Projektni zadatak v1

APLIKACIJA VREMENSKE PROGNOZE I KVALITETE ZRAKA

Opis aplikacije:

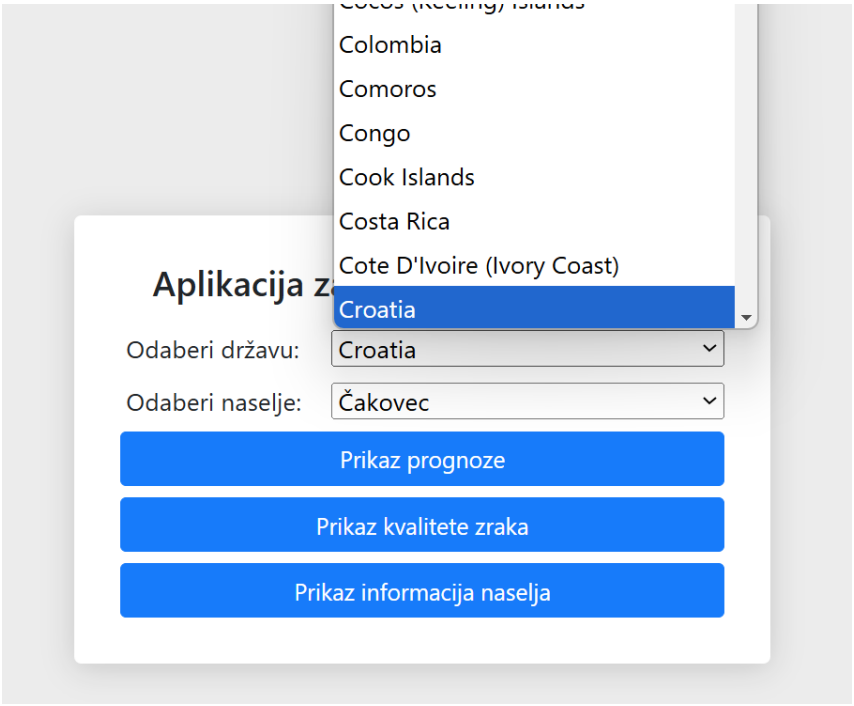
Tema završnog zadatka je web platforma koja omogućava sinkronizaciju (povlačenje) vremenske prognoze za naselja, a prema želji i mogućnosti autora, može uključivati i izbor putem Google Maps-a (samo za entuzijaste). Ova web aplikacija ima za ulogu povlačenje podataka putem API poziva iz „open API“ sustava „OpenWeatherMap“.

Struktura aplikacije:

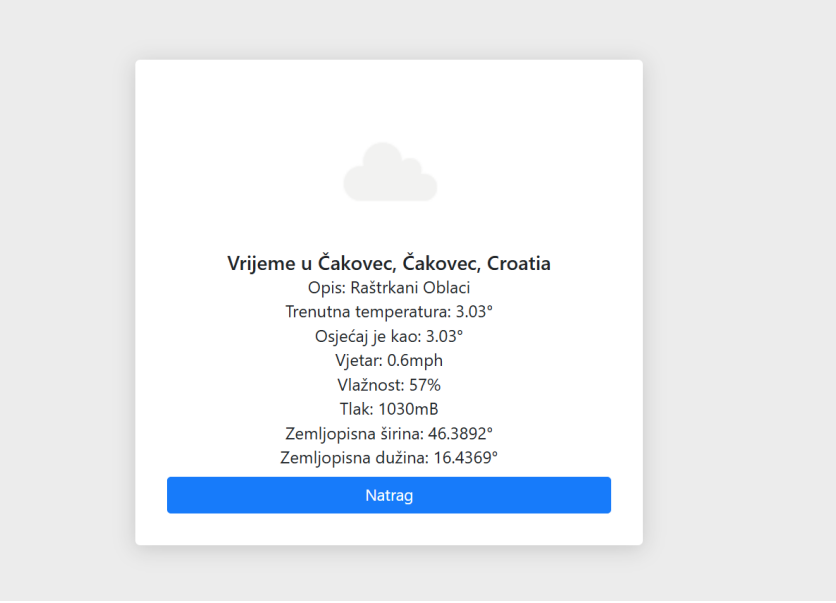
Aplikacija treba imati izbornik:

