Ohsiha harjoitustyö, kulukorvausjärjestelmä
vaihe 3

Teemu Mikkonen

1. Lyhyesti

Harjoituksessa tuotetaan yhdistykselle web-pohjainen kulukorvaushakemusjärjestelmä. Tuottamani järjestelmä ei kerää ulkoista dataa, joten datan näyttäminen käyttäjälle tehdään käyttäjän omien syötteiden avulla.

Harjoitustyön kolmannessa vaiheessa on toteutettu käyttäjän ja järjestelmävalvojan pääpiirteisen käyttöliittymät. Rekisteröitynyt käyttäjä voi lisätä kulukorvaushakemuksia, jotka tallentuvat tietokantaan. Järjestelmävalvojalle on toteutettu toiminnallisuus hakemuksen hyväksymisestä ja hylkäämisestä. Käyttäjä näkee kulukorvaushakemuksensa tilan käyttöliittymästä.

Harjoitustyössä on aloitettu myös visuaalinen kehitys datan näyttämiseksi käyttäen Bootsrapia.

2. Tietokantarakenne

Tietokantarakenteeseen lisättiin toiminnallisuuden takia kaksi uutta ominaisuutta.

```
korvaussumma = models.DecimalField(max_digits = 8, decimal_places = 2, default=0)
status = models.CharField(max_length=3, default = 'LAH', choices = STATUKSET)
```

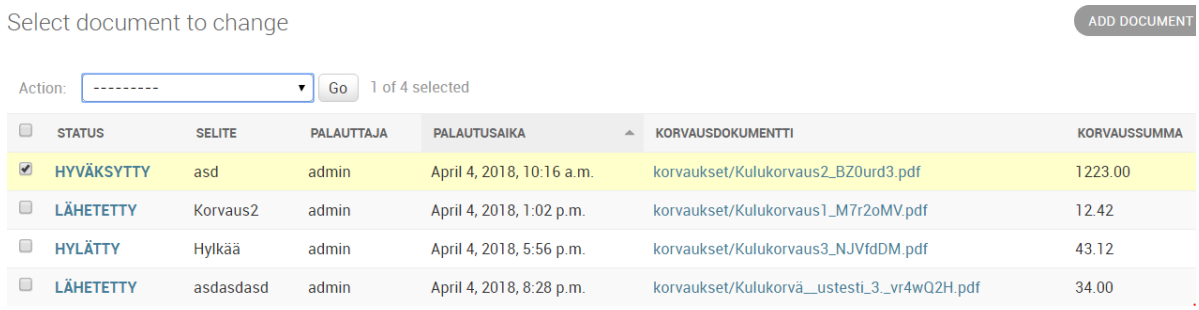
Statukselle annettiin eri vaihtoehdot: lähetetty, hyväksytty, hylätty. Korvaussumma lisättiin käyttäjän täytettäväksi syötteen upload-templateen.

3. Admin-toiminnallisuus

Admin-toiminnallisuus on tähän mennessä ollut hyvin rajoittunut ja sillä on voinut vain poistaa syötteitä tai käyttäjiä. Nyt admin.py on laajennettu seuraavasti.

```python
from django.contrib import admin
from .models import Document

def hyvaksy_korvaus(modeladmin, request, queryset):

    queryset.update(status = 'HYV')
    hyvaksy_korvaus.short_description = "Hyväksy kulukorvaushakemus"

def hylkaa_korvaus(modeladmin, request, queryset):

    queryset.update(status = 'HYL')
    hylkaa_korvaus.short_description = "Hylkää kulukorvaushakemus"

class DocumentAdmin(admin.ModelAdmin):
    list_display = ['status', 'selite', 'palauttaja', 'palautusaika',
    'korvausdokumentti', 'korvaussumma']
    ordering = ['palautusaika']
    actions = [hyvaksy_korvaus, hylkaa_korvaus]

admin.site.register(Document, DocumentAdmin)
```

Adminille on lisätty funktiot, joiden avulla kulukorvauksen status muuttuu. Luotiin myös luokka DocumentAdmin, joka näyttää pelkkien tiedostojen sijaan palautukset taulukkomuodossa ja järjestää ne palautusajan mukaan. Tässä myös rekisteröidään Adminin toiminnallisuudet. Lopulta luokka rekisteröidään admin.site.register -funktion avulla.



