Librería Respect/Validation

Instalación:

El paquete está disponible en Packagist, se puede instalar usando Composer, previamente descargado e instalado.

Para la instalación se requiere tener instalada la versión PHP 7.3 o superior. Descargar e instalar Symfony para la extensión Mbstring.

Luego es necesario instalar stringifier. Esto puede hacerse desde la terminal de Visual Studio Code o de comandos del sistema con el siguiente comando:
composer require respect/stringifier

Luego de haber hecho todo eso, en la terminal copiar el siguiente comando:
composer require respect/validation

Y luego de esto ya se puede utilizar la librería.

Guía de Funciones:

Importación de espacio de nombre

Respect \ Validation tiene un espacio de nombres, pero puede hacer su vida más fácil importando una sola clase en su contexto:

```
use Respect\Validation\Validator as v;
```

Validación en Cadena

Es posible utilizar validadores en cadena. El siguiente ejemplo muestra una cadena que contiene números y letras, sin espacios en blanco y una longitud entre 1 y 15.

```
$usernameValidator = v::alnum()->noWhitespace()->length(1, 15);
$usernameValidator->validate('alganet1'); // true
```

Funciones:

A continuación se detalla una lista de las funciones utilizadas en el proyecto. Para la documentación completa de la librería dirigirse al siguiente enlace: Respect/Validation.

Positive

Valida que la entrada sea un número positivo

```
v::positive()->validate(1); // true
v::positive()->validate(0); // false
v::positive()->validate(-15); // false
```

NoWhitespace

- NoWhitespace()

Valida si una cadena no contiene espacios en blanco(espacios, tabulaciones y saltos de línea);

```
v::noWhitespace()->validate('foo bar');  //false
v::noWhitespace()->validate("foo\nbar"); // false
```

Length

- Length(int $min, int $max)
- Length(int $min, null)

- Length(null, int $max)
- Length(int $min, int $max, bool $inclusive)

Valida la longitud de la entrada dada.

Ejemplo más simple:
```
v::stringType()->length(1, 5)->validate('abc'); // true
```

También puede validar solo la longitud mínima:
```
v::stringType()->length(5, null)->validate('abcdef'); // true
```

Solo longitud máxima:
```
v::stringType()->length(null, 5)->validate('abc'); // true
```

El tipo como primer validador de una cadena es una buena práctica, ya que la longitud acepta muchos tipos:
```
v::arrayVal()->length(1, 5)->validate(['foo', 'bar']); // true
```

Se puede pasar un tercer parámetro para validar los valores pasados inclusive:
```
v::stringType()->length(1, 5, true)->validate('a'); // true
```

La plantilla de mensaje para este validador incluye {{minValue}} y {{maxValue}}.

Alpha

- Alpha()
- Alpha(string …$additionalChars)

Valida que la entrada contenga solo caracteres alfabéticos.

```
v::alpha()->validate('some name'); // false
v::alpha(' ')->validate('some name'); // true
v::alpha()->validate('Cedric-Fabian'); // false
v::alpha('-')->validate('Cedric-Fabian'); // true
v::alpha('-', '\'')->validate('\'s-Gravenhage'); // true
```

Puede restringir el uso de uppercase() y lowercase() mediante las reglas de lowercase() y uppercase().

```
v::alpha()->lowercase()->validate('EXAMPLE'); // false
v::alpha()->uppercase()->validate('example'); // false
```

Regex

- Regex(string $regex)

Valida que la entrada coincida con una expresión regular definida.

```
v::regex('/[a-z]/')->validate('a'); // true
```

La plantilla de mensaje para este validador incluye {{regex}}.

MinAge

- MinAge(int $age)
- MinAge(int $age, string $format)

Valida una edad mínima para una fecha determinada. El argumento $format debe estar de acuerdo con la función date() de PHP. Cuando $format no se proporciona, esta regla acepta los formatos de fecha y hora admitidos por PHP.

```
v::minAge(18)->validate('18 years ago'); // true
v::minAge(18, 'Y-m-d')->validate('1987-01-01'); // true

v::minAge(18)->validate('17 years ago'); // false
v::minAge(18, 'Y-m-d')->validate('2010-09-07'); // false
```

Date

- Date()
- Date(string $format)

Valida que la entrada sea una fecha. El argumento $format debe estar de acuerdo con la función date() de PHP, pero solo se permiten:

Formato	Descripción	Valores
d	Día del mes, 2 dígitos con ceros a la izquierda	01 al 31
j	Día del mes sin ceros a la izquierda	1 hasta 31
S	Sufijo ordinal en inglés para el día del mes, 2 caracteres	st, nd, rd o th
F	Una representación textual completa de un mes, como enero o marzo	Enero a diciembre
m	Representación numérica de un mes, con ceros a la izquierda	01 a 12
M	Una breve representación textual de un mes, tres letras.	De enero a diciembre
n	Representación numérica de un mes, sin ceros a la izquierda	1 hasta 12
Y	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1988 o 2017
y	Una representación de dos dígitos de un año	Ejemplos: 88 o 17

Cuando $format no se proporciona, su valor predeterminado es Y-m-d.

```
v::date()->validate('2017-12-31'); // true
v::date()->validate('2020-02-29'); // true
v::date()->validate('2019-02-29'); // false
v::date('m/d/y')->validate('12/31/17'); // true
v::date('F jS, Y')->validate('May 1st, 2017'); // true
v::date('Ydm')->validate(20173112); // true
```

Phone

- Phone()

Valida que la entrada sea un número de teléfono válido.

Valida un número de teléfono válido de 7, 10, 11 dígitos (América del Norte, Europa y la mayoría de los países de Asia y Medio Oriente), que admiten códigos de país y área (en anotaciones de puntos, espacios o guiones) como:

- (555) 555-5555
- 555 555 5555
- +5 (555) 555.5555
- 33 (1) 22 22 22 22
- +33 (1) 22 22 22 22
- +33 (020) 7777 7777
- 03-6106666

StringType

- StringType()

Valida si el tipo de entrada es cadena o no.

```
v::stringType()->validate('hi'); // true
```

Integrantes:
Graff Rocío FAI-2158
Scantamburlo Santiago FAI-2238