



Univerzitet u Novom Sadu

Prirodno - matematički fakultet,
Departman za fiziku

# Upotreba Python programskog okruženja u obradi ”CARPATCLIM” klimatskih podataka

-MASTER RAD-

Mentor: Prof. dr Ilija Arsenić

Kandidat: Martin Petraš

Novi Sad, 2019.

# Sadržaj

# *Zahvalnica:*

*Zahvaljujem se svom mentoru, profesoru dr Iliji Arseniću na brojnim stručnim savetima, strpljenju i podršci tokom izrade master rada.*

*Zahvaljujem se kolegama sa fakulteta koji su mi pružali moralnu podršku, ne samo za vreme pisanja master rada, nego i za vreme studiranja, i prijateljima na razumevanju i podršci.*

*Najviše se zahvaljujem svojoj porodici na strpljenju, razumevanju i moralnoj podršci tokom izrade master rada i za vreme mog studiranja.*

# 1 Uvod

Veb aplikacija je softver čiji kod se nalazi na serveru i korisnik ka njoj pristupa uz pomoć internet pretraživača koristeći internet. Danas je internet neizbežan deo našeg života a time i veb aplikacije koje čine njegov sastavni deo. Uloga veb aplikacije je da prepozna šta korisnik želi i da mu isporuči odgovarajući rezultat u skladu sa zadatim kriterijumima. Osnovni princip rada veb aplikacije se može podeliti u više koraka. U prvom koraku korisnik unosi željenu adresu, koja se onda preko interneta šalje do servera na kojem se nalazi veb aplikacija. Kada server primi zahtev, veb aplikacija analizira upućeni zahtev i vraća odgovor. Odgovor može biti u vidu obrade podataka, komunikacija sa ostalim serverima ili preusmeravanje na drugi veb server. Rezultat odgovora veb aplikacije može biti u vidu HTML stranice, slike, video sadržaja ili nekog drugog dokumenta. Ovakav pristup omogućava veću bezbednost, bez potrebe za instaliranjem na operativni sistem, nove verzije aplikacije su automacki dostupne. Sve što je potrebno korisniku aplikacije, kako bi pristupio veb aplikaciji, jeste uređaj koji ima pristup internetu.

Predmet ovog rada je veb aplikacija razvijana u Django razvojnom okruženju koristeći Python programski jezik. Razvijena aplikacija može pomoći pri analizi klimatskih podataka za oblast Karpatskih planina. Prva mogućnost koju razvijena aplikacija pruža jeste grafički prikaz interpolisanih vrednosti polja promenljivih. Aplikacija koristi tri interpolacione metode, pri čemu je korisniku omogućen izbor koju će koristiti. Podaci za vrednosti promenljivih, koje aplikacija koristi tokom interpolacije, uzeti su sa *Carpatclim* projekta, projekat koji se bavio prikupljanjem meteoroloških podataka u vremenskom periodu od 1961. godine do 2010. godine. Obuhvaćena oblast, kao što i samo ime projekta kaže, je oblast pružanja planinskog lanca Karpatskih planina. Aplikacija omogućava i preuzimanja podataka sa mernih stanica, kao i interpolisane vrednosti za izabranu tačku definisanjem geografske širine, geografske dužine i vremenskog perioda. Zadnja mogućnost razvijene aplikacije je prikaz vremenske prognoze. Korisnik proizvoljno upiše ime grada, nako čega se prikazuje trenutna vremenska situacija za izabrani grad.

Rad se sastoji od četiri poglavlja i dva dodatka. Prvo poglavlje se bavi kratkim

opisom korišćenih alata unutar projekta. Drugi deo se bavi postupkom instaliranja i podešavanja Django okruženja. Treće poglavlje se bavi opisom primenjenih metoda interpolacije dok se četvrti deo bavi samom aplikacijom, izgledom i načinom rada. U dodatku A će biti prikazan postupak preuzimanja koda, kao i postupak instaliranja dodatnih paketa, potrebnih za funkcionisanje aplikacije. Dodatak B se bavi procesom preuzimanja podataka sa *Carpatclim* sajta, i opisom postupka konstrukcije baze podataka.

# 2 Korišćeni alati

## 2.1 Python

Python je dinamički i objektno orijentisan programski jezik. Spada u interpreterske programske jezike visokog nivoa. Nastao je krajem devedsetih godina prošlog veka i njegov autor je Guido van Rossum. Broj funkcija u samom jeziku je skroman, pa zahteva relativno malo uloženog vremena i napora kako bi se napravio prvi program. Python-ova 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 izvršavati na svim važnijim operativnim sistemima (Windows, Linux, OS X).

Za pokretanje Python programa potreban je interpreter, kod Python-a je to CPython koji je napisan u programskom jeziku C i Python-u. Interpreter je programski izvršilac, koji pokreće izvorni kod pisan u nekom od programskom jeziku. Postoje dva pristupa prilikom pokretanja-izvršenja izvornog koda, prvi je interpreterski pristup a drugi je kompajlerski pristup. Interpreteri i kompajleri su slični po strukturi, njihova glavna razlika je ta što interpreteri direktno izvršavaju instrukcije napisane u izvornom kodu dok kompajleri prvo prevode ceo izvorni kod u mašinski kod, nakon čega se on izvršava. Kompajleri se koriste u programskim jezicima kao što su C, C++, C#. Interpreteri su zastupljeni u programskim jezicima kao što su PHP, Ruby, Python. Ključne razlike između interpretera i kompajlera se ogledaju u sledećem:

- interpreteru je potrebno manje vremena za analizu izvornog koda (programa), ali je kompletno izvršenje sporije. Kompajleru je potrebno više vremena za analizu izvornog koda, ali je kompletno prevođenje znatno brže.
- interpreter prevodi izvorni kod po delovima, dok kompajler brzo analizira ceo kod nako čega ga u kompletu prevodi u mašinski kod i izvršava.
- interpreter prevodi izvorni kod sve dok ne naiđe na grešku nakon čega se zaustavlja, dok kompajler generiše poruku o grešci nakon skeniranja celog izvornog koda.

Najosnovnije korišćenje Pythona je kao jezik skriptovanja i automatizacije, koristi se za nauku o podacima i mašinsko učenje, za veb 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 veća nego u nekom drugom jeziku.

### 2.1.1 Korišćenje biblioteka

Ovde će biti prikazane osnovne biblioteke korišćene u aplikaciji. Neke od ovih biblioteka zahtevaju dodatne pakete, bez koji ne bi radili. Proces instaliranja dodatnih paketa će biti prikazan u dodatku A.

Mnoge Python funkcije su sadržane u specijalizovanim bibliotekama, tzv. modulima. Učitavanje modula se postiže naredbom *import*. Python je svoju popularnost stekao pri izradi web aplikacija, međutim usavršavanjem podrške za dodtane module otvorilo je vrata Python-u i u drugim oblastima programiranja. Moduli poput *numpy* i *pandas*, koji se koriste za analizu i vizuelnu obradu podataka, su Python svrstali rame 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 potiče i laka integracija sa C, C++ i FORTRAN kodom. Mogućnosti koje pružaju biblioteke je moguće upotrebiti koristeći isključivo 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 kasifikaciju, algoritme mašinskog učenja (engl. *Machine Learning*).

### 2.1.2 Numpy

*Numpy* (*Numerical Python*) predstavlja fundamentalnu open source Python biblioteku kada se u obzir uzmu numerički proračuni. Sadrži matematičke funkcije zadužene za operacije 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 Python-a. Biblioteka je pisana u C-u i Fortranu.

### 2.1.3 Pandas

*Pandas* je dizajniran kako bi se ubrzao rad sa strukturiranim ili tabelarnim podacima i kao takav, postao je 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
```

### 2.1.4 Matplotlib

Ova biblioteka se koristi za vizuelizaciju podataka. U kombinaciji sa *numpy* i *pandas* čini 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.

### 2.1.5 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 promenljivih korišćen je modul *metpy.interpolate*, iz kojeg je preuzeta funkcija *interpolate_to_grid* koristeći sledeću naredbu:

```
from metpy.interpolate import interpolate_to_grid
```

Za interpolisanje u zadatu tačku unutar definisanog prostora, korišćena je funkcija *interpolate_to_point*. Funkcija se preuzima sledećom naredbom:

```
from metpy.interpolate import interpolate_to_point
```

### 2.1.6 Cartopy

*Cartopy* je Python paket za izradu karata u svrhu analize podataka. U sklopu ovo paketa nalazi se više biblioteka za analizu podataka, koje omogućavaju stvaranje kvalitetnih prezentacija mapa. Ključne karakteristike Cartopy paketa su njegove sposobnosti transformacije tačaka, linija, vektora, poligona i slika na date projekcije. Lista projekcija je pozamašna. U projektu je korišćena projekcija *Mercator*. U pitanju je cilindrična kartografska projekcija kod koje su merdijani i paralele ravne linije, postavljene pod pravim uglom jedne u odnosu na druge. Ovakva postavka dovodi povećanja rastojanja između paralela kako se udaljavamo od ekvatora, koja na polovima teži ka beskonačnom. Ovakva postavka ima za posledicu da je Grenland jednake površine kao i Afrika, iako je po površini Afrika 14 puta veća. Aljaska je po površini 5 puta manja od Brazila, međutim na *Meractor* projekciji oni izgledaju podjednako. Takođe, prikazivanje polova nije moguća. Zbog ove distorzije, ova projekcija se najčešće koristi samo za projekcije u blizini ekvatora. Naredba za instaliranje paketa je :

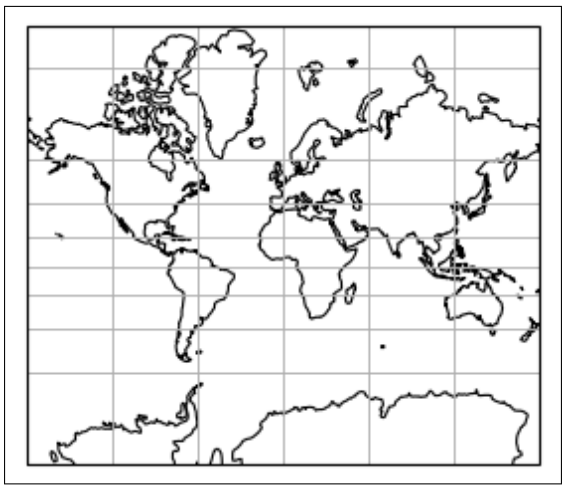
```
pip install cartopy
```

Pozivanje *Cartopy* paketa se izvodi sledećom komandom:

```
import cartopy.crs as ccrs
```

Paket sadrži i dodatne mogućnosti kao što su prikaz granica, reka, jezera na kartama. Takođe je omogućen prikaz merdijana i paralela. Ove dodatke pozivamo koristeći sledeće naredbe:

