|  |  |
| --- | --- |
|  | **2017** |
|  | **GEMA SAS**  Edgar Andres Rios Infante |

|  |
| --- |
| **[Manual Gema sas]** |
| Instructivo con las funcionalidades del sistema |

**Este proyecto está realizado en PHP Y MSQL Utilizando el Framework Symfony Versión 3.3.10 el cual nos permite adoptar el patrón de diseño MVC con el objetivo de tener un proyecto más seguro y mejor estructurado. En cuanto a la base de datos se utiliza MariaDB. Y para el diseño de Front-end el Framework Bootstrap en su versión 4.0.**

**Tiempo planeado:** Para el diseño e implementación del proyecto el tiempo inicial estimado fue de 4 horas. En cuanto al manual o documento de procedimientos se contemplaron 2 horas aproximadamente.

**Solución:** Para la solución de este proyecto se creó una entidad llamada Usuarios con los campos que se requerían, además se empleó una Clase para validar el contenido del archivo de texto, teniendo en cuenta su extensión, formato y validaciones de los atributos de la entidad Usuario en el caso de que fueran necesarias.

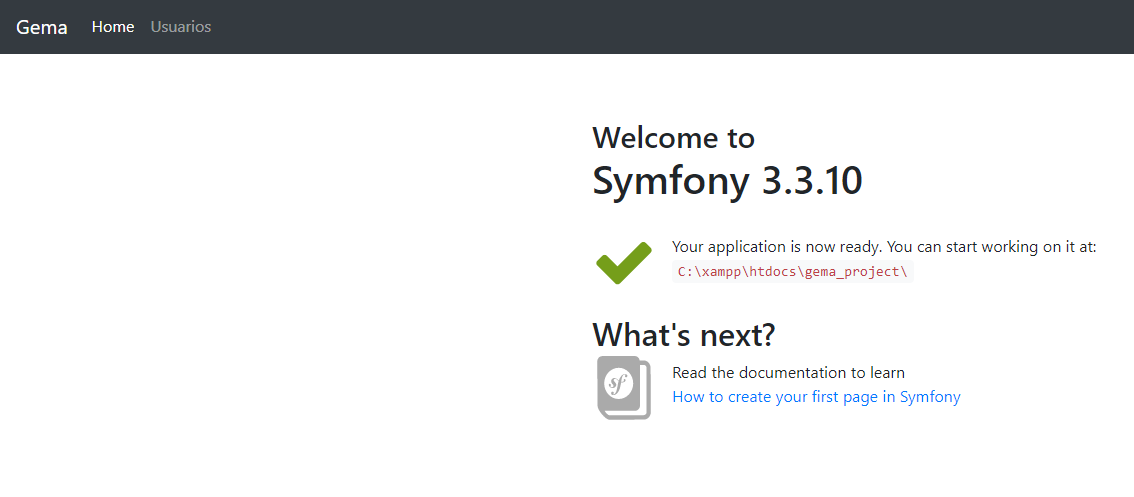
Para cargar el archivo se realizó un formulario con un botón sbmit el cual llamaría a un controlador, el cual valida dicho archivo haciendo uso de la clase ValidarArchivo, en el caso que este contenga errores se retornara nuevamente al formulario pero desplegando una lista de errores encontrados. Por otra parte si el archivo es correcto la clase ValidarArchivo retorna una lista de usuarios mediante un ArrayCollection con la información ya validada y sin errores para almacenar en la base de datos.

Finalmente la información ya podrá ser visualizada según el orden requerido.

