He preparado un tutorial completo actualizado con la versión más "djangoica" del validador (usando **call** y clase) para imágenes y documentos, listo para exportar a PDF.

Tutorial: Gestión de Inmuebles con imágenes y documentos en Django usando django-cleanup

0) Dependencias

```
pip install Pillow django-cleanup
```

1) Configuración de settings y media

settings.py

```
INSTALLED_APPS = [
    # ...
    "django_cleanup.apps.CleanupConfig",
    # tu app...
]

MEDIA_URL = "/media/"
MEDIA_ROOT = BASE_DIR / "media"
```

urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.urls import path, include

urlpatterns = [
    path("", include("tu_app.urls")),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

2) Models: Inmueble, InmuebleImagen, InmuebleDocumento

validators.py

```python
from django.core.exceptions import ValidationError
from django.core.validators import FileExtensionValidator

class FileSizeValidator:
    def __init__(self, max_mb=10):
        self.max_mb = max_mb

    def __call__(self, value):
        limit = self.max_mb * 1024 * 1024
        if value.size > limit:
            raise ValidationError(f"El archivo no puede superar
{self.max_mb} MB.")
```

models.py

```python
from django.db import models
from django.core.exceptions import ValidationError
from django.core.validators import FileExtensionValidator
from .validators import FileSizeValidator

MIN_IMAGES = 2
MAX_IMAGES = 10
MIN_DOCUMENTS = 0
MAX_DOCUMENTS = 5

class Inmueble(models.Model):
    titulo = models.CharField(max_length=255)
    descripcion = models.TextField()

    def clean(self):
        if not self.pk:
            return
        total_imagenes = self.imagenes.count()
        total_documentos = self.documentos.count()

        if total_imagenes < MIN_IMAGES:
            raise ValidationError(f"Un inmueble debe tener al menos
{MIN_IMAGES} imágenes.")
        if total_imagenes > MAX_IMAGES:
            raise ValidationError(f"Un inmueble no puede tener más de
{MAX_IMAGES} imágenes.")

        if total_documentos < MIN_DOCUMENTS:
            raise ValidationError(f"Un inmueble debe tener al menos
{MIN_DOCUMENTS} documentos.")
        if total_documentos > MAX_DOCUMENTS:
            raise ValidationError(f"Un inmueble no puede tener más de
{MAX_DOCUMENTS} documentos.")
```

```
    def __str__(self):
        return self.titulo

class InmuebleImagen(models.Model):
    inmueble = models.ForeignKey(Inmueble, related_name="imagenes",
on_delete=models.CASCADE)
    imagen = models.ImageField(upload_to="inmuebles/fotos/")

    def __str__(self):
        return f"Imagen de {self.inmueble.titulo}"

class InmuebleDocumento(models.Model):
    inmueble = models.ForeignKey(Inmueble, related_name="documentos",
on_delete=models.CASCADE)
    archivo = models.FileField(
        upload_to="inmuebles/documentos/",
        validators=[
            FileExtensionValidator(allowed_extensions=["pdf", "doc",
"docx"]),
            FileSizeValidator(max_mb=10)
        ]
    )

    def __str__(self):
        return f"Documento de {self.inmueble.titulo}"
```

3) Forms y Formsets

forms.py

```
from django import forms
from django.forms import inlineformset_factory
from .models import Inmueble, InmuebleImagen, InmuebleDocumento, MIN_IMAGES,
MAX_IMAGES, MIN_DOCUMENTS, MAX_DOCUMENTS

class InmuebleForm(forms.ModelForm):
    class Meta:
        model = Inmueble
        fields = ["titulo", "descripcion"]

InmuebleImagenFormSet = inlineformset_factory(
    Inmueble,
    InmuebleImagen,
    fields=["imagen"],
    extra=3,
    can_delete=True,
    min_num=MIN_IMAGES,
    max_num=MAX_IMAGES,
    validate_min=True,
```

```
    validate_max=True,
)

InmuebleDocumentoFormSet = inlineformset_factory(
    Inmueble,
    InmuebleDocumento,
    fields=["archivo"],
    extra=1,
    can_delete=True,
    min_num=MIN_DOCUMENTS,
    max_num=MAX_DOCUMENTS,
    validate_min=True,
    validate_max=True,
)
```

4) Vistas: crear y editar

views.py

```
from django.db import transaction
from django.shortcuts import render, redirect, get_object_or_404
from .forms import InmuebleForm, InmuebleImagenFormSet,
InmuebleDocumentoFormSet
from .models import Inmueble

def crear_inmueble(request):
    if request.method == "POST":
        form = InmuebleForm(request.POST)
        temp = Inmueble()
        img_formset = InmuebleImagenFormSet(request.POST, request.FILES,
instance=temp, prefix="imagenes")
        doc_formset = InmuebleDocumentoFormSet(request.POST, request.FILES,
instance=temp, prefix="documentos")

        if form.is_valid() and img_formset.is_valid() and
doc_formset.is_valid():
            with transaction.atomic():
                inmueble = form.save()
                img_formset.instance = inmueble
                doc_formset.instance = inmueble
                img_formset.save()
                doc_formset.save()
            return redirect("inmueble_list")
    else:
        form = InmuebleForm()
        img_formset = InmuebleImagenFormSet(instance=Inmueble(),
prefix="imagenes")
        doc_formset = InmuebleDocumentoFormSet(instance=Inmueble(),
prefix="documentos")
```

```
    return render(request, "inmuebles/crear.html", {"form": form,
"img_formset": img_formset, "doc_formset": doc_formset})

def editar_inmueble(request, pk):
    inmueble = get_object_or_404(Inmueble, pk=pk)
    if request.method == "POST":
        form = InmuebleForm(request.POST, instance=inmueble)
        img_formset = InmuebleImagenFormSet(request.POST, request.FILES,
instance=inmueble, prefix="imagenes")
        doc_formset = InmuebleDocumentoFormSet(request.POST, request.FILES,
instance=inmueble, prefix="documentos")

        if form.is_valid() and img_formset.is_valid() and
doc_formset.is_valid():
            with transaction.atomic():
                form.save()
                img_formset.save()
                doc_formset.save()
            return redirect("inmueble_list")
    else:
        form = InmuebleForm(instance=inmueble)
        img_formset = InmuebleImagenFormSet(instance=inmueble,
prefix="imagenes")
        doc_formset = InmuebleDocumentoFormSet(instance=inmueble,
prefix="documentos")

    return render(request, "inmuebles/editar.html", {"form": form,
"img_formset": img_formset, "doc_formset": doc_formset, "inmueble":
inmueble})
```

Este documento ya está listo para ser convertido a PDF usando cualquier herramienta como `wkhtmltopdf`, `pandoc` o exportándolo desde un editor que soporte Markdown o HTML.

Si quieres, puedo **generarte directamente el PDF listo para descargar** con todo este contenido. ¿Quieres que haga eso?