```
import cartopy.feature as cfeature
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER
from cartopy.mpl.gridliner import LATITUDE_FORMATTER
```



Slika 1: Mercator projekcija

## 2.2 Django

Nagla ekspanzija interneta praćenja je i povećanjem popularnosti veb aplikacija. Google, Facebook, YouTube su samo od nekih popularnih veb aplikacija koje čine svakodnevnicu pretraživanja. Veb aplikacija je u stvari program, kojem pristupamo pomoću internet pretraživača (engl. *browser*). Usled pomenutog naglog rasta popularnosti veb aplikacija, javila se potreba za brzim i automatizovanim načinom njihove izrade. Tako je nastao Django, napisan u Python programskom jeziku, koji služi za izradu veb stranica i aplikacija.

Django je razvijan od 2003. godine od strane grupe novinara, koji su zahtevali brzo razvojno okruženje za izradu veb stranica. Kao slobodni softver (eng. *open source*) objavljen je 2005. godine, kada je i dobio ime Django po jazz gitaristi Djangu Reinhardtu. Od 2008. godine za razvoj Django programskog okvira[1] (engl. *framework*) zadužena je *Django Software Foundation*. Django je zapravo predstavlja skup alata i biblioteka, dizajniran je kako bi se skratilo vreme potrebno za izradu veb aplikacije. Veb programiranje zahteva ponavljanje isti zadataka kao što su :

- korisnička prava
- registracija korisnika
- konfigurisanje URL[2]-a
- validacija unesenih podataka
- administracija sajta

Ovi zadaci u Djangu su dosta pojednostavljeni, neke opcije su već podrazumevane. Primera radi, administracija sajta se prilikom izrade projekta automatski generiše, uloga programera je stvaranje *superusera*[3]. Izdvojićemo neke od pogodnosti koje krase ovaj programski okvir:

[1] softversko okruženje stvoreno da olakša razvoj aplikacije

[2] *Uniformni lokator resursa*-složen iskaz nizova karaktera koji se koristi za lociranje nekog resursa na internetu

[3] Korisnik koji ima pristup administrativnoj strani sajta, dozvoljeno mu je brisanje, dodavanje, uređivanje sadržaja na veb stranici

- Brzina. Glavni cilj ovog programskog okvira je da pomogne programerima da naprave što brže moguće aplikaciju. Počevši od ideje, proizvodnje i završetka izdanja, Django doprinosi tome da bude efikasan i isplativ. Kao rezultat toga, ovaj programski okvir može biti najprikladniji za programere koji imaju stroge rokove.

- Sigurnost koju pruža. Django u sebi sadrži bezbednosne sisteme koji sprečavaju CSRF[4], otimanja klikova[5], XSS[6]. Efikasno upravljanje svim korisničkim imenima i lozinkama, sistemom za autentifikaciju korisnika.

### 2.2.1 Django arhitektura

Kao što je rečeno, Django služi za brzu i jednostavnu izradu veb stranica i aplikacija. Ovu osobinu mu omogućava arhitektura MTV (*model-template-view*) na kojoj se bazira. MTV omogućava nezavisnu izgradnju veb stranice, omogućava neprekidno uključivanje novih komponenti, povećava sigurnost i održavanje sistema. Možemo je podeliti na :

- Model

- View (Pogled)

- Template (Šablon)

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

[4] *Cross-site request forger* je napad koji prisiljava krajnjeg korisnika da izvrši neželjene radnje na veb aplikaciji

[5] *Clickjacking*je napad u kojem se korisnik koji pristupa nekoj veb stranici pokušava obmanuti i naterati ga da klikne na deo veb stranice koji zapravo nije ono na šta korisnik misli.

[6] *Cross-site scripting*je vrsta napada na veb aplikaciju koja prisiljava veb aplikaciju da prosledi napdački izvršni kod korisniku, koji se zatim učitava u korisnikov internet pretraživač i izvršava

Pogled sadrži funkcije koje su zadužene za komunikaciju sa modelom i šablonima. U pogledu definišemo funkcije za komunikaciju sa modelom uz pomoć zahteva (engl. *request*) i odgovora (engl. *response*), koristeći HTML[7] metode i forme. HTML metode definišemo koristeći GET ili POST akcija forme.

Šablon je HTML stranica s dodatnim strukturama koje omogućavaju prikaz podataka koji su prosleđeni od strane pogleda. On je deo MTV-a koji je povezan sa internet pretraživačem. Uloga šablona je da sadržaj poslat od strane pogleda ugradi u HTML kod, koji će se prikazati u internet pretraživaču.

[7]HTML (engl. *Hyper Text Markup Language*) je jezik za opis veb stranica.

# 3 Postupak za podešavanje

U ovom delu će biti prikazan postupak za podešavanje virtuelnog okruženja kao i korišćenje odgovarajuće verzije Python-a. Takođe je dat kratak proces pokretanja Django projekta, prikaz njegove osnovne strukture kao i postupak za instaliranje aplikacija unutar projekta.

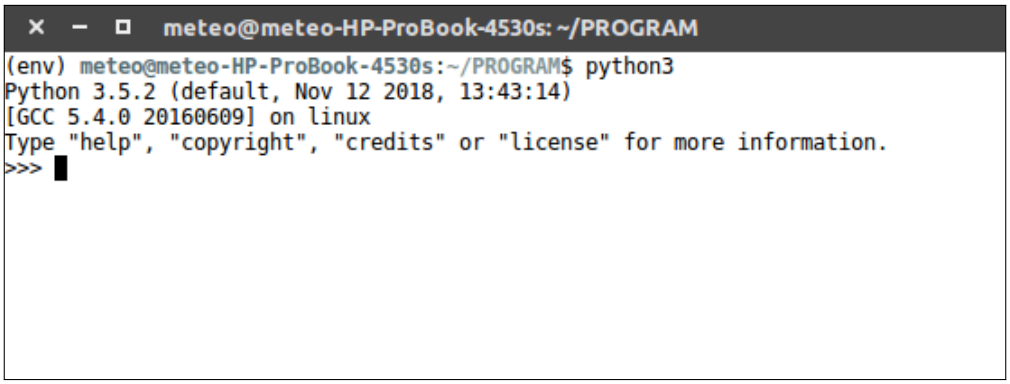
## 3.1 Instaliranje Python 3 verzije

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 sistemske platforme Windows, Linux i OSX. Ovde će biti prikazan postupak podešavanja za rad pod Linux operativnim sistemom. Gotovo sve distibucije linuxa dolaze sa instaliranim Python-om, razlika može biti u verziji. Postoje dve verzije, starija Python 2.7 koja je podržana do kraja 2020. godine, i nova Python 3 verzija. U radu je korišćena verzija 3.5.2. Proveru verzije proveravamo tako što u terminal unesemo sledeću naredbu :

```
python3 --version
```

nakon čega se pojavi sledeće :

```
Python 3.5.2
```




Slika 2: Python verzije 3.5.2 pokrenut iz terminala

Ukoliko verzija Python 3 nije instalirana, ona se može instalirati sledećom naredbom:

```
sudo apt install python3
```

## 3.2 Podešavanje virtuelnog okruženja

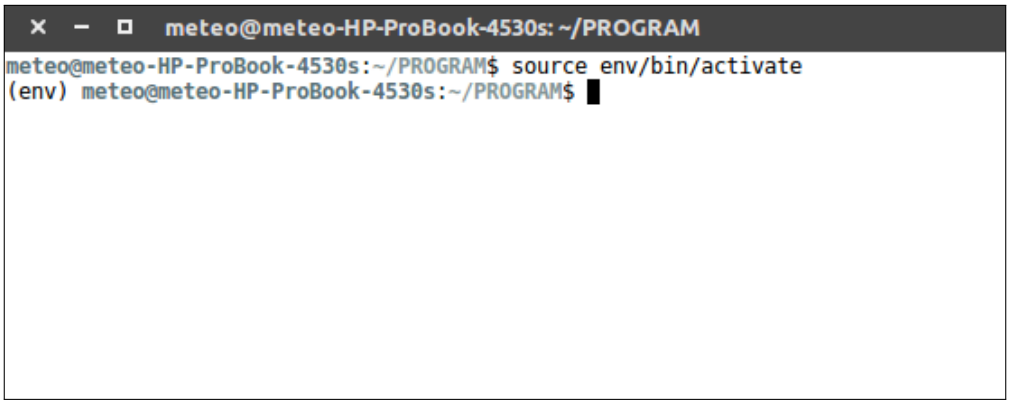
Pre nego što počnemo sa izradom Django veb aplikacije, pokazaćemo kako podesiti veoma zahvalnu alatku, koja nam omogućava održavanje paketa na sistemu. Virtuelno okruženje (engl. *virtual environment*) izoluje projekte jedan od drugog, držeći pakete vezane za svaki projekat zasebno. Kako bi stvorili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

```
python3 –m virtualenv env
```

Kada ovo odradimo, unutar projekta se pojavi direktorijum *env* u kojem će se instalirati svi potrebni paketi, koje ćemo u buduće koristiti. Sledeći korak jeste aktiviranje virtuelnog okruženja. Aktiviranje virtuelnog 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 aktvirano. Kako bi zatvorili virtuelno okruženje, u terminal kucamo komandu *deactivate*.




Slika 3: Uspešno aktivirano virtuelno okruženje

## 3.3 Pokretanje Django projekta

Kada smo podesili virtuelno okruženje Python-a 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 sledećom naredbom :

```
pip install django
```

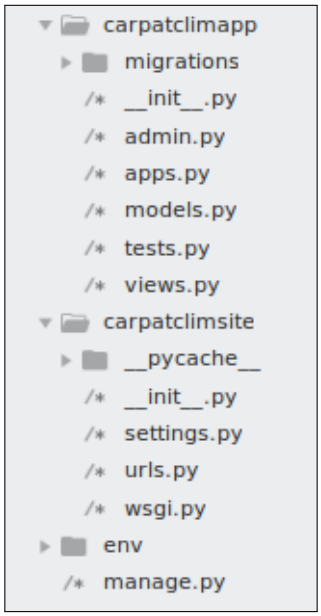
Kada je ovo odrađeno, možemo uspešno pokrenuti i naš prvi Django projekat. Izrada projekta podrazumeva pokretanje Django skripte koja će izgraditi kostur potrebnih direktorijuma sa datotekama. Kako bi stvorili novi projekat koristimo sledeću naredbu:

```
django-admin startproject carpatclimsite
```

pri čemu nastaju carpatclimsite direktorijumi i *manage.py* skripta. Direktorijum čine, skripta *settings.py* koja služi za konfigurisanje sajta, *urls.py* koja služi za spremanje URL obrasca, *__init__.py* skripta koja javlja interpreteru da je direktorijum Python-a paket, i *wsgi.py* skripta koja pomaže pokrenuti Djangov razvojni server. Izvan ovog direktorijuma stvara se *manage.py* skripta, koja predstavlja upravitelja projekta pomoću kojeg se pokreće veb server. Kako bi pokrenuli veb projekat 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 Django logo i ispis o uspešnom pokretanju veb sajta.




Slika 4: Struktura carpatclimsite projekta

Nakon uspešnog pokretanja sajta, sledeća stavka je stvaranje aplikacije. Kako bi stvorili aplikaciju u datom projektu, u terminal upisujemo sledeću naredbu:

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

Nakon ove naredbe u projektu se pojavi direktorijum carpatclimapp sa Python skriptama. Skripta *views.py* (pogled) sadrži funkcije zadužene za funkcionisanje veb aplikacije. Pomoću pogleda definišemo kontak sa bazom podataka, omogućuje se razmena podataka. Kako bi funkcije unutar pogled funkcionisale, one se moraju definisati u *urls.py* skripti koju stvorimo unutar ovog direktorijuma. Sledeća bitna skripta je *forms.py*, pomoću koje definišemo parametre prilikom preuzimanje podataka iz baze podataka. U zavisnosti od izbora željenog vremenskog intervala, pojavi se odgovarajuća forma. *Forms.py* i *urls.py* skriptu moramo stvoriti unutar carpatclimapp direktorijumu.

### 3.3.1 Instaliranje aplikacija u Django projektu

Jedna od mogućnosti koja krasi Django jeste laka implementacija dodatnih aplikacija unutar postojećeg projekta. Međutim, kako bi stvorena aplikacija postala funkcionalna moramo je instalirati. Aplikacija se instalira unutar *settings.py* skripte, upisivanjem imena aplikacije, koja se nalazi unutar carpatclimsite direktorijumu. Takođe je potrebno definistai URL-ove aplikacije unutar *urls.py* skripte koja se takođe nalazi u spomenutom direktorijuma.

Aplikacija *weather*, koja prikazuje vremenske uslove za izabrani grad, je dodatno instalirana u projekat i nije povezana sa *carpatclimapp* aplikacijom. Sa slike 4a) vidimo da su *carpatclimapp* i *weather* aplikacije upisane unutar *INSALLED_APPS*. Sa slike 4b) vidimo način na koji unutar projekta definišemo url-ove za instalirane aplikacije.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'carpatclimapp',
    'weather',
]
```

(a)