**Tiempo real de solución:** Para cumplir con todos los requerimientos fueron necesarias 10 horas Aproximadamente.

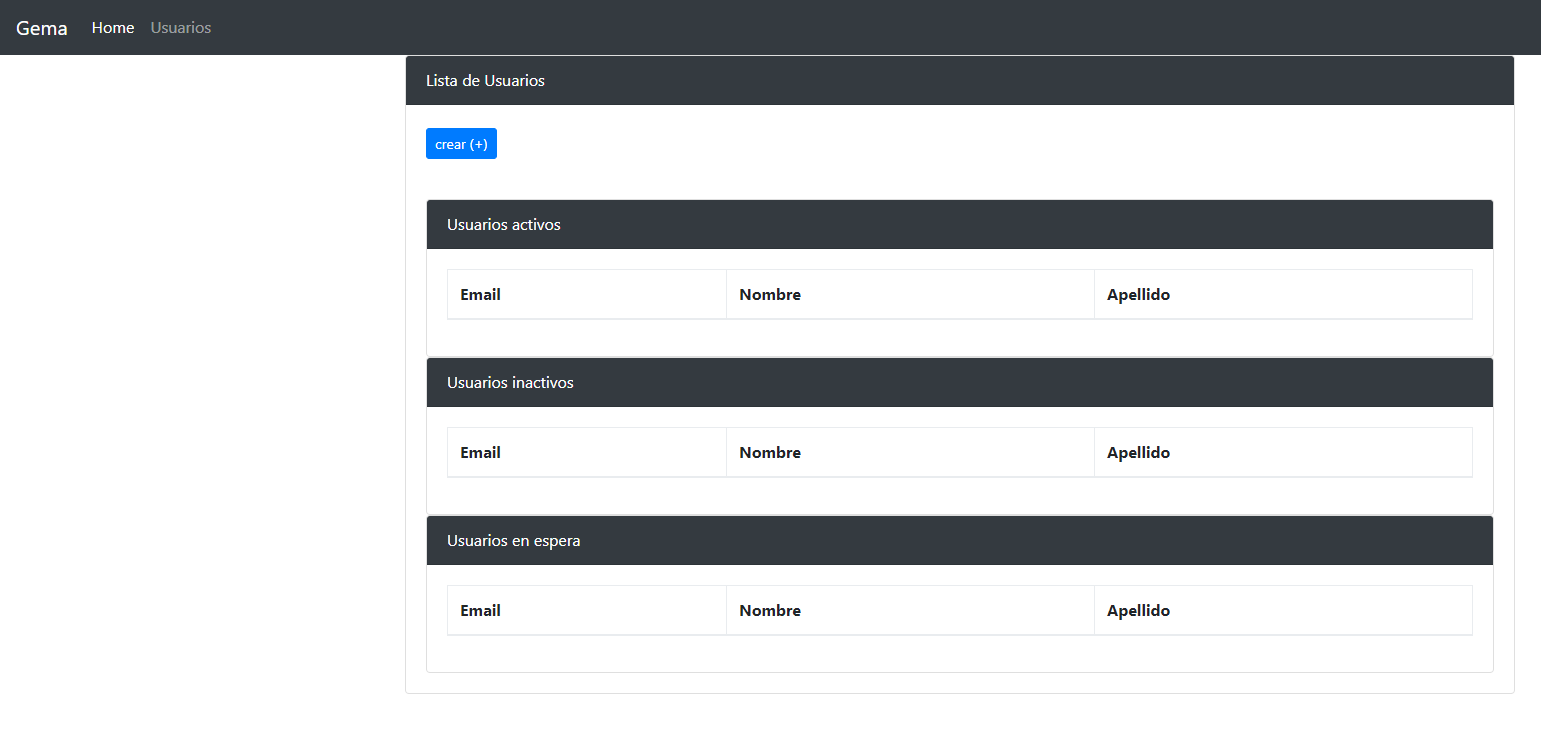
URL o ruta inicial de nuestro proyecto



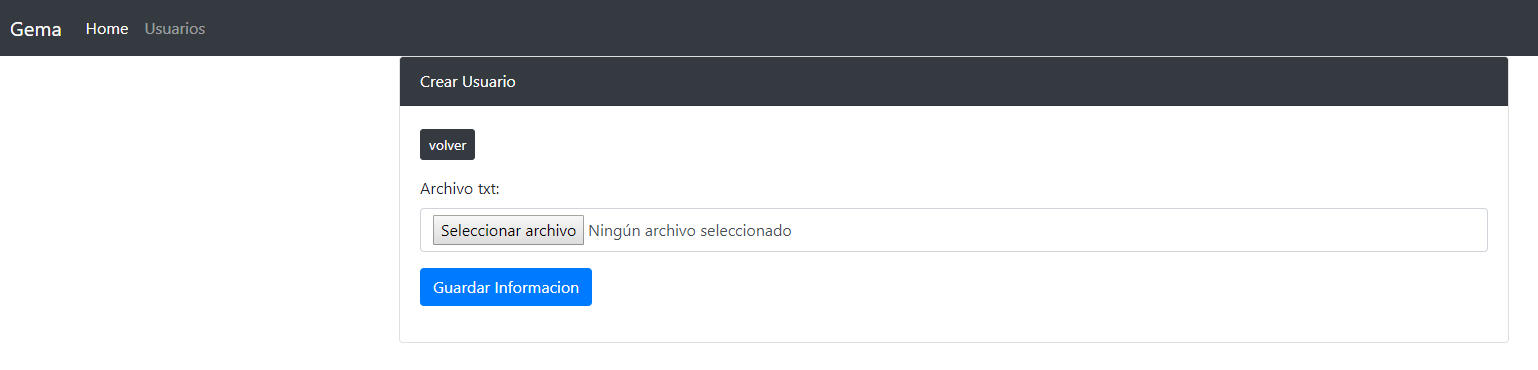


Menú de usuario, al presionar sobre el botón Home nos dirigiremos al inicio de nuestra aplicación, mientras que si presionamos el botón Usuarios nos desplegara los datos almacenado en la base de datos, en este caso no hay elementos creados.