HOME	**Prognoza i kvaliteta zraka**	**Države**	**Naselja**	**API**			

- HOME
    - Početna stranica neka po vašem izboru sadrži informacije o aplikaciji. Dizajn NIJE presudan u radu, jer se temeljimo na funkcionalnosti u smislu vještine upotrebe programskog jezika Python i tehnologija: algoritmi, korištenje GIT-a, objektno orijentirano programiranje, rad s SQLite bazom podataka, korištenje funkcija i složenih struktura podataka te implementaciju Flask radnog okvira (*framework*).
- Prognoza i kvaliteta zraka
    - Stranica koja prikazuje polja za pretraživanje prema ključnim poljima
    - Polja za pretraživanje trebaju izgledati ovako:
        1. Država
            - Padajući/drop-down izbornik s popisom država pohranjenih u SQLite bazi podataka
        2. Naselja
            - Padajući/drop-down izbornik s popisom naselja pohranjenih u SQLite bazi podataka koji su filtrirani prema prethodno odabranoj državi
        3. Gumbi (ili tekstualni linkovi)
            - “Prikaz prognoze”
            - “Prikaz kvalitete zraka”
            - “Prikaz informacija naselja”
    - Primjer mogućeg prikaza:



- Klikom na gumb:
  1. "Prikaz prognoze":
     - Prikazuje se trenutna prognoza u formi koju odabire autor
     - Primjer mogućeg prikaza:



  2. "Prikaz kvalitete zraka":
     - Prikazuje se trenutna kvaliteta zraka u formi koju odabire autor
  3. "Prikaz informacija naselja":
     - Prikazuju se informacije o naselju i državi u formi koju odabire autor
     - Potrebno je kreirati novu HTML stranicu koja će prikazivati grupne informacije o odabranom naselju i državi

- Države
    - Stranica koja omogućuje:
        1. Tabelarni prikaz unesenih država u SQLite bazi
        2. Treba biti omogućene CRUD operacije
            - Tabela Drzava treba minimalno sadržavati
                - ID
                - Naziv
                - Iso2
                    - Popis iso2 kodova: https://en.wikipedia.org/wiki/ISO_3166-1#:~:text=native%20name%20%22Deutschland%22.-,Codes,-%5Bedit%5D
                - Površina
                - Broj stanovnika
                - Kontinent (tekst)
                - Valuta
                - Jezici (tekst)
            - Obavezni podaci kod unosa novog zapisa su: Naziv, Iso2, Kontinent, Valuta i Jezici
            - ***Pripazite:*** Kod brisanja određene države, automatski se trebaju obrisati i svi gradovi povezani s tom državom
- Naselje
    - Stranica koja omogućuje:
        1. Tabelarni prikaz unesenih naselja u SQLite bazi
        2. Treba biti omogućene CRUD operacije
            - Tabela Naselje treba minimalno sadržavati
                - ID
                - Naziv
                - Država ID
                - Zemljopisna širina (latitude – koristite Geocoding API)
                - Zemljopisna dužina (longitude – koristite Geocoding API)
                - Kategorija naselja (npr. padajući izbornik s oznakama Grad i Selo)
                - Površina
                - Broj stanovnika
                - Poznate osobe
            - Obavezni podaci kod unosa novog zapisa su: Naziv, Država ID i Kategorija naselja
                - ***Pripazite:*** Polja Zemljopisna širina i Zemljopisna dužina se trebaju automatski popuniti pozivanjem Geocoding API-ja prilikom kreiranja zapisa. Također, prilikom ažuriranja postojećeg zapisa, ta polja moraju biti zaključana kako bi se onemogućilo ručno uređivanje. Na autoru je odgovornost da na prikladan način prikaže eventualne pogreške (npr. obavijest na stranici) u slučaju kada Geocoding API ne uspije dohvatiti zemljopisne koordinate.

- ***Dodatno (nije nužno):*** Ako Geocoding API ne uspije pronaći zemljopisne koordinate za određeno naselje, omogućite korisniku da ručno unese podatke za zemljopisnu širinu i dužinu. Također, implementirajte opciju koja omogućuje zaključavanje i otključavanje tih polja prilikom uređivanja zapisa.

