MEDIA kartotek

Perfekt timing. Vi bygger “mediadatabase” oppå `docs`-appen du allerede har — med opplasting, nedlasting og mappe-baserte rettigheter (grupper). Vi gjør det enkelt nå, men robust nok til å vokse (f.eks. RBAC på fylke/lokallag senere).

---

1) Modeller – legg på mappe-ACL (les/skriv)

1) Modeller (mappe-ACL)
Vi utvider `DocFolder` med to mange-til-mange-felter mot `Group`:

- `can_read` = grupper som kan **se/nedlaste** filer i mappen
- `can_write` = grupper som kan **laste opp** til mappen
  Du kjører `makemigrations` + `migrate` for å få feltene i DB.

`docs/models.py` – utvid `DocFolder` med gruppe-rettigheter:

```
from django.db import models
from django.contrib.auth import get_user_model
from django.contrib.auth.models import Group

User = get_user_model()

class DocFolder(models.Model):
    navn = models.CharField(max_length=200)
    beskrivelse = models.TextField(blank=True)
    opprettet_av = models.ForeignKey(User, on_delete=models.SET_NULL, null=True,
blank=True)
    opprettet = models.DateTimeField(auto_now_add=True)

    # NYTT: ACL per mappe
    can_read = models.ManyToManyField(Group, blank=True,
related_name="docfolders_can_read", verbose_name="Kan lese (grupper)")
    can_write = models.ManyToManyField(Group, blank=True,
related_name="docfolders_can_write", verbose_name="Kan laste opp (grupper)")

    class Meta:
        verbose_name = "Mappe"
        verbose_name_plural = "Mapper"
        ordering = ["navn"]

    def __str__(self):
        return self.navn

class Document(models.Model):
    folder = models.ForeignKey(DocFolder, on_delete=models.CASCADE,
related_name="docs")
    tittel = models.CharField(max_length=255)
```

```
    fil = models.FileField(upload_to="docs/%Y/%m/")
    merknad = models.TextField(blank=True)
    opprettet_av = models.ForeignKey(User, on_delete=models.SET_NULL, null=True,
blank=True)
    opprettet = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name = "Dokument"
        verbose_name_plural = "Dokumenter"
        ordering = ["-opprettet"]

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

Kjør:

```
python manage.py makemigrations docs
python manage.py migrate
```

---

2) Admin – lettere å styre rettigheter

`docs/admin.py`

`2) Admin (styring av rettigheter)`

`I Django-admin kan du:`

- Opprette **Groups** (“Sentralt”, “Fylkesstyre”, …)
- På hver **DocFolder** velger du hvilke grupper som kan lese/skrive (enkelt via to felt). Dette er den praktiske måten å sette tilgang uten kode.

```
from django.contrib import admin
from .models import DocFolder, Document

@admin.register(DocFolder)
class DocFolderAdmin(admin.ModelAdmin):
    list_display = ("navn", "opprettet_av", "opprettet")
    search_fields = ("navn", "beskrivelse")
    filter_horizontal = ("can_read", "can_write")  # NYTT

@admin.register(Document)
class DocumentAdmin(admin.ModelAdmin):
    list_display = ("tittel", "folder", "opprettet_av", "opprettet")
    search_fields = ("tittel", "merknad")
    list_filter = ("folder",)
```

I **Admin → Auth → Groups** kan du lage grupper som “Sentralt”, “Fylkesstyre”, “Lokallag …”, “Økonomi” osv., og så gi mapper les/skriv til disse.

---

3) Views – filtrering + sikker nedlasting

`docs/views.py` – legg til hjelpers + oppdatér views, inkl. **beskyttet nedlasting**:

3) Views (sjekker tilgang + trygg nedlasting)

- Lister viser **kun mapper/dokumenter** brukeren har lov å lese.
- Upload-skjemaet viser **bare mapper** brukeren kan skrive til.
- Nedlasting går via en egen view som **autentiserer/autoriserer** før filen leveres.

