



Univerzitet u Novom Sadu
Prirodno - matematički fakultet,
Departman za fiziku

UPOTREBA PROGRAMSKI JEZIK PYTHON U ANALIZI METEOROLOŠKIH PODATAKA

-MASTER RAD-

Mentor: Prof.dr Ilija Arsenić

Kandidat: Martin Petraš

Novi Sad, 2018.

Sadržaj

1	Korišćeni alati	3
1.1	Python	3
1.1.1	Korišćenje biblioteka	3
1.1.2	Scipy	4
1.1.3	Numpy	4
1.1.4	Pandas	4
1.1.5	Matplotlib	5
1.1.6	Metpy	5
1.2	Django	6
1.2.1	Django arhitektura	6
2	Postupak za instaliranje	8
2.1	Instaliranje Python3 verzije	8
2.2	Podešavanje virtuelnog okruženja	8
2.3	Pokretanje Django projekta	9
2.3.1	POST i GET zahtev	11
2.3.2	Grafički prikaz polja temperature	12
2.3.3	Definisanje URL-a	12
3	Interpolacija	14
3.1	Korišćene metode interpolacije	14
3.1.1	Linearna interpolacija	14
3.1.2	Natural neighbor interpolacija	15
3.1.3	Barnes i Cressman interpolacija	15
4	Carpacclim aplikacija	17
4.1	Godišnje, mesečne, dnevne srednje vrednosti temperature	18
4.2	Vrednosti tačke	19
4.3	Trenutno vreme	20

Zahvalnica:

mami,tati,.....

Uvod

Predmet ovog rada je izrada web aplikacije u Django razvojnom okruženju za analizu srednje temperature na visini od dva metra od površine tla. Web aplikacija koristi podatke iz *Carpatoclim* projekta, projekat koji se bavio prikupljanjem meteoroloških podataka u vremenskom periodu od 1961-2010 godine. Obuhvaćena oblast, kao što i samo ime projekta kaže, je oblast pružanja planinskog lanca Karpatskih planina. Aplikacija će korisniku omogućiti prikaz podataka u vizuelnom grafičkom obliku za navedeni vremenski interval. Kako bi ubrzali proces grafičkog crtanja, podatke o srednjoj temperaturi smo upakovali u SQLite data bazu, koja je podrazumevana data baza za Django programski okvir¹. Aplikacija omogućava informativni prikaz za zadate koordinate tačke, u kojoj državi odnosno altitudi se nalazi. Rad će biti podeljen na četiri dela. Prvi deo se bavi kratkim opisom korišćenih paketa. Drugi deo se bavi postupkom instaliranja i podešavanja okruženja, konfigurisanjem Django projekta. Treći deo opisuje primenjen metod interpolacije dok se četvrti deo bavi samom aplikacijom.

¹Programski okvir ili na engleskom framework predstavlja univerzalano softversko okruženje koje može biti korišćeno iznova i koje omogućava određenu funkcionalnost kao deo veće softverske platforme da olakša razvoj aplikacija, proizvoda i rešenja

1 Korišćeni alati

1.1 Python

Python je dinamički i objektno orijentisan programski jezik. Spada u interpreterske programske jezike visokog nivoa. Nastao je krajem devedesetih godina prošlog veka i njegov autor je Gvido van Rosum. Broj funkcija u samom jeziku je skroman, pa zahteva relativno malo uloženog vremena i napora kako bi se napravio prvi programi. Pythonova sintaksa je dizajnirana da bude čitljiva i jednostavna, što ga čini idealnim nastavnim jezikom i omogućava početnicima brzo napredovanje. Programeri provode više vremena razmišljajući o problemu koji pokušavaju da reše, a manje vremena razmišljaju o kompleksnosti jezika. Python se može izvoditi na svim važnijim operativnim sistemima (Windows, Linux, OS X). Za pokretanje Python programa potreban je Python interpreter, koji je pisan u programskom jeziku C. Najosnovnije korišćenje Pythona je kao jezik skriptovanja i automatizacije, koristi se za nauku o podacima i mašinsko učenje, za web usluge, metaprogramiranje. Njegova popularnost, sem čiste i pregledne sintakse, ogleda se i u velikom broju standardnih biblioteka koje su najčešće pisane u C i samom Pythonu jeziku. Kao nedostatak ovog programskog jezika istakli bi njegovu brzinu, međutim brzina kojom se može napisati funkcionalan program je daleko brži nego u nekom drugom jeziku.

1.1.1 Korišćenje biblioteka

Mnoge Python funkcije su sadržane u specijalizovanim bibliotekama, tzv. modulima. Učitavanje modula se postiže naredbom *import*. Python je prvo svoju popularnost stekao pri izradi web aplikacija, međutim usavršavanjem podrške za dodtane module otvorilo je vrata Pythonu i u drugim oblastima programiranja. Moduli poput *numpy* i *pandas*, koji se koriste za analizu i vizuelno obradu podataka, su Python svrstali uz rame sa ostalim kako komercijalnim programima tako i sa programima otvorenog koda kao što su R, MATLAB, SAS, Stata. Kada uzmemo u obzir da Python spada u besplatni programski jezik, njegova popularnost nije slučajnost. Jedan deo ove popularnosti u obradi podataka pripada i laka integracija sa C, C++ i FORTRAN kodom. Mogućnosti koje pružaju biblioteke je moguće izvesti koristeći isključivo samo Pythonov kod. Prednost biblioteka je u efikasnosti i lakoći kojim se postižu isti rezultati, uz manje utrošenog vremena. Jedna od značajnih biblioteka otvorenog koda je i SciKit. Ona sadrži različite klase objekata za klasifikaciju, algoritme mašinskog učenja (engl. *Machine Learning*). U SciKit paketu se nalaze NumPy i SciPy biblioteke koje sadrže klase za razne standardne matematičke i numeričke funkcije.