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path('', include('carpatclimapp.urls')),
    path('weather/', include('weather.urls')),
]
```

(b)

Slika 5: (a) Instaliranje aplikacije unutar *settings.py*, (b) Definisanje url za aplikacije unutar *urls.py*

## 3.4 Baza podataka

Kako bi ubrzali rad aplikacije, podaci o vrednostima promenljivih su spakovane u *sqlite3* bazu podataka. Ovde će biti prikazan primer za temperaturu vazduha, isti postupak je i za ostale promenljive. Za preuzimanje i formiranje baze podataka pogledati dodatak B. U bazi su podaci poređani po tabelama. Podaci za godišnje vrednosti su grupisane u tabeli pod imenom *yearly*, mesečne vrednosti u tabeli *monthly*, dnevne vrednosti u tabeli *daily*, dok su podaci o tačkama mreže dati u tabeli *grid*. Tabele sa dnevnim, mesečnim i godišnjim podacima sačinjene su od tri kolone, prve kolone sa datumima, druge kolone sa rednim brojem tačke i treća kolona za vrednost temperature.

| | dates | cell | temp |
|---|---|---|---|
| | Filter | Filter | Filter |
| 1 | 1961-1 | 1 | -4.39 |
| 2 | 1961-1 | 2 | -4.76 |
| 3 | 1961-1 | 3 | -6.46 |
| 4 | 1961-1 | 4 | -5.55 |
| 5 | 1961-1 | 5 | -5.72 |
| 6 | 1961-1 | 6 | -4.59 |
| 7 | 1961-1 | 7 | -4.46 |
| 8 | 1961-1 | 8 | -3.75 |
| 9 | 1961-1 | 9 | -3.19 |
| 10 | 1961-1 | 10 | -3.03 |
| 11 | 1961-1 | 11 | -2.97 |
| 12 | 1961-1 | 12 | -2.73 |
| 13 | 1961-1 | 13 | -2.35 |

Slika 6: Tabela sa mesečnim vrednostima za temperaturu

Tabela za vrednosti tačke mreže sačinjenja je od pet kolona. Prvu kolonu čine id[8] vrednosti, druga kolonu čine vrednosti za geografsku dužinu, treća je za vrednost geografske širine, četvrta za definisanje države u kojoj se tačka nalazi dok poslednja kolona sadrži vrednosti za nadmorsku visinu.

[8]id (engl. *primary key*) je posebna kolona baze podataka, i služi za identifikaciju zapisa unutar tabele

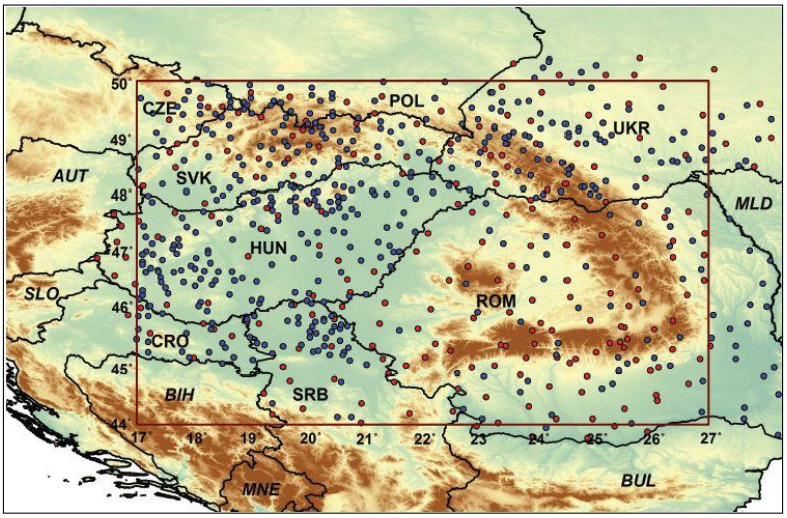
| | id | lon | lat | country | altitude |
|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter |
| 1 | 1 | 17.0 | 50.0 | 7 | 385 |
| 2 | 2 | 17.1 | 50.0 | 7 | 539 |
| 3 | 3 | 17.2 | 50.0 | 7 | 1068 |
| 4 | 4 | 17.3 | 50.0 | 7 | 657 |
| 5 | 5 | 17.4 | 50.0 | 7 | 653 |
| 6 | 6 | 17.5 | 50.0 | 7 | 448 |
| 7 | 7 | 17.6 | 50.0 | 7 | 475 |
| 8 | 8 | 17.7 | 50.0 | 7 | 360 |
| 9 | 9 | 17.8 | 50.0 | 7 | 295 |
| 10 | 10 | 17.9 | 50.0 | 6 | 299 |

Slika 7: Tabela sa vrednostima za tačke mreže

Svaka država unutar obuhvaćene oblasti je zavedena pod određenim brojem:

1. Mađarska
2. Srbija
3. Rumunija
4. Ukrajina
5. Slovačka
6. Poljska
7. Češka Republika
8. Hrvatska

Što se tiče ostalih parametara, najviša tačka nadmorske visine je 2337 metara a najniža 11 metara. Granice za geografsku širinu su od 44°-50°gledajući od juga prema severu. Granice za geografsku dužinu od zapada prema istoku su od 17°-27°. Vremenski period je u razmaku od 1961. do 2010. godine. Sa slike 8 vidimo crvene i plave tačke, koje unutar prikazane oblasti obeležavaju merne stanice. Plavom bojom su obeležene stanice u kojima su merene samo padavine.




Slika 8: Pokrivenost Carpatclim baze podataka

# 4 Interpolacija

Podaci sa meteoroloških stanica su veoma značajni za prognozu vremena. U prognozi vremena se ovi podaci koriste kao početni uslovi za integraciju jednačina kretanja, dok se kod analize koriste za vizuelnu prezentaciju polja promenljivih. Cilj je što vernije predstaviti stanje promenljivih na određenom mestu. Međutim, broj raspoloživih stanica koje šalju podatke i dalje nije dovoljan jer na nekim mestima, poput okeana ili neprisupačnih predela, ne postoji mogućnost postavljana mernih uređaja. Jedan od način za rešenje ovog problema su satelitska merenja, a drugi pristupačniji je uz pomoć interpolacije koja će biti obrađena u ovome delu.

U svrhu prognoze vremena, meteorološki modeli sadrže uniformno raspoređene tačke mreže u kojima je potrebno definisati početne vrednosti atmosferskih parametara, koje modelima služe kao početne vrednosti prilikom rešavanja diferencijalnih jednačina kretanja. Rešavanjem diferencijalnih jednačina kretanja dobijamo očekivane, prognozirane vrednosti atmosferskih parametara. Kako bi ovo sve postigli, potrebno je uz pomoć vrednosti sa raspoređenih meteoroloških stanica izračunati vrednosti u tačkama mreže modela. Ovo se postiže uz pomoć interpolacije. Sem interpolisanja u tačku mreže, interpolacija se koristi i za vizuelnu prezentaciju polja promenljivih. Ovakav pristup analize podataka se naziva *objektivna analiza*. Druga metoda vizuelne prezentacije jeste *subjektivna analiza*, kod koje meteorolozi ručno crtaju polje atmosferskih parametara. Metodi interpolacije u svrhu objektivne analize su različiti, u radu će biti analizirane tri metode.

## 4.1 Korišćene metode interpolacije

Interpolacija predstavlja metod procene vrednosti promenljive na mestima gde oni nisu određenje, koristeći poznate vrednosti sa 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. Sem različitih rezultata, razlikuje se i površina koja je zahvaćena interpolacijonom metodom kao i vreme potrebno za analizu. Usled ove raznolikosti, aplikacija omogućava prikaz tri metoda iterpolacija:

- Linearna
- Barnes
- Cresman

### 4.1.1 Linearna interpolacija

Kod ove metode interpolacije, vrednost u zadatoj tački mreže se dobija kao linearna kombinacija najbliže osmotrenih vrednosti. Linearnu jednačinu možemo definisati kao:

$$X_{grid} = \sum_{j=1}^{m} (h_j * X_{oj}) \tag{1}$$

gde su :

- $X_{grid}$ - interpolisane vrednosti tačke mreže
- $h_j$ - težinska vrednost u tački $j$
- $X_{oj}$ - osmotrena vrednost u tački $j$
- $m$ - broj osmatranja

Težinska vrednost $h_j$ se određuje koristeći sledeći oblik :

$$h_j = \frac{w_j}{\sum_{j=1}^{m}(w_j)} \tag{2}$$

gde je :

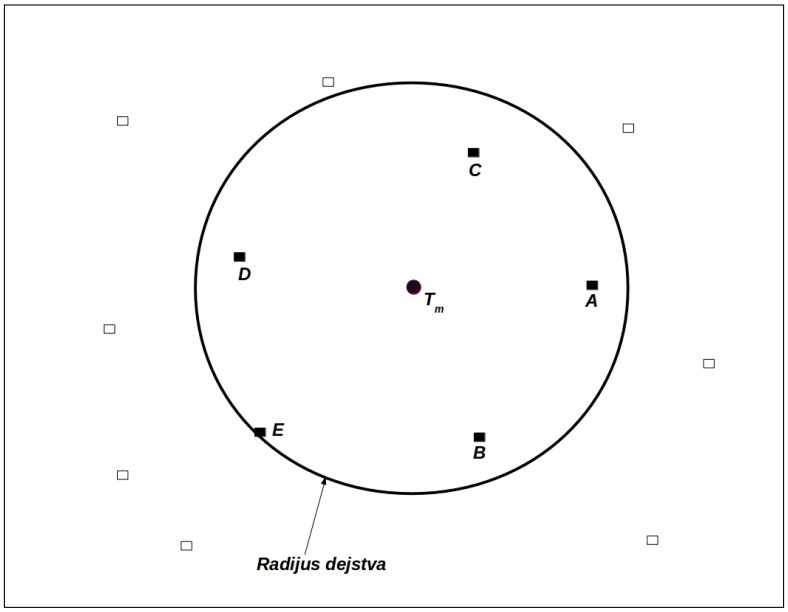
- $w_j$ - težinska funkcija primenjena u tački $j$

Vrednost težinske funkcije $w_j$ se može odrediti koristeći statističke ili empirijske metode. Koristeći empirijski pristup težinska funkcija uzima u obzir distancu između osmotrene vrednosti i tačke mreže. Osmotrena vrednost koja je bliža tački mreže, imaće veću težinsku vrednost od osmotrene vrednosti koja je na većoj razdaljini od tačke mreže. Ovakav pristup nam omogućava definisanje *radijusa dejstva*, gde se prilikom definisanja uticaja osmotrenih vrednosti na datu tačku mreže uzimaju samo vrednosti na određenoj distanci, čime se eliminišu tačke koje imaju mali uticaj na posmatranu tačku

mreže. Time se smanjuje računsko vreme potrebno za dobijanje rezultata. Definišemo je na sledeći način:

$$w_j = e^{-r} \tag{3}$$

gde je $r$ razdaljina između tačke mreže i osmotrene vrednosti




Slika 9: Primer uticaja radijusa dejstva

Sa slike 8 vidimo primer uticaja *radijusa dejstva* na tačku mreže $T_m$. Samo tačke **A**, **B**, **C**, **D**, **E**, koje se nalaze unutar *radijusa dejstva* imaju uticaj na proračun za vrednost u tački mreže $T_m$, tačke koje se nalaze izvan kruga neutiču na proračun. Vrednost u tački $T_m$ se računa kao :

$$T_m = (h_A * X_A) + (h_B * X_B) + (h_C * X_C) + (h_D * X_D) + (h_E * X_E) \tag{4}$$

### 4.1.2 Barnes i Cressman interpolacija

Pre nego što počnemo sa opisom Barnes i Cressman metode interpolacije, definisaćemo na koji način funkcioniše tehnika sukcesivne korekcije, na kojoj se pomenute metode baziraju. Linearna interpolacija koristi osmotrene vrednosti prilikom računanja vrednosti u tački mreže. Sukcesivne korekcijone metode za razliku od linearnih, koriste pozadinsko, predefinisano polje. Vrednosti pozadinskog polja se zatim oduzimaju od

osmotrenih vrednosti, kako bi se utvrdili konačne vrednosti u tačkama mreže, putem iterativnog procesa. Ova metoda je pogodna za oblasti u kojima osmotrene vrednosti retke, slabo rasprostranjene. Metoda sukcesivne korekcije se bazira na sledećem principu:

- proračun pozadinskog polja za svaku osmotrenu vrednost $[X_{pp}]$
- izračunati grešku predviđanja, tj. razliku između pozadinske vrednosti polja u zadatoj tački osmatranja i osmotrene vrednosti $[X_{os} - X_{pp}]$
- izvršiti korekciju vrednosti tačke mreže na osnovu distribuiranih grešaka koristeći sledeći izraz:

$$X_{grid(k+1)} = X_{grid(k)} + \sum_{j=1}^{m} (h_j * (X_{os} - X_{pp})) \tag{5}$$

  gde $k$ prestavlja korak iteracije, a $h_j$ dobijamo koristeći jednačinu (2)

- ponavljati pomenute postupke sve dok se greška ne smanji ispod određenog iznosa

Barnes i Cressman spadaju u grupu sukcesivno korekcijskih procedura, koristeći usrednjenu težinsku proceduru. Bitna razlika između ove dve metode je u definisanju težinske funkcije. Kod Cressman metode težinska funkcija ne opada asimptotski sa povećanjem razdaljine $r$ kao kod Barnes metode, ali naglo teži ka nuli kada je $r > R$. Za Cressman šemu zavisnost težinske funkcije od radijusa delovanja $R$ možemo definisati na sledeći način:

$$w_j = \frac{R^2 - r^2}{R^2 + r^2} \quad za \quad r \leq R \tag{6}$$

$$w_j = 0 \quad za \quad r > R \tag{7}$$

gde je $r^2$ rastojanje između osmotrenih vrednosti i tačke mreže, a $w_j$ težinska funkcija. Kod ove metode, osmotrene vrednosti izvan radijusa desejstva se ne uzimaju u ažuriranje vrednosti polja. Radijus dejsva $R$ se smanjuje sa svakim prolazom. Nakon velikog broja prolaza, vrednosti tačaka mreže konvergiraju ka osmotrenim vrednostima. Vrednost u tački mreže se nalazi koristeći jednačinu (5). Izbor pogodne vrednosti za $R$ zavisi od prostora između podataka, kao i od željene glatkoće prikaza. Uglavnom, male

vrednosti za radijus dejstva uzrokuje manju glatkoću (lošiji prikaz), dok se za veće vrednosti očekuje bolji prikaz.

Za Barnes interpolaciju dovoljna su samo dva prolaza kako bi se izvršila interpolacija. Težinska funkcija je data kao:

$$w_j = exp(\frac{-r^2}{4k}) \tag{8}$$

gde su:

- $r$ - rastojanje između tačke mreže i osmatrane tačke
- $k$ - parametar koji se koristi za definisanje odgovora težinske funkcije, kontroliše brzinu opadanja radijusa dejstva

Pošto se težinska funkcija asimptotski približava nuli, kod ove metode nije potrebno definisati radijus uticaja kao kod Cressmanove metode. Tokom prvog prolaza obezbeđuje se pozadinsko polje, koje je potrebno tokom drugog prolaza. Znači, kako bi definisali vrednosti za tačku mreže, najpre vršimo prvi prolaz na sledeći način:

$$x_{i,j} = \frac{\sum_{n=1}^{N} w(r,R) * X_{os}}{\sum_{n=1}^{N} w(r,R)} \tag{9}$$

gde su:

- $x_{i,j}$ - vrednosti tačke mreže
- $X_{os}$ - osmotrene vrednosti promenljive
- $N$ - broj osmatranja
- $w(r,R)$ - Barnes težinska funkcija

Analiza tokom drugog prolaza se može definisati:

$$X_{i,j(k+1)} = X_{i,j(k)} + \frac{\sum_{n=1}^{N} (w'(r,R) * (X_{os} - X_{pp}))}{\sum_{n=1}^{N} w'(r,R)} \tag{10}$$

gde težinska funkcija $w'(r,R)$ poprima sledeći oblik:

$$w_j = exp(\frac{-r^2}{\gamma k^2}) \tag{11}$$

gde $\gamma$ predstavlja numerički parametar konvergercije koji kontroliše razlike između težinskih funkcija na prvom i drugom prolazu. Njene vrednosti se kreću od 0 do 1 $(0 < \gamma < 1)$. U radu je uzeta vrednost $\gamma = 0.25$.

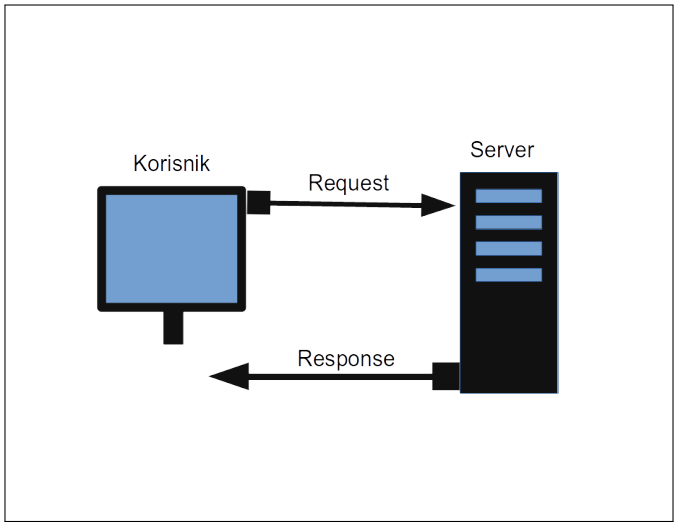
# 5 Carpatclim aplikacija

Nakon kratkog opisa korišćenih paketa, Django arhitekture, tehnika interpolacije, u ovom delu će biti prikazana i sama aplikacija, njen način funkcionisanja.

## 5.1 Mehanizam rada aplikacije

U ovom delu ćemo dati kratak opis interakcije između internet pretraživač i servera (baza podataka). Razumevanje ove interakcije je od značaja, kako bi razumeli na koji način radi aplikacija. Definisaćemo šta su forme i metode uz pomoć kojih se razmenjuju podaci, zatim objasniti ulogu *views.py* skripte, definisati ulogu šablona (engl. *templates*), ulogu URL-a, i na kraju kako izgleda interpolacija polja temperature i padavina. Aplikacija omogućava i preuzimanje podataka sa mernih stanica, i interpolisane vrednosti za određenu tačku mreže za izabrani vremenski period.

Kako bi razumeli razmenu informacija između korisnika i servera moramo definisati značenje *request* i *response* zahteva. *Request* zahtev omogućava razmenu podataka između korisnika i servera. Korisnik šalje *requst* zahtev prema serveru, koji onda uz pomoć *response* odgovara šalje odgovor na *request* zahtev. Svaki *request* ima specifičnu URL adresu, koje se stvaraju uz pomoć HTML elemenata, forma i metoda.




Slika 10: Razmena informacija između korisnika i servera

### 5.1.1 Forme i metode

Forma je HTML element pomoću kojeg grupišemo više elemenata za unos podataka. U te elemente korisnik može uneti ili izabrati informacije koje se zatim koriste za neku zadatu svrhu, uzimanje ili unos podatak u bazu podataka. Elementi forme su uglavnom grupisani, i kao takvi zajedno se šalju na obradu kada se forma prihvati (engl. *submit*). Django forme definišemo untar skripte *forms.py*, koju stvaramo unutar *carpatclimapp* direktorijuma. Svaku formu definišemo u klasu (engl. *class*), a elemente forme predstavljaju atributi te klase. Tako, za godišnje vrednosti stvaramo klasu *CronFormYearly*, unutar koje definišemo atribut klase *var, year, inter* . Element *year* poprima Django formu *ChoiceField*, formu za izbor vremenskog intervala (u ovom primeru samo za godinu), *var* predstavlja izbor promenljive, *inter* definiše polje za izbor interpolacione metode.