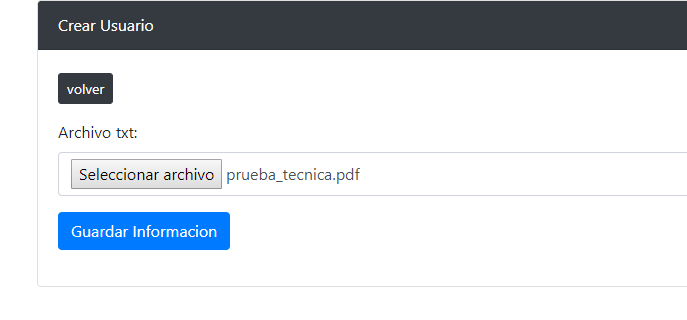


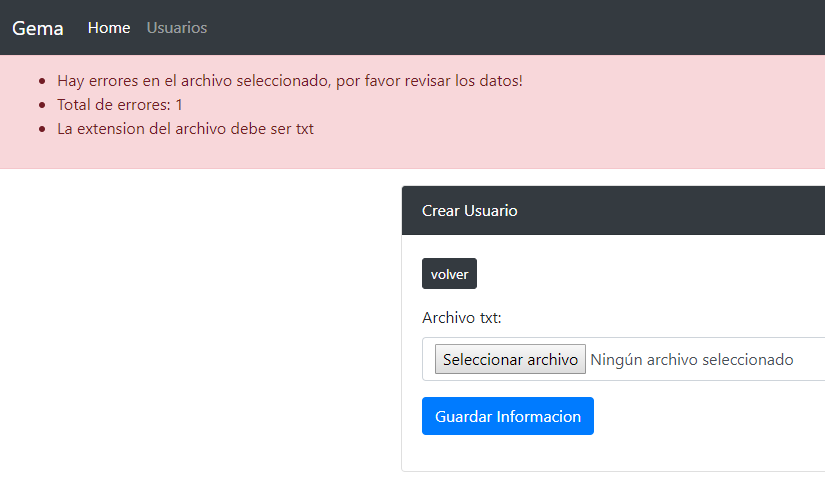


Al presionar sobre el botón crear la aplicación nos llevara al formulario para realizar la carga de archivos.

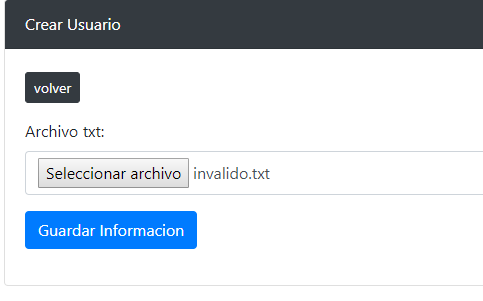


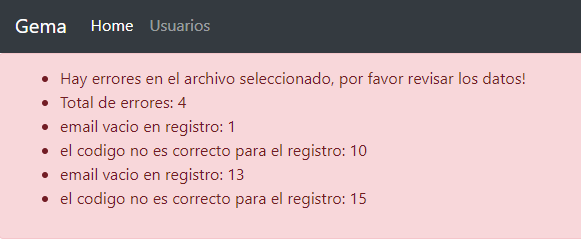
Al subir un archivo con una extensión diferente, la aplicación desde el servidor nos retornara un error antes de realizar la lectura del mismo. Hay que tener en cuenta que se pueden hacer validaciones también desde el cliente.



