
}
}
```

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Pest, PHPStorm y Laravel Sail

## Configurar nuestro entorno de trabajo

La comodidad de Laravel Sail es impresionante para quienes trabajamos con decenas de proyectos de versiones distintas, y de software diferente de PHP. Pero tiene sus cosillas. Aquí te dejo como configurar Laravel Sail + PHPStorm + PestPHP.

## Configuración

Bien supongo que ya tienes instalado Pest como plugin de PHPStorm. Ahora falta configurarlo.

Click en **[Command ⌘][,]** para abrir las preferencias.

Se abrirá la imagen de abajo, y deberas rellenarla

PHP Preferencias

Haciendo click en los tres puntitos del **CLI Interpreter** podrás seleccionar el que usarás con Laravel Sail.

CLI Interpreters

Es importante seleccionar la opción **Always start a new container ("docker-compose run")** ya que he visto algún video los super bloggers gurús que te lo dicen al revés y te saldrá un error.

```
[docker-compose:///Users/abkrim/Sites/swissknife_v3/docker-compose.yml]:laravel.test/]:php
vendor/pestphp/pest/bin/pest --teamcity --configuration phpunit.xml
/var/www/html/tests/Feature/Jobs/CpanelUsersSynchroJobTest.php "--
filter=^(P\\)?Tests\\Feature\\Jobs\\CpanelUsersSynchroJobTest::it\\sexample(\\swith\\s(data\\sset
\\s\\\".*\\\"|\\(.*\\))(\\s\\s(data\\sset\\s\\\".*\\\"|\\(.*\\)))*(\\s#\\d+)?)?$/\"
WARNING: Compose V1 is no longer supported and will be removed from Docker Desktop in an
upcoming release. See https://docs.docker.com/go/compose-v1-eol/
the input device is not a TTY

Process finished with exit code 1
```

También es importante que selecciones TÚ php de trabajo, para el proyecto. NO es copiar y pegar.





En test Frameworks, tendrás el plugin de Pest, y veras la configuración. Aunque pone local, no pongas el path completo sino el relativo al proyecto como en la imagen de abajo.

PHP > Test Framework

Cuando ejecutes los test desde el runner de PHPStorm, tendrás un erro que realmente es un warning. No he tenido tiempo de solventarlo, pero si te apetece, escíbeme y lo publico.  
[abdelkarim.mateos arroba castris.com]

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Laravel Herd y cosas que no estan documentadas.

## Introduccion

El cambio de Laravel Sail a usar Laravel Herd en MacOS, tuvo su mas y sus menos. Aqui dejo unos tips que fueron saliendo.

## Redis

```
Your requirements could not be resolved to an installable set of packages.
```

```
Problem 1
```

```
- Root composer.json requires PHP extension ext-redis * but it is missing from your system. Install or enable PHP's redis extension.
```

```
To enable extensions, verify that they are enabled in your .ini files:
```

- /opt/homebrew/etc/php/8.2/php.ini
- /opt/homebrew/etc/php/8.2/conf.d/ext-opcache.ini

```
You can also run `php --ini` in a terminal to see which files are used by PHP in CLI mode. Alternatively, you can run Composer with `--ignore-platform-req=ext-redis` to temporarily ignore these required extensions.
```

Bien, uso [DBngin](#) y no hay problema con Redis.

La solución con Homebrew es sencilla.

Necesitamos instalar en la versión de PHP que estemos necesitados de redis, la extension via pecl. (Se entiende que ya tenemos instalado redis como servidor)

```
/opt/homebrew/opt/php@8.2/bin/pecl install redis
```

```
downloading redis-6.0.2.tgz ...
```

```
Starting to download redis-6.0.2.tgz (365,966 bytes)
```

```
.....done: 365,966 bytes
```

```
43 source files, building
```

```
running: phpize
Configuring for:
PHP Api Version:      20220829
Zend Module Api No:   20220829
Zend Extension Api No: 420220829
enable igbinary serializer support? [no] :
enable lzf compression support? [no] :
enable zstd compression support? [no] :
enable msgpack serializer support? [no] :
enable lz4 compression? [no] :
use system liblz4? [yes] :
building in /private/tmp/pear/temp/pear-build-abkrimKX1ljZ/redis-6.0.2
...
...
252974137 1512 -rwxr-xr-x 1 abkrim wheel 772392 Nov 30 06:39 /private/tmp/pear/temp/pear-
build-abkrimKX1ljZ/install-redis-
6.0.2/opt/homebrew/Cellar/php@8.2/8.2.13/pecl/20220829/redis.so

