Haaga-Helia
University of Applied Sciences

Ohjelmistokehityksen teknologioita - *S*eminaarityö

Sää-applikaatio

Python

Mari Paltiala

6.12.2022

Sisältö

Tiivistelmä

Työn tavoitteena oli tehdä sääsovellus, joka tarjoaa käyttäjille ajantasaista ja luotettavaa tietoa säätilasta ja sääennusteista. Sovellus toteutettiin Python-kielisenä web-sovelluksena, jonka avulla käyttäjä voi hakea säätietoja syöttämällä kaupungin nimen. Sovellus käyttää Flask-kirjastoa sekä OpenWeatherMapin API:a hakiessaan säätietoja. Jos kaupunkia ei ole annettu, sovellus ilmoittaa virheestä. Sovellus näyttää säätiedot index.html-nimisessä HTML-tiedostossa.

Seminaarityön tuloksena saatiin luotua halutunlainen sääsovellus, joka antaa käyttäjälle tietoina ajankohdan, lämpötilan, kosteusprosentin sekä tuulennopeuden ja -suunnan. Tiedot on rajattu siihen mitä OpenWeatherMapin API -rajapinta tarjoaa.

1 Johdanto

Tavoitteena oli tehdä sääsovellus, joka tarjoaa käyttäjille ajantasaista ja luotettavaa tietoa säätilasta ja sääennusteista.

Toteutin sovelluksen seuraamalla kahta eri tutoriaalia ja käyttämällä niistä opittuja taitoja luoden itselleni mieleisen sovelluksen. Pääpaino oli opetella Python Flaskin ja HTML-tiedoston yhdistämistä sekä API-pyyntöjen tekemistä, sillä nämä asiat jäivät kurssilla itselläni vähäisiksi. Lisäksi minua kiinnostaa Pythonin opettelu (en ennen tätä kurssia ole Pythonia käyttänyt).

Aloitin työn teon tekemällä taustatutkimusta siitä mikä ylipäätään on Python Flask ja mihin sitä voi hyödyntää. Flask on Python- pohjainen kirjasto, jonka avulla voi luoda verkkosovelluksia. Flaskin avulla voi luoda esimerkiksi RESTful API:n tai interaktiivisen verkkosivuston, jota itse hyödynnän seminaarityössäni. Flask toimii hyvin yhdessä muiden Python-kirjastojen kanssa.

Taustatutkimuksen jälkeen löysin kaksi hyvää tutoriaalia, jotka olivat itsessäänkin hyviä, mutta en halunnut suoraan kopioida ja täten jättää oppimista minimaaliseksi. Analysoin molemmat koodit ja päätin minkälaisen haluan omasta tuotoksestani. Molempien tutoriaalien läpikäynti auttoi oppimaan API-pyyntöjen tekemistä ja datan käsittelyä, sillä se oli toteutettu eri tavalla molemmissa tutoriaaleissa, vaikka pohjana oli sama OpenWeatherMap-rajapinta. Toisessa tutoriaalissa käyttäjällä ei ollut mahdollista kirjoittaa haluamaansa kaupunkia, vaan se tuli tehdä URL-osoitteeseen. Pidin kyseisen tutoriaalin lopputuloksesta muuten,

mutta halusin lisätä käyttäjän interaktiivisuutta ja luoda HTML-puolella formin, johon käyttäjän on mahdollista kirjoittaa haluamansa kaupungin nimi. Kävin huolellisesti läpi molemmat tutoriaalit ja mitä koodi tarkalleen kummassakin tekee ja onnistuin tuottamaan halutunlaisen lopputuloksen. Seuraavassa kappaleessa ”käytetyt tekniikat” käydään läpi mitä koodissa tapahtuu ja miten sovellus toimii.