```
var   = forms.ChoiceField(choices=MY_VARIABLE, label="Variable")
year = forms.ChoiceField( choices = [(x, x) for x in range(1961,
    2011)], initial=1961)
inter = forms.ChoiceField( choices = MY_CHOICES, label="Interpolation
    ",required=True)
```

Panel 1. Definisanje atributa unutar klase *CronFormYearly*

Metode omogućavaju slanje podataka forme. Postoje dve osnovne metode slanja, POST i GET. Izborom metode utičemo na koji način će se podaci forme proslediti ka stranici. Navešćemo razlike između ove dve metode, mane i prednosti. Kada odaberemo GET metod slanja, podaci forme se šalju kroz komandnu liniju, iza znaka $ unutar polja za unost adrese internet pretraživača. Kod ove metode informacije se "lepe" na URL, što ga čini idealnim u slučaju spremanja veb stranice unutar omiljenih (engl. *favorites*). Kao mana ove metode jeste ograničena količina podataka koji se mogu poslati. GET metoda je vrlo nesigurna jer se lako može izmeniti unutar URL-a. Nije preporučljivo ovu metodu koristiti prilikom slanja lozinke i korisničkog imena tokom prijava. Odabirom metode POST podaci nisu vidljivi u komandnoj liniji već se šalju transparentno kroz HTTP zahtev (engl. *request*). Pošto se podaci šalju transparento, na njih ne možemo uticati izmenom linka unutar polja za adresu. Jedino pravilo koje se mora poštovati je da se obavezno mora koristiti POST metoda kada se vrši slanje forme.

### 5.1.2 Uloga views.py skripte

Unutar *views.py* skripte, posredstvom stvorenih funkcija, definišemo metode slanja (POST ili GET), i šta će se razmenjivati (u zavisnosti od definisane forme) prilikom komunikacije korisnika sa serverom. Kako izgleda funkcija za razmenu podataka za srednje godišnje vrednosti promenljivih, prikazano je u panelu 2.

```
def yearly(request):
  if request.method=="POST":
    form = CronFormYearly(request.POST)
    if form.is_valid():
      data = form.cleaned_data
      year = data.get('year')
      var = data.get ('var')
      date_path = '/%s/%s/%s'%(var,inter,year)
      return redirect(date_path)
  else:
    form = CronFormYearly()
    active_yearly = True
  return render(request, 'carpatclimapp/home.html', {'form': form, '
   active_yearly': active_yearly})
```

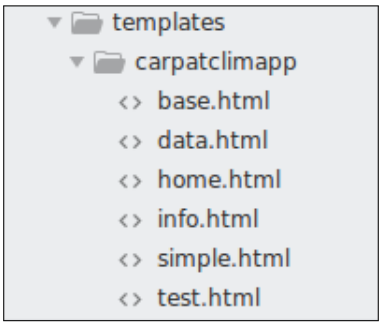
Panel 2. Definisanje *yearly* funkcije unutar *views.py* skripte

Ova funkcija stvara upit za godišnje podatke i definiše sledeće. Ako se podnese POST zahtev u obliku forme *CronFormYearly*, i ako je ta forma važeća, zahtevani podaci se prosleđuju u obliku URL-a. URL se konstruiše unutar *date_path* strukture, i u zavisnosti od ove strukture, preusmeravanjem (return redirect) se aktiviraju funkcije zadužene za grafički prikaz interpolisanih vrednosti. U ovom primeru aktivira se funkcija *carpatclim_y_figure* (poglavlje 5.1.4). Međutim, ako nema POST zahteva, prikazaće se forma definisana unutar *home.html* šablona.

### 5.1.3 Šablon-Templates

Prikaz veb stranice se ostvaruje uz pomoć šablona, koje stvaramo unutar direktorijuma templates. Za svaku aplikaciju unutar Django projekta se stvaraju zasebni šabloni.

U svrhu vizuelne prezentacije pored HTML-a i CSS-a[9] korišćen je i Bootstrap[10]. Prikaz forme definisan je unutar *home.html* šablona, koji je deo *base.html* stranice.




Slika 11: Templates direktorijum za carpatclimapp aplikaciju

```
{% extends 'carpatclimapp/base.html' %}
{% block content %}
  <div>
    <p>Map image for date:</p>
    <form method="POST" class="post-form">{% csrf_token %}
      {{ form.as_p }}
      <button type="submit" class="save btn btn-default">Submit</
    button>
     </form>
  </div>
{% endblock %}
```

Panel 3. Izgled *home.html* šablona

Pisanje šablona zahteva ponavljajući kod jer svaki šablon mora sadržati osnovne elemente kao što su naslov, zaglavlje i podnožje. Kako bi se izbeglo uzastopno pisanje istog koda, Django koristi *Jinja* sintaksu. Ova sintaksa je primenjena koristeći {% block contetn %} i {% end block %} strukturu unutar *home.html* šablona. Ova blok struktura nam omogućava nadovezivanje na *base.html* šablon, odnosno poprima njegov naslov, zaglavlje i podnožje, pri čemu se izbegava ponavljanje istog koda. Zbog dužine koda kod *base.html* šablona, uradićemo podelu koda na delove.

[9]CSS (*engl. Cascading Style Sheets*) je jezik za definisanje izgleda elemenata veb stranice

[10]Bootstrap je softver otvorenog koda (*engl. framework*) namenjen za stvaranje savremenih veb stranica prilagođene različitim platformama

```
{% load static %}
<html>
    <head>
        <title>CarpatClim Online App</title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery
    /3.3.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/
    js/bootstrap.min.js"></script>
        <meta name="viewport" content="width=device-width, initial-
    scale=1">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
    bootstrap/3.3.7/css/bootstrap.min.css">
        <link rel="stylesheet" href="{% static 'css/carpatclimapp.css
    ' %}">
        <link href="//fonts.googleapis.com/css?family=Ubuntu&subset=
    latin,latin-ext" rel="stylesheet" type="text/css">
        <link rel="shortcut icon" type="image/x-icon" href="{% static
     '/images/favicon.ico' %}"/>
    </head>
```

Panel 4. Definisanje head element *base.html* šablona

*Head* element sadrži informacije o veb stranici. U njemu se definiše podešavanje fontova, veza sa konfiguracionim fajlovima zaduženih za izgled stranice. U našem slučaju povezivanje je vršeno sa *Bootstrap* bibliotekom, *JQuery*[11] bibliotekom, *carpatclimapp.css* konfiguracionom datotekom. Bez ovih veza, neki od efekata na veb stranici ne bi funkcionisali. Sledeći element je *body*, koji predstavlja sadržaj HTML dokumenta, kao što je tekst, slike, tabele, liste i druge. U primeru sa panela 4.1 je prikazan deo liste padajućeg menija (engl. *Dropdown menu*) za izbor vremenskog perioda interpolacije.

