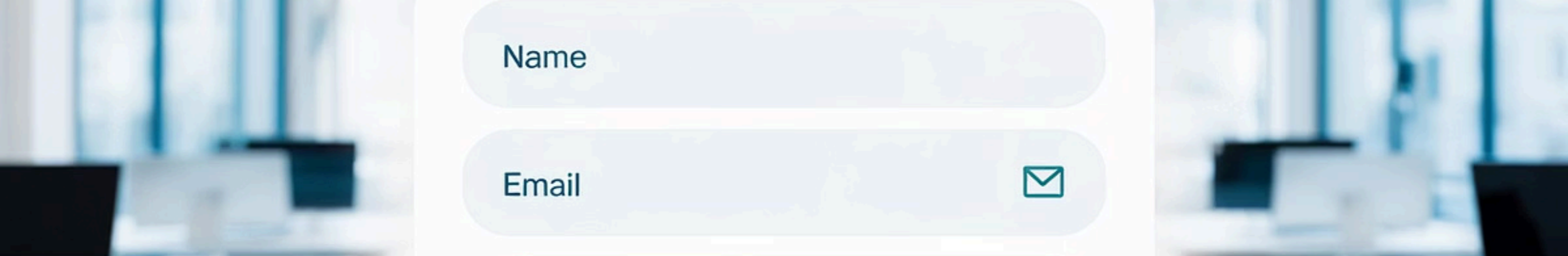



AE4 – Formularios en Django

Objetivo de la clase: Implementar formularios en un aplicativo web utilizando Django para capturar y procesar información.



Name

Email 

Introducción a los formularios

Un formulario es la puerta de entrada de datos en una aplicación web.

Django nos facilita la vida con herramientas listas para usar: validación, seguridad y persistencia en la base de datos.

Ejemplo: registro de usuario, contacto, comentarios, compras.

Actividad: ¿Qué formulario web usaron la última vez? (chat en clase).

```

n <coe="ahocurtclei""boaiitcoie[]]>
i   yucnudoat""[]>
z   yuauaiommtne^"lpgiisiitente[]]
</>
<[]eilunitelt"=clon="[]>
1  <["uarlq'go=">
z   <_fog-"destároenll[][]>
7   itug= "etari:">

```

```

001 yvcounfbct:[]>
002 ytenantteet""lppillaittu ittu[]>
003
004 <">
005 <[]ang"oue=">
006 pp""aiceuuttatts""=acor[]>
007 p'lwyificianvianalol-"liileustet[]>
008 alioelile"si=-is""padiiiter[]>
009 pjm iisiumtuievl-"lunic:zeste[]>
010 <[]>

```

Formularios HTML vs Django Forms

Formularios HTML

necesitas escribir `<form>`, `<input>`, `<button>`,
manejar validaciones manualmente.

Ejemplo práctico:

```

# Django
correo = forms.EmailField()

```

Django Forms

abstraen el proceso → defino los campos en Python y
Django genera HTML + validación.



Estructura básica de un formulario Django

01

Definir la clase en forms.py

02

Conectarlo con una vista (views.py)

03

Mostrarlo en una plantilla (.html)

04

Procesar datos (GET/POST)

Nuestra primera clase Form

forms.py

```
from django import forms
```

```
class ContactoForm(forms.Form):
```