1.1.2 Scipy

Scipy (engl. *Scientific Python*) proširuje funkcionalnost Numpy-a sa značajnom zbirkom algoritama za Fourijeovu transformaciju, regresiju i druge matematičke tehnike. Pored modula za interpolaciju, koji je nama značajan, sadrži još neke module poput :

- `scipy.integrate` - integracija funkcija
- `scipy.special` - specijalne funkcije
- `scipy.optimize` - optimizacija

1.1.3 Numpy

Numpy (*Numerical Python*) predstavlja fundamentalnu open source Python biblioteku kada se u obzir uzmu numerički proračuni. Sadrži matematičke funkcije zadužene za operaciju na podacima, koje se veoma brzo i efikasno izvršavaju. Poziva se komandom :

```
import numpy as np
```

Za numeričke proračune, formirani nizovi korišćenjem Numpy su dosta efikasniji prilikom njihove manipulacije nego bilo koja druga izgrađena struktura unutar Pythona. Biblioteka je pisana u C-u i Fortranu.

1.1.4 Pandas

Pandas je dizajniran kako bi se ubrzao rad sa strukturiranim ili tabelarnim podacima i kao takav, postala jako moćna i produktivna alatka za analizu podataka. Ono što krasi ovu biblioteku je i velika podrška od strane zajednice, njen aktivan razvoj, odličan rad sa ostalim bibliotekama. Neke od mogućnosti koje pruža ova biblioteka su:

- ulaz-izlaz podataka u različitim formatima (csv, txt, SQL...)
- indeksiranje, sortiranje, rangiranje
- filtriranje podataka
- grupisanje
- vizuelizacija

Sačinjenja je od dve strukture podataka, *dataframe* i *series*. *Series* predstavljaju jednodimenzioni objekat sačinjen od tabele sa vrednostima i njihovim indeksima, dok *dataframe* takođe predstavlja tabelarnu strukturu definisanu u dve ili više dimenzija. Najčešće korišćenja skraćenjica za *pandas* je *pd*, poziva se korišćenjem komande :

```
import pandas as pd
```

Za čitanje baze podataka korišćena je komanda :

```
pd.read_sql_query
```

1.1.5 Matplotlib

Ova biblioteka se koristi za vizuelizaciju podataka. U kombinaciji sa *numpy* i *pandas* čine jako moćne alate, koji se mogu nositi sa komercijalnim i dosta skupljim alatima kao što su *Matlab* i *Mathematica*. Mana ove biblioteke je potreba za pisanjem većeg broja linija koda radi dobijanja naprednije vizuelizacije. Povećanjem potrebe za kvalitetnijom vizuelizacijom, nastale su biblioteke, koje se u većoj ili manjoj meri oslanjaju na *matplotlib*. Jedna od takvih biblioteka je *seaborn*, čiji je fokus na atraktivnijoj izradi statističkih grafika. *Seaborn* se potpuno oslanja na *matplotlib*. Naveo bih još i *bokeh* i *altair* koje nisu zavisne od *matplotlib*, a takođe se koriste za vizuelizaciju podataka.

1.1.6 Metpy

Metpy predstavlja kolekciju alata koje dopunjuju Python module i koriste se za pregled, vizuilizaciju i proračune meteoroloških podataka. Paket je nastao od strane *Unidata* udruženja, koja se bavi izradom softverskih alata u naučne svrhe. Za instaliranje Metpy alata, u terminal kucamo sledeću komandu :

```
pip install metpy
```

Prilikom interpolacije polja temperature korišćena je funkcija *metpy.interpolate*, koja je se bazira na *scipy.interpolate* funkciji, razlika je u brzini interpolacije.

1.2 Django

Nagla ekspanzija interneta praćenja je i povećanjem popularnosti web aplikacija. Google, Facebook, YouTube su samo od nekih popularnih web aplikacija koje čine svakodnevnicu pretraživanja. Web aplikacija je u stvari program, kojim pristupamo pomoću internet pretraživača (browser). Ovakav pristup omogućava veću bezbednost, sigurnost bez potrebe za instaliranjem na operativni sistem. Sve što je potrebno jeste uređaj koji ima pristup internetu. Usled spomenutog naglog rasta popularnosti web aplikacija, javila se potreba za brzim i automatizovanim načinom njihove izrade. Tako je nastao Django, web framework napisan u Pythonu koji služi za izradu web stranica i aplikacija.