2 Käytetyt tekniikat

2.1 Python Flask

Alussa tuodaan Flask-kirjasto, jota käytetään web-sovelluksen toteuttamiseen.sekä requests-kirjasto, jota käytetään HTTP-pyyntöjen lähettämiseen OpenWeatherMapin API:lle.

```
from flask import Flask, request, render_template
import requests

app = Flask(__name__)
```

Tämän jälkeen luodaan Flask-sovelluksen ilmentymä nimeltä app ja määritetään reititys (route) Flask-sovellukselle osoitteeseen '/', johon voi tehdä sekä GET- että POST-pyyntöjä. Tämä reititys suorittaa index-nimisen funktion.

```
@app.route('/', methods=["GET", "POST"])
def index():
```

Index-funktiossa luodaan muuttujat data, error ja city, joita käytetään myöhemmin säätietojen näyttämiseen.

```
def index():
    data = ''
    error = 0
    city = ''
```

Tämän jälkeen tarkistetaan, onko pyyntö tehty POST-menetelmällä. Jos näin on, city-muuttujaan tallennetaan käyttäjän syöttämä kaupungin nimi lomakkeesta.

Jos city on annettu, luodaan uusi muuttuja weatherApi, johon tallennetaan OpenWeatherMapin API-avain, ja url-muuttuja, johon luodaan osoite OpenWeatherMapin API:lle haettaessa säätietoja annetusta kaupungista. Sitten lähetetään pyyntö OpenWeatherMapin API:lle ja tallennetaan vastauksena saadut säätiedot data-muuttujaan JSON-muodossa.

Jos city-muuttuja on tyhjä, asetetaan error-muuttuja arvoon 1.

```
if request.method == "POST":
        city = request.form.get("cityName")
        if city:
            weatherApi = '7e491f82ad6ab793e093d82c83936de2'
            url = "https://api.openweathermap.org/data/2.5/fore
cast?q="+city+"&appid=" + weatherApi
            data = requests.get(url).json()
        else:
            error = 1
```

Lopuksi palautetaan renderöity (render_template) HTML-sivu index.html, johon lähetetään data-, city- ja error-muuttujien arvot.

```
    return render_template('index.html', data=data, cityName=city, error=er-
ror)
```

2.2 HTML

Käyn HTML-tiedostosta läpi olennaisimmat osat, jotka liittyvät sovelluksen toimintaan. HTML-tiedostossa määritellään <form>-elementti, jonka avulla käyttäjä voi lähettää tiedot lomakkeella sekä <input>-elementti, jonka avulla käyttäjä voi syöttää kaupungin nimen. <input>-elementillä määritetään id, jota käytetään Pythonin puolella ja joka tunnistaa käyttäjän syöttämän kaupungin. <button>-elementillä luodaan käyttäjälle painettava nappula ”get weather”, joka on tyyppiä ”submit” eli tieto lähetetään käyttäjältä sovellukselle. {% if %}-komennolla, tarkistetaan, onko virhe määritelty ja esiintyykö se. Jos virhe on olemassa, koodi näyttää viestin "Error: Please enter valid city name".

```
            <form action="{{ url_for('index')}}" method="post">
            <div class="form-group">
                <label for="cityName">City:</label>
                <input type="text" id="cityName" name="cityName"
value="{{cityName}}" placeholder="Type city">
                <button class="submit">Get weather</button>
                {% if error is defined and error %}
                <br><br><span class="alert alert-danger">Error: Please enter
valid city name</span></br>
                {% endif %}
            </div>
```

Jos käyttäjä on kirjoittanut kaupungin, renderöidään taulukko. Taulukkoon haetut tiedot haetaan JSON-muotoisena datana OpenWeatherApi -rajapinnasta API-avaimella. Alla esimerkki API-rajapinnan tarjoamasta datasta:


```
{
  "cod": "200",
  "message": 0,
  "cnt": 40,
  "list": [
    {
      "dt": 1670274000,
      "main": {
        "temp": 272.55,
        "feels_like": 267.7,
        "temp_min": 272.39,
        "temp_max": 272.55,
        "pressure": 1022,
        "sea_level": 1022,
        "grnd_level": 1019,
        "humidity": 80,
        "temp_kf": 0.16
      },
      "weather": [
        {
          "id": 600,
          "main": "Snow",
          "description": "light snow",
          "icon": "13n"
```