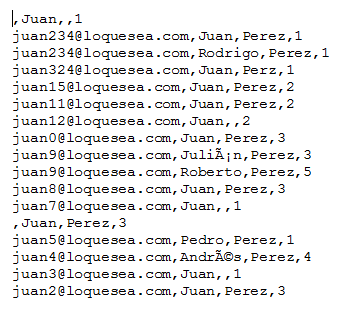


Si se sube un archivo con extensión correcta, pero que contenga errores de validación de los atributos, o que su estructura del archivo no esté acorde a lo contemplado, la aplicación desplegara otra seria de errores como se ve a continuación.

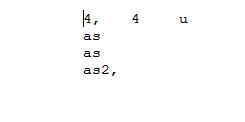


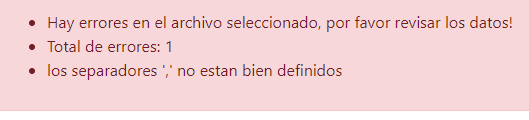


Como se puede observar los errores detectados para este archivo de prueba se encuentran encerrados en un círculo.

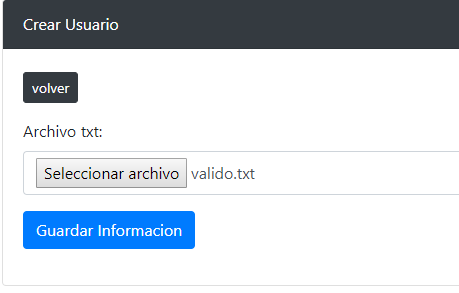


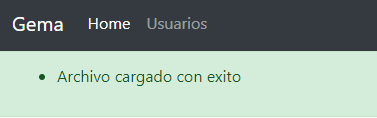
A continuación se examina un archivo que no tiene la estructura contemplada