Action-valikosta päästään muuttamaan hakemuksen tilaa.

4. Käyttöliittymä

Käyttöliittymässä on tapahtunut suurimmat muutokset. Suurimpana haasteena oli näyttää käyttäjälle vain ne kulukorvaukset, joita käyttäjä itse on palveluun tallentanut. Tämän kehittämiseksi tehtiin ensin muutoksia upload-funktioon.

```python
if form.is_valid():
    instance = form.save(commit=False)
    instance.palauttaja = request.user
    instance.save()
```

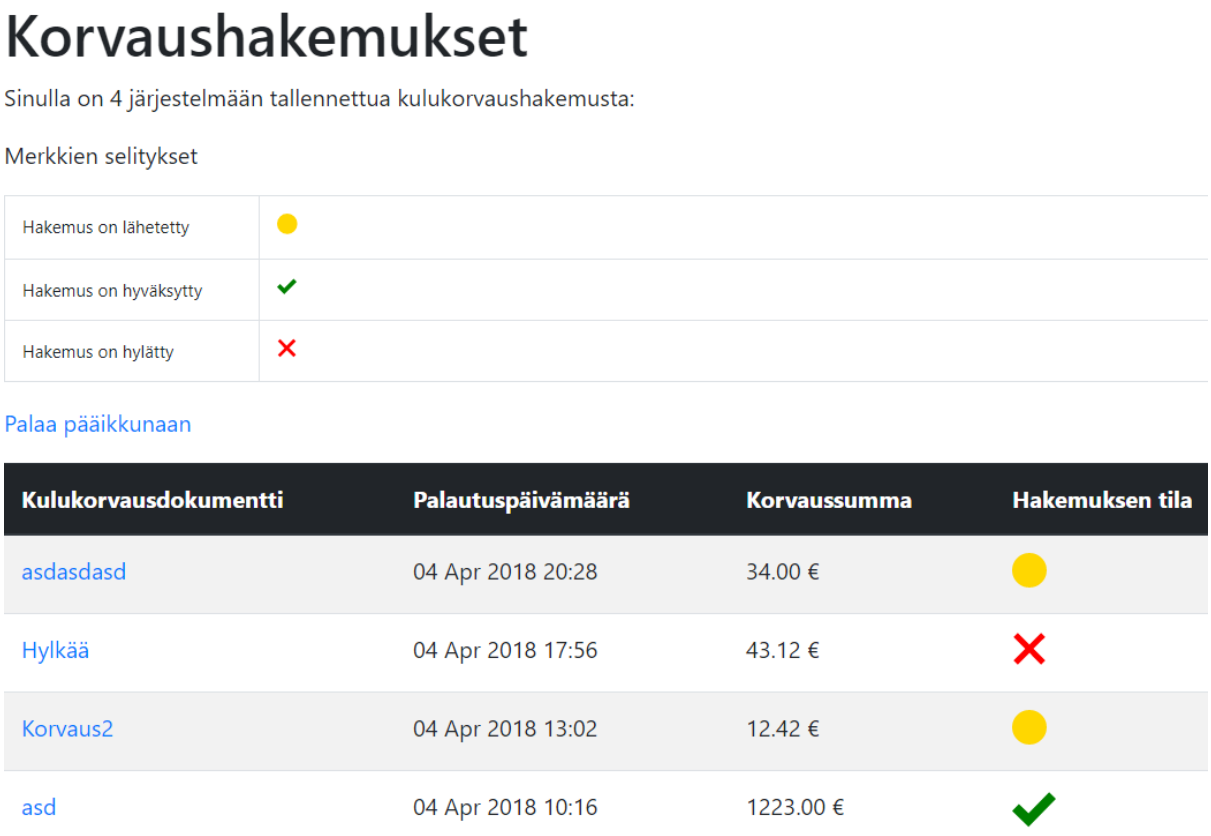
Instancen avulla tallennetaan palauttaja-ominaisuuteen sen hetkinen käyttäjä. Tämän jälkeen voidaan kysellä querysetin avulla tietokannasta ne palautukset, jotka käyttäjä on itse tehnyt. Queryset filtteröidään siten, että palauttaj_id vastaa sen hetkistä käyttäjää. Myös korvaussummat kysellään querysetin avulla käyttäjän kokonaissummaa varten. Kulukorvaukset annetaan korvaukset.html -templatelle.

```
def korvaukset(request):

    kulukorvaukset = Document.objects.filter(palauttaja_id=request.user.id)
    korvaussummat = Document.objects.filter(palauttaja_id=request.user.id).values('korvaussumma')

    total_list=[]
    total = 0

    for summa in korvaussummat:
        total_list.append(list(summa.values()))

    for korvaussumma in total_list:
        for alkio in korvaussumma:
            total += alkio

    template_data = {'kulukorvaukset':kulukorvaukset,
                     'total':total,
                     }

    return render(request, 'korvaukset.html', template_data)
```