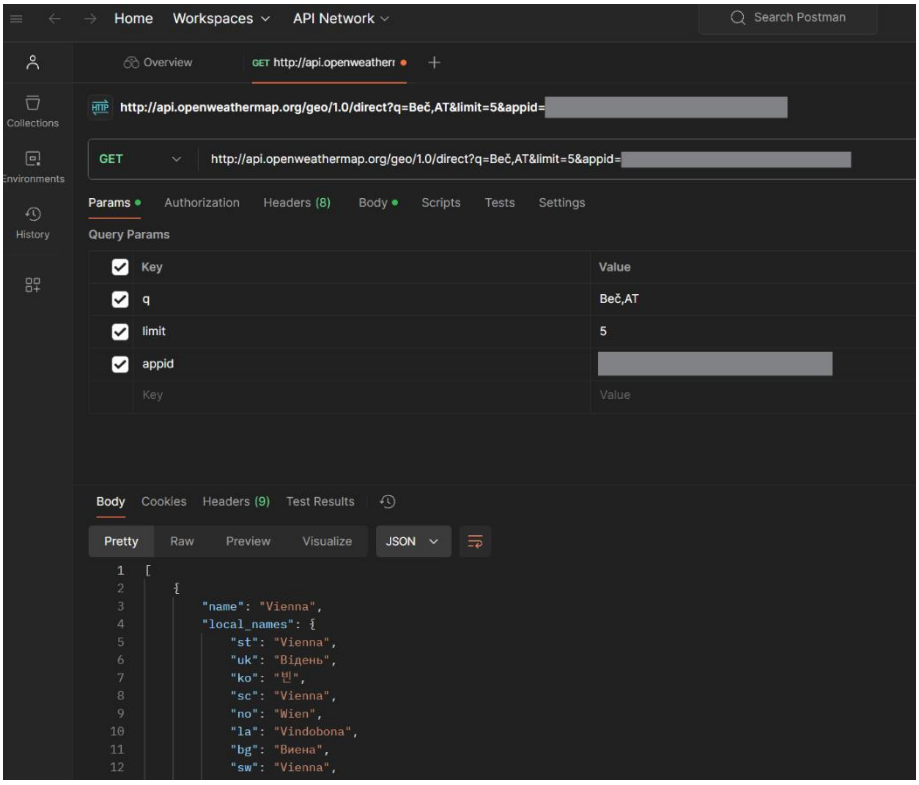
Aplikacija treba sadržavati SQLite bazu s minimalnim brojem tablica:

- Drzava
- Naselje

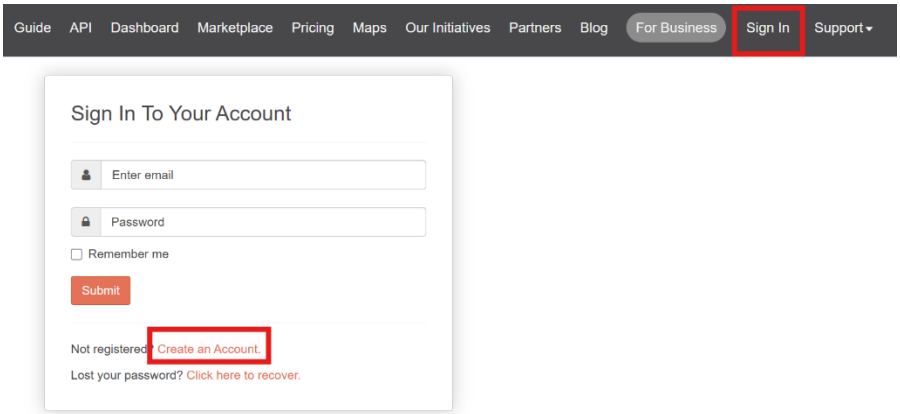
Svaka tablica treba imati primarni ključ, jedinstvena polja prema zahtjevu i procjeni. Tablica Naselje ima „strani ključ“ (po)vezan uz tablicu Drzava.

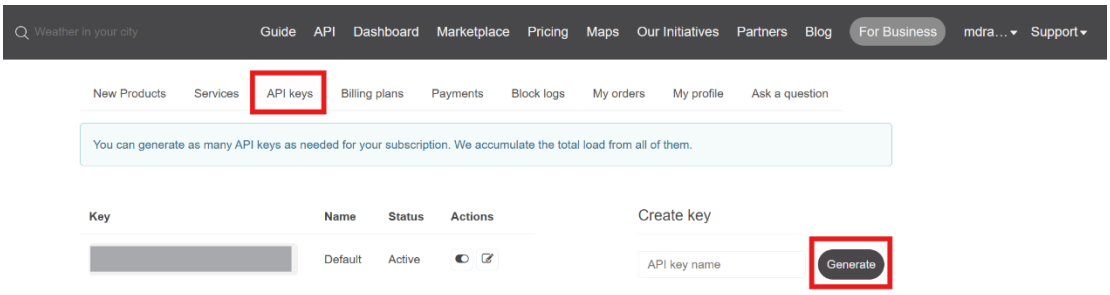
Za API pozive i dodane funkcionalnosti proučite priložene stranice OpenWeatherMap API-a:

