Proyecto Final de Python

Comisión 18695

Francisco Martín Carranza

Creación de usuario

Se plantea la necesidad de no permitir que el usuario introduzca datos inválidos en la creación de usuarios para la administración de la web. Deben cumplirse los siguientes requerimientos:

- **Usuario:** Solo se permiten letras, dígitos y los símbolos @, . (punto), +, -, _.
- **E-mail:** Debe cumplir el formato de e-mail con un @.
- **Contraseña:** No puede similar al nombre del usuario o cualquier otra información personal, debe tener al menos 8 caracteres y no puede ser enteramente numérica. Por supuesto, el segundo campo donde se introduce la contraseña para confirmarla debe coincidir con el primero.

Además, todos los campos son obligatorios. Primero se presenta la función completa de **views.py**, y luego se detalla cada sección:

```
   def register_request(request):

    if request.method == 'POST':
        form = RegistroUsuarioForm(request.POST)

        if form.is_valid():
            form.save()
            return render (request, 'AppCoder/index.html', {'flag': True, 'mensaje': 'Te
registraste con éxito!'})

        else:
            return render (request,'AppCoder/register.html', {'form': form, 'mensaje': 'Por
favor, revisa los requisitos para crear el usuario.', 'error_user': True})

    form = RegistroUsuarioForm()
    return render(request, 'AppCoder/register.html', {'form': form, 'mensaje': '', 'error':
False})
```

Si el método es **POST**, se crea el formulario con los datos ingresados:

```
def register_request(request):

 if request.method == 'POST':
     form = RegistroUsuarioForm(request.POST)
```

Con ayuda de la función .**is_valid()** se verifica si el formulario es válido. En ese caso, este se guarda en la base de datos y devuelve el mensaje de "registro exitoso", con la bandera **flag** (que detallará más adelante) con valor **True**. Además, se devuelve al usuario al **Home** del sitio.

En caso contrario, el usuario no ha cumplido con los requisitos especificados. Entonces, se lo redirige a la misma página de registro, se devuelve el mismo **form** creado con el mensaje de que "deben revisarse los requisitos":

```
if form.is_valid():
            form.save()
            return render (request, 'AppCoder/index.html', {'flag': True, 'mensaje': 'Te
registraste con éxito!'})

        else:
            return render (request,'AppCoder/register.html', {'form': form, 'mensaje': 'Por
favor, revisa los requisitos para crear el usuario.', 'error_user': True})
```

Si el método no fuera **POST**, se crea el formulario vacío y bajo **render** se dirige al usuario a la página de registro inicial:

```
form = RegistroUsuarioForm()
    return render(request, 'AppCoder/register.html', {'form': form, 'mensaje':
'', 'error': False})
```

A continuación el formulario de registro, donde se customiza el **UserCreationForm**:

```
class RegistroUsuarioForm(UserCreationForm):
    username = forms.CharField(required=True, label= 'Usuario')
    email = forms.EmailField(required=True, label= 'E-mail')
    password1 = forms.CharField(required=True, label= 'Contraseña',
widget=forms.PasswordInput)
    password2 = forms.CharField(required=True, label= '', widget=forms.PasswordInput)

    error_messages = {
        'password_mismatch': 'Las contraseñas introducidas no coinciden.',
        }

    def __init__(self, *args, **kwargs):
        super(UserCreationForm, self).__init__(*args, **kwargs)
```

```
        self.fields['username'].help_text = 'Se permiten letras, dígitos y los símbolos
@/./+/-/_'
        self.fields['password1'].help_text = 'No se permite similitud con información
personal. Debe tener al menos 8 caracteres y no se permiten números únicamente'
        self.fields['password2'].help_text = 'Repite la contraseña.'

    class Meta:
         model = User
         fields = ['username', 'email', 'password1', 'password2']
```

Se han customizado los **help_text**, traduciendo los mensajes al español.

En cuanto a la URL **register.html** :

```
{% extends 'AppCoder/index.html' %}

{% block head_block %}
<title>Register</title>
{% endblock head_block %}

{% block body_block %}
    {% if mensaje %}
        <h3 style = color:red>{{mensaje}}</h3>
    {% endif %}

    <form action= '' method = "POST">
        {% csrf_token %}
        <table>
            {{form}}
        </table>
        <input type = "submit" value="Registrar"/>
    </form>

{% endblock body_block %}
```

Se ha realizado una herencia del **Home** index.html, y en caso de que hubiera un **mensaje** (que sería el caso de error en el registro), se presenta en color **rojo**.