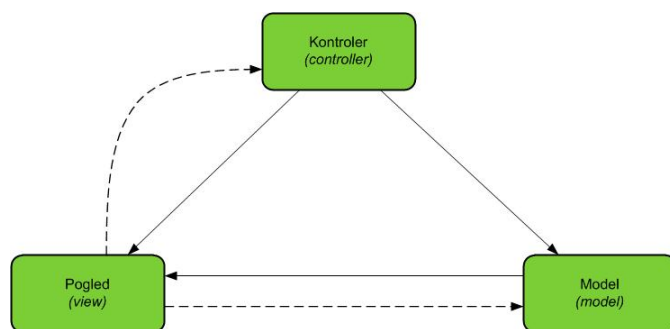
Razvijan je od 2003. godine od strane novinara, koji su zahtevali brzo razvojno okruženje za izradu web stranica. Kao slobodni softver (eng. *open source*) objavljen je 2005. godine, kada je i dobio ime Django po jazz gitaristi Djangu Reinhardt. Od 2008. godine za razvoj Django frameworka zadužena je *Django Software Foundation*. Django je zapravo skup alata i biblioteka koje programerima web aplikacija olakšavaju rad. Dizajniran je kako bi se skratilo vreme potrebno za izradu web aplikacije, držeći se strogih zahteva iskusnih web programera koji su ga stvorili. Web programiranje zahteva ponavljanje isti zadataka kao što su :

- korisnička prava
- registracija korisnika
- konfigurisanje URL-ova
- validacija unesenih podataka
- administracija sajta

1.2.1 Django arhitektura

Kao što je rečeno, Django služi za brzu i jednostavnu izradu web stranica i aplikacija. Ovu osobinu mu omogućava arhitektura MVC (*model-view-controller*) na kojoj se bazira, i koja razdvaja logiku same aplikacije od prikazanog dela aplikacije. Možemo je podeliti na tri dela:

- Model
- View (Pogled)
- Controller(Kontroler)



Slika 1. Koncept modela MVC

Pune linije predstavljaju vezu izmedju navedenih elemenata za razmenu podataka, dok isprekidane linije predstavljaju vezu kod koje neki drugi objekt vrši tu funkciju. Django koristi drugačije nazive elemenata modela MVC. Tako *view* predstavlja kontroler a *template* (šablon) predstavlja *view* iz MVC modela. Implementaciju MVC modela kod Django frameworka možem nazvati MTV (*model-template-view*). Ova promena je objašnjena na takav način, što se kod Django pogledom predstavljaju podaci koji se prezentuju, dok šablon predstavlja način na koji se podaci prikazuju.

Model je zadužen za komunikaciju, odnosno opis baze podataka sa kojom je Django povezan. Model je u Django okruženju klasa (engl. *class*) i određuje varijable i metode pridružene određenim tipovima podataka. Pridružene varijable predstavljaju kolone u tablici dok metode definišu relacije između varijabli. Ulog modela je da dohvati tražene podatke i prosledi ih ka pogledu. Model nema informacija o šablonima i funkcijama izvedenih u pogledima.

Uloga pogleda je prikazivanje podataka dohvaćenih iz baze podataka u internet pretraživaču. Priliko izrade web stranice, svaka kreirana aplikacija ima svoj pogled. Pogled u stvari sačinjavaju funkcije u Python programskom jeziku. Ove funkcije imaju ulogu prenosnika podataka koji će biti prikazani.

Pregled je HTML stranica s dodatnim strukturama koje omogućavaju prikaz podataka koji su poslani od pogleda. Uloga pregleda je da sadržaj poslat od strane pogleda ugradi u HTML kod, koji će se prikazati u internet pretraživaču.

2 Postupak za instaliranje

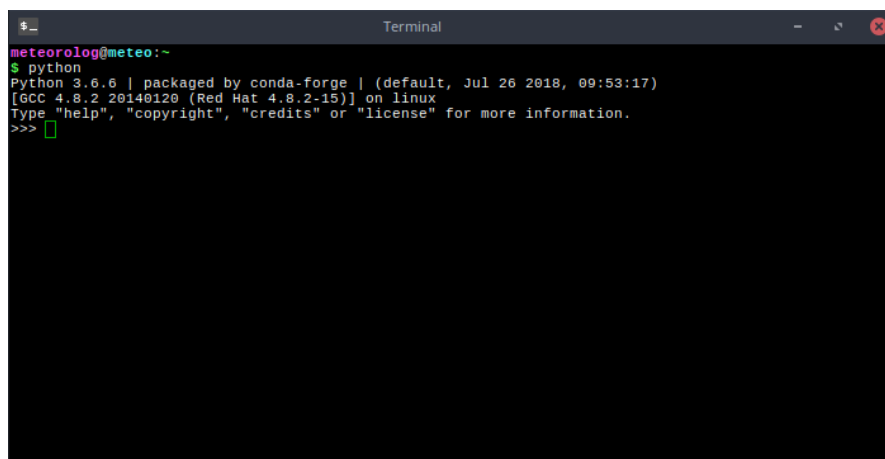
2.1 Instaliranje Python3 verzije

Django je pisan u Python-u i kako bi ga koristili potrebno je imati instaliran Python. Python kao programski jezik postoji za sve tri popularne systemske platforme Windows, Linux i OSX. Ovde će biti prikazan postupak instaliranja potrebnih paketa za rad pod Linux operativnim sistemom. Gotov sve distribucije linuxa dolaze sa instaliranim Python-om, razlika može biti u verziji. Postoje dve verzije, starija 2.7 verzija koja je podržana do kraja 2020. godine, i novija verzija 3.6. U radu je korišćena verzija 3.5. Proveru verzije radimo tako što otvorimo terminal i kucamo sledeću komandu :

```
python3 --version
```