- https://openweathermap.org/current
  - Koristite query kao: **“lat={lat}&lon={lon}&appid={API key }&units=metric&lang=hr”**
- https://openweathermap.org/api/air-pollution#:~:text=levels%20scale%22%20page.-,Current%20air%20pollution%20data,-API%20call
- https://openweathermap.org/api/geocoding-api
  - Koristite query kao: **“q={city name},{country code}&limit=1&appid={API key}”**
    - **“{city name}”** omogućuje unos naziva grada ili naseljenog područja, bilo prema izvornom nazivu grada ili nazivu koji se koristi u drugom jeziku (npr. Beč umjesto Vienna)
  - Zbog pojednostavljenja, limit je postavljen na 1 kako bi se uvijek uzimao samo prvi zapis
- Za testiranje API poziva možete koristiti aplikaciju **Postman**: https://www.postman.com/

- 

Otvorite besplatan račun na OpenWeatherMap platformi i generirajte API ključ koji ćete koristiti kod slanja upita na API sučelje:

- 



- ***Važno upozorenje:*** Prije predaje projekta (tj. uploadanja projekta na GitHub repozitorij), obavezno uklonite sadržaj API ključa, budući da se radi o vašem privatnom ključu
    - Kako biste sigurno pohranili API ključ, preporučuje se kreiranje posebne datoteke pod nazivom "api_kljuc.env", u koju ćete unijeti sadržaj vašeg ključa.
        - Primjer sadržaja "api_kljuc.env" datoteke:
            - **API_KLJUC_OPENWEATHERMAP=<vaš_privatni_API_ključ>**
    - Za učitavanje API ključa u vaš Python projekt, možete koristiti biblioteku **dotenv**, koja omogućuje dohvaćanje varijabli iz .env datoteke:

```
# pip install python-dotenv
from dotenv import load_dotenv

# python-dotenv čita parove ključ-vrijednost iz .env datoteke i postavlja ih kao varijable okruženja
load_dotenv("api_kljuc.env")
API_KLJUC_OPENWEATHERMAP = os.environ.get("API_KLJUC_OPENWEATHERMAP")
```

    - Također, vrlo je važno dodati datoteku "api_kljuc.env" u ".gitignore" kako bi se osiguralo da ona neće biti uključena prilikom postavljanja projekta na GitHub. Na taj način vaš API ključ ostaje siguran i zaštićen od neovlaštenog pristupa.
        - **.gitignore** je posebna konfiguracijska datoteka u Git projektima koja definira koje datoteke ili direktoriji Git treba ignorirati

Od autora se očekuje odgovoran pristup u rukovanju korisničkim pogreškama. U slučaju da korisnik napravi grešku (npr. unese neispravan podatak ili izostavi obvezno polje), autor bi trebao osigurati da sustav jasno i razumljivo prikaže obavijest o pogrešci.

Svaki polaznik obvezan je otvoriti vlastiti privatni GIT repozitorij za projekt (u koji će se „uploadati" projektna implementacija), u koji je potrebno dodati predavače s pravom pregleda (read).

Za sva pitanja i komunikaciju koristit će se Timska stranica s kanalom „ZAVRŠNI PROJEKT".

Rok izrade projekta je do **15.2.2025. godine do ponoći**. Planirani rok za obranu je **17.2.2025. godine u 17:00**.

Ocjenjuje se funkcionalnost i razumijevanje sadržaja predanog projekta. Projekt mora sadržavati opisane dijelove. Za prolaz je neophodno da projekt sadrži barem 50% opisane funkcionalnosti projekta.

POMOĆ: Specifikacija OpenWeatherMap API poziva s primjerima u Pythonu

Da biste koristili OpenWeatherMap API, prvo se trebate registrirati na njihovoj web stranici i dobiti **API ključ**.

---

1. Trenutna vremenska prognoza

API za dohvaćanje trenutnih vremenskih podataka za grad.

- **Endpoint**: http://api.openweathermap.org/data/2.5/weather
- **Parametri**:
    - q: Naziv grada (npr. "Zagreb").
    - appid: Vaš API ključ.
    - units: Jedinice mjerenja (metric za Celzijus, imperial za Fahrenheit, standard za Kelvin).