Si el registro fuera exitoso, se colocó la bandera **flag** en **index.html**:

```
        {% if flag %}
            <h3 style = color:blueviolet>{{mensaje}}</h3>
        {% endif %}
        {% endblock body_block %}
```

Customización de los errores que aparecen en **bold** luego de un error de creación de usuario

Se utiliza el **UnicodeUsernameValidatorCustom()** de **django > contrib > auth > models.py (class AbstractUser)** para traducir el mensaje al español. Además, también se modifica en este el mensaje **unique**, que indica que no puede repetirse un usuario ("El usuario introducido ya existe"):

```
username_validator = UnicodeUsernameValidatorCustom()

    username = models.CharField(
        _('username'),
        max_length=150,
        unique=True,
        help_text=_('Required. 150 characters or fewer. Letters, digits and
@/./+/-/_ only.'),
        validators=[username_validator],
        error_messages={
            #'unique': _("A user with that username already exists."), Se
customiza para traducir el mensaje
            'unique': _("El usuario introducido ya existe.")
        },
    )
    first_
```

```
@deconstructible
class UnicodeUsernameValidatorCustom(validators.RegexValidator): #Se crea esta
clase para customizar el mensaje en inglés
    regex = r'^[\w.@+-]+\Z'
    message = _(
        'Introduzca un usuario válido. Sólo se permiten letras, dígitos y los
símbolos @/./+/-/_'
    )
    flags = 0
```

Luego se modificaron los mensajes de las clases dentro de **django > contrib > auth > password_validation.py**:

```
def validate(self, password, user=None):
        if len(password) < self.min_length:
            raise ValidationError(
                ngettext(
                    # "This password is too short. It must contain at least
%(min_length)d character.",
                    # "This password is too short. It must contain at least
%(min_length)d characters.", Se customiza para traducir el mensaje
```

```
                    "La contraseña introducida es muy corta. Debe contener al
menos %(min_length)d caracteres.",
                    "La contraseña introducida es muy corta. Debe contener al
menos %(min_length)d caracteres.",
                    self.min_length
                ),
                code='password_too_short',
                params={'min_length': self.min_length},
            )
```

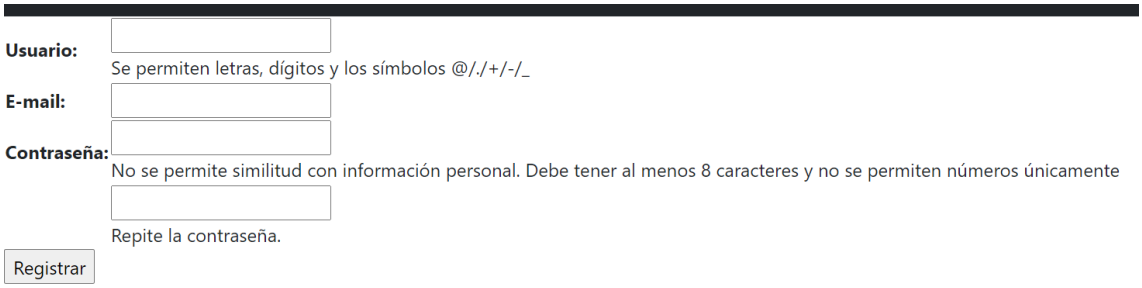
```
if SequenceMatcher(a=password.lower(), b=value_part.lower()).quick_ratio() >=
self.max_similarity:
                    try:
                        verbose_name =
str(user._meta.get_field(attribute_name).verbose_name)
                    except FieldDoesNotExist:
                        verbose_name = attribute_name
                    raise ValidationError(
                        # _("The password is too similar to the
%(verbose_name)s."), Se customiza para traducir el mensaje
                        _("La contraseña introducida es muy similar al nombre del
usuario."),
                        code='password_too_similar',
                        params={'verbose_name': verbose_name},
                    )
```

```
def validate(self, password, user=None):
        if password.lower().strip() in self.passwords:
            raise ValidationError(
                # _("This password is too common."), Se customiza para traducir
el mensaje
                _("La contraseña introducida es muy común."),
                code='password_too_common',
            )
```