Build process completed successfully
Installing '/opt/homebrew/Cellar/php@8.2/8.2.13/pecl/20220829/redis.so'
install ok: channel://pecl.php.net/redis-6.0.2
Extension redis enabled in php.ini
```

Eso es todo.

# PHP enums un gran aliado.

## Ejemplo con fechas

Una enumeracion PHP con el método `->dates()`

```
enum Range: string
{
    case Year = 'year';
    case Last_30 = 'last30';
    case Last_7 = 'last7';
    case Today = 'today';

    // esto hace super sencillo al como esto:
    // $query->whereBetween(Range::Last_30->date())

    return match ($this) {
        [static::Year => [Carbon::now()->startOfYear(), now()],
         static::Last_30 => [Carbon::today()->subDays(29), now()],
         static::Last_7 => [Carbon::today()->subDays(6), now()],
         static::Today => [Carbon::today(), now()],
        };
    }
}
```

### Enum Tip

```
> App\Enums\RangeDates::Last_30->dates()
= [
    Illuminate\Support\Carbon @1700352000 {#9470
        date: 2023-11-19 00:00:00.0 UTC (+00:00),
    },
    Illuminate\Support\Carbon @1702920948 {#9471
        date: 2023-12-18 17:35:48.748993 UTC (+00:00),
    },
]
```

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Manera elegante de obtener el phpinfo() en tu proyecto

## phpinfo()

A veces es necesario saber por que lugar andamos con el php, sobre todo cuando no es nuestra máquina, o no tenemos todo claro sobre el sistema en el que esta el proyecto en el que estamos tranado de solventar algún problema

La manera más elegante que conozco es en el archivo de rutas `routes/web.php`, añadir una para ver el phpinfo

```
Route::get('phpinfo', function () { phpinfo(); }->name('phpinfo'));
```

Después ya esta claro, `https://misitios.com/phpinfo`

## Agradecimientos

A [Brennan James](#) por su respuesta [How to display phpinfo\(\) within Laravel for debugging PHP?](#)

### Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Instalar ionCube en Laravel Herd en un mac Silicon

## Introducción

Hace poco que abandone el desarrollo con Laravel Sail tras la aparición de Laravel Herd. Pero hoy me enfrente a un tema del que no había mucha documentación. Era la necesidad de instalar las extensiones de Ioncube para el desarrollo de un Addon de WHMCS en entorno local.

## PHP e Ioncube en un Mac con chip Silicon M1/M2

Si escribo esto es porque a la fecha, 15/02/2024 el instalador de IonCube no funciona en el Mac.

## Versión de PHP

WHMCS pese al año en que estamos sigue con soporte solo hasta PHP 8.1 y encima esta ofuscado con Ioncube, así que instale PHP 8.1 en Herd.

Mi primera sorpresa fue que no se instalaba nada o no encontraba el path por lo que opte por instalar el PHP 8.1 como hago siempre en mac, via [Homebrew](#).

```
brew install php@8.1
```

## Directorio de extensiones

Después saber donde esta el directorio de extensiones

```
php -i | grep extension_dir
extension_dir => /lib/php/extensions/no-debug-non-zts-20210902 => /lib/php/extensions/no-
debug-non-zts-20210902
sqlite3.extension_dir => no value => no value
```

Ojo, esta información es relativa al PHP que se ejecuta en el shell, así que por favor, en [Laravel Herd](#), deberemos poner como PHP global esa version, o en su defecto usar un `phpinfo()` en un documento visible en el public de nuestro proyecto, para ver el path para esa versión.

## Descarga de Ioncube

### [Loaders](#)

Elegimos [macOS ARM M1 \(arm64 64 bits\) 13.0.2](#) que descargaremos y descomprimiremos.

Después debemos copiar o mover los ficheros de la extension a ese path.