Primjer u Pythonu:

```
import requests


def get_current_weather(city_name, api_key):
    url = "http://api.openweathermap.org/data/2.5/weather"
    params = {
        "q": city_name,
        "appid": api_key,
        "units": "metric"
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
```

```
    return response.json()
  else:
    return f"Error: {response.status_code}, {response.json()}"

# API ključ i primjer poziva
api_key = "YOUR_API_KEY"
city = "Zagreb"
weather = get_current_weather(city, api_key)
print(weather)
```

---

2. Satna prognoza (48 sati)

API za dohvaćanje vremenske prognoze u satnim intervalima.

- **Endpoint**: http://api.openweathermap.org/data/2.5/forecast
- **Parametri**:
    - q: Naziv grada.
    - appid: Vaš API ključ.
    - units: Jedinice mjerenja.

Primjer u Pythonu:

```
def get_hourly_forecast(city_name, api_key):
  url = "http://api.openweathermap.org/data/2.5/forecast"
  params = {
    "q": city_name,
    "appid": api_key,
```

```
    "units": "metric"
  }
  response = requests.get(url, params=params)
  if response.status_code == 200:
    return response.json()
  else:
    return f"Error: {response.status_code}, {response.json()}"

# API ključ i primjer poziva
forecast = get_hourly_forecast(city, api_key)
print(forecast)
```

---

3. Povijesni podaci

API za dohvaćanje povijesnih vremenskih podataka (plaćena opcija).

- **Endpoint**: http://api.openweathermap.org/data/2.5/onecall/timemachine
- **Parametri**:
    - lat: Geografska širina grada.
    - lon: Geografska dužina grada.
    - dt: Vrijeme u UNIX formatu (sekunde od 1. siječnja 1970.).
    - appid: Vaš API ključ.

Primjer u Pythonu:

```
def get_historical_weather(lat, lon, unix_time, api_key):
  url = "http://api.openweathermap.org/data/2.5/onecall/timemachine"
```

```
    params = {
        "lat": lat,
        "lon": lon,
        "dt": unix_time,
        "appid": api_key,
        "units": "metric"
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        return f"Error: {response.status_code}, {response.json()}"

# API ključ i primjer poziva
latitude = 45.815
longitude = 15.981
unix_time = 1672531200  # UNIX timestamp
historical_weather = get_historical_weather(latitude, longitude, unix_time, api_key)
print(historical_weather)
```

---

4. Kvaliteta zraka

API za dohvaćanje podataka o kvaliteti zraka.

- **Endpoint**: http://api.openweathermap.org/data/2.5/air_pollution

- **Parametri**:
    - lat: Geografska širina grada.
    - lon: Geografska dužina grada.
    - appid: Vaš API ključ.

Primjer u Pythonu:

```
def get_air_quality(lat, lon, api_key):
    url = "http://api.openweathermap.org/data/2.5/air_pollution"
    params = {
        "lat": lat,
        "lon": lon,
        "appid": api_key
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        return f"Error: {response.status_code}, {response.json()}"


# API ključ i primjer poziva
air_quality = get_air_quality(latitude, longitude, api_key)
print(air_quality)
```

---

5. Geokodiranje

Za pretvaranje imena grada u koordinate i obrnuto.

- **Endpoint**: http://api.openweathermap.org/geo/1.0/direct (za naziv u koordinate).
- **Endpoint**: http://api.openweathermap.org/geo/1.0/reverse (za koordinate u naziv).
- **Parametri**:
    - q ili lat i lon: Naziv grada ili koordinate.
    - appid: Vaš API ključ.

Primjer u Pythonu:

```
def get_coordinates(city_name, api_key):
    url = "http://api.openweathermap.org/geo/1.0/direct"
    params = {
        "q": city_name,
        "appid": api_key,
        "limit": 1  # Maksimalan broj rezultata
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        return f"Error: {response.status_code}, {response.json()}"


coordinates = get_coordinates(city, api_key)
print(coordinates)
```

Ključni savjeti

- **Error handling**: Uvijek provjerite status kôd odgovora API-ja.
- **Rate limiting**: OpenWeatherMap ima ograničenja poziva za besplatne planove, pa optimizirajte pozive.
- **Sigurnost API ključa**: Držite svoj API ključ sigurnim (npr. koristite .env datoteku za pohranu).

Napomena:

- 100% se ne jamči da je gore opisana specifikacija funkcionalna, zbog promjena izvornog sustava. Stoga se predlaže da se ista provjeri na službenim stranicama Geocoding API – OpenWeatherMap.