

Harjoitustyö vaihe 3: Data rajapinnan käyttö

Kuvaan tässä harjoitustyön osan 3 raportissa, miten otan käyttöön datarajapinnan, jonka avulla käyttäjälle näytetään jotakin palvelun kannalta olennaista dataa. En ollut vielä edellisessä harjoitustyön vaiheessa päättänyt, mistä aiheesta haluaisin työn toteuttaa. Aloitin tämän osion tutkimalla erilaisia datarajapintoja ja millaista dataa haluan käyttää harjoitustyössä. Tutkin Tampereen kaupungin tarjoamia rajapintoja, jossa oli paljon mielenkiintoisia vaihtoehtoja sovelluksen tekemiseen.

Valitsin kuitenkin rajapinnaksi OpenWeatherMapin tarjoaman Current Weather Data API:n, sillä API:sta on tarjolla paljon hyödyllistä ohjeistusta. En halunnut valita liian vaikeata rajapintaa, sillä tähän mennessä minulla on ollut hankaluuksia ymmärtää kokonaisuuden merkitystä ja API-kehittäminen on minulle täysin uutta. Koostepalveluni idea on nähdä tietyn alueen reaaliaikainen sää. Otan tässä vaiheessa käyttöön GitHubin käyttöön versionhallintaan varten.

Rajapinnan käyttöönotto

Käytin tovin aikaa sopivan rajapinnan löytämiseen, sillä halusin löytää sen liittyen aiheeseen, joka kiinnostaa minua. Katsoin muutaman tutoriaalin ennen kuin ryhdyin itse testaamaan rajapintaa. Tutustuin rajapintaan ja sen tarjoamaan dataan ensin huolella. Ennen kuin pääsin käsiksi rajapintaan, rekisteröidyin käyttäjäksi openweathermap.org-sivustolle.

Aloitan rajapinnan käyttöönoton luomalla *mysite*-projektin alle uuden applikaation *weather*.

```
python3 manage.py startapp weather
```

Lisäsin projektitason settings.py-tiedostoon uuden applikaation INSTALLED_APPS-kohtaan.

Jotta käytettävän API:n kautta tulevaa dataa voidaan käsitellä Pythonilla, tuli ensin ladata komentorivillä requests.

```
pip3 install requests
```

Seuraavaksi lisäsin API:n tiedot *weather* applikaation *views.py*-tiedostoon.

```
weather > views.py > ...
1  from django.shortcuts import render
2  import requests
3
4  # Create your views here.
5
6  def index(request):
7      url = 'http://api.openweathermap.org/data/2.5/weather?q={}&units=metric&appid=d89d87f40af6acfc405f16c215f274f'
8
```

Sitten tein paljon samoja vaiheita kuin aikaisemmissa applikaatioissa. Lisäsin projektitason *urls.py* tiedostoon uuden */weather*-urlin ja loin *weather*-applikaatioon *urls.py*-tiedoston. Loin projektitasoon tiedoston *templates/weather/weather.html*, jota myöhemmin muokkasin. Tein myös muutoksia applikaation *admin.py*-tiedostoon.

Ohjelman pää rakenne perustuu valmiiseen OpenWeatherMap API:a hyödyntävään ohjeeseen, mutta tein siihen paljon omia muutoksia ja lisäsin toimintoja.

Ohjelman toiminnan kannalta merkittävimmät muutokset tehtiin *weather*-applikaation tason tiedostoihin *views.py* ja *models.py*. Lisäksi luotiin *forms.py*-tiedosto, jotta käyttäjä voi syöttää haluamansa kaupungin nimen.