Ukoliko je Python verzije 3.6 instaliran, pojavi se sledeće

```
Python 3.6.6
```



Slika 2. Python verzije 3.6 pokrenut iz terminala

Ukoliko nemate instaliran Python verzije 3.6, možete ga instalirati :

```
sudo apt install python3.6
```

2.2 Podešavanje virtuelnog okruženja

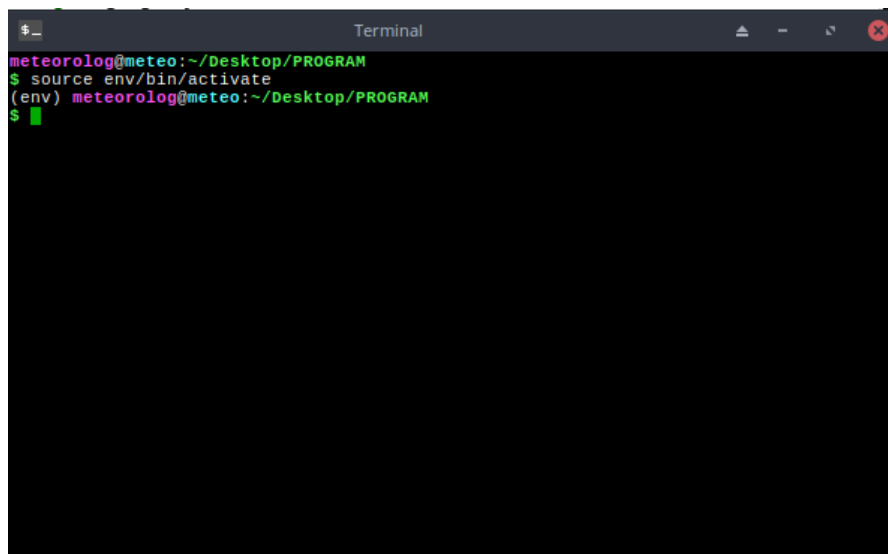
Pre nego što instaliramo Django pokazaćemo kako instalirati veoma zahvalnu alatku, koja nam omogućava održavanje čistoće paketa na sistemu. Virtuelno okruženje (engl. *virtual environment*) izoluje projekte jedan od drugog, držeći pakete vezane za svaki projekat zasebno. Kako bi kreirali virtuelno okruženje zvano *env*, u direktorijum u kome smo započeli projekat, u terminal kucamo :

```
python3 -m virtualenv env
```

Kada ovo odradimo, u dokumentu se pojavi dokument *env* u koji će se instalirati svi paketi koje u buduće budemo koristili. Kada ovo sve odradimo, ostalo nam je još da ovo virtuelno okruženje aktiviramo. Aktiviranje virtuelno okruženja radimo na sledeći način :

```
source env/bin/activate
```

nako čega se u terminalu pojavljuje nastavak (*env*). Ovaj nastavak nam potvrđuje da je virtuelno okruženje aktivirano. Kako bi zatvorili virtuelno okruženje, u terminalu kucamo komandu *deactivate*.



Slika 3. Postupak aktiviranje virtuelnog okruženja

2.3 Pokretanje Django projekta

Kada smo podesili virtuelno okruženje možemo instalirati Django. Instaliranje radimo pomoću *pip* menadžera. Prvo uradimo nadogradnju *pip* paketa, kako bi imali poslednju verziju. To radimo na sledeći način :

```
python3 -m pip install --upgrade pip
```

Kada smo uradili nadogradnju *pip*-a, Django instaliramo :

```
pip install django
```

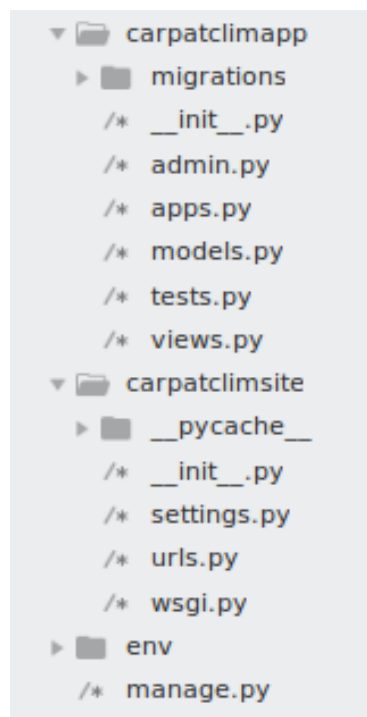
Kada ovo odradimo možemo uspešno pokrenuti naš prvi Django projekat. Izrada projekta podrazumeva pokretanje Django skripte koja će izgraditi kostur potrebnih datoteka. Kucamo sledeću komandu :

```
django-admin startproject carpatclimsite .
```

i u stvorenom direktorijum dobijamo strukturu kao sa slike 4. U strukturi django projekta *manage.py* skripta predstavlja upravitelja projekta pomoću kojeg se pokreće web server. Skripta *settings.py* služi za konfigurisanje našeg sajta. *urls.py* sadrži šablone (preglede) koje koristimo. Kako bi pokrenuli web server u terminal kucamo :

```
python manage.py runserver
```

nako čega u internet pretraživač unesemo adresu *http://127.0.0.1:8000/*. Ako je sve urađeno kako treba u internet pretraživaču bi trebalo da se pojavi čestitka o uspešnom pokretanju web sajta. Nakon uspešnog pokretanja sajta, sledeća stavka