```
from django.contrib.auth.decorators import login_required, user_passes_test
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib import messages
from django.http import HttpResponse, FileResponse, HttpResponseForbidden
from django.utils.encoding import iri_to_uri
import mimetypes
from .models import DocFolder, Document
from .forms import DocumentForm

def staff_required(u): return u.is_active and u.is_staff

def user_can_read_folder(user, folder: DocFolder) -> bool:
    if user.is_superuser: return True
    if folder.can_read.filter(id__in=user.groups.values_list("id",
flat=True)).exists():
        return True
    # fallback: eier av mappen
    return folder.opprettet_av_id == user.id

def user_can_write_folder(user, folder: DocFolder) -> bool:
    if user.is_superuser: return True
    if folder.can_write.filter(id__in=user.groups.values_list("id",
flat=True)).exists():
        return True
    return folder.opprettet_av_id == user.id

@login_required
@user_passes_test(staff_required)
def doc_home(request):
    # vis bare mapper brukeren kan lese
    folders = [f for f in DocFolder.objects.all().order_by("navn") if
user_can_read_folder(request.user, f)]
    q = request.GET.get("q", "").strip()
    docs = Document.objects.select_related("folder").all()
    # filtrer dokumenter til bare de i mapper bruker kan lese
    docs = [d for d in docs if user_can_read_folder(request.user, d.folder)]
    if q:
        docs = [d for d in docs if q.lower() in d.tittel.lower()]
    return render(request, "docs/home.html", {"folders": folders, "docs": docs,
"q": q})

@login_required
@user_passes_test(staff_required)
def doc_folder(request, folder_id: int):
```

```
    folder = get_object_or_404(DocFolder, id=folder_id)
    if not user_can_read_folder(request.user, folder):
        return HttpResponseForbidden("Ingen tilgang til denne mappen.")
    return render(request, "docs/folder.html", {"folder": folder, "docs":
folder.docs.all()})

@login_required
@user_passes_test(staff_required)
def doc_upload(request):
    if request.method == "POST":
        form = DocumentForm(request.POST, request.FILES)
        if form.is_valid():
            doc = form.save(commit=False)
            if not user_can_write_folder(request.user, doc.folder):
                return HttpResponseForbidden("Ingen opplastingsrett i valgt
mappe.")
            doc.opprettet_av = request.user
            doc.save()
            messages.success(request, "Dokument lastet opp.")
            return redirect("docs:home")
    else:
        form = DocumentForm()
    # Begrens valg i skjema til mapper bruker kan skrive til
    form.fields["folder"].queryset = DocFolder.objects.all()
    form.fields["folder"].queryset = form.fields["folder"].queryset.filter(
        id__in=[f.id for f in DocFolder.objects.all() if
user_can_write_folder(request.user, f)]
    )
    return render(request, "docs/upload.html", {"form": form})

@login_required
@user_passes_test(staff_required)
def doc_download(request, doc_id: int):
    doc = get_object_or_404(Document, id=doc_id)
    if not user_can_read_folder(request.user, doc.folder):
        return HttpResponseForbidden("Ingen tilgang.")

    # Bruk Nginx X-Accel-Redirect i prod for rask/trygg levering
    # NB: Nginx må ha 'location ^~ /konservativt/protected/ { internal; alias
<MEDIA_ROOT>/; }'
    protected_url = "/konservativt/protected/" + iri_to_uri(doc.fil.name)
    content_type, _ = mimetypes.guess_type(doc.fil.name)
    resp = HttpResponse()
    resp["Content-Type"] = content_type or "application/octet-stream"
    resp["X-Accel-Redirect"] = protected_url
    # last ned med opprinnelig filnavn
    resp["Content-Disposition"] = f'inline; filename="{doc.fil.name.rsplit("/",1)
[-1]}"'
    return resp
```

`docs/urls.py` – legg til nedlastingsrute:

```
from django.urls import path
from . import views

app_name = "docs"
urlpatterns = [
    path("", views.doc_home, name="home"),
    path("folder/<int:folder_id>/", views.doc_folder, name="folder"),
```

```
    path("upload/", views.doc_upload, name="upload"),
    path("d/<int:doc_id>/", views.doc_download, name="download"),  # NY
]
```

Templates – bytt nedlastingslenker til den nye, sikre ruten:

`templates/docs/home.html` og `templates/docs/folder.html`:

```
<a href="{% url 'docs:download' d.id %}">Last ned</a>

home.html ser slik ut nå: {% extends "base.html" %} {% block content %}
<h1>Dokumenter – home</h1>{% endblock %} Hvordan skal den se ut?
Her er en enkel, fungerende templates/docs/home.html som passer til viewene vi
satte opp (mapper + søk + siste dokumenter + nedlasting via docs:download):


{% extends "base.html" %}
{% block title %}Dokumenter{% endblock %}

{% block content %}
<h1>Dokumenter</h1>

<form method="get" style="margin-bottom:1rem;">
  <input type="text" name="q" value="{{ q }}" placeholder="Søk i titler…" />
  <button type="submit">Søk</button>
  <a href="{% url 'docs:upload' %}" style="margin-left:1rem;">Last opp</a>
</form>

<h2>Mapper</h2>
<ul>
  {% for f in folders %}
    <li><a href="{% url 'docs:folder' f.id %}">{{ f.navn }}</a>{% if f.beskrivelse
%} – {{ f.beskrivelse }}{% endif %}</li>
  {% empty %}
    <li>Ingen mapper tilgjengelig. Opprett i admin og/eller tildel
rettigheter.</li>
  {% endfor %}
</ul>

<h2 style="margin-top:1.5rem;">Siste dokumenter</h2>
{% if docs %}
  <table style="width:100%; border-collapse:collapse;">
    <thead>
      <tr>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Tittel</th>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Mappe</th>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Dato</th>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Handling</th>
      </tr>
    </thead>
    <tbody>
      {% for d in docs %}
        <tr>
          <td style="padding:.4rem;">{{ d.tittel }}</td>
          <td style="padding:.4rem;"><a href="{% url 'docs:folder' d.folder.id
%}">{{ d.folder.navn }}</a></td>
```

```
          <td style="padding:.4rem;">{{ d.opprettet|date:"Y-m-d H:i" }}</td>
          <td style="padding:.4rem;"><a href="{% url 'docs:download' d.id %}">Last
ned</a></td>
        </tr>
      {% endfor %}
    </tbody>
  </table>
{% else %}
  <p>Ingen dokumenter funnet.</p>
{% endif %}
{% endblock %}
```

Kort fortalt:

- Bruker `folders`, `docs` og `q` som viewet sender inn.
- Last ned-lenken går via `docs:download` (ACL-sjekk + X-Accel-Redirect i Nginx).
- "Last opp" peker til `docs:upload` (skjema filtrert til mapper du kan skrive til).

Rop ut om du vil at jeg skal gi tilsvarende oppfrisket `folder.html` og `upload.html` også.

Jepp! Her er en enkel og riktig **`templates/docs/folder.html`** som passer til oppsettet du har nå:

```
{% extends "base.html" %}
{% block title %}{{ folder.navn }}{% endblock %}

{% block content %}
<p><a href="{% url 'docs:home' %}">← Tilbake til dokumenter</a></p>

<h1>{{ folder.navn }}</h1>
{% if folder.beskrivelse %}<p>{{ folder.beskrivelse }}</p>{% endif %}

<p style="margin: .75rem 0;">
  <a href="{% url 'docs:upload' %}">Last opp dokument</a>
</p>

{% if docs %}
  <table style="width:100%; border-collapse:collapse;">
    <thead>
      <tr>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Tittel</th>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Dato</th>
        <th style="text-align:left; border-bottom:1px solid #ddd;
padding:.4rem;">Handling</th>
      </tr>
    </thead>
    <tbody>
      {% for d in docs %}
        <tr>
          <td style="padding:.4rem;">{{ d.tittel }}</td>
          <td style="padding:.4rem;">{{ d.opprettet|date:"Y-m-d H:i" }}</td>
          <td style="padding:.4rem;"><a href="{% url 'docs:download' d.id %}">Last
ned</a></td>
        </tr>
```

```
      {% endfor %}
    </tbody>
  </table>
{% else %}
  <p>Ingen dokumenter i denne mappen ennå.</p>
{% endif %}
{% endblock %}
```

Tips: Upload-lenken er alltid synlig her, men selve `doc_upload`-viewet håndhever skrivetilgang, så det er trygt.

Vil du at knappen kun vises for de som faktisk kan skrive i mappen, legg til én linje i viewet og en liten sjekk:

`docs/views.py` (i `doc_folder`)

```
return render(request, "docs/folder.html", {
    "folder": folder,
    "docs": folder.docs.all(),
    "can_write": user_can_write_folder(request.user, folder),
})
```

`templates/docs/folder.html`

```
{% if can_write %}
  <p style="margin: .75rem 0;">
    <a href="{% url 'docs:upload' %}">Last opp dokument</a>
  </p>
{% endif %}
```

Rop ut om du vil ha en tilsvarende oppgradert `upload.html` også.

```
Ja, ta upload også.
```

Her er en enkel og riktig **`templates/docs/upload.html`** som passer til oppsettet ditt (skjema med CSRF, multipart, feilmeldinger, og “Tilbake”-lenke). Den bruker feltene fra DocumentForm:

```
{% extends "base.html" %}
{% block title %}Last opp dokument{% endblock %}

{% block content %}
<p><a href="{% url 'docs:home' %}">← Tilbake til dokumenter</a></p>

<h1>Last opp dokument</h1>

<form method="post" enctype="multipart/form-data" style="max-width:720px;">
  {% csrf_token %}

  {% if form.non_field_errors %}
    <div style="color:#b00020; margin:.5rem 0;">
      {{ form.non_field_errors }}
    </div>
  {% endif %}

  <p>
```

```
    <label for="{{ form.folder.id_for_label }}"><strong>Mappe</strong></label><br>
    {{ form.folder }}
    <small>Kun mapper du har skrivetilgang til vises.</small>
    {{ form.folder.errors }}
  </p>

  <p>
    <label for="{{ form.tittel.id_for_label }}"><strong>Tittel</strong></label><br>
    {{ form.tittel }}
    {{ form.tittel.errors }}
  </p>

  <p>
    <label for="{{ form.fil.id_for_label }}"><strong>Fil</strong></label><br>
    {{ form.fil }}
    <small>Maks ~50 MB (jfr. Nginx <code>client_max_body_size</code>).</small>
    {{ form.fil.errors }}
  </p>

  <p>
    <label for="{{ form.merknad.id_for_label
}}"><strong>Merknad</strong></label><br>
    {{ form.merknad }}
    {{ form.merknad.errors }}
  </p>

  <button type="submit">Last opp</button>
</form>
{% endblock %}
```

Valgfritt (for litt penere filfelt): legg til widget-attributter i **`docs/forms.py`**:

```
from django import forms
from .models import Document

class DocumentForm(forms.ModelForm):
    class Meta:
        model = Document
        fields = ["folder", "tittel", "fil", "merknad"]
        widgets = {
            "tittel": forms.TextInput(attrs={"placeholder": "Kort tittel"}),
            "fil": forms.ClearableFileInput(attrs={"accept": "*/*"}),  # snevr inn
ved behov
            "merknad": forms.Textarea(attrs={"rows": 4, "placeholder": "Valgfri
merknad"}),
        }
```

Det er alt. Med viewet ditt som allerede filtrerer mapper etter skrivetilgang og sjekker ACL ved lagring, er du good to go. 👌

---

4) Nginx – beskyttet levering (X-Accel-Redirect)

Legg denne **i HTTPS-serverblokka for q1.no** (sammen med de andre `/konservativt-` locationene):

`4) Nginx (X-Accel-Redirect, rask & sikker fillevering)`
`Når Django har godkjent forespørselen, svarer det med headeren X-Accel-Redirect som peker inn i en` **`intern`** `Nginx-location.`

- Nginx henter fila direkte fra `MEDIA_ROOT` og sender den effektivt til klienten.
- Uten gyldig Django-sjekk kommer man **ikke** inn i den interne locationen.

```
# Kun appen kan be Nginx levere filer herfra
location ^~ /konservativt/protected/ {
    internal;
    alias /home/bjornno11/q1.no/konservativt/media/;  # må matche MEDIA_ROOT
    add_header X-Content-Type-Options nosniff;
}
```

Test/reload:

```
sudo nginx -t && sudo systemctl reload nginx
sudo systemctl restart konservativt-q1
```

> I **dev (runserver)** har du ikke Nginx – da kan vi evt. falle tilbake til `FileResponse`. Si fra hvis du ønsker at `doc_download` skal ha dev-fallback, så legger jeg inn 3 linjer for det.

---

5) Hurtigbruk

5) Hurtigbruk (flyt i praksis)

1. Lag grupper → 2) Gi mapper les/skriv til grupper → 3) Legg brukere i grupper →
2. Last opp via `/doc/upload` → 5) Last ned via `/doc/d/<id>/`.
   Resultat: Brukere ser bare “sine” mapper og kan bare laste opp der de har skrivetilgang.

1. Lag grupper i **Admin** → **Groups** (f.eks. “Sentralt”, “Fylkesstyre”, “Lokallag Oslo”, “Økonomi”).
2. **Mapper**: sett *Kan lese*/*Kan laste opp* til riktige grupper.
3. Legg brukere i grupper (Admin → Users → velg bruker → Groups).
4. Last opp via **/konservativt/doc/upload** – bare mapper med skrivetilgang vises.

5. Last ned via **/konservativt/doc/d//** – sjekker rettigheter og lar Nginx levere fila.

---

6) Videreutvikling (når du vil)

6) Videreutvikling (når du trenger mer)

- Eierskap (mappe-eier har alltid tilgang)
- Finmasket ACL også på **enkelt-dokumenter**
- Koble grupper/tilgang til **fylke/lokallag** når `org/roles` er på plass.

- **Eierskap/overstyring:** gi `opprettet_av` alltid tilgang selv utenfor gruppe.
- **Per-dokument ACL:** legg `can_read`/`can_write` også på `Document` hvis du vil være mer finmasket.
- **RBAC på nivåer:** når `org`/`roles` kommer, kan vi auto-sette mappe-tilgang basert på fylke/lokallag.

Hvis du vil, kan jeg også legge inn *dev-fallback* i `doc_download` nå, eller skrive en liten “Opprett standardmapper og grupper”-management-kommando.

Ja — hvis du kjører via Gunicorn/systemd må du restarte tjenesten for at den nye `docs/forms.py` skal lastes.

Hvis du kjører via systemd (anbefalt nå)

```
sudo systemctl restart konservativt-q1
sudo systemctl status konservativt-q1 --no-pager
# ved feil:
journalctl -u konservativt-q1 -n 50 --no-pager
```

(Nginx trenger ikke reload for denne endringen.)

Hvis du kjører med runserver (dev)

Dev-serveren reloader vanligvis automatisk. Hvis ikke, stopp og start:

```
./venv/bin/python manage.py runserver 0.0.0.0:8011
```

Rask verifisering

1. Åpne: `https://q1.no/konservativt/doc/upload`
2. Se at “Mappe”-feltet viser bare mapper du har skrivetilgang til.
3. Last opp en fil → skal dukke opp på …`/doc/` og i mappen, med “Last ned”-lenke.

Typiske feilmeldinger og hva du gjør

- **403 ved opplasting:** mappen mangler skrivetilgang → gi brukerens gruppe tilgang i admin (DocFolder → “Kan laste opp (grupper)”).
- **TemplateDoesNotExist:** sjekk at `templates/docs/upload.html` finnes og at `TEMPLATES[0]["DIRS"]` inkluderer `BASE_DIR / "templates"` i `settings.py`.
- **ImportError: DocumentForm:** bekreft at fila heter `docs/forms.py` og at `views.py` har `from .forms import DocumentForm`. Restart Gunicorn igjen.

Kort og greit – sjekk dette:

1) Bruker serveren systemd?

```
ps -p 1 -o comm=
```

Hvis dette skriver `systemd`, har du systemd.

2) Kjører denne appen via systemd?

Prøv status på tjenesten vi laget:

```
sudo systemctl status konservativt-q1
sudo systemctl is-enabled konservativt-q1
journalctl -u konservativt-q1 -n 30 --no-pager
```

- Får du `active (running)` → ja, den kjører via systemd.
- Får du `Unit konservativt-q1.service could not be found` → ikke via systemd (eller feil navn).

3) Kjører den som dev-server i stedet?

```
ps aux | grep -E "manage.py runserver" | grep -v grep
```

Hvis du ser en prosess med `manage.py runserver` …, da er det *runserver* (ikke systemd).

4) Kjører Gunicorn manuelt/annet navn?

```
pgrep -a gunicorn
```

Ser du noe som `gunicorn ... konservativt.wsgi:application` uten at systemd har en unit → den er startet manuelt (tmux/screen) eller via en annen service.

5) Lytter backend-porten vår?

(vi satte 9010 i oppsettet)

```
sudo ss -lptn 'sport = :9010'
```

Lister den `127.0.0.1:9010` med `gunicorn` → backend er oppe.

Hva betyr dette i praksis?

- **Via systemd:** start/stop/restart med
  `sudo systemctl restart konservativt-q1`
- **Via runserver:** stopp med Ctrl-C i terminalen der den kjører.
- **Manuelt gunicorn:** stopp prosessen (PID) eller legg den inn som systemd-unit for ryddig drift.