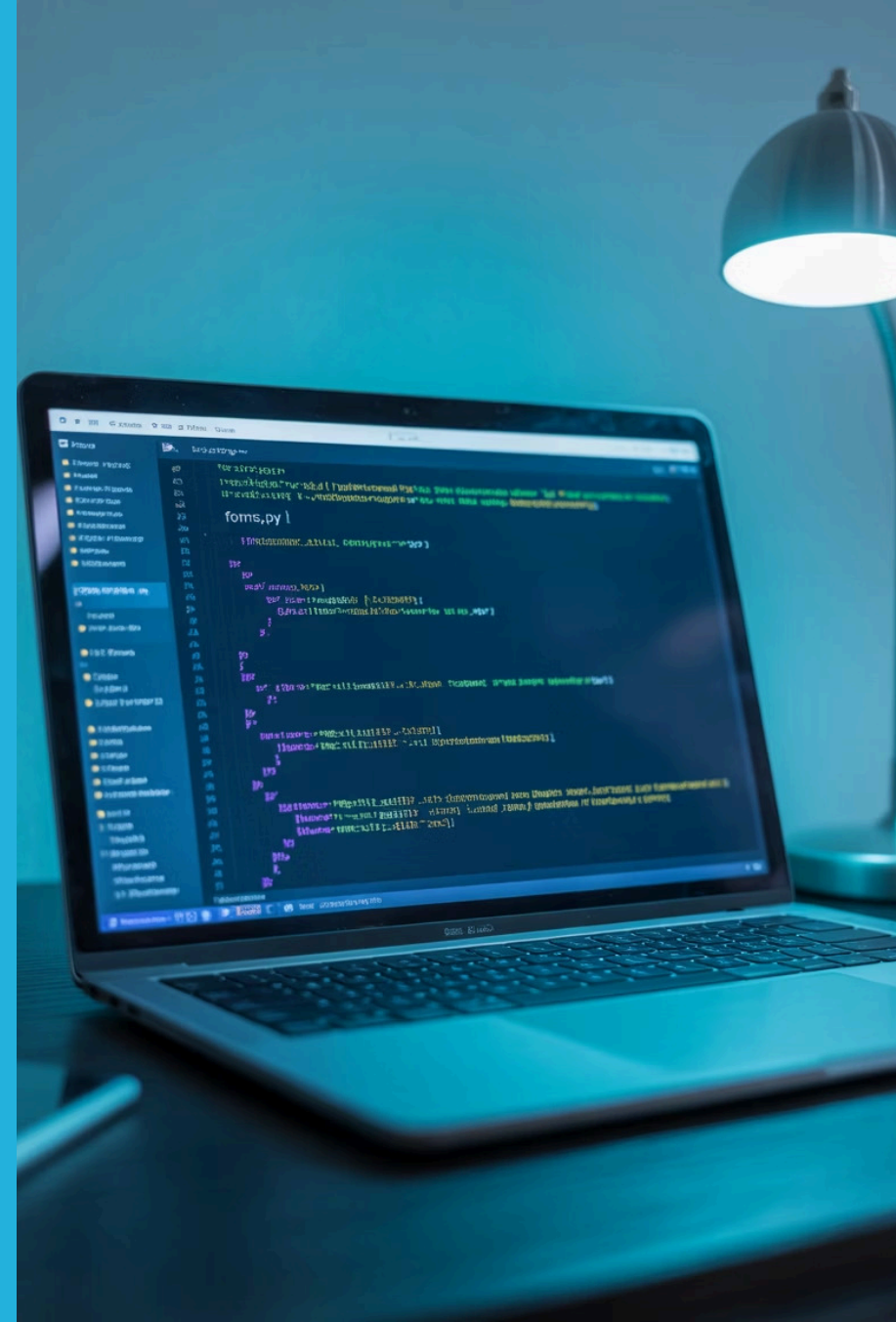
```
    nombre = forms.CharField(max_length=100)
```

```
    correo = forms.EmailField()
```

```
    mensaje = forms.CharField(widget=forms.Textarea)
```

Simple, en Python puro.

Campos: CharField, EmailField, Textarea.



Procesando el formulario en la vista

```
# views.py
def contacto(request):
    if request.method == 'POST':
        form = ContactoForm(request.POST)
        if form.is_valid():
            nombre = form.cleaned_data['nombre']
            # usar datos...
        else:
            form = ContactoForm()
    return render(request, 'contacto.html', {'form': form})
```

