# Practica

| **Historia de Usuario** |
| --- |
| Yo como estudiante, necesito ingresar mi nombre y validar que el nombre no tenga números o caracteres extraños.<br><br>Criterios de aceptación<br>• Se debe imprimir el nombre que se ingresa<br>• El texto que se imprime debe estar en mayúsculas<br>• Retorna una alerta cuando sobrepasa los 100 caracteres<br>• Debe retornar un mensaje de alerta cuando ingrese un número<br>• Debe imprimir también la cantidad de palabras en el nombre |

| **TO-DO** |
| --- |
| ➢ "cristian" devuelve "CRISTIAN"<br>➢ "pedro" devuelve "PEDRO"<br>➢ Si valido la cantidad de caracteres de la cadena "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam" me retorna falso<br>➢ Si valido la cantidad de caracteres de la cadena "Lorem ipsum dolor sit amet, consectetur adipiscing elit" retorna true<br>➢ La cadena "Crist1an Bot1na" devuelve un mensaje de error<br>➢ "Cristian Leandro Botina" retorna un 3 |

**PRACTICA TDD**

1. Instalar un proyecto Laravel.

- Abrir cmd y dirigirse a xampp/htdocs, ejemplo:
  cd xampp/htdocs

- Clonar el repositorio:
  git clone https://github.com/krihs323/practica1.git practica1

- Después de ejecutar el comando, en la consola vamos a la carpeta practica1:
  cd practica1

- Instalar dependencias:
  composer install

- Abrir Xampp y habilitar el servicio Apache

2. Se accede a las vista ejemplo1 en el navegador web, se puede acceder desde la siguiente ruta:
http://localhost/practica1/public/comprobarnombre

3. Crear el test case desde la consola:
```
php artisan make:test EstudianteTest --unit
```

4. Abrir EstudianteTest que se creó anteriormente dentro de la carpeta test y crear los test unitarios, usar la librería EstudianteController:
```
use App\Http\Controllers\EstudianteController;
```

5. Cada vez que se desee ejecutar las pruebas unitarias, se debe dirigir desde la consola a vendor/bin/ y se ejecutan las pruebas con el siguiente comando:
```
phpunit --colors ../../tests
```

6. Para la primera prueba ***"cristian" devuelve "CRISTIAN"***. Agregar el siguiente test en EstudianteTest para probar la función que me debe convertir el nombre a mayúsculas y ejecutar PHPUNIT

```
public function testRetornaMayusculas(){
      //AAA
      //ARRANGE / ACT / ASSER
      $estudiante = new EstudianteController();

      $resultado = $estudiante->retornaMayuscula("cristian");

      $this->assertEquals("CRISTIAN" , $resultado);
    }
```

7. Crear la función retornaMayuscula dentro de la clase EstudianteController que se encuentra dentro de la carpeta Http/Controllers/ y probar en PHPUnit, para que pase la prueba

```
  public function retornaMayuscula( $cadena ){
      return "CRISTIAN";
    }
```

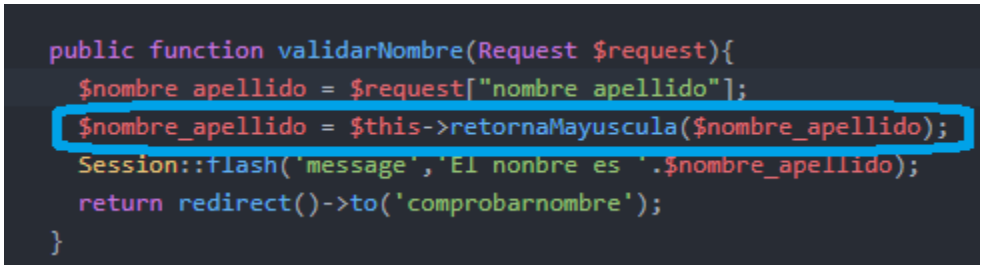
8. En la segunda prueba ***"pedro" devuelve "PEDRO"*** se agrega en la clase EstudianteTest la segunda prueba unitaria y luego ejecutar el PHPUnit

```
public function testRetornaMayusculasCadenaPedro(){
        //AAA
        //ARRANGE / ACT / ASSER
        $estudiante = new EstudianteController();

        $resultado = $estudiante->retornaMayuscula("pedro") ;

        $this->assertEquals("PEDRO",$resultado) ;
    }
```

9. Se aplica Refactorización en la función retornaMayuscula en la clase EstudianteController ya que solo esta funcionando para la variable "cristian" pero no para la variable "pedro"

```
public function retornaMayuscula($cadena){
        return strtoupper($cadena);
}
```

10. En EstudianteController llamar la función que se creó y se probó para convertir letras minúsculas a mayúsculas, la función principal validarNombre(), debería quedar así:



```
public function validarNombre(Request $request){
  $nombre_apellido = $request["nombre_apellido"];
  $nombre_apellido = $this->retornaMayuscula($nombre_apellido);
  Session::flash('message','El nonbre es '.$nombre_apellido);
  return redirect()->to('comprobarnombre');
}
```

```
$nombre_apellido = $this->retornaMayuscula($nombre_apellido);
```

11. Crear el resto de funciones necesarias para cumplir con los criterios de aceptación de la historia de usuario aplicando Test Driven Development (TDD)

Algunas funciones PHP para realizar la práctica, obtenidos de http://php.net/manual/es/

| Funciones PHP | Descripción |
|---|---|
| strtoupper | Devuelve el string con todos los caracteres alfabéticos convertidos a mayúsculas. |
| str_split | Convierte un string en un array. |
| is_numeric | Comprueba si una variable es un número o un string numérico |
| Strlen | Obtiene la longitud de un string |

| | |
|---|---|
| Explode | Divide un string en varios string |
| empty | Determina si una variable está vacía |

Algunos Asserts de PHPUnit para realizar la práctica, obtenidos de https://phpunit.readthedocs.io/es/latest/assertions.html

| Aserciones | Descripción |
|---|---|
| assertEquals(`$expected, $actual, $message`) | Reporta un error identificado por el $message si las dos variables $expected y $actual no son iguales. |
| assertTrue(`$condition, $message`) | Reporta un error identificado por el $message si $condition es false. |
| assertFalse(`$condition, $message`) | Reporta un error identificado por el $message si $condition es true. |