Slika 4. Struktura carpatclimsite projekta

je stvaranje aplikacije. Kako bi stvorili aplikaciju u datom projektu, u terminal upisujemo sledeću komandu :

```
python manage.py startapp carpatclimapp
```

Nakon ove komande u projektu se pojavi dokument carpatclimapp sa Python skriptama. U Python skripti *views.py* (pogled) skripta sadrži funkcije zadužene za funkcionisanje web aplikacije. Pomoću pogleda definišemo kontak sa bazom podataka, vršimo razmenu podataka. Svaki pogled se mora definisati u *urls.py* skripti

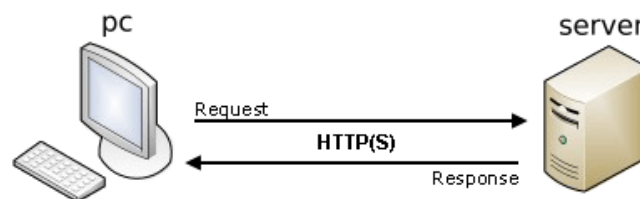
koju stvorimo u ovom dokumentu. Sledeća bitna skripta je `forms.py`, pomoću koje vršimo definisanje godine, meseca, dana za pružimanje podataka iz baze podataka. Ovu skriptu kao i `urls.py` moramo stvoriti u `carpatclimapp` dokumentu. U `forms.py` skripti definišemo klase za :

- Godišenje (`CronFormYearly`)
- Mesečne (`CronFormMonthly`)
- Dnevne (`CronFormDaily`)

vrednosti srednjih temperatura. U zavisnosti od izbora željenog vremenskog intervala, pojavi se odgovarajuća forma.

2.3.1 POST i GET zahtev

Kako bi što bolje razumeli interakciju između forme i pogleda, njihovo funkcionalnost ćemo objasniti uz pomoć POST i GET zahteva. Koristeći formu stvaramo podatke koje uz pomoć POST metoda šaljemo ka serveru u vidu zahteva (*Request*), nakon čega server vraća zahtevane podatke u vidu odgovora (*Response*). Forma stvara zahtev koje podatke želimo primiti kada se stisne dugme "Submit". GET metoda se koristi za dostavljanje podataka u niz i stvaranje URL adrese. Uloga pogleda je ostvarivanje kontakta sa serverom i stvaranje prikaza podataka



Slika 5. Razmena podataka između korisnika i servera

kada oni stignu sa servera. Prikaz se ostvaruje uz pomoć šablona (*templates*), koje se moraju stvoriti u dokumentu date aplikacije. Za svaku aplikaciju unutar Django projekta se stvaraju zasebni šabloni čija uloga je vizuelni u sklopu internet pretraživača. U te svrhe pored HTML i CSS korišćen je i Bootstrap.

Pogledaćemo sada kako izgleda pogled za godišnje vrednosti temperature.

```
def yearly(request):
    if request.method == "POST":
        form = CronFormYearly(request.POST)
        if form.is_valid():
            data = form.cleaned_data
```

```

        year = data.get('year')
        date_path = year
        return redirect('/%s/' % (date_path))
    else:
        form = CronFormYearly()
        active_yearly = True
        return render(request, 'carpatclimapp/home.html', {'form': form, '
            active_yearly': active_yearly})

```

Ovaj pogled, koji definiše upit za godišnje podatke, nam kaže sledeće, ako je aktiviran POST metod, uzmi formu u obliku *CronFormYearly*, ako je forma važeća uzmi podatke za definisanu godinu. Ako POST zahtev nije aktiviran, prikaži formu čija struktura je definisana u *home.html* datoteci, koja je deo bloka *base.html*.

2.3.2 Grafički prikaz polja temperature

Kada smo poslali zahtev za podatake, i kada su ovi podaci uspešno prosleđeni, sa njima želim nešto dobiti. U našem slučaju mi želimo dobiti grafički prikaz polja srednje temperature u sklopu Karpata za zadatu godinu. Kako bi dobili grafički prikaz moramo stvoriti funkciju koja će ovaj zadatak uraditi za nas. U te svrhe je zadužena funkcija *carpatclim_y_figure*.

```

def carpatclim_y_figure(request, year):

    map = create_map(year, month=None, day=None)
    buffer = BytesIO()
    canvas = FigureCanvas(map)
    canvas.print_png(buffer)

    response = HttpResponse(buffer.getvalue(), content_type='image/png')
    response['Content-Length'] = str(len(response.content))
    return response

```

Uzimajući vrednosti za godinu, koristeći funkciju *create_map* dobija se grafički prikaz polja temperature.