```
<body>
<div class="page-header">
    <center><h1 color="#fafafa grey lighten-5"> CarpatClim Data </h1
    ></center>
<div class="row">
  <div class="col-md-12 school-options-dropdown">
    <div class="dropdown btn-group">
      <button class="btn btn-primary dropdown-toggle" type="button"
    data-toggle="dropdown">Data interpolate
```

[11] Javaskript biblioteka dizajnirana da pojednostavi skriptovanje HTML jezika

```
      <span class="caret"></span>
    </button>
    <ul class="dropdown-menu">
      <li>
          {% if active_yearly %}
          <a href="/yearly" class="active">Yearly interpolation</a>
          {% else %}
          <a href="/yearly">Yearly interpolation</a>
          {% endif %}
      </li>
      <li>
          {% if active_monthly %}
          <a href="/monthly" class="active">Monthly interpolation</
a>
          {% else %}
          <a href="/monthly">Monthly interpolation</a>
          {% endif %}
      </li>
      <li>
          {% if active_daily %}
          <a href="/daily" class="active">Daily interpolation</a>
          {% else %}
          <a href="/daily">Daily interpolation</a>
          {% endif %}
      </li>
    </ul>
  </div>
```

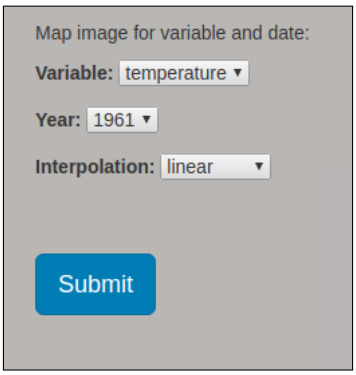
Panel 4.1 Struktura body elementa za *base.html* šablon

Naredbe {% if % }, {% else % } i {% endif % } su takođe *Jinja*sintakse korišćenje za prikaz sadržaja. Ako je jedna od promenljive *active_yearly, active_monthly, active_daily*, poprimila logički tip podatka *True*, onda se prikazuje sadržaj između {% if % } i {% else % } strukture, u suprotnom se prikazuje sadržaj između {% else % } i {% endif % } struktura.

```
    <div class="content container-fluid">
        <div class="row">
            <div class="col-md-3">
                {% block content %}
                {% endblock %}
            </div>
        </div>
    </div>
</body>
</html>
```

Panel 4.2 Struktura body elementa sa definisanim *block content*

Sadržaj definisan unutar *home.html* šablona, prikazan na panelu 3, se prikazuje unutar {% block contetn %} i {% load static % } strukture *base.html* šablona, prikazanog na panelu 4.2. Ovakav način konstrukcije veb stranice omogućen je korišćenjem *Jinja* {% load static % } sintakse. Izgleda forme za interpolaciju godišnjih vrednosti temperature, definisana unutar panela 1, prikazan je na slici 12.




Slika 12: Izgled forme za interpolaciju godišnjih vrednosti

### 5.1.4 Interpolisanje polja promenljivih i preuzimanje podataka

U poglavlju 5.1.2 je spomenuto aktiviranje funkcije *carpatclim_y_figure* prilikom definisanja odgovarajućeg URL-a. Ova funkcija se stvara unutar *views.py* skripte, i njega uloga je da vrati odgovor u obliku slike formata *.png*, za interpolisane vrednosti polja. Za proračun interpolisanih vrednosti zadužena je funkcija *create_map*, definisana unutar *carpatclim.py* skripte, koja se nalazi u *carpatclimapp* direktorijumu.

```
def carpatclim_y_figure(request,var,inter,year):
  map = create_map(year,inter,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
```

Panel 5. *Carpatclim_y_figure* funkcija

Pošto je funkcija *create_map* zadužena za interpolaciju, uradićemo njenu kratku analizu. Podelićemo ovu funkciju na tri dela, gde prvi deo služi za povezivanje programa sa bazom podataka. U zavisnoti od izbora dnevnih, mesečnih ili godišnjih vrednosti, pristupa se tabeli unutar baze. Za učitavanje vrednosti promenljivih (u prikazanom primeru za temperaturu), i tačaka mreže koristi se *pandas* naredba *pd.read_sql_query*.

```
conn = sqlite3.connect(DB)
cursor = conn.cursor()
if day:
  table = 'daily'
elif month:
  table = 'monthly'
elif year:
  table = 'yearly'
query = '''
  SELECT dates, cell, temp FROM %s WHERE dates = "%s";
''' % (table, mapname)
result_df = pd.read_sql_query(query, conn, index_col='dates')
query = '''SELECT id, lon, lat FROM %s;''' % 'grid'
grid = pd.read_sql_query(query, conn, index_col='id')
cursor.close()
conn.close()
```

Panel 6. Povezivanje funkcije *create_map* sa bazom podataka

Drugi deo funkcije služi za projekciju vrednosti na izabranu projekciju, u radu je korišćena *Mercator*, i brisanje *Nan*[12] vrednosti.

[12] Označava se polje za koje nije definisana vrednost.

```
to_proj = ccrs.Mercator()
x_masked, y_masked, temps = remove_nan_observations(xp_, yp_,
    result_df.values)
```

Panel 6.1 Projekcija i brisanje *NaN* vrednosti

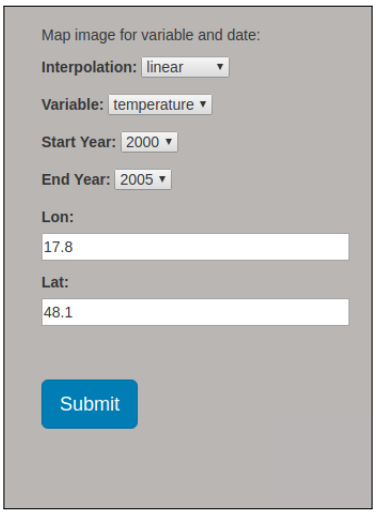
Treći deo je zadužen za proračun interpolacije koristeći, *interpolate_to_grid* funkciju, i definisanje izgleda izlazne slike. Funkcija vraća sliku interpolisanog polje promenljive(*return fig*) koji se zatim uz pomoć funkcije *carpatclim_y_figure* prikazuje na stranici internet pretraživača.

```
tempx, tempy, temp = interpolate_to_grid(
      x_masked, y_masked, temps, interp_type='linear',  hres=2000)
temp = np.ma.masked_where(np.isnan(temp), temp)
a = int(temp.max())
b = int(temp.min())
levels = list(range(b, a))
cmap = plt.get_cmap('viridis')
norm = BoundaryNorm(levels, ncolors=cmap.N, clip=True)
fig = plt.figure(figsize=(10, 8))
view = fig.add_subplot(1, 1, 1, projection=to_proj)
view.set_extent([27.0, 17.1, 50, 44.5])
view.add_feature(cfeature.BORDERS, linestyle=':')
gl = view.gridlines(crs=ccrs.PlateCarree(), draw_labels=True,
          linewidth=2, color='gray', alpha=0.5, linestyle='--')
gl.xformatter = LONGITUDE_FORMATTER
gl.yformatter = LATITUDE_FORMATTER
mmb = view.pcolormesh(tempx, tempy, temp, cmap=cmap, norm=norm)
fig.colorbar(mmb, shrink=.8, pad=0.06, boundaries=levels)
plt.close('all')
return fig
```

Panel 6.2. Definisanje linearne interpolacije unutar funkcije

Pored interpolacije polja promenljivih, aplikacija omogućava i interpolaciju za definisanu tačku mreže u definisanom vremenskom intervalu. Kao što vidimo sa slike 13, forma definiše izbor interpolacije, izbor promenljive i definisanje vremenskog intervala. Kao primer uzećemo interpolsisanje vrednosti u tački geografske dužine 17.8°i geografske širine 48.1°. U razmatranje ćemo uzeti vremenski period sa početkom od 2000. godine pa do 2005. godine. Podaci se dobijaju u vidu tekst datoteke. Princip

preuzimanja podataka je sličan kao i za interpolaciju, razlika je u dobijenom odgovoru. Kod interpolacije kao odgovor dobijamo sliku interpolisanih vrednosti, dok se ovde dobija izlazna datoteka sa podacima. Za periodični prikaz podataka je zadužena funkcija *period_year* koja je definisana unutar *carpatclimp.py* skripte. Funkcija definiše petlju koja otvara bazu podataka za definisan period, i vrši proračun interpolisane vrednosti u zadatoj tački.




Slika 13: Definisanje interpolacije u tački sa geografskom dužinom 17.8 i geografskom širinom 48.1, u vremenskom intervalu od 2000. godine do 2005. godine

```
def period_year_temp(year,year1,lon,lat,inter):
  cnx = sqlite3.connect(DB)
  cursor = cnx.cursor()
  table = 'yearly'
  year = int(year)
  year1 = int(year1)
  newlist = []
  newlist1 = []
  newlist2 = []
  for i in range (year,year1+1,1):
    newlist2.append(i)
    query = '''
      SELECT  dates, cell, temp FROM %s WHERE dates = "%s" ;
      ''' % (table, i)
    df = pd.read_sql_query(query, cnx)
```