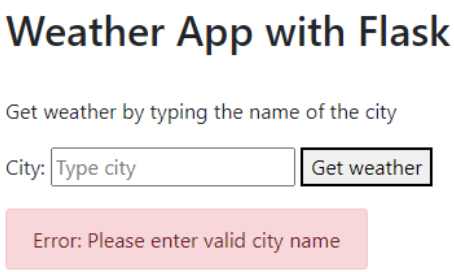
Valitsin työhöni seuraavat tiedot:

```
                <tr>
                    <td>{{ item["dt_txt"] }}</td>
                    <td>{{ item["main"]["temp"] }}</td>
                    <td>{{ item["main"]["temp_kf"] }}</td>
                    <td>{{ item["main"]["humidity"] }}%</td>
                    <td>{{ '%.2f' % (item["wind"]["speed"]*3.6) }} km/h ({{
item["wind"]["deg"] }} deg)</td>
                </tr>
```

3 Tulokset ja pohdinta

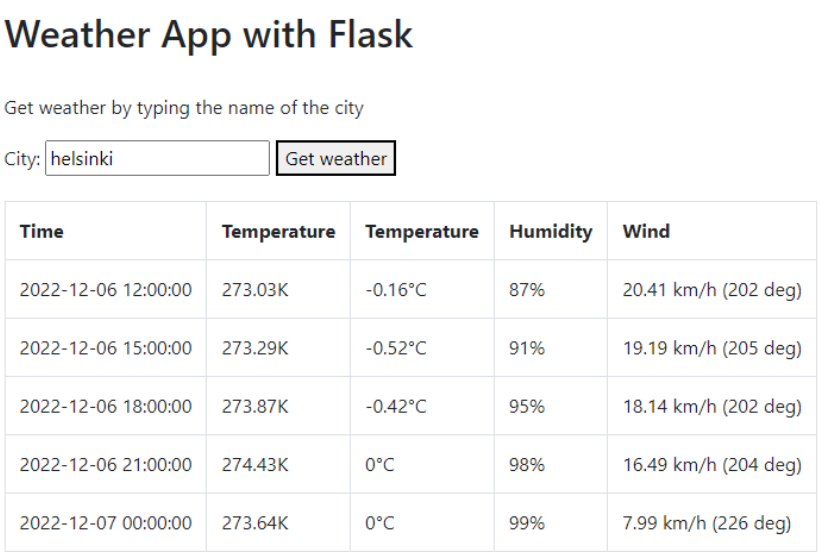
3.1 Lopputulos

Sovellus aukeaa näkymään, jossa käyttäjä voi kirjoittaa haluamansa kaupungin nimen syöttökenttään. Mikäli kentän jättää tyhjäksi, sovellus herjaa ja kehottaa syöttämään validin kaupungin.



Kuva 1. Ensimmäinen näkymä

Kaupungin syöttämisen jälkeen, käyttäjä ohjataan näkymään, jossa on listattu kyseisen kaupungin säätiedot.



Kuva 2. Näkymä kaupungin syöttämisen jälkeen.

3.2 Pohdinta

Koin tehtävän mielekkääksi, sillä opin sen myötä käsittelemään Flaskin avulla erilaisia pyyntöjä (GET- ja POST-pyyntöjä) sekä käsittelemään Flaskilla JSON-dataa. Pythonia en ennen tätä kurssia ole käyttänyt, joten tämä seminaarityö auttoi oppimaan Pythonia entistä paremmin. Mielestäni tekemäni sovellus on hyödyllinen. Sovellusta voisi jatkaa sillä tavoin, että käyttäjä voisi lisätä tietokantaan eniten hakemiaan kaupunkeja, jolloin niiden säätietojen hakeminen helpottuisi.

https://github.com/Mardehei/WeatherApp

Kiitos!

Lähteet

https://dev.to/cscarpitta/creating-a-simple-weather-app-with-python-and-flask-5bpd

https://home.openweathermap.org/

https://webdamn.com/develop-weather-app-using-flask-in-python/