Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

Projektna dokumentacija

Dokumentacija pri Principih programskih jezikov

Mentorja:
Matej Moravec, mag.
doc. dr. Niko Lukač

Avtorji:
Luka Golčman (vodja)
Jure Dolar
Matija Šiško

Maribor, junij 2022

1 UVOD

Namen dokumenta je predstavitev projekta in značilnosti implementacije, ki je nastajala v okviru projekta v 2. letniku programa RIT UN, predvsem pri predmetu Principi programskih jezikov. V besedilu so povzete zahteve naloge in celotna integracija v projekt, opisane so naše rešitve, ter implementacija, vse omenjeno pa je predstavljeno tudi za čim lažjo uporabo uporabnika.

2 ZAHTEVE IN NAČRTOVANJE

Glavna zahteva celotnega projekta je bila izdelava digitalnega dvojčka mesta Maribor. V naši skupini smo se odločili, da se pri tem osredotočimo na onesnaženost zraka, nevarne delce in njihov grafični prikaz. Celotno delo je bilo razdeljeno na 4 predmete, od katerih smo pri vsakem pokrili del implementacije, ter jih na koncu združili v celoto. Pri ostalih predmetih smo izdelali podatkovno bazo, grafični vmesnik, spletno stran, ter vzpostavili gostovanje, pri principih programskih jezikov pa smo izdelali programa za pridobivanje in generiranje podatkov ter grafični vmesnik za lažje delo z omenjenima programoma.

Po zahtevah na navodilih, našem modelu aplikacije in dostopnosti orodij smo se nato odločili katera orodja bomo uporabili. Za izdelavo programa za pridobivanje in razčlenjevanje podatkov, ter za program za generacijo naključnih podatkov smo zaradi dobrega poznavanja in jasnosti kode uporabili Python. Vse skupaj pa smo nato povezali in predstavili z grafičnim vmesnikom implementiranim v programskem jeziku Kotlin. Ta nam namreč omogoča lažje delo s programoma in prijaznejšo uporabniško izkušnjo.

3 IMPLEMENTACIJA

Kot omenjeno smo rešitev implementirali v programskih jezikih Python in Kotlin, ter jo povezali s programom implementiranim pri predmetu Spletno programiranje, natančneje s tamkajšnjim API-jem. Tako smo zagotovili pravilno shranjevanje podatkov v podatkovno bazo MongoDB in povezanost rešitve z ostalimi deli projekta.

3.1 PODATKOVNA BAZA IN PODATKI

Podatke za aplikacijo trenutno pridobivamo primarno iz spletne strani Agencije Republike Slovenije za okolje, ki na svojih spletnih straneh vsako uro objavi nove podatke meritev njihovih senzorjev o delcih v ozračju. Podatki se nahajajo na spletnem naslovu https://www.arso.gov.si/xml/zrak/ones_zrak_urni_podatki_zadnji.xml, v .xml obliki. V dokumentu je podano ime merilnega mesta, datum začetka in datum konca meritve, ter vrednosti za sedem različnih urnih koncentracij snovi v ozračju, ki so vidne na sliki Slika 3.1. Izgled .xml datoteke, pridobljene s spletne strani, v obliki v kateri se posreduje v program, pa je lahko viden na sliki Slika 3.2.

11.3.2022 - ver 1.3

Spremembe:

- elementu <postaja> dodan podelement **<benzen>**
- atribut verzija spremenjen v 1.3 (**<arsopodatki verzija="1.3" >**)

Podelementi elementa <postaja> po spremembi:

<merilno_mesto> - ime merilnega mesta
<datum_od> - datum začetka meritve
<datum_do> - datum konca meritve
<so2 > - urna koncentracija žveplovega dioksida v µg/m³
<