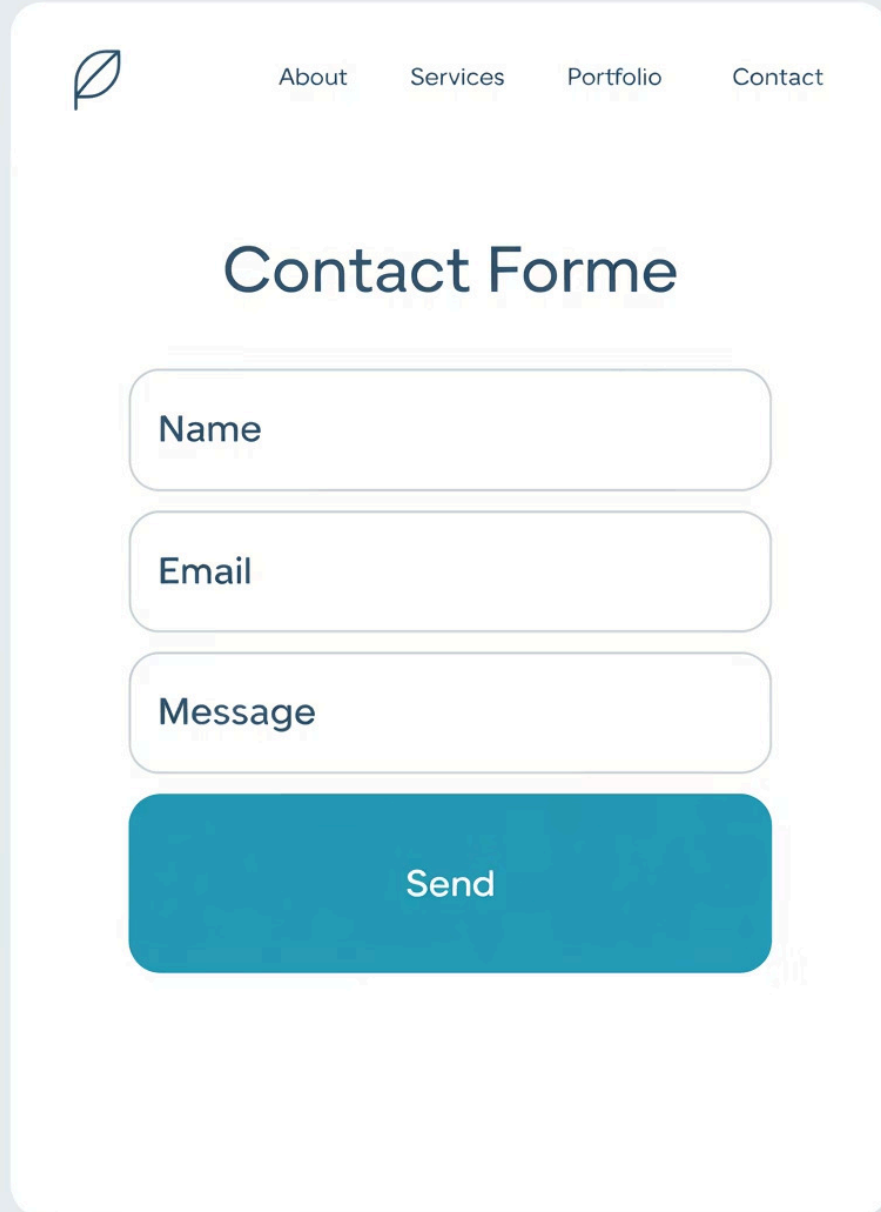
- `request.POST` captura datos.
- `.is_valid()` valida automáticamente.
- `.cleaned_data` entrega datos listos para usar.



El template de despliegue

```
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Enviar</button>
</form>
```

- `{{ form.as_p }}` → Django renderiza los campos con `<p>`.
- CSRF token = seguridad contra ataques.



Logo

[About](#) [Services](#) [Portfolio](#) [Contact](#)

Contact Forme

Name

Email

Message

Send

Get in Touch

We'd love to hear from you

Renderizado manual de campos

```
<label>Nombre:</label>
{{ form.nombre }}
```

Control total sobre estilos y etiquetas.

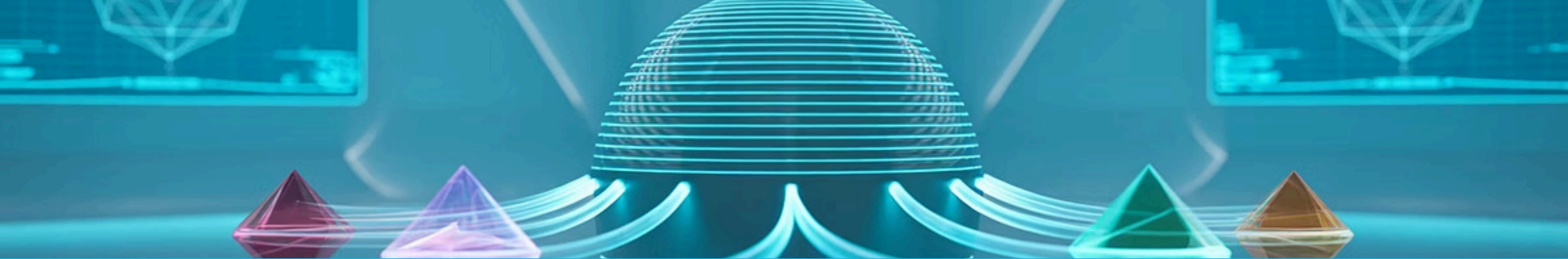
Permite personalizar Bootstrap o Tailwind.



Ejercicio práctico 1

Reto en vivo:

- Crear un formulario de registro de usuario con nombre, correo y contraseña.
- Renderizarlo con `{{ form.as_p }}`.
- Probar el envío en el navegador.



Plantillas reutilizables

Crear form_base.html con la estructura del formulario.

Extenderla en otras plantillas con `{% block %}`.

Ahorra tiempo y mantiene código limpio.