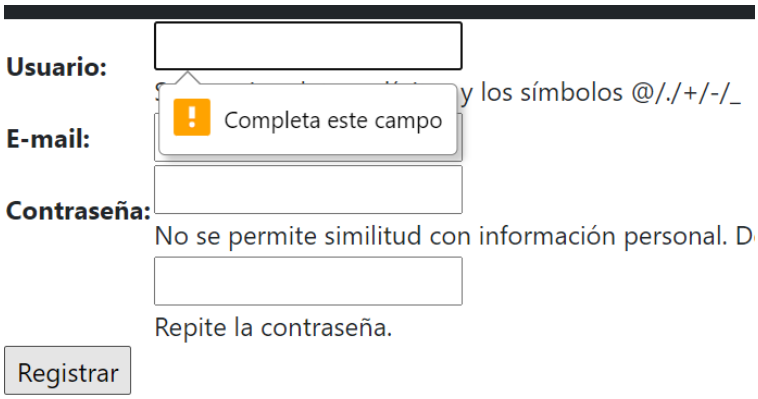
```
def validate(self, password, user=None):
        if password.isdigit():
            raise ValidationError(
                # _("This password is entirely numeric."), Se customiza para
traducir el mensaje
                _("La contraseña introducida es enteramente numérica."),
                code='password_entirely_numeric',
            )
```

Ejemplos:

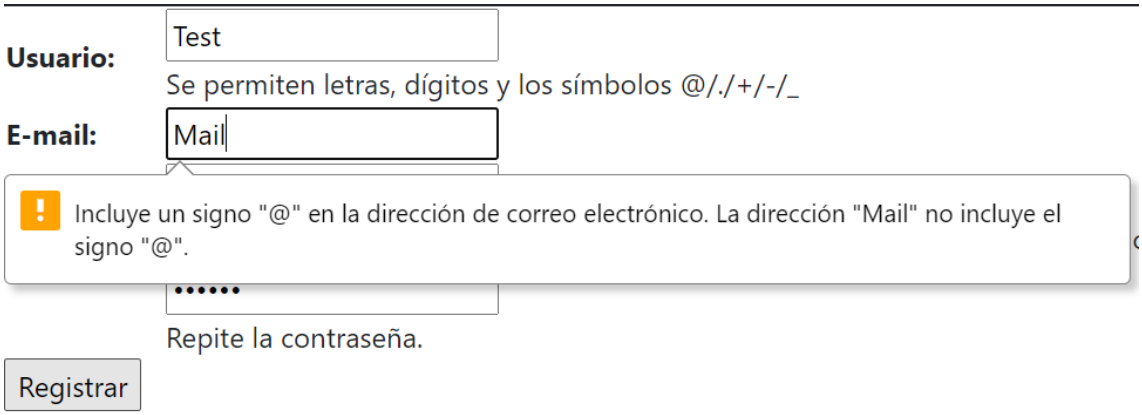
1. Ingreso por primera vez a la URL:



2. Campo **usuario** obligatorio:



3. Requisito de **e-mail** con el formato correcto, con @ y dominio:



4. Ejemplo de **usuario** no válido:

Por favor, revisa los requisitos para crear el usuario.

Introduzca un usuario válido. Sólo se permiten letras, dígitos y los símbolos @/./+/-/_

Usuario: Test!

Se permiten letras, dígitos y los símbolos @/./+/-/_

E-mail: Mail@mail.com

5. Ejemplo de contraseñas que no coinciden:

Por favor, revisa los requisitos para crear el usuario.

Usuario: Test

Se permiten letras, dígitos y los símbolos @/./+/-/_

E-mail: Mail@mail.com

Contraseña:

No se permite similitud con información personal. Debe tener al menos 8 caracteres y no se permiten números únicamente

Las contraseñas introducidas no coinciden.

Repite la contraseña.

Registrar

6. Ejemplo de contraseña parecida al usuario (coinciden 5 caracteres), no cumple la longitud mínima, muy común (bajo comparación de campos precargados por django) y enteramente numérica:

Por favor, revisa los requisitos para crear el usuario.

Usuario: 123asd

Se permiten letras, dígitos y los símbolos @/./+/-/_

E-mail: Mail@mail.com

Contraseña:

No se permite similitud con información personal. Debe tener al menos 8 caracteres y no se permiten números únicamente

La contraseña introducida es muy corta. Debe contener al menos 8 caracteres.

La contraseña introducida es muy común.

La contraseña introducida es enteramente numérica.

Repite la contraseña.

Registrar

7. El usuario ya existe:

Por favor, revisa los requisitos para crear el usuario.

El usuario introducido ya existe.

Usuario: testing

Se permiten letras, dígitos y los símbolos @/./+/-/_

E-mail: Mail@mail.com

Contraseña:

No se permite similitud con información personal. Debe tener al menos 8 caracteres y no se permiten números únicamente

Repite la contraseña.

Registrar

8. Registro exitoso: