IT1901 Gruppe 1 Dokumentasjon

Morten Skandsen, Sander Lindberg, Hanne Brynildsrud, Halvor Horge,
Sara Haugen, Magnus Tidemann & Marius Sjøberg

November 2018

Innhold

Figurer

1 Rammeverk

1.1 Django

Rammeverket vi var brukt for produket kalles Django. Hva er Django? Django er et rammeverk programmert i Python. Django hjelper webutviklere med en kjappere og smidigere måte å utvilke nettsider på. Når en utvikler en nettside er det mye en må tenke på: Innlogging, Utlogging, registrering, sikkerhet med mer. Dette sørger Django automatisk for. Rammeverket har innebygd databasesikkerhet, passordvalidering, skjemavalidering og mye, mye mer.

1.1.1 Hvorfor valgte vi Django?

Vi valgte å bruke Django som rammeverk da vi hadde hørt at det var et verktøy som var lett å bruke, samt hadde støtte for databaser. I tilegg bruker Django programmeringsspråket Python (og HTML, CSS og JavaScript), som var kjent for samtlige i gruppen.
Da ingen i gruppen hadde tildigere erfaring med databaser og sikkerthet, så vi Django som er godt valg for rammeverk.

2 Kode

2.1 Programmeringsspråk

Som nevnt har vi brukt rammeverket Django, som hovedsaklig bruker programmeringsspråket Python. I tilegg bruker Django **HTML**, **CSS** og **JavaScript**. (De vanligste webspråkene).

2.1.1 Versjoner

Versjonene til programmeringsspråkene vi har brukt er som følger:

- **Python 3.7**
- **HTML 5**
- **CSS 3**
- **JavaScript 1.8.5**

2.2 Sammenheng mellom språkene

Siden vi bruker Django, er det mye forskjellig kode. Hovedsaklig brukes HTML, CSS og JavaScript til front end (det som synes på siden + interaktivitet), mens Python brukes til back end (behandling av data, databaser osv...) Når en bruker går inn på en av sidene på nettsiden, sendes en GET request til serveren (Django). Denne GET requesten behandles av Python, som deretter sier ifra til serveren hvilken HTML som skal vises og hva i den som skal vises. På samme måte behandler Python POST requests (når brukeren sender et skjema, som f.eks innlogging). Databaser bruker som regel språket SQL, men i Django skrives også databasekommandoer i Python, som Django deretter automatisk gjør om til SQL.

2.2.1 Eksempel på sammenheng (Registrering)

Her vil du se et eksempel på hvordan Django håndterer POST og GET requests, samt en sammenheng mellom programmeringsspråkene brukt.

Dette er et eksempel på hvordan registreringen på siden er implementert. Det er 6 filer som spiller en rolle her, hvor fire av de er nødvendige Django-filer.
Filen som behandler GET og POST request kalles **views.py**:

Views.py

```
def register(request):
    if request.method == 'POST':
        form = RegForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            form.save()
            profil = Profile(user=form.save())
            profil.save()
            args = {'form': form, 'username': username}
            return redirect('/home', args)

    else:
        form = RegForm()

    args = {'form': form}
    return render(request, 'home/register.html', args)
```

Vi ser her at hvis requesten er POST skal skjemaet på siden valideres og lagres. Det skal opprettes en profil for den ny-registrerte brukeren og deretter skal brukeren omdirigeres til hjemmesiden. Hvis requesten er GET, skal skjemaet bare være tomt og "/home/register.htmlskal vises. Vi ser også at denne funksjonen bruker et form, som er definert i **Forms.py**:

Forms.py

```
class RegForm(UserCreationForm):

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

        for fieldname in ['username', 'password1', 'password2']:
            self.fields[fieldname].help_text = None

    class Meta:
        model = User
        fields = (
            'username',
            'first_name',
            'last_name',
            'email',
            'password1',
            'password2',)

    def save(self, commit=True):
        user = super(RegForm, self).save(commit=False)
        user.first_name = self.cleaned_data['first_name']
        user.last_name = self.cleaned_data['last_name']
        user.email = self.cleaned_data['email']

        if commit:
            user.save()

        return user
```

Skjemaet bruker Djangos innebygde registreringsskjema. **Fields** under **class Meta** beskriver hvilke felter av skjemaet som skal vises og **save()** lagrer informasjonen i en ny databasemodell. Modeller er oppgitt i **Models.py**:

Models.py

```
class Profile(models.Model):
    user = models.ForeignKey(User, related_name='user', on_delete=models.CASCADE)
    alder = models.CharField(max_length=3, blank=True)
    interesser = models.CharField(max_length=500000, null=True, blank=True)
    tlf = models.CharField(max_length=12, blank=True, null=True)

    def __str__(self):
        return self.user.username

    class Meta:
        verbose_name_plural = "Profiler"
```

Dette er en databasemodell for en brukerprofil. Den er skrevet i Python, men for at databasen skal skjønne hva det betyr, gjør Django denne koden om til SQL.

Den siste Django-filen kalles **Urls.py**. Denne sier hva som skal skje når brukeren skiver noe i adressefeltet:

Urls.py

```
url(r'^register/', views.register, name='register')
```

Denne linjen sier at hvis brukeren skriver inn 'localhost:8000/home/register' i addressefeltet, skal **register** viewet fra **views.py** kjøres. Deretter sier **register** viewet ifra at det er **register.html** som skal vises:

Register.html

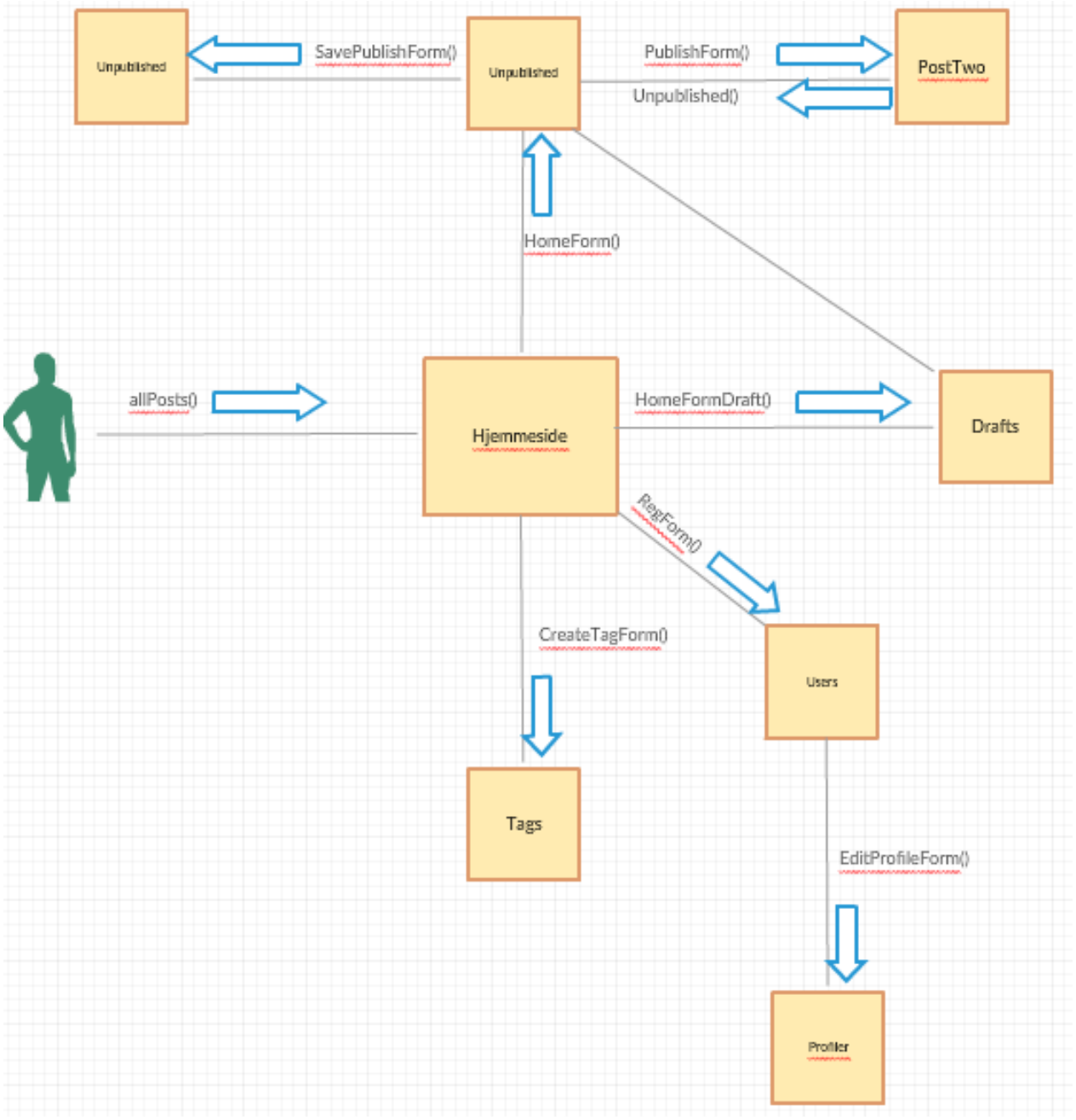
```
{% extends 'base.html' %}

{% block content %}
    <div class="page-container">
        <form class="container" id="article" method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
            <button type="submit" class="submitBtn">Register</button>
    </form>
    </div>
{% endblock %}
```

Her ser vi igjen et eksempel på Django rammeverket. alle linjene med %% eller {{ }} er 'Django kode'. Øverste linjen sier at HTML-filen skal arve fra en annen HTML fil. **{{ form.as_p }}** sier at skjemaet som kommer fra **Views.py** skal vises som er skjema. **{% csrf_token %}** er et eksempel på hvordan Django håndterer sikkerhet. Uten denne vil ikke formet vises, da den mangler verifikasjon i hendhold til Djangos regler.

Den siste filen er helt vanlig **CSS**.

2.3 UML

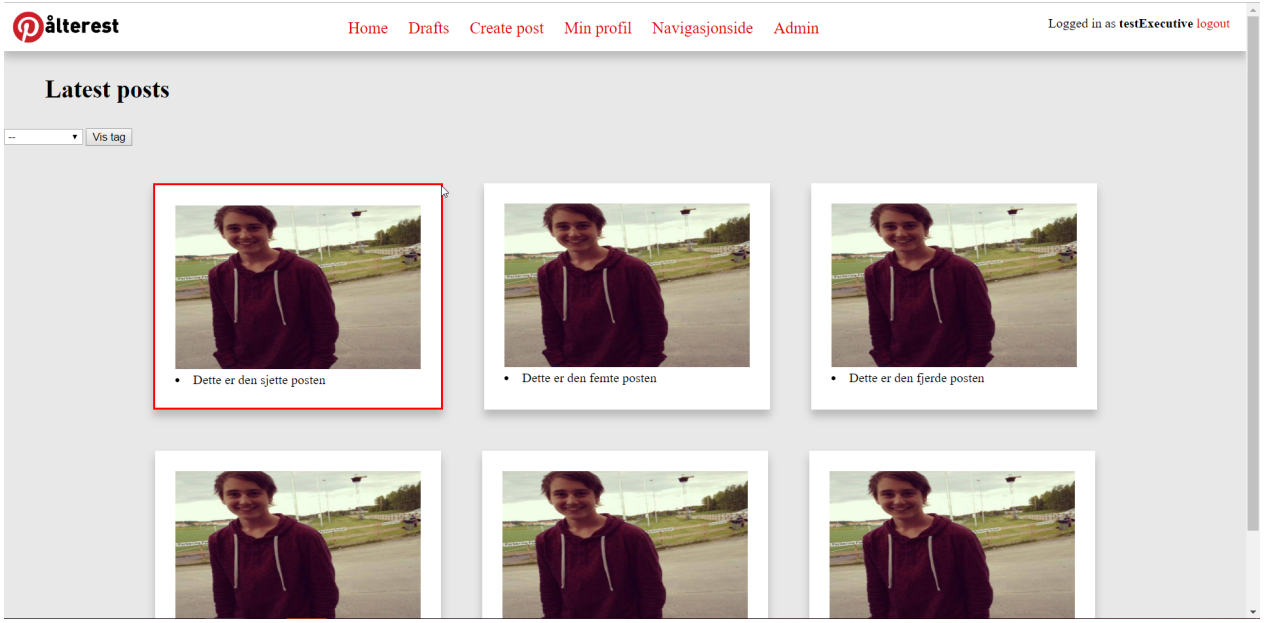


Figur 1: UML diagram av nettsiden

Figur 1 viser et UML diagram av nettsiden. Den viser hvilke modeller objektene lagres i i databasen og hvilke funksjoner og skjemaer som kjøres når brukeren utfører ulike operasjoner på nettsiden

3 Design

Designet for nettsiden var inspirert av Pinterest. Derfor har nettsiden et simpelt design med rød og hvit som primære farger. Vi var enige fra starten av at vi ønsket å designe nettsiden med brukervennlighet som øverste mål. Derfor har vi prøvd å få nettsiden så oversiktlig som mulig. Designet på nettsiden er hovedsaklig skrevet i CSS.



Figur 2: Forside av nettsiden

4 Brukermanual

4.1 Teknisk

4.1.1 Python

Det første en må gjøre for å ta i bruk produktet er å innstallere Python 3.7. Python kan lastes ned fra Pythons hjemmested

4.1.2 Git

Deretter må git installeres. Dette kan gjøres på flere måter:

- For Windows, Mac og Linux
  - Denne nettsiden
- For Mac
  - Åpne terminalen og skriv inn følgende:

```
$ sudo dnf install git-all
```

- For linux
  - Åpne terminalen og skriv inn følgende:

```
$ sudo apt install git-all
```

4.1.3 Klone prosjektet fra GitLab

Åpne terminalen og naviger til stedet du vil at mappen med prosjektet skal lastes ned. Dette gjøres med **cd** kommandoen. F.eks:

```
$ cd Users/Downloads
```

for Mac og Linux
Eller for Windows:

```
$ cd c:\Users\User\Downloads
```

Videre må en skrive inn kommandoen:

```
$ git clone https://gitlab.stud.idi.ntnu.no/it1901-2018/01.git
```

4.1.4 Kjøre serveren

For å kjøre serveren må du igjen kjøre **cd** kommandoen fra forrige punkt, bare denne gangen, naviger deg inn i mappen der **manage.py** filen fra det nedlastede prosjektet ligger. Deretter:

```
$ python3 manage.py runserver
```

og du er igang!

4.1.5 Avslutte serveren

For å avslutte serveren går du inn i terminalen hvor serveren kjører og klikker **CTRL+C**

4.2 Ikke-tekninsk

4.2.1 Komme deg inn på nettsiden

Start serveren ved instruksjoner under 4.1.4. Åpne deretter din favotittnettleser og gå inn på localhost:8000

4.2.2 Registrer deg

1. Hvis du allerede er logget inn med en av testbrukerene, klikk logg ut
2. Klikk logg inn øverst til høyre
3. Klikk **Er du ikke registrert? Registrer her"**
4. Skriv inn et passende brukernavn og passord og klikk **registrer** [1]

4.2.3 Oppdatere profil

1. Logg deg inn med ønsket bruker
2. Klikk **"min profil"**
3. Klikk **edit"**
4. Fyll inn ønsket innhold
5. Klikk **submit"**

[1] Er passordet for svakt eller brukernavnet opptatt vil du blir sendt til registreringssiden på nytt. Hvis registreringen er vellykket, vil du bli sendt til hovedsiden

4.2.4 Se en post

1. Start serveren og gå inn på nettsiden (4.1.4, 4.2.1)
2. Du vil se en mengde poster, klikk deg inn på en av de

4.2.5 Legge ut en post

1. Klikk logg inn øverst til høyre
2. Logg inn med en av brukerene gitt i 4.3.1[2]
3. Klikk på "**Create post**"
4. Lag en post og klikk submit [3]
5. Logg ut og logg inn med en adminbruker [4]
6. Klikk på navigasjonsside og deretter "**Review posts**"
7. Du vil nå se en liste over alle poster som ligger til godkjenning. Klikk deg inn på den du lagde i 4
8. Klikk submit

4.2.6 Lagre en post

1. Gjenta steg 1 - 3 i 4.2.5
2. Istedenfor å klikke **submit**, klikker du **save**

4.2.7 Lage tag

1. Logg deg inn med en bruker med executive editor rettigheter eller høyere (4.3.1)
2. Klikk **navigasjonsside"**
3. Klikk **"create tag"**
4. Skriv inn ønsket tag og klikk **save tag"**

[2] Merk: Brukeren må være author eller høyere. Nylig registrerte brukere vil ikke være authors
[3] Du blir sendt til hjemmesiden, men posten er lagt til godkjenning av admin/executive editor
[4] Bruk "adminsom brukernavn og "adminsom passord

4.2.8 Fjern en post fra hjemmesiden

1. Logg deg inn med en bruker med executive editor rettigheter eller høyere (4.3.1)
2. Gå inn på en post (4.2.4)
3. Klikk **delete** [5]

4.2.9 Endre rollen til en bruker

1. For dette **må** en være innlogget som Admin. (4.3.1)
2. Klikk admin
3. Naviger deg inn på **"users"**
4. Klikk på brukeren du vil endre rollen til
5. Bla deg nedover til du finner **groups"**
6. Legg til og fjern grupper her
7. Bla nederst på siden og klikk **save"**

4.3 Nyttige tips til nettsiden

4.3.1 Brukere

Her er en liste over brukere som ligger i databasen, hvis en har lyst til å teste med flere forskjellige roller:

Gruppe	**Brukernavn**	**Passord**
Author	testAuthor	qwertyuiopå
Editor	testEditor	qwertyuiopå
Executive editor	testExecutive	qwertyuiopå
Admin	admin	admin

[5] Sletter ikke posten, bare fjerner den fra hjemme siden. Den ender opp i "unpublished"