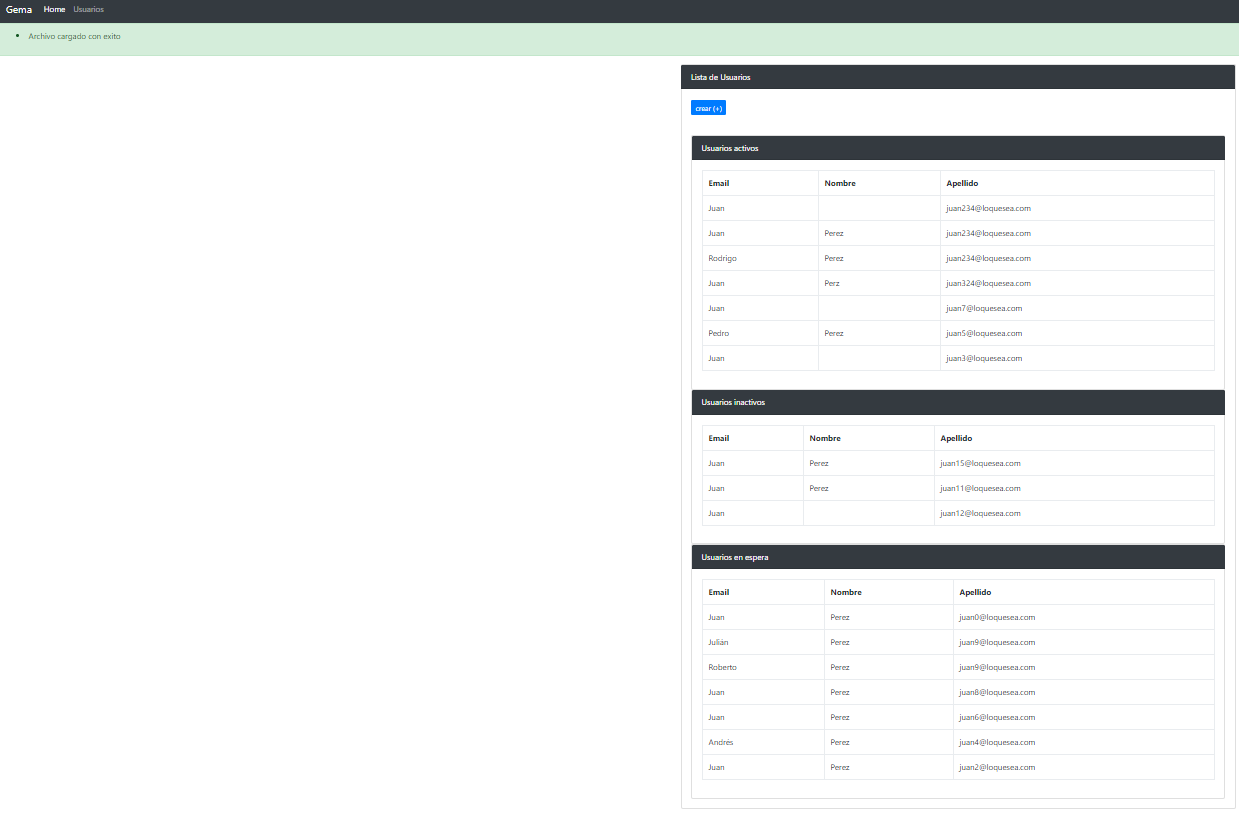


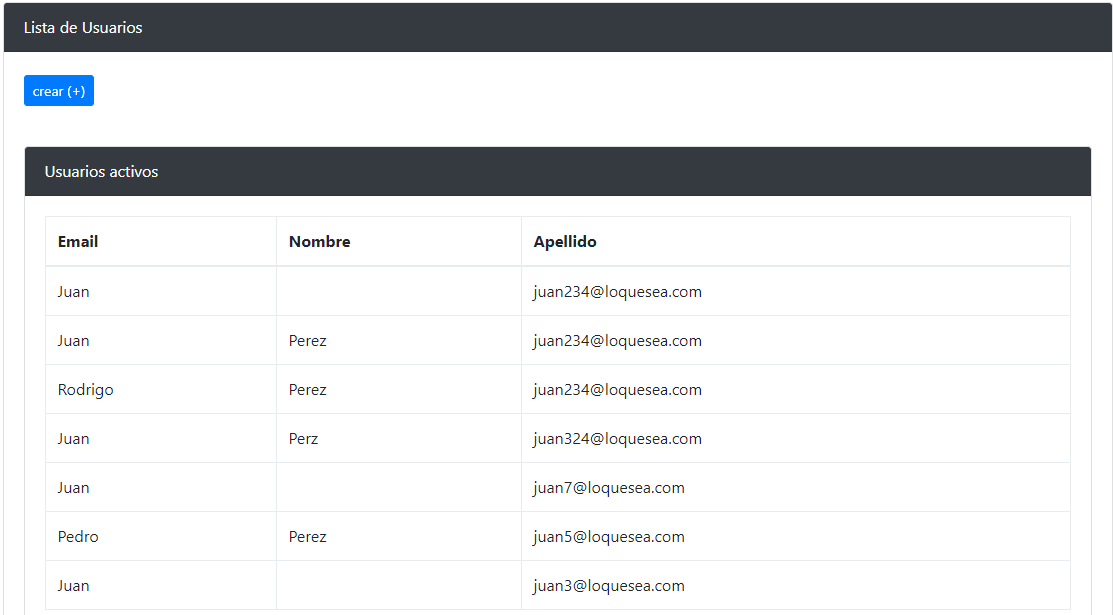


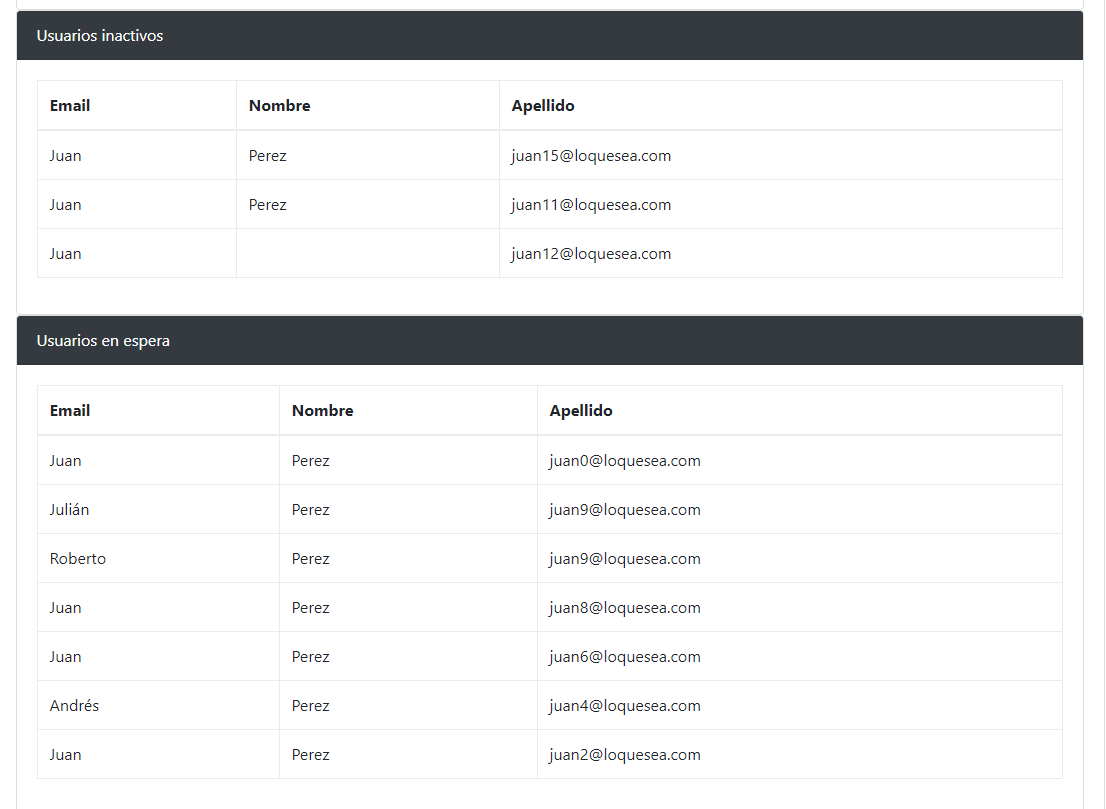
Cuando se sube un archivo ya validado y que cumple los requerimientos se almacena la información en la base de datos y por consiguiente ya estará lista para visualizarse.

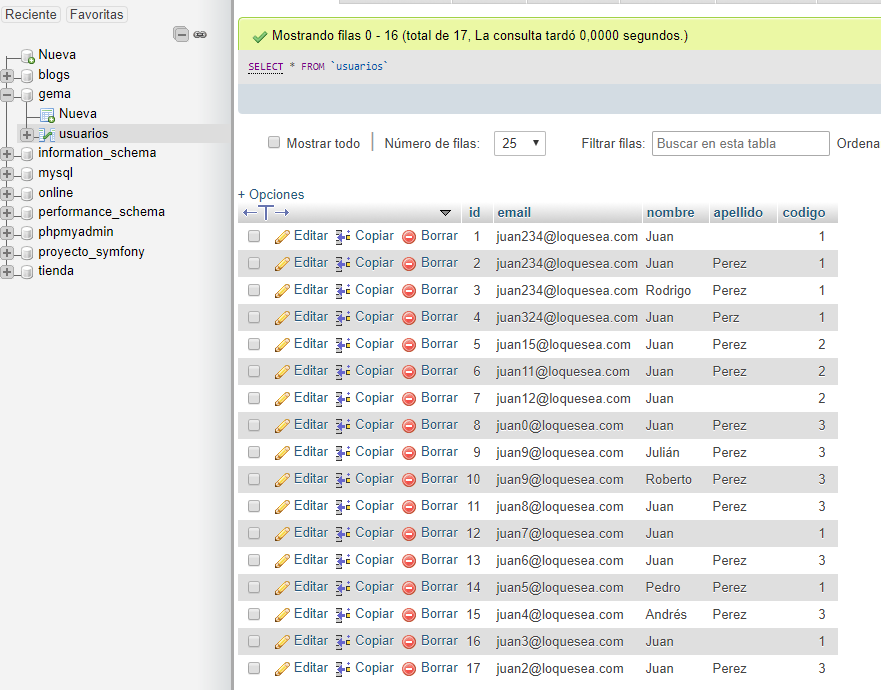












La lógica se encuentra prácticamente en estas tres partes Entidad,Controlador,ValidarArchivo

**<?php**

namespace AppBundle\Servicio;

use Doctrine\Common\Collections\ArrayCollection;

use AppBundle\Entity\Usuario;

class **ValidarArchivo** {

private $errores;

private $extension;

private $registros;

private $cantErr;

private $usuarios;

public function **\_\_construct**($file) {

$this->cantErr = 0;

$this->errores = new ArrayCollection;

$this->extension = $file->guessExtension();

$this->registros = file($file);

$this->usuarios = new ArrayCollection;

$this->validarExtension();

}

public function **validarExtension**() {

if ($this->getExtension() == "txt") {

$this->leerArchivo();