```
sudo cp ioncube_loader_dar_8.1* /opt/homebrew/Cellar/php@8.1/8.1.27/lib/php/20210902/  
sudo chown $(whoami):admin  
/opt/homebrew/Cellar/php@8.1/8.1.27/lib/php/20210902/ioncube_loader_dar_8.1*
```

## Incorporar el modulo zend al php.ini

Desde Laravel Herd lo tenemos fácil.

Simplemente hay que ir al menu **Show php.ini** y seleccionar el `php.ini` de la versión que usamos.

Añadimos al final.

```
zend_extension=/opt/homebrew/Cellar/php@8.1/8.1.27/lib/php/20210902/ioncube_loader_dar_8.1.so
```

Salvamos el php.ini y reiniciamos Laravel Herd, y ya esta. Ya tenemos nuestra versión PHP 8.1 para FPM, preparada para ionCube.

### Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).



# El cast y su importancia en el modelo Eloquent de Laravel

## La importancia del cast

A veces, aprendemos lecciones inesperadas durante el trabajo y la práctica cotidiana. Así me ocurrió con el tema del cast de fechas en Laravel, específicamente en lo que respecta a los modelos Eloquent.

### Cast `datetime` vs. `datetime Y-m-d H:i:s`

Al realizar pruebas, el formato de salida de los campos `dateTime` solía frustrarme. No entraré en debates infructuosos sobre qué tipo de datos es mejor manejar. Creo firmemente en la madurez del ecosistema Laravel, y prefiero adaptarme a él, donde prevalecen el sentido común y la pragmática sobre las preferencias personales.

Por lo tanto, acostumbraba a hacer cast de los campos `dateTime` a `datetime Y-m-d H:i:s`.

Gran error.

Este enfoque ignora toda la elegancia interna del núcleo de Laravel, especialmente en lo que respecta a los setters y getters para los campos `datetime`, dejando de lado la posibilidad de utilizar funciones adicionales como `isToday()` y muchas otras, que permiten escribir más y mejor código con menos esfuerzo.

Recientemente descubrí mi error en este aspecto, y, afortunadamente, no fue tan grave. Solo necesité ajustar todos los tests donde aplicaba ese cast para verificar (por pereza) que ahora no funcionan precisamente porque el formato no coincide. Pero, como resultado, he logrado limpiar bastante código spaghetti.

### Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Phpredis en Laravel 10/11

## Introducción

Siempre he preferido en sistema el uso de sockets por que hay numerosa literatura y benchmarks, además de estar indicado pro [Redis - Benchmarks](#)

“ Cuando los programas de referencia del servidor y del cliente se ejecutan en el mismo equipo, se pueden utilizar tanto el loopback TCP/IP como los sockets de dominio Unix. Dependiendo de la plataforma, los sockets de dominio Unix pueden alcanzar alrededor de un 50% más de rendimiento que el loopback TCP/IP (en Linux, por ejemplo). El comportamiento predeterminado de redis-benchmark es utilizar el bucle invertido TCP/IP.

Pero esto no esta bien documentado en Laravel desde hace tiempo. Por eso mucha gente instala `predis/predis` en lugar de [PhpRedis](#). Si le añadimos el proceso de instalación de este último que en Mac Os se puede torcer un poco, y los primeros errores la gente abandona. Y la verdad es que tanto los sockets como PhpRedis, con más rápidos.

## Instalación en Macos M1/M2

Te recomiendo la lectura de [PhpRedis - Instalación](#)

En mi caso que uso intensivamente [Homebrew](#) es sencillo en el caso de usar [Laravel Herd - Extensiones adicionales](#)

```
pecl install redis
```

Si tienes más de una versión (Laravel Herd o similar) y deseas instalarlo en otras versiones

```
/opt/homebrew/Cellar/php@8.2/8.2.17/bin/pecl install redis
```

Después es añadirlo (activarlo) en el php.ini de la versión. Abajo te dejo un ejemplo de un php.ini con varias extensiones y formatos para añadirlas.