Templatessa käsitellään korvausten näyttämistä html:n sisäisten python-skriptien avulla. Tässä vaiheessa on jo toteutettu visuaalinen ulottuvuus bootstrapin ja yksinkertaisten svg-tiedostojen avulla. Koko html-tiedostoa en tähän pura, mutta alla ovat tärkeimmät rakenteet ja ulkoasu.

```
  {% for korvaus in kulukorvaukset %}
      <tr>
        <td>
          <a href= "{%get_media_prefix%}{{korvaus.korvausdokumentti}}">
            {{korvaus.selite}}</a></td>
        <td>
          {{korvaus.palautusaika |date:"d M Y H:i"}}</td>
        <td>{{korvaus.korvaussumma}} €</td>
        {% if korvaus.status == 'HYV' %}
        <td>
          <svg xmlns="http://www.w3.org/2000/svg" width="30" height="30", fill="green"
          </svg>
        </td>
        {% elif korvaus.status == 'HYL' %}
        <td>
          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill = "red"
          </svg>
        </td>
        {% elif korvaus.status == 'LAH' %}
        <td>
          <svg width="24" height="24" fill = "gold" xmlns="http://www.w3.org/2000/svg"
          </svg>
        </td>
        {% endif %}
      </tr>

  {% endfor %}
  </table>
```

Ulkoasu käyttöliittymässä näyttää tällä hetkellä tältä.



Korvaushakemukset

Sinulla on 4 järjestelmään tallennettua kulukorvaushakemusta:

Merkkien selitykset

Hakemus on lähetetty	●
Hakemus on hyväksytty	✔
Hakemus on hylätty	✘

Palaa pääikkunaan

Kulukorvausdokumentti	Palautuspäivämäärä	Korvaussumma	Hakemuksen tila
asdasdasd	04 Apr 2018 20:28	34.00 €	●
Hylkää	04 Apr 2018 17:56	43.12 €	✘
Korvaus2	04 Apr 2018 13:02	12.42 €	●
asd	04 Apr 2018 10:16	1223.00 €	✔

Korvaushakemustesi summa on yhteensä 1312.54 €

Palaa pääikkunaan

Seuraavassa vaiheessa toteutetaan käyttäjälle mahdollisuus poistaa omia syötteitään, sillä tämä osoittautui yllättävän vaikeaksi.

5. Kolme helppoa/hankalaa tekijää

1. Bootstrap-kirjastojen avulla oli todella helppo muokata hyvän näköisiä ulkoasuja
2. Admin-toiminnallisuudet on rakennettu hyvin djangoon, hyvin pienillä muutoksilla saatiin aikaan toiminnallisuuksia adminille
3. HTML-dokumentissa djangon sisäänrakennetun python skriptin toiminnallsuuden ja syntaksin opettelu vaati pitkää testausta ja istumista djangon dokumentaation ääressä.

6. Lähteet

Tärkeimpinä lähteinä toimivat seuraavat:

Ohjelmallinen sisällönhallinta – luentodiat + teknologiademot

Djangon dokumentaatio

Simpleisbetterthancomplexin tutoriaali admin-rakenteiden muuttamiseen

Django Girlsin tutoriaali Querysetteihin

Näiden lisäksi ongelmissa on turvauduttu etenkin Stackoverflown ohjeisiin.