TUTORIAL CODEIGNITER PARTE 3

Validación de campos

1. Crear formulario tipo Bootstrap. Insertar CSS y Javascript. Se llamara form.php

```
<form method="POST">
    <div class="form-group">
        <label for="my-input">Email</label>
        <input id="my-input" class="form-control" type="text" name="email" placeholder="Email"
    </div>
    <div class="form-group">
        <label for="password">Password</label>
        <input id="password" class="form-control" type="password" name="password" placeholder=
    </div>
    <div class="form-group">
        <label for="categoria">Categoria</label>
        <select name="categoria" id="categoria" class="form-control">
            <option value=""></option>
            <?php foreach ($categorias as $cate): ?>
                <option value="<?= $cate ?>"><?= $cate ?> </option>
            <?php endforeach; ?>
        </select>
    </div>
    <button class="btn btn-dark" type="submit">Guardar</button>
</form>
```

2. **Crear el controlador Form.php**

```
use CodeIgniter\Controller;

class Form extends Controller{

    public function index()
    {
        //paso 4 se implementa un helper
        helper(['form']);

        //paso 1
        $data = [];
        $data['categorias'] = [
            'Estudiante',
            'Profesor',
            'Coordinador'
        ];
```

```php
        //paso 5. Implementar reglas

        if($_POST){
            $rules = [
                'email' => 'required'
            ];
            if($this->validate($rules)){
                //entonces se haria la inserción de datos y el login del usuario
            }else{
                $data['validacion'] = $this->validator;
            }

        }
```

```php
    /*con la etiqueta pre puedo imprimir exactamente como se ve en el código, incluido
    etiquetas html
            //paso 3
            if($_POST){
                echo '<pre>';
                    print_r($_POST);
                echo '</pre>';
            }
            */

            //paso 2
            return view('form', $data);
        }
    }
```

3. Añadir la parte respectiva en el formulario

```php
<body>
    <div class="container">
        <h3>Formulario con validaciones</h3>
        <!--Paso 6 -->
        <?php if(isset($validacion)): ?>
            <div class="text-danger">
                <?= $validacion->listErrors(); ?>
            </div>
        <?php endif; ?>
        <form method="POST">
            <div class="form-group">
                <label for="my-input">Email</label>
                <input id="my-input" class="form-control" type="text
```

4. Cambio de idioma

Validation.php × App.php

system > Language > en > Validation.php

```php
    'required'              => 'El campo {field} es requerido.',
    'required_with'         => 'The {field} field is required when
    'required_without'      => 'The {field} field is required when
```

5. Luego validamos email

```
    $rules = [
        'email' => 'required|valid_email'
    ];
```

Otras validaciones. Por ejemplo, que solo haya unos valores permitidos para la selección o para lo escrito en un input:

```
$rules = [
    'email' => 'required|valid_email',
    'password' => 'required|min_length[8]',
    'categoria' => 'in_list[Estudiante, Profesor]'
];
```

6. Si queremos conservar lo ingresado en los input y select después de aplicada la validación, utilizaríamos set_value para los input y set_select para los select, así:

```
.v class="form-group">
  <label for="my-input">Email</label>
  <input id="my-input" value="<?=set_value('email')?>" class="form-control"
iv>
.v class="form-group">
  <label for="password">Password</label>
  <input id="password" value="<?=set_value('password')?>" class="form-contr
iv>
.v class="form-group">
  <label for="categoria">Categoria</label>
  <select name="categoria" id="categoria" class="form-control">
      <option value=""></option>
      <?php foreach ($categorias as $cate): ?>
          <option <?=set_select('categoria', $cate)?> value="<?= $cate ?>">
      <?php endforeach; ?>
  </select>
iv>
```

7. Devolviendo un mensaje cuando la validación es correcta:

Al final del método index() se digita:

```
    }

    function exitoso(){
        return 'Waw, has pasado la validación. Felicidades!';
    }
}
```

Y luego el if de la validación se digita:

```
            'categoria' => 'in_list[Estudiante, Profesor]'
        ];
        if($this->validate($rules)){
            //entonces se haría la inserción de datos y el login del usuario
            return redirect()->to('/form/exitoso');
        }else{
            $data['validacion'] = $this->validator;
```

Creando una validación personalizada

1. Experimentamos con algunas cosas nuevas en la validación, por ejemplo, del email:

```
if($_POST){
    $rules = [
        //'email' => 'required|valid_email',
        /*la siguiente es otra manera de validad los campos, más personalizada
        en este caso, el email */
        'email' => [
            'rules' => 'required|valid_email',
            'label' => 'Dirección de correo'
            /* 'errors' => [
                'required' => 'Ey, '
            ] */
        ],
        'password' => 'required|min_length[8]',
```

2. Ahora lo intentamos con las fechas

2.1. Primero creamos el nuevo input y hacemos un print_r

```
iv class="form-group">
  <label for="fecha">Fecha</label>
  <input id="fecha" value="<?=set_value('fecha')?>" class="form-control" type="date" name="fecha"
div>
php
  echo '<pre>';
  print_r($_POST);
  echo '<pre>';

utton class="btn btn-dark" type="submit">Guardar</button>
>
```

Notamos que la fecha puede tener un formato diferente en su envío a como lo pusimos en el frontend.

2.2. Digitamos lo respectivo en el controlador Form.php

```
'password' => 'required|min_length[8]',
'categoria' => 'in_list[Estudiante, Profesor]',
'fecha' => [
    'rules' => 'required|check_date',
    'label' => 'Fecha',
    'errors' => [
        'check_date' => 'No puedes añadir una fecha antes de hoy'
    ]
]
```

2.3. En app creamos Validations/customRules.php

```
app > Validations > customRules.php > PHP Intelephense > customRules
<?php
namespace App\Validations;

class customRules{
    function check_date(string $str, string &$error = null): bool
    {
        if($str < date('Y-m-d')){
            return false;
        }
        return true;
    }
}
```

2.4. Y se especifica esta regla en Config/Validation.php

```
Form.php   Validation.php ...\Config   customRules.php
app > Config > Validation.php > Validation > $ruleSets
use CodeIgniter\Validation\FormatRules;
use CodeIgniter\Validation\Rules;
use App\Validations\customRules;

class Validation
{
```

```
public $ruleSets = [
    Rules::class,
    FormatRules::class,
    FileRules::class,
    CreditCardRules::class,
    customRules::class
];
```