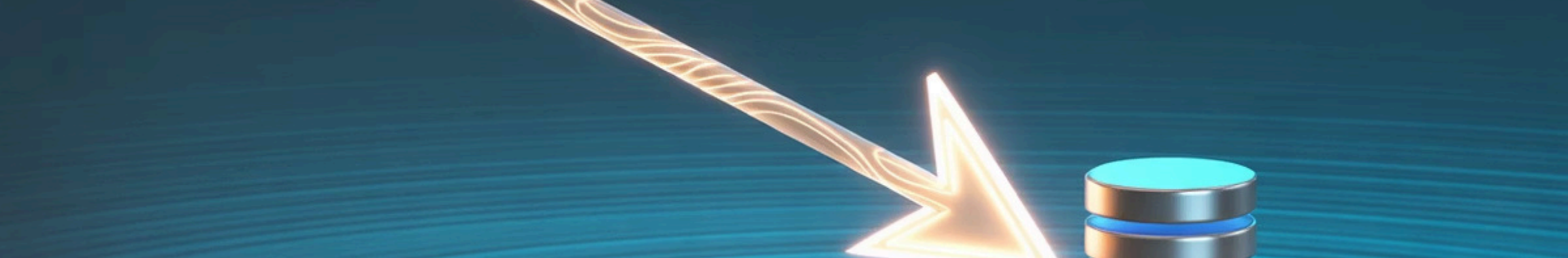


ModelForms

Enlazan directamente un modelo de Django con un formulario.

Reducen líneas de código.

```
class ProductoForm(forms.ModelForm):  
    class Meta:  
        model = Producto  
        fields = ['nombre', 'precio', 'categoria']
```



Procesamiento con ModelForms

```
def nuevo_producto(request):  
    if request.method == 'POST':  
        form = ProductoForm(request.POST)  
        if form.is_valid():  
            form.save() # Inserta en DB
```

.save() guarda el objeto directamente.

Password,

Login

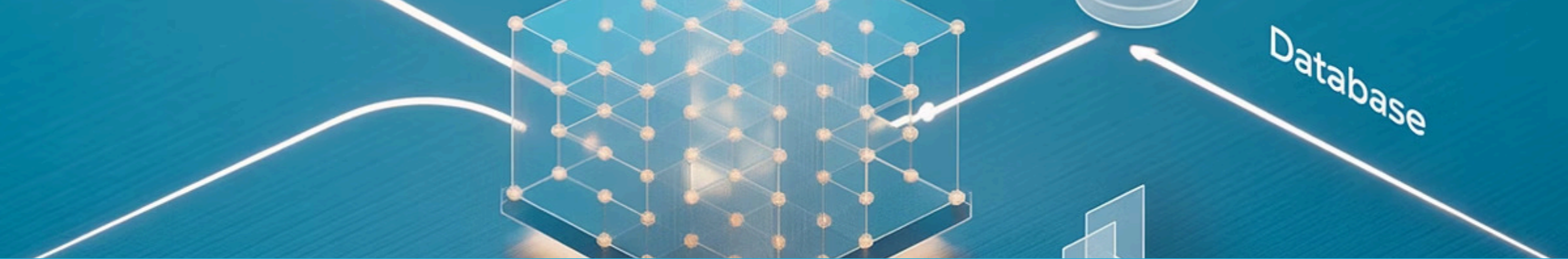
Validación automática y errores

Django valida tipos de datos, longitud, formato de email.

Muestra mensajes de error junto a cada campo.

Ejemplo en HTML:

```
{% if form.errors %}  
  <div class="errors">  
    {{ form.errors }}  
  </div>  
{% endif %}
```



Ejercicio práctico 2

Reto:

- Crear un modelo Cliente con campos nombre, correo y fecha_registro.
- Hacer un ModelForm.
- Insertar datos en la base de datos desde el formulario.

InlineFormsets

Sirven para manejar relaciones con claves foráneas.

Ejemplo: Producto pertenece a Categoría.

Con un solo formulario puedes crear varios productos relacionados.





Binding de datos

Django hace "binding" → asocia automáticamente datos enviados con campos.

Evita reescribir código.

Ejemplo:

```
form = ContactoForm(request.POST or None)
```



Seguridad en formularios

- CSRF token protege de ataques.
- Validación protege contra inyecciones.
- Nunca confíes en datos del usuario → Django ya piensa en eso.



Actividad de grupo

Mini-proyecto en parejas:

- Crear una página "Contacto" con un formulario (nombre, email, mensaje).
- Validar los datos.
- Guardar en base de datos con un ModelForm.

Resumen de la clase

Formularios = entrada de datos en la web

Django simplifica creación, validación y guardado

forms.Form y ModelForm

Templates reutilizables

Manejo de errores y seguridad integrada



Cierre y tarea

Tarea: Crear un formulario de registro de productos con ModelForm.

Extra: personalizar estilos con Bootstrap.

Autenticación y login con formularios.