2.3.3 Definisane URL-a

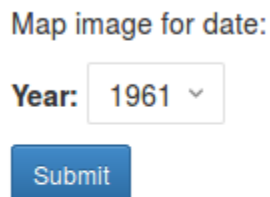
Kako bi pokrenuli funkciju *yearly* i *carpatclim_y_figure*, koje se nalaze u sklopu pogleda, moramo definisati njihov url. To se radi tako što unutar *carpatclimapp* stvorimo skriptu *urls.py*, unutra koje definišemo urlpatterns za funkcije:

```

path('yearly/', views.yearly, name='yearly'),
path('<int:year>/figure.png', views.carpatclim_y_figure, name='
    carpatclim_y_figure')

```

Ovim pristup postizemo da za svaku funkciju postoji zadana adresa. Recimo, korišćenjem `yearly/` u adresi prikazuje se forma u obliku kao na slici 6 :



Map image for date:

Year: 1961 ▾

Submit

Slika 6. Izgled forme za godišnje vrednosti

U ovom primeru je prikazan url za godišnje vrednosti. Za mesečne i dnevne vrednosti adrese se razlikuju po tome što se osim godine, u adresi pojavljuju vrednosti za mesec odnosno dan.

3 Interpolacija

Podaci sa meteoroloških stanica su veoma značajni za prognozu vremena. U prognozi vremena ovi podaci se koriste kao početni uslovi za jednačine dok se kod analize koriste za vizuelni prezentaciju polja promenljivih. Cilj je što vernije predstaviti stanje varijabli na određenom nivou i mestu. Ako bi postojala merenja u svim tačkama prostora i uz to dovoljno tačna, problem tačnosti ne bi postojao. Međutim, broj raspoloživih stanica koje šalju podatke i dalje nije dovoljan jer na nekim mestima, poput okeana ili neprisupačnim predela, ne postoji mogućnost postavljanja mernih uređaja. Jedan način rešenje ovog problema su satelitska merenja a drugi pristupačniji je uz pomoć interpolacije.

3.1 Korišćene metode interpolacije

Interpolacija predstavlja metod procene vrednosti varijable na mestima gde oni nisu određeni, koristeći poznate vrednosti iz lokalnih lokacija. Interpolacija je moćna metoda za analizu podataka kod kojih su merenja rasuta. Problem interpolacije je njena implementacija. Za nekoliko poznatih primera, koristeći različite metode interpolacije dobićemo različite rezultate. Zbog toga, aplikacija će omogućivati prikaz četiri metoda interpolacije :

- Linearna interpolacija
- Interpolacija najbližeg suseda
- Barnes interpolacija
- Cressman interpolacija

Metode interpolacije možemo podeliti u dve kategorije u zavisnosti koje tačke su uzete za definisanje interpolacije u određenoj tački. Ove dve kategorije su globalna i lokalna interpolacija.

3.1.1 Linearna interpolacija

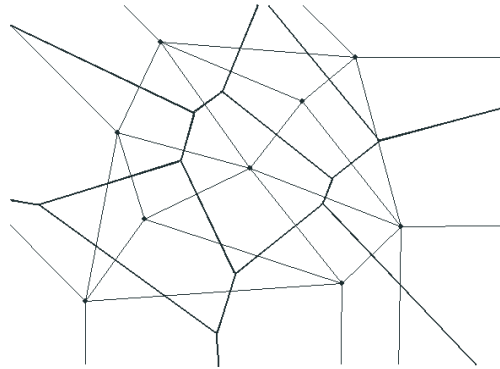
U pitanju je najjednostavnija i najmanje zahtevna metoda interpolacije. Interpolant se dobija uzimajući dve najbliže tačke, recimo na pozicijama (x_1, y_1) i (x_2, y_2) na sledeći način :

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{(x_2 - x_1)} \quad (1)$$

Mana ove metode je njena preciznost. Greška raste proporcijalno kvadratu udaljenosti između tačaka.

3.1.2 Natural neighbor interpolacija

Postoje razne implementacije natural neighbor interpolacija. Ovde će biti obrađena lokalna metoda koja se bazira na Voronojevom dijagramu i Deloneovoj trijangulaciji. Pokazala se kao vremenski malo zahtevna metoda, jer se bazira na ideji ponovnog korišćenja rezultata tokom proračuna u što većem obimu. Prvo ćemo definisati teoriju Voronoj dijagrama i Delonove trijangulacije. Za dati skup tačaka u ravni, koje ćemo u daljem tekstu nazvati centrima, konstrukcija Voronoj dijagrama podrazumeva podelu ravni na ćelije, tako da svaka ćelija sadrži jedan centar. Deloneov graf nad datim skupom tačaka u ravni je dualan graf Voronoj dijagramu. Drugim rečima, čvorovi ovog grafa su centri, a grane povezuju čvorove, koji odgovaraju centrima susednih Voronoj ćelija. Na slici 5 su prikazani Voronoj dijagram (podebljane linije) i Deloneova triangulacija (obične linije) skupa od devet centara. Ako dve Voronoj ćelije dele zajedničke ivice onda za njih kažemo