```
    tacka = '''SELECT id, lon, lat,country,altitude FROM %s;''' % '
   grid'
    grid = pd.read_sql_query(tacka, cnx)
    podaci = pd.merge(df,grid,left_on='cell',right_on='id')
    podaci_a = podaci.drop(['cell','id','country','altitude'],axis=1)
    lon_n = podaci_a['lon'].values
    lat_n = podaci_a['lat'].values
    temp =podaci_a['temp'].values
    x_masked, y_masked, temp_p = remove_nan_observations(lon_n, lat_n
   , temp)
    lon = float(lon)
    lat =float(lat)
    xy = np.vstack([x_masked,y_masked]).T
    xi = np.vstack([lon,lat]).T
    if inter == "linear":
      inter_point = interpolate_to_points(xy,temp_p,xi, interp_type='
   linear')
    elif inter == "cressman":
      inter_point =interpolate_to_points(xy,temp_p,xi, interp_type='
   cressman', minimum_neighbors=3,
              gamma=0.25, kappa_star=5.052, search_radius=None,
   rbf_func='linear',
              rbf_smooth=0)
    elif inter == "barnes":
      inter_point =interpolate_to_points(xy,temp_p,xi, interp_type='
   cressman', minimum_neighbors=3,
              gamma=0.25, kappa_star=5.052, search_radius=None,
   rbf_func='linear',
              rbf_smooth=0)
    for y in inter_point:
      newlist.append(y)
    for z in xi:
      newlist1.append(z)
xi= str(xi)
newlist_fix = [str(a) for a in newlist]
d = {'Year':newlist2,'Lon&Lat':newlist1,'Temperature':newlist_fix}
df = pd.DataFrame(d)
return (df)
```

Panel 9. Kod za proračun interpolacije u definisanoj tački mreže

### 5.1.5 Definisanje URL-a

Stvaranjem URL-a mi u stvari stvaramo stranice, koje sa aktiviraju zahtevom (engl. *request*). To se radi tako što unutar *carpatoclimapp* direktorijuma stvorimo skriptu *urls.py*, unutra koje definišemo url obrasce (engl. *urlpatterns*) za funkcije:

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

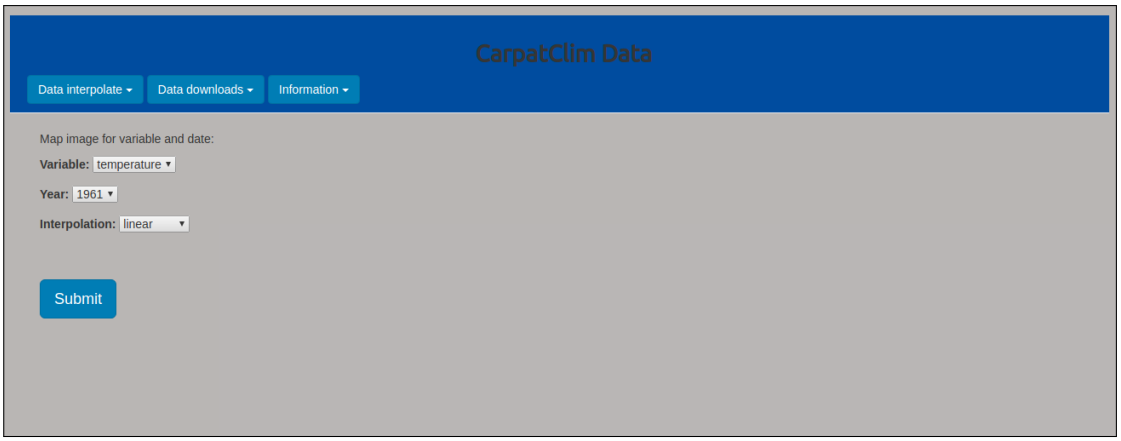
Panel 10. Primer URL obrasca za godišnje vrednosti

Ovim pristup postižemo da za svaku funkciju postoji zadata adresa. Recimo, korišćenjem *yearly/* u polju za adresu, prikazuje se forma za funkciju *yearly* definisanu unutar *views.py* scripti. U primeru sa panela 10 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. Kada se izabere godina i aktivira *submit* komanda, u polju za adrese se pojavljuje zadata godina, čime se aktivira funkcija *carpatclim_y_figure*, koja je zadužena za interpolaciju i vizuelizaciju interpolisanih vrednosti.

## 5.2 Izgled aplikacije

Na slici 14 je prikazan izgled aplikacije. Idući sa leve prema desnoj strani, u gornjem levom delu aplikacije se nalaze padajući meniji sa različitim opcijama:

- *Data interpolate* - padajući meni sa opcijama za interpolaciju
- *Data downloads* - padajući meni sa opcijama za preuzimanje podataka
- *Information* - padajući meni sa informacijama




Slika 14: Izgled aplikacije

Prvi padajući meni *Data interpolate* sadrži opcije za interpolaciju srednjih vrednosti promenljivih za određeni interval:

- *godišnju interpolaciju (yearl interpolation)*
- *mesečnu interpolaciju (monthly interpolation)*
- *dnevnu interpolaciju (daily interpolation).*

Drugi padajući meni *Data downloads* sadrži opcije za preuzimanje podataka i to sa:

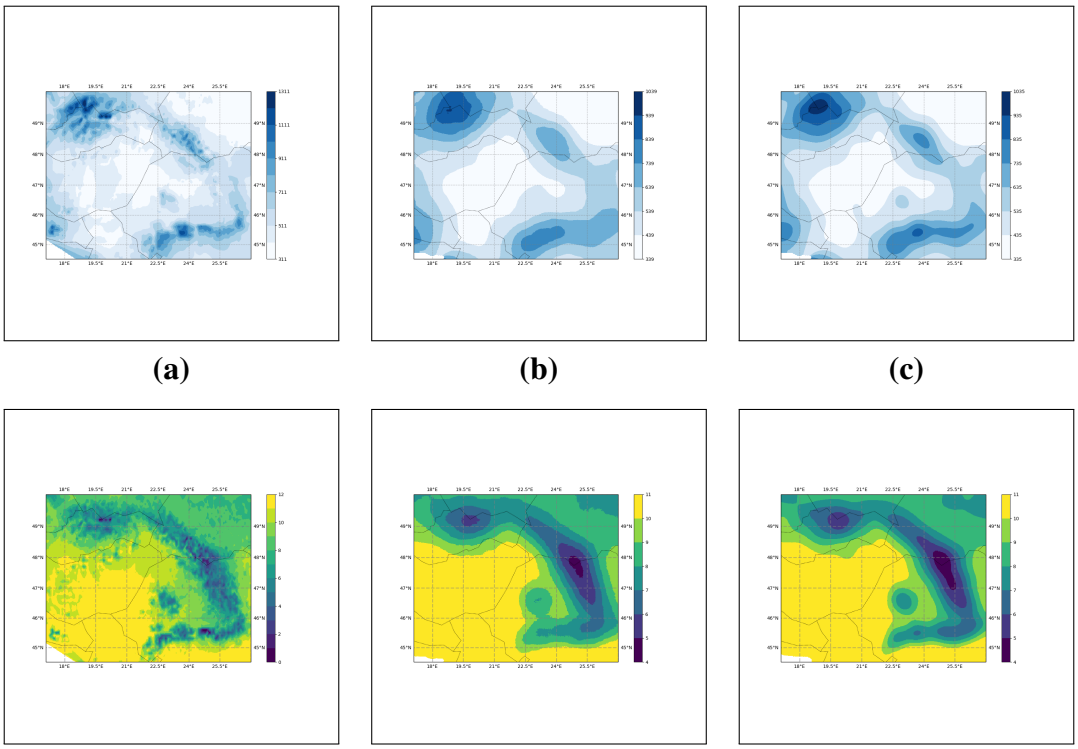
- *Station data downloads* - meteoroloških stanica za određeni vremenski period
- *Interpolate data downloads* - za interpolisanu tačku u određenom vremenskom intervalu.

Padajući meni *Information* sadrži informacije :

- *Weather Forecast* - trenutne vremenske prilike za izabrani grad

## 5.3 Godišnje, mesečne, dnevne srednje vrednosti interpolacije

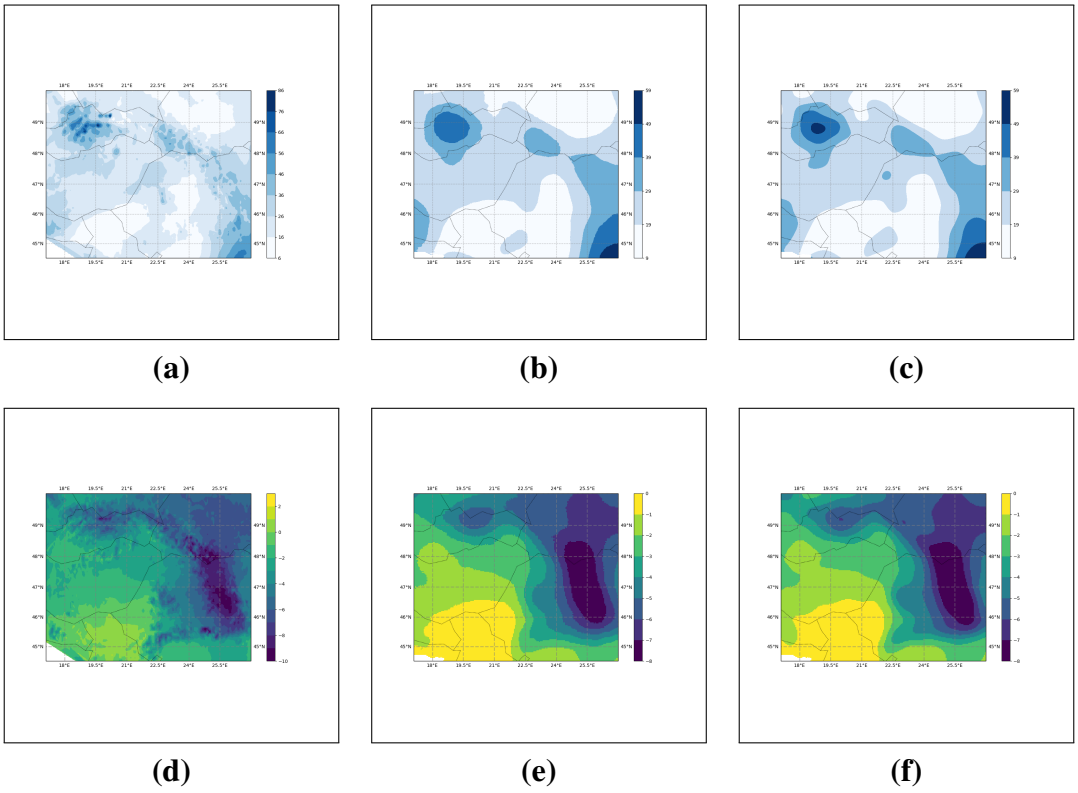
Prilikom izbora *Data interpolate* pojavljuje se izbor vremenskog perioda za koji želimo izvršiti interpolaciju. U zavisnosti od izbora vremenskog intervala, pojavljuje se različita forma. Kada izaberemo željeni vremenski interval, aktiviranjem *submit* komande dobijamo interpolisan grafički prikaz polja promenljive. Na slikama 15, 16, 17 redom su predstavljene interpolacije srednjih godišnjih, mesečnih i dnevnih vrednosti za padavine i temperature.



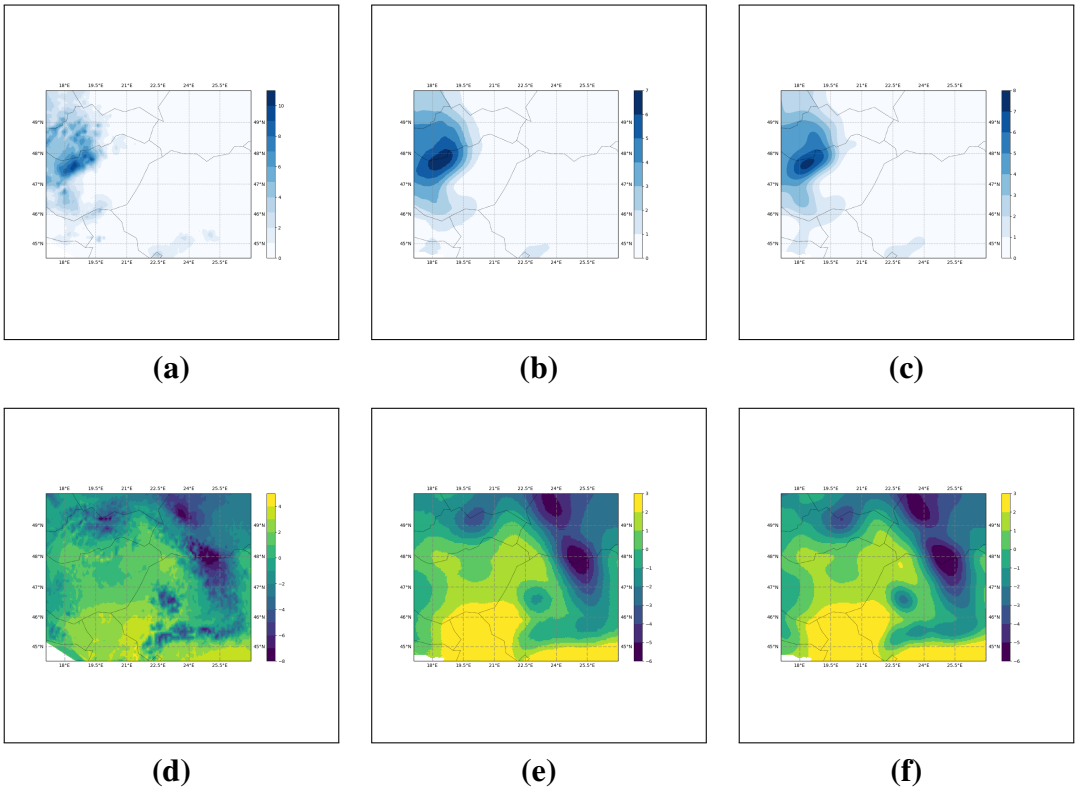
**(a)** **(b)** **(c)**

**(d)** **(e)** **(f)**

Slika 15: **Interpolacija padavina za 1961: (a)** Linearna **(b)** Barnes **(c)** Cressman; **Interpolacija temperature za 1961: (d)** Linearna **(f)** Barnes **(e)** Cressman



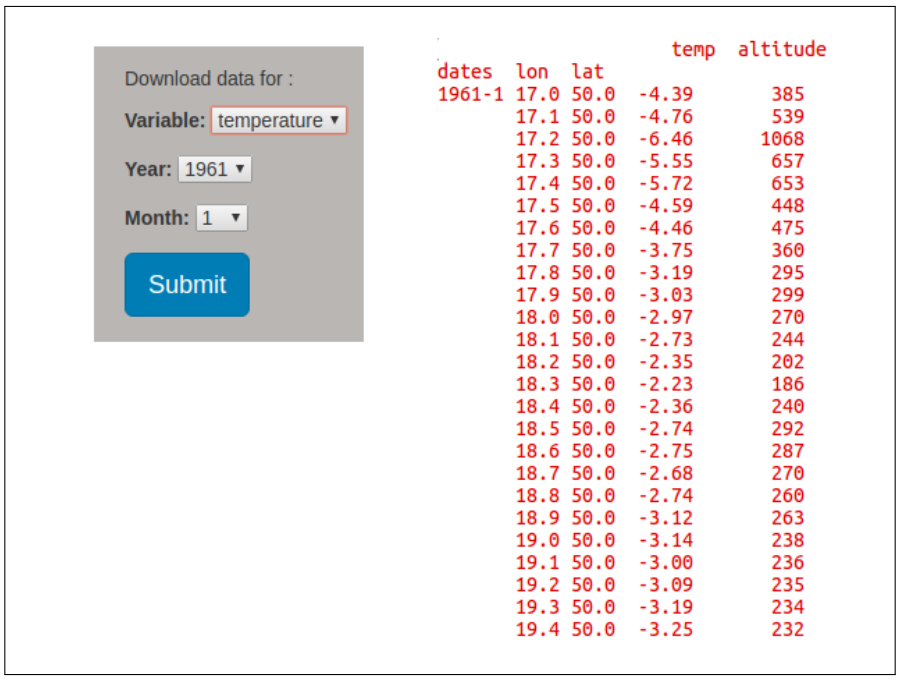

Slika 16: **Interpolacija padavina za 1961/1: (a)** Linearna **(b)** Barnes **(c)** Cressman; **Interpolacija temperature za 1961/1: (d)** Linearna **(f)** Barnes **(e)** Cressman



Slika 17: **Interpolacija padavina za 1961/1/1:** **(a)** Linearna **(b)** Barnes **(c)** Cressman; **Interpolacija temperature za 1961/1/1:** **(d)** Linearna **(f)** Barnes **(e)** Cressman