Views.py-tiedostoa muokattiin, jotta sovellus hakee käyttäjän syöttämät kaupungit tietokannasta. Views pyytää request-pyyntöä sää tiedot API:sta ja palauttaa ne weather templatesille. Context-kohdan tarkoitus on mahdollistaa datan hyödyntäminen weather templatesissa. Ohjelman koodi luo tietojen keräämisen kaupungin, lämpötilan, sään kuvauksen, sään kuvakkeen, kosteusprosentin, pilvisuusprosentin ja tuulennopeuden osalta API:n tiedoista.

```
weather > views.py > ...
1  from django.shortcuts import render
2  import requests
3  from .models import City
4  from .forms import CityForm
5
6  # Create your views here.
7
8  def index(request):
9      url = 'http://api.openweathermap.org/data/2.5/weather?q={}&units=metric&appid='
10
11      if request.method == 'POST':
12          form = CityForm(request.POST)
13          form.save()
14
15          form = CityForm()
16
17          cities = City.objects.all()
18
19          weather_data = []
20
21          for city in cities:
22
23              r = requests.get(url.format(city)).json()
24
25              city_weather = {
26                  'city': city.name,
27                  'temperature': r['main']['temp'],
28                  'description': r['weather'][0]['description'],
29                  'icon': r['weather'][0]['icon'],
30                  'humidity': r['main']['humidity'],
31                  'wind': r['wind']['speed'],
32                  'clouds': r['clouds']['all'],
33
34              }
35
36              weather_data.append(city_weather)
37
38          context = {'weather_data': weather_data, 'form': form}
39          return render(request, 'weather/weather.html', context)
```

Applikaatioon luotiin *forms.py*-tiedosto, jonka avulla kaupunkeja voidaan lisätä sivuston kautta.

```
weather > forms.py > ...
1  from django.forms import ModelForm, TextInput
2  from .models import City
3
4  class CityForm(ModelForm):
5      class Meta:
6          model = City
7          fields = ['name']
8          widgets = {'name': TextInput(attrs={'class': 'input', 'placeholder': 'City Name'})}
9
```

Models.py-tiedostoon täydennettiin tarvittava koodi, jotta tiedot säilyivät ja tallentuivat tietokantaan.

```
weather > models.py > ...
1  from django.db import models
2
3  # Create your models here.
4
5  class City(models.Model):
6      name = models.CharField(max_length=25)
7
8      def __str__(self):
9          return self.name
10
11     class Meta:
12         verbose_name_plural = 'cities'
```

Sivuston muotoilu

Kun rajapinnan käyttö oli saatu toimimaan onnistuneesti, tutustuin sovelluksen visuaalisuuteen. Tein *weather.html* koodiin muutoksia, jotta sain sivuston ulkonäöstä mukavamman. Tutustuin bootstrapin käyttöön, mutta päätin hyödyntää muotoilussa Bulmaa. Aion kuitenkin harjoitustyön seuraavassa vaiheessa perehtyä vielä kunnolla bootstrapiin ja tehdä sivustoani visuaalisemmaksi sen avulla. Alla on kuva *weather.html* tiedostosta, joka määrittelee sivuston ulkonäön. HUOM! Kuvassa ei ole koko koodi.

```
<form method="POST">
  {% csrf_token %}
  <div class="field has-addons">
    <div class="control is-expanded">
      {{ form.name }}
    </div>
    <div class="control">
      <button type="submit" class="button is-link is-rounded">
        Add Location
      </button>
    </div>
  </div>
</form>
</div>
</div>
</section>
<section class="section">
  <div class="container">
    <div class="columns">
      <div class="column is-offset-4 is-4">
        {% for city_weather in weather_data %}
        <div class="box">
          <article class="media">
            <div class="media-left">
              <figure class="image is-55x55">
                
              </figure>
            </div>
            <div class="media-content">
              <div class="content">
                <p>
                  <span class="title is-2">{{ city_weather.city }} </span>
                  <br>
                  <span class="subtitle is-4">Temperature: {{ city_weather.temperature }}° C</span>
                  <br>Weather: {{ city_weather.description }}
                  <br>Clouds: {{ city_weather.clouds }}
                  <br>Humidity: {{ city_weather.humidity }} %
                  <br>Wind speed: {{ city_weather.wind }} m/s
                </p>
              </div>
            </div>
          </article>
        </div>
        </div>
      </div>
    </div>
  </section>
```

Seuraavana on kuvana varsinaisesta sivustosta sen jälkeen, kun sen muotoilua parannettiin.