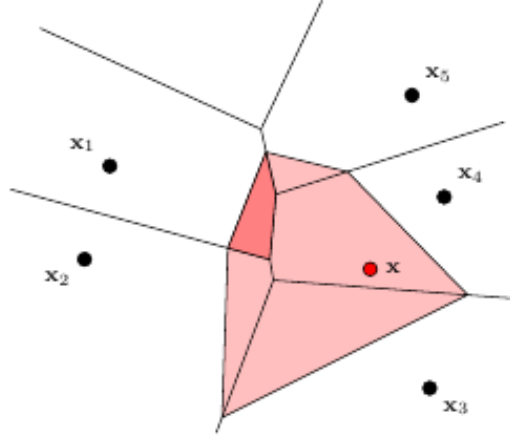


Slika 5. Voronoj dijagram i Deloneova triangulacija

da su prirodni susedi(natural neighbor). Kako bi se definisali susedi za određene tačke interpolacije x , prvo formiramo Voronoj ćeliju u kojoj se ova tačka nalazi. Formiranjem ćelije definišemo zapreminu $V(x)$, koja će imati nove susede. Na slici 6 možemo videti da su za tačku x prirodni susedi x_1, x_2, x_3, x_4, x_5 , odnosno da Voronoj ćelija $V(x)$ deli ivice sa $V(x_1), V(x_2), V(x_3), V(x_4), V(x_5)$. Vrednost interpolacije u tački x dobijamo tako što usrednjimo vrednosti ovih definisanih okolnih, susednih tačaka.

3.1.3 Barnes i Cressman interpolacija

Barnes i Cressman spadaju u grupu sukcesivno korekcionih metoda, kod kojih se polje varijabli modifikuje koristeći osmotrene podatke iterativnim postupkom. Metod se zasniva na prolazu kroz svaku tačku mreže, ažuriranj varijabli za svaku



Slika 6. Najbliži susedi za tačku x

od njih na osnovu osmotrenih podataka koje obuhvataju tu tačku mreže. Nakon prvog prolaza kroz zadato područje, radi se drugi prolaz, modifikujući polje svake tačke mreže u odnosu na okolne osmotrene tačke. Ove interakcije se rade sve dok razlike između dve iteracije ne budu manje od zadanog praga. Polje mreže u tački i se ažurira na sledeći način:

$$f_i^{m+1} = f_i^m + \frac{\sum_{k=1}^K w_{ik}^m (O_k - f_i^m)}{\sum_{k=1}^K w_{ik}^m} \quad (2)$$

gde f_i^m je vrednost varijable u i -toj tački mreže za m iteraciju. O_k je vrednost k -te osmotrene vrednosti u okolini tačke mreže, a w_{ik}^m je težinska funkcija i određuje se na sledeći način:

$$w_{ik}^m = \exp\left(\frac{-r_{ik}^2}{2R_m^2}\right) \quad (3)$$

Za Cressman šemu težinska funkcija zavisi od radijusa delovanja R_m kao:

$$w_{ik}^m = \frac{R_m^2 - r_{ik}^2}{R_m^2 + r_{ik}^2} \quad \text{za} \quad r_{ik}^2 \leq R_m^2 \quad (4)$$

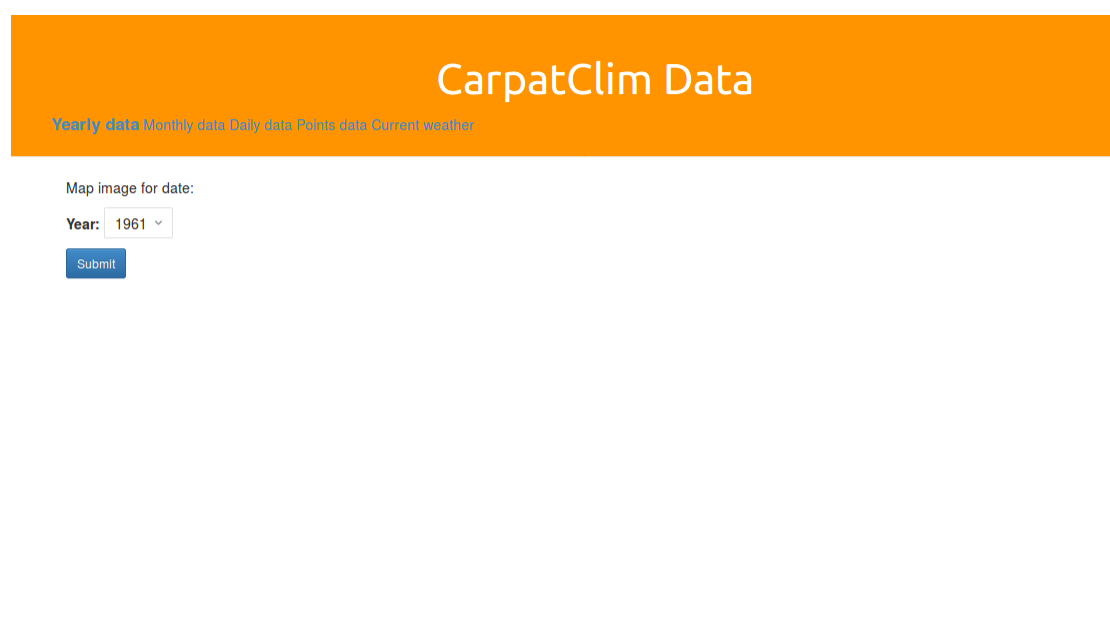
$$w_{ik}^m = 0 \quad \text{za} \quad r_{ik}^2 > R_m^2 \quad (5)$$

gde je r_{ik}^2 dužina između osmotrenih vrednosti i tačke mreže. Kod ove metode, osmotrene vrednosti izvan radijusa dešavanja se ne uzimaju u ažuriranje vrednosti polja.