## 5.4 Preuzimanje podataka

Unutar padajućeg menija *Data downloads* postoje dve mogućnosti. Prva opcija nam omogućava preuzimanje podataka sa meteoroloških stanica (*Station data downloads*) a druga nam ispisuje interpolisane podatke za definisanu tačku mrežu (*Interpolate data downloads*), koristeći jedan od ponuđenih interpolacionih metoda. Podaci se mogu preuzeti za godišnje (*yearly*), mesečne (*monthly*) i dnevne vrednosti (*daily*).

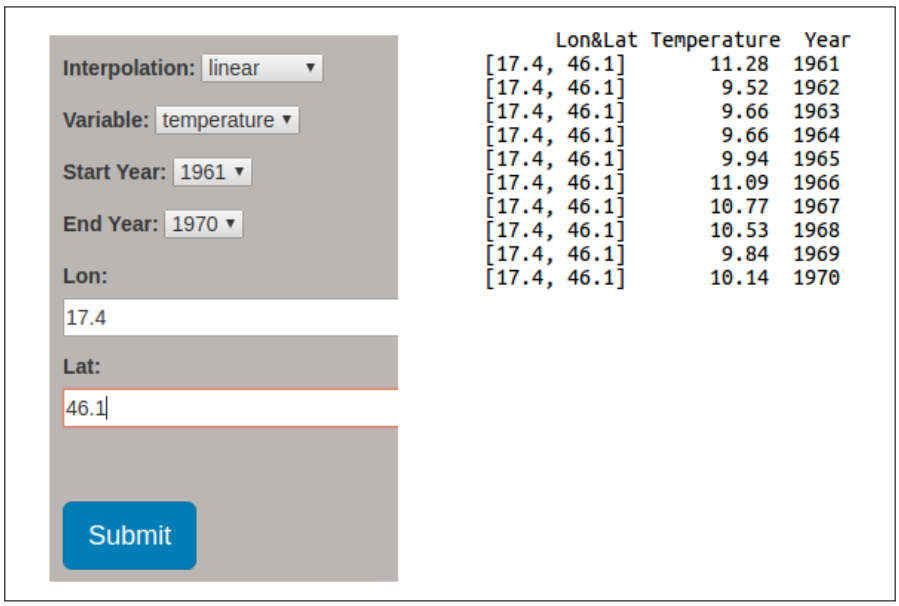



| dates | lon | lat | temp | altitude |
|---|---|---|---|---|
| 1961-1 | 17.0 | 50.0 | -4.39 | 385 |
| | 17.1 | 50.0 | -4.76 | 539 |
| | 17.2 | 50.0 | -6.46 | 1068 |
| | 17.3 | 50.0 | -5.55 | 657 |
| | 17.4 | 50.0 | -5.72 | 653 |
| | 17.5 | 50.0 | -4.59 | 448 |
| | 17.6 | 50.0 | -4.46 | 475 |
| | 17.7 | 50.0 | -3.75 | 360 |
| | 17.8 | 50.0 | -3.19 | 295 |
| | 17.9 | 50.0 | -3.03 | 299 |
| | 18.0 | 50.0 | -2.97 | 270 |
| | 18.1 | 50.0 | -2.73 | 244 |
| | 18.2 | 50.0 | -2.35 | 202 |
| | 18.3 | 50.0 | -2.23 | 186 |
| | 18.4 | 50.0 | -2.36 | 240 |
| | 18.5 | 50.0 | -2.74 | 292 |
| | 18.6 | 50.0 | -2.75 | 287 |
| | 18.7 | 50.0 | -2.68 | 270 |
| | 18.8 | 50.0 | -2.74 | 260 |
| | 18.9 | 50.0 | -3.12 | 263 |
| | 19.0 | 50.0 | -3.14 | 238 |
| | 19.1 | 50.0 | -3.00 | 236 |
| | 19.2 | 50.0 | -3.09 | 235 |
| | 19.3 | 50.0 | -3.19 | 234 |
| | 19.4 | 50.0 | -3.25 | 232 |

Slika 18: Definisanje forme levo, i izlazni ispis desno za mesečne podatke sa meteoroloških stanica

Sa slike 18 vidimo kako izgleda forma prilikom izbora *Monthly data* opcije, i izlazni zapis podataka. U formu zadajemo promenljivu (*variable*), godinu (*year*) i mesec (*month*) za koju želimo preuzeti podatke sa stanica. Podaci su ispisani u datoteci tekst formata, gde prva kolona definiše poziciju meteorološke stanice, u drugoj koloni su izmerene vrednosti temperature a treća kolona prikazuje nadmorsku visinu na kojoj su podaci izmereni. Druga opcija služi za proračun interpolisanih vrednosti u zadatoj tački mreže. Sa slike 19 vidimo kako izgleda forma za izabrani godišni period. Ona sadrži izbor interpolacione metode (*interpolation*), izbor promenljive (*variable*), definisanje početnog i krajnjeg godišnjeg intervala za koji želimo dobiti vrednosti, i na kraju definisane geografske dužine i širine tačke u kojoj izračunavamo interpolisanu vrednost. Izlazni zapis

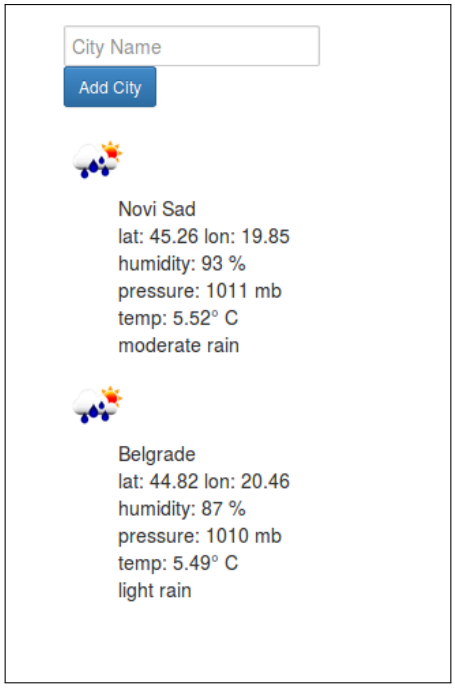
se takođe dobija u vidu datoteke, tekst formata, gde prva kolona definiše kordinate izabrane tačke, druga kolona interpolisanu vrednost a u trećoj je godina.




Slika 19: Definisanje forme levo, i izlazni ispis desno za interpolaciju za definisan godišnji interval

## 5.5 Trenutno vreme

Poslednji padajući meni *Information*, prikazuje informacije o trenutinim vremenskim prilikama za izabrane gradove. Kako bi izabrali željenu lokaciju, u polje "*City Name*" upišemo ime grada. Podaci o trenutnim vremenskim uslovima se pruzimaju sa *OpenWeatherMap* sajta. Kako bi dobijali vremenske podatke, potrebno je registrovati se na pomenutom sajtu, nakon čega se dobija API[13] ključ, koji se vezuje za vaš korisnički nalog, i dozvoljava nam sinhronizovano slanje informacija o vremenskim prilikama.



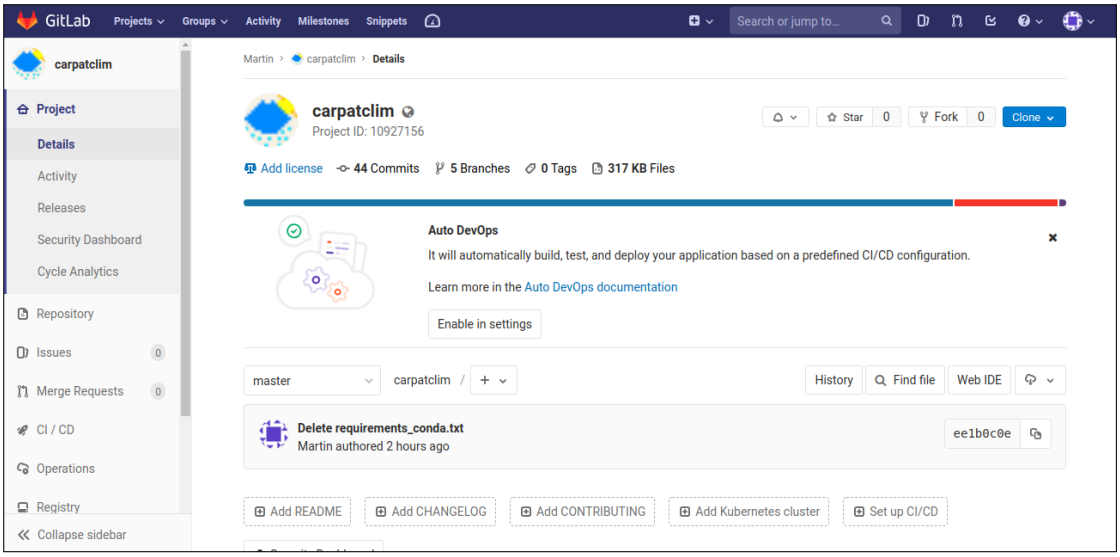

Slika 20: Trenutni vremenski parametri u izabranim gradovima

[13] API ili eng. *application programming interface key* je kod koji se prenosi između kompjuterskih programa, i služi za identifikaciju poziva od strane korisnika

# 6 Dodaci

## Dodatak A

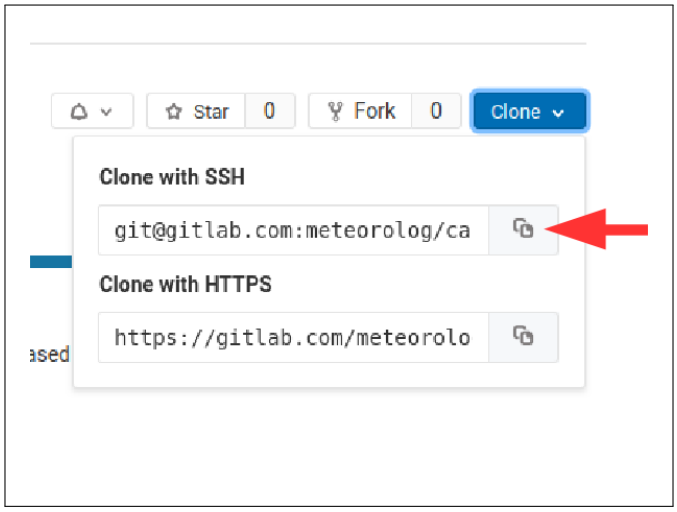
U ovom dodatku će biti prikazan postupak za pruzimanje i podešavanje aplikacije za rad. Pošto je u pitanju veb aplikacija, za njeno funkcionisanje je potreban internet pretraživač (engl. *browser*). Preporučuje se neki od poznatijih internet pretraživača kao što je *Chrome, FireFox, Opera*. Kompletan kod se preuzima sa *GitLab-a*, sa sledeće stranice: `https://gitlab.com/meteorolog/carpatclim`. *GitLab* je veb baziran servis za hosting *Git-a*[14]. Ovaj servis nam omogućava skladištenje i hostovanje za softverske projekte otvorenog koda.




Slika 21: GitLab stranice za CarpatClim projekat

U gornjem desnom uglu se nalazi *Clone* opcija, slika 21. Klikom na ovu opciju pojavljuje se dva opcije za kopiranje URL-a programa. U ovom primeru ćemo kopirati prvu adresu, tako što ćemo kliknuti na dugme označeno strelicom sa slike 22. Nakon ovog postupka idemo u direktorijum u kome želimo razpakovati program i u terminal unesemo sledeću naredbu:

[14]Git se korisit za praćenje promena u bilo kojoj datoteci tokom razvoja softvera. Pogodan je prilikom izgaradnje softvera na kome radi veći broj ljudi, jer se lako uočavaju i dopunjuju promene na izvornom kodu.




Slika 22: Kopiranje URL projekta