} else {

$this->addError("La extension del archivo debe ser txt");

}

}

public function **addError**($error) {

$this->errores->add($error);

$this->cantErr++;

}

public function **leerArchivo**() {

foreach ($this->getRegistros() as $num\_reg => $registro) {

$col = explode(",", $registro);

if (count($col) != 4) {

$this->addError("los separadores ',' no estan bien definidos");

return;

} else {

$usr = new Usuario();

foreach ($col as $key => $value) {

$this->validarCampos($value, $key, $num\_reg);

}

if ($this->getCantErr() == 0) {

$usr->setEmail($col[0]);

$usr->setNombre($col[1]);

$usr->setApellido($col[2]);

$usr->setCodigo($col[3]);

$this->addUsuario($usr);

}

}

}

}

public function **validarCampos**($value, $key, $num\_reg) {

if ($key == 0) {

if ($value == null || empty($value)) {

$this->addError("email vacio en registro: " . ($num\_reg + 1));

}

} elseif ($key == 1) {

} elseif ($key == 2) {

} elseif ($key == 3) {

if ($value == 1 || $value == 2 || $value == 3) {

} else {

$this->addError("el codigo no es correcto para el registro: " . ($num\_reg + 1));

}

} else {

$this->addError("los separadores ',' no estan bien definidos");

return;

}

}

public function **addUsuario**(Usuario $user) {

$this->usuarios->add($user);

}

public function **getErrores**() {

return $this->errores;

}

public function **getRegistros**() {

return $this->registros;

}

public function **getCantErr**() {

return $this->cantErr;

}

function **getExtension**() {

return $this->extension;

}

function **getUsuarios**() {

return $this->usuarios;

}

}

public function **guardarAction**(Request $request) {

if (!$request->isMethod('POST')) {

return $this->redirectToRoute('usuario.listar');

}

$usuario = new Usuario();

$form = $this->createForm(UsuarioType::*class*, $usuario);

$form->handleRequest($request);

$file = $form['nombre']->getData();

$obj = new ValidarArchivo($file);

if ($form->isValid() && $obj->getCantErr() == 0) {

foreach ($obj->getUsuarios() as $usr) {

$this->getUsuarioDao()->guardar($usr);

}

$this->addFlash('success', "Archivo cargado con exito");

return $this->redirectToRoute('usuario.listar');

} else {

$this->addFlash('error', 'Hay errores en el archivo seleccionado, por favor revisar los datos!');

$this->addFlash('error', "Total de errores: " . $obj->getCantErr());

foreach ($obj->getErrores() as $error) {

$this->addFlash('error', $error);

}

return $this->render('usuario/form.html.twig', [

'form' => $form->createView(),

'titulo' => 'Crear Usuario',]);

}

}

**<?php**

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

use Symfony\Component\Validator\Constraints as Assert;

/\*\*

\* **@ORM\Entity**

\* **@ORM\Table**(name="usuarios")

\*/

class **Usuario** {

/\*\*

\* **@ORM\Id**

\* **@ORM\Column**(name="id", type="integer")

\* **@ORM\GeneratedValue**(strategy="AUTO")

\*/

private $id;

/\*\*

\* **@ORM\Column**(length=45)

\*/

private $nombre;

/\*\*

\* **@ORM\Column**(length=45)

\*/

private $apellido;

/\*\*

\* **@ORM\Column**(length=45)

\*/

private $email;

/\*\*

\* **@ORM\Column**(name="codigo", type="integer")

\*/

private $codigo;

public function **\_\_construct**($id = null, $nombre = null, $apellido = null, $email = null, $codigo = null) {

$this->id = $id;

$this->nombre = $nombre;

$this->apellido = $apellido;

$this->email = $email;

$this->codigo = $codigo;

}

function **getId**() {

return $this->id;

}

function **getNombre**() {

return $this->nombre;

}

function **getApellido**() {

return $this->apellido;

}

function **getEmail**() {

return $this->email;

}

function **getCodigo**() {

return $this->codigo;

}

function **setId**($id) {

$this->id = $id;

}

function **setNombre**($nombre) {

$this->nombre = $nombre;

}

function **setApellido**($apellido) {

$this->apellido = $apellido;

}

function **setEmail**($email) {

$this->email = $email;

}

function **setCodigo**($codigo) {

$this->codigo = $codigo;

}

}