```
;extension=/opt/homebrew/Cellar/php@8.2/8.2.16_1/pecl/20220829/mongodb.so
extension=/opt/homebrew/lib/php/pecl/20220829/pcov.so
extension=/opt/homebrew/lib/php/pecl/20220829/redis.so
```

# Adaptar la configuración de Laravel

La otra cuestión es el como decirle a Laravel use el socket ya que la configuración y el mecanismo es distinto en **PhpRedis** que en **Predis**

Con **Predis** Laravel usa el path y con **PhpRedis** usa el host para decirle el socket. Asi que lo mejor es usar la varaibale REDIS\_HOST

## Phpredis

## Laravel 12

Solo pomnga la de phpredis que es la que uso pro eficacia y que cambio en Laravel 12 [Unix Socket Connections](#)

```
REDIS_CLIENT=phpredis
REDIS_SCHEME=unix
REDIS_PASSWORD=null
REDIS_HOST=/home/USER/.redis/redis.sock # Configuracion redis socket en Directadmin
REDIS_PORT=0
```

## Old laravel

```
REDIS_SCHEME=unix
REDIS_CLIENT=phpredis
# Redis host must be set to the path of the Redis socket and phpredis
REDIS_HOST=/tmp/redis.sock
#REDIS_PORT=null # is deprecated in php 8.3
REDIS_PASSWORD=null
REDIS_PATH=null
```

## Predis

```
REDIS_CLIENT=predis
REDIS_SCHEME=unix
REDIS_PASSWORD=null
```

```
#REDIS_PORT=null # is deprecated in php 8.3  
REDIS_PATH=/tmp/redis.sock
```

# Prueba de concepto

```
php artisan cache:clear
```

Si no te falla, es que ya estas usando tu **Redis** via **PhpRedis**

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

# Livewire & Laravel Localization: The GET method is not supported for route livewire/update 404

## Livewire & mcamara/laravel- localization: The GET method is not supported for route livewire/update 404

### Introducción

Es terrible cuando aparecen los errores 404 en tu app con livewire. El mundo se apaga a tu alrededor y comienzas un viaje en el que no hay mucha información, hay mucha mezcla de datos inconexos y tu no sabes que hacer.

Voy a tratar este escenario, con todos los datos para que el que llegue aquí, sepa de que estamos hablando, por que es un caso concreto para un mismo enunciado: **The GET method is not supported for route livewire/update 404**

### Escenario

En un aplicación que usa [Laravel 11](#), [FilamentAdmin](#) y por extensión [Livewire](#) multi idioma, basado en la librería [LaravelLocalization](#).

La cuestión es que al hacer click en el icono de expansión de un acordeón (accordion) típico de una sección FAQ, me saltaba un terrible error 404.

Barra de depuración : Herramientas del desarrollador

En la barra de debugger al margen de ver `POST livewire/update` vemos su error.

The route es could not be found.

vendor/laravel/framework/src/Illuminate/Routing/AbstractRouteCollection.php#44

Symfony\Component\HttpKernel\Exception\NotFoundHttpException

```
        return $this->getRouteForMethods($request, $others);
    }

    throw new NotFoundHttpException(sprintf(
        'The route %s could not be found.',
        $request->path()
    ));
```

Lo primero que pensamos es: bueno, esto trata de añadir a la configuración de Laravel Localization, una exclusión de ese path aunque habría que pensar que el método POST ya esta excluido, luego no tiene sentido.

```
'urlsIgnored' => [
    '/admin',
    '/admin/*',
    '/admin/multimedia',
    '/storage/*',
    '/articles/*',
    '/_debugbar/*',
    '/colours',
    '//'/livewire/*',
],

'httpMethodsIgnored' => ['POST', 'PUT', 'PATCH', 'DELETE'],
```

Así que no va por ahí.

Recuerdos de un 404 en [Problem with site in production: livewire.js and app.js 404](#) y de otro tip con FilamentAdmin v3, para añadir al composer.json en la sección scripts.post-update-cmd "@php artisan vendor:publish --tag=livewire:assets --ansi --force"

Pero no. Eso ya lo uso.

La solución pasa por algo que esta en la documentación de Livewire, pero de esas cosas que pasan desapercibidas, como [Configuring Livewire's update endpoint](#)

- Añadir a la ruta que usa la localización en el frontend

```
Route::group(['prefix' => LaravelLocalization::setLocale()], function() {  
    Route::get('/', [HomeController::class, 'index']);  
  
    Route::get('/blog', [BlogController::class, 'index'])->name('blog.index');  
    Route::get('/blog/{article:slug}', [BlogController::class, 'article'])->  
>name('blog.article');  
  
    // Esto es lo que hay que añadir  
    Livewire::setUpdateRoute(function ($handle) {  
        return Route::post('/livewire/update', $handle);  
    });  
});
```

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).