Hello! Welcome to the weather app.

Check out the latest weather conditions and forecasts around the world.

City Name Add Location



Tampere

Temperature: -1.7° C

Weather: broken clouds

Clouds: 51

Humidity: 58 %

Wind speed: 3.6 m/s



Helsinki

Temperature: -0.39° C

127.0.0.1:8000/weather/weather/



Tampere

Temperature: -1.7° C

Weather: broken clouds

Clouds: 51

Humidity: 58 %

Wind speed: 3.6 m/s



Helsinki

Temperature: -0.39° C

Weather: shower rain

Clouds: 20

Humidity: 58 %

Wind speed: 2.1 m/s



Vaasa

Temperature: -1.29° C

Weather: clear sky

Clouds: 0

Humidity: 33 %

Wind speed: 2.6 m/s

← → ↺ 127.0.0.1:8000/admin/weather/city/

Django administration

Home › Weather › Cities

Select city to change

Action: Go 0 of 3 selected

☐ CITY

☐ Vaasa

☐ Helsinki

☐ Tampere

Projektin vieminen GitHubiin

Koin mielekkääksi tässä vaiheessa harjoitustyötä lisätä tiedostot GitHubiin. Tämä tapahtui komentorivillä ensin kloonaamalla nykyinen repositorio, alustamalla Git projektin kansioon ja luotiin uusi commit. Lopulta tehtiin push kyseiseen repoon. En huomannut ottaa vaiheesta kuvia, mutta komentorivillä projektin vieminen Gitiin tapahtui suurin piirtein alla olevien vaiheiden mukaisesti.

```
git clone
git init
git . add
git commit -m "API:sta saatiedot saava saapalvelu"
git push origin
```

Harjoitustyön 3. vaiheen projektin versio löytyy GitHubistani
<https://github.com/anutamminen/ohsiha>.

Helppoa/haasteellista

- Alkuun pääseminen oli vaikeaa, vaikka aloitin tekemään hyvissä ajoin. Oli vaikeaa päättää, millaisen rajapinnan kanssa haluan työn toteuttaa. Kokeilen kahden eri rajapinnan käyttöä ennen kuin päädyin nykyiseen.
- Paljon oli erroraita matkassa, mutta positiivista on, että sain ne itse vaivattomasti ratkaistua ja ymmärrän Django toiminnan paljon paremmin. Edellisissä vaiheissa oli hankaluksia ymmärtää Django toiminnasta, mutta koen oppineeni nyt paljon.
- Bootstrapin tutustuminen oli mukavaa! Sivuston ulkoasun muotoilu oli työn hauskin vaihe ja aion ehdottomasti lisätä visuaalisuutta seuraavassa vaiheessa.
- Projektin pushaus GitHubiin ei sujunut ongelmitta, mutta dokumentaatiota ja ohjeita tarkasti seuraten se onnistui lopulta hienosti.
- Opin tässä vaiheessa valtavasti uutta. Seuraavassa vaiheessa ajatuksena olisi lisätä datan näyttämiseen ominaisuuksia.

Lähteet

API:n dokumentaatio

<https://openweathermap.org/current>

Vinkkejä rajapinnan käyttöönottoon

<https://simpleisbetterthancomplex.com/tutorial/2018/02/03/how-to-use-restful-apis-with-django.html>

Säärajapinnan käyttäminen

<https://www.youtube.com/watch?v=v7xjdXWZafY>
https://www.youtube.com/watch?v=lcWfSn6-m_8&t=2s

Pythonin requests

<https://realpython.com/python-requests/>

Sivuston ulkoasu ja bootstrap

<https://bulma.io/documentation/elements/button/>
https://www.w3schools.com/bootstrap4/bootstrap_containers.asp

GitHubin käyttäminen

<https://help.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository>
<https://help.github.com/en/github/managing-files-in-a-repository/adding-a-file-to-a-repository-using-the-command-line>