4 Carpatclim aplikacija

Na slici 7. je prikazan izgled aplikacije. Idući sa leve prema desnoj strani, u gornjem levom delu aplikacije mogućnosti koje aplikacija omogućava su sledeće:

- Yearly data (godišnje vrednosti temeperature)
- Monthly data (mesečne vrednosti temperature)
- Daily data (dnevne vrednosti temperature)
- Points data (vrednosti tačke)
- Current weather (trenutno vreme)



Slika 7. Izgled aplikacije

Prve tri opcije daju grafički prikaz srednjih temperatura za određeni interval, *godišnje (yearl)* srednje temperature za izabranu godinu dok ostale dve opcije daju srednje vrednosti za *mesečne (monthly)*, odnosno *dnevne (daily)* vrednosti. *Vrednosti tačke (Points data)* opcija daje informacije o izabranoj tački, izborom vrednosti longitude i latitude u mogućnosti smo videti u kojoj zemlji se nalazi tačka i na kojoj je visini. Opcija *trenutno vreme (current weather)* daje trenutne vremenske podatke za izabrani grad.

4.1 Godišnje, mesečne, dnevne srednje vrednosti temperature

Prilikom izbora jedno od ove tri opcije, pojavljuje se forma za izbor godine, meseca ili dana u zavisnosti da li želimo godišnje, dnevne ili mesečne vrednosti. Na slici 8. je prikazana forma za srednje dnevne vrednosti temperature.

Map image for date:

Year: 1961 ▾

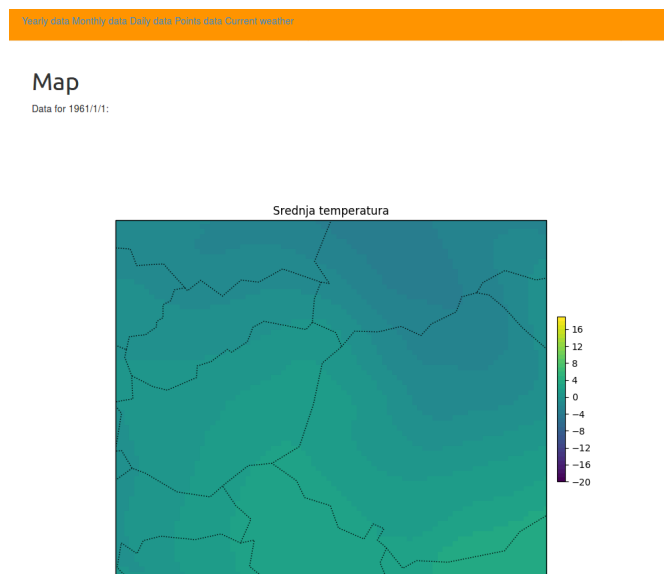
Month: 1 ▾

Day: 1 ▾

Submit

Slika 8. Forma za izbor godine, meseca, dana za prikaz srednje dnevne temperature

Kao što sa slike 8. vidimo, postoji izbor za godinu koje se kreću u rasponu od 1961-2010, zatim izbor meseca i dana. Kada izaberemo željeni vremenski interval, aktiviranjem dugmeta *submit* dobijamo grafički prikaz.



Slika 9. Grafički prikaz polja srednje dnevne temperature za izabrani period 1961/1/1

4.2 Vrednosti tačke

Sledeća opcija *vrednost tačke(points data)* na daje opis tačke mreže za definisanu longitudu i latitudu.

Data for coordinates:

Lat:

Lon:

Slika 10. Izbor latitude i longitude za tačku mreže

Moguće vrednosti za latitudu su od 44-50 dok su za longitudu od 17-27. Izborom neke vrednosti za latitudu i longitudu dobijaju se podaci kao sa slike 11:

Country: 1	Altitude: 183	Latitude: 48.0	Longitude: 19.1
Country	Number		
Hungary	1		
Serbia	2		
Romania	3		
Ukraine	4		
Slovakia	5		
Poland	6		
Czech Republic	7		
Croatia	8		

Slika 11. Podaci za latitudu 48.0 i longitudu 19.1

Sa slike 11. vidimo da se tačka sa datim kordinatam nalazi u Mađarskoj, i visina na kojoj su merene vrednosti meteoroloških promenljivih je 183.

4.3 Trenutno vreme

Ova aplikacija nam daje informacije o trenutnim vremenskim parametrima u izabranim gradovima. Kako bi izabrali željenu lokaciju, u polje ” *City Name*” upišemo ime grada.

Add City



Novi Sad

lat: 45.26 lon: 19.85

humidity: 93 %

pressure: 1011 mb

temp: 5.52° C

moderate rain



Belgrade

lat: 44.82 lon: 20.46

humidity: 87 %

pressure: 1010 mb

temp: 5.49° C

light rain

Slika 12. Trenutni vremenski parametri u izabranim gradovima