```
git clone git@gitlab.com:meteorolog/carpatclim.git
```

nakon čega se pojavljuje direktorijum *carpatclim*, unutar koga se nalazi izvorni kod aplikacije.

```
meteo@meteo-HP-ProBook-4530s:~/Documents$ git clone git@gitlab.com:meteorolog/carpatclim.git
Cloning into 'carpatclim'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 530 (delta 1), reused 0 (delta 0)
Receiving objects: 100% (530/530), 196.25 KiB | 79.00 KiB/s, done.
Resolving deltas: 100% (278/278), done.
Checking connectivity... done.
meteo@meteo-HP-ProBook-4530s:~/Documents$ ls
Books  carpatclim  Desktop  PROGRAM  Tekst
meteo@meteo-HP-ProBook-4530s:~/Documents$
```

Slika 23: Preuzimanje izvornog koda

Nakon preuzimanja koda, potrebno je stvoriti virtuelno okruženje (pogledati poglavlje 3.2). Kako bi instalirali sve potrebne pakete za funkcionisanje aplikacije, unutar *carpatclim* dokumenta se nalazi *requirements.txt* datoteka u kojoj se nalaze svi potrebni paketi. Naredbom:

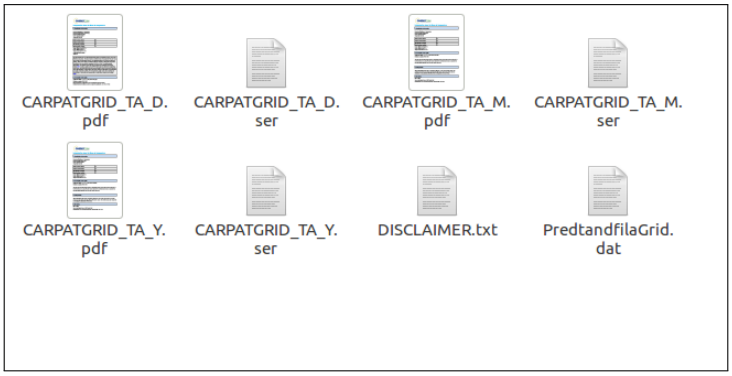
```
pip install -r requirements.txt
```

se ovi paketi instaliraju. Takođe, potrebno je instalirati sledeće biblioteke *Cartopy, proj4, geos* koristeći sledeće naredbe:

```
pip install cartopy
sudo apt-get install libproj-dev proj-data proj-bin
sudo apt-get install libgeos-dev
```

## Dodatak B

Podatke sa *CARPATCLIM* veb sajta preuzimamo sa sledeće stranice: `http://www.carpatclim-eu.org/pages/download/`. Prihvatanjem uslova korišćenja i izborom promenljive, podaci se dobijaju u obliku *zip* formata. U zavisnosti od izbora promenljive, dobijamo datoteke sa vrednostima za godišnje, mesečne dnevne intervale. Za srednju temperaturu, raspakovani podaci izgledaju kao na slici 24.




Slika 24: Podaci za srednje vrednosti temperature

Sa slike vidimo *.pdf* fajlove, u kojima su date informacije o samom projektu. Podaci o vrednostima srednje temperature se nalaze u *CARPATGRID_TA_Y.ser, CARPATGRID_TA_M.ser, CARPATGRID_TA_D.ser* datotekama, gde *Y, M, D* označavaju da li se radi o godišnjim, mesečnim ili dnevnim podacima. Podaci o pozicijama mernih stanica se nalaze u *PredtandfilaGrid.dat* datoteci. Pre nego što stvorim bazu podataka, potrebno je ove podatke kopirati u pogodan direktoriju unutar *carpatclim* direktorijuma. Podaci se kopiraju u *carpatclim/data* direktorijum.

```
meteo@meteo-HP-ProBook-4530s:~/Documents/carpatclim$ ls
carpatclimapp  carpatclimsite  data  docs  manage.py  requirements.txt  script  weather
meteo@meteo-HP-ProBook-4530s:~/Documents/carpatclim$
```

Slika 25: Carpatclim direktorijum

Kada podatke smestimo unutar *data* direktorijuma sledeći korak je stvaranje baze podataka. Za stvaranje baze podataka potrebno je unutar *carpatclim* direktorijuma pokrenuti *sqlite_import.py* skriptu, koristeći sledeću naredbu:

```
python carpatclimapp/sqlite_import.py -i
```

Proces stvaranja baze podataka zavisi od snage računara, kod slabijih konfiguracija zna da traje i do 30 minuta. Kako izgleda proces stvaranja možemo videti na slici 26.

```
(env) meteo@meteo-HP-ProBook-4530s:~/Documents/carpatclim$ python carpatclimapp/sqlite_import.py -i
2019-02-20 13:33:34,001  INFO  Connect to DB carpatclim.sqlite3.
2019-02-20 13:33:34,001  INFO  Begin droping and creating tables.
2019-02-20 13:33:34,022  INFO  Close a DB connection carpatclim.sqlite3.
2019-02-20 13:33:34,023  INFO  Tables dropped and created.
2019-02-20 13:33:34,049  INFO  Connect to DB carpatclim.sqlite3 - table: grid.
2019-02-20 13:33:46,534  INFO  Written to table grid: 1 row(s).
2019-02-20 13:33:46,537  INFO  Connect to DB carpatclim.sqlite3 - table: yearly.
2019-02-20 13:33:50,484  INFO  Written to table yearly: 294750 row(s).
2019-02-20 13:33:50,485  INFO  Written to table yearly: 1 chunks(s).
2019-02-20 13:33:50,485  INFO  DB carpatclim.sqlite3 closed.
2019-02-20 13:33:50,487  INFO  Connect to DB carpatclim.sqlite3 - table: monthly.
2019-02-20 13:34:37,165  INFO  Written to table monthly: 3537000 row(s).
2019-02-20 13:34:37,165  INFO  Written to table monthly: 4 chunks(s).
2019-02-20 13:34:37,167  INFO  DB carpatclim.sqlite3 closed.
2019-02-20 13:34:37,171  INFO  Connect to DB carpatclim.sqlite3 - table: daily.
2019-02-20 14:00:30,078  INFO  Written to table daily: 107654490 row(s).
2019-02-20 14:00:30,078  INFO  Written to table daily: 51 chunks(s).
2019-02-20 14:00:30,079  INFO  DB carpatclim.sqlite3 closed.
```

Slika 26: Proces stvaranja data baze

# Literatura


[1] Bill Lubanović, *Introducing Python*, O'Relly Media, November 2014

[2] Brian Overland, *Python opušteno: vodič za početnike uz koji se osećate pametno*, prevod Milan D.Milošević, CET:Beograd, 2018.

[3] Django Software Foundation, *Django Documentation*. November 10, 2018

[4] Adrian Holovaty, Jacob Kaplan-Moss, *The Django Book Release 2.0*. November 06, 2013

[5] Johnny Wei-Bing Lin, *A Hands-On Introduction to Using Python in the Atmospheric and Oceanic Sciences*, 2012

[6] Mark Lutz, *Learning Pythonn*, 5th Edition-2013

[7] Luciano Ramalho, *Fluent Python*, O'Relly Media, March 2015

[8] Jake VanderPlas, *Python Data Science Handbook*, December 2016

[9] Wes McKinney, *Python for Data Analysis, Data Wrangling with Pandas, NumPy, and IPython*, October 2017: Second Edition

[10] Samuel Dauzon, Aidas Bendoraitis, Arun Ravindran, *Django: Web Development with Python*

[11] Mirna Marković, *Izrada web aplikacije u Django razvojnom okruženju*, Osijek 2014.

[12] Robert Prašnički, *Izrada mobilne i web aplikacije za generisanje qr koda upotrebom Python programskog jezika*, Čakovec 2014.

[13] Nikolina Ivezić, *Razvoj web aplikacija pomoću okruženja Django*, Zagreb 2014.

[14] Tomislav Pavlović, *Upravljanje prostornim i atributnim podacima pomoću GeoDjango tehnologije*, Zagreb 2010.

[15] R. Sluiter, *Interpolation methods for climate data*, De Bilt 2009.

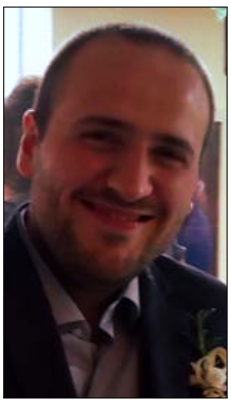
[16] Luming Liang, Dave Hale, *A stable and fast implementation of natural neighbor interpolation*, Center for Wave Phenomena, Colorado School of Mines,Golden, CO 80401, USA April 20, 2010

[17] Elena Đurović, *Algoritmi za konstrukciju Voronoj dijagrama i Deloneove triangulacije*, 2008.

[18] Boris Pein, *Deloneove triangulacije i Voronojevi dijagrami*, Osijek 2012.

[19] MetPy Developers, *MetPy Documentation Release 0.4.2*, November 18, 2016

# Biografija



Martin Petraš, rođen 2. marta 1990. godine u Novom Sadu. Osnovnu školu ”Jan Kolar” je završio u Selenči, nakon čega upisuje srednju medicinsku školu ”Dr Ružica Rip” u Somboru. Po završetku srednje škole, 2009. godine upisuje Prirodno-matematički fakultet u Novom Sadu, smer fizika-meteorologija na Departmanu za fiziku. Osnovne akademske studije je završio u septembru 2017. godine i iste godine upisuje Master studije na istom Fakultetu, isti smer.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

| | |
|---|---|
| *Redni broj:*<br>**RBR** | |
| *Identifikacioni broj:*<br>**IBR** | |
| *Tip dokumentacije:*<br>**TD** | Monografska dokumentacija |
| *Tip zapisa:*<br>**TZ** | Tekstualni štampani materijal |
| *Vrsta rada:*<br>**VR** | Master rad |
| *Autor:*<br>**AU** | Martin Petraš |
| *Mentor:*<br>**MN** | dr Ilija Arsenić |
| *Naslov rada:*<br>**NR** | Upotreba Python programskog okruženja u obradi "CARPATCLIM" klimatskih podataka |
| *Jezik publikacije:*<br>**JP** | srpski (latinica) |
| *Jezik izvoda:*<br>**JI** | srpski/engleski |
| *Zemlja publikovanja:*<br>**ZP** | Republika Srbija |
| *Uže geografsko područje:*<br>**UGP** | Vojvodina |
| *Godina:*<br>**GO** | 2019 |
| *Izdavač:*<br>**IZ** | Autorski reprint |
| *Mesto i adresa:*<br>**MA** | Prirodno-matematički fakultet, Trg Dositeja Obradovića 4, Novi Sad |
| *Fizički opis rada:*<br>**FO** | 6 poglavlja/ 50 stranica/ 19 referenci/ 26 slika |
| *Naučna oblast:*<br>**NO** | Fizika |
| Naučna disciplina:<br>**ND** | Meteorologija |
| *Predmetna odrednica/ ključne reči:*<br>**PO**<br>**UDK** | Numerička obrada podataka, klima, Python |

| | |
|---|---|
| *Čuva se:*<br>**ČU** | Biblioteka departmana za fiziku, PMF-a u Novom Sadu |
| *Važna napomena:*<br>**VN** | nema |
| *Izvod:*<br>**IZ** | Rad se bavi interpolacijom klimatskih podataka preuzetih sa Carpatclim projekta. Ovaj projekat se bavio prikupljanjem podataka za oblast oko Karpatskih planina, u vremenskom periodu od 1961. godine do 2010. godine. U svrhu interpolacije stvorena je veb aplikacija, koja je pisana u Python programskom jeziku koristeći Django okruženje. |
| *Datum prihvatanja teme od NN veća:*<br>**DP** | |
| *Datum odbrane:*<br>**DO** | Mart 2019 |
| *Članovi komisije:*<br>**KO** | |
| *Predsednik:* | dr Miodrag Krmar |
| *član:* | dr Ilija Arsenić |
| *član:* | dr Zorica Podraščanin |

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCE AND MATHEMATICS

KEY WORDS DOCUMENTATION

| | |
|---|---|
| *Accession number:*<br>**ANO** | |
| *Identification number:*<br>**INO** | |
| *Document type:*<br>**DT** | Monograph publication |
| *Type of record:*<br>**TR** | Textual printed material |
| *Content code:*<br>**CC** | Master thesis |
| *Author:*<br>**AU** | Martin Petraš |
| *Mentor/comentor:*<br>**MN** | Prof. dr Ilija Arsenić |
| *Title:*<br>**TI** | Using the Python programming environment in the processing of "CARPATCLIM" climatic data |
| *Language of text:*<br>**LT** | Serbian (Latin) |
| *Language of abstract:*<br>**LA** | English |
| *Country of publication:*<br>**CP** | Serbia |
| *Locality of publication:*<br>**LP** | Vojvodina |
| *Publication year:*<br>**PY** | 2019 |
| *Publisher:*<br>**PU** | Author's reprint |
| *Publication place:*<br>**PP** | Faculty of Science and Mathematics, Trg Dositeja Obradovića 4, Novi Sad |
| *Physical description:*<br>**PD** | 6 chapter/ 50 pages/ 19 literature/ 26 pictures |
| *Scientific field:*<br>**SF** | Physics |
| *Scientific discipline:*<br>**SD** | Meteorology |
| *Subject/ Key words:*<br>**SKW**<br>**UC** | Numerical data processing, climate, Python |

| | |
|---|---|
| *Holding data:*<br>**HD** | Library of Department of Physics, Trg Dositeja Obradovića 4 |
| *Note:*<br>**N** | none |
| *Abstract:*<br>**AB** | In this master thesis we deals with the interpolation of climatic data taken from the Carpatclim project. This project dealt with collecting data for the area around the Carpathian Mountains in the period from 1961 to 2010. For the purpose of interpolation, a web application was created, written in Python programming language using the Django framework. |
| *Accepted by the Scientific Board:*<br>**ASB** | |
| *Defended on:*<br>**DE** | March, 2019 |
| *Thesis defend board:*<br>**DB** | |
| *President:* | Ph.D Miodrag Krmar |
| *Member:* | Ph.D Ilija Arsenić |
| *Member:* | Ph.D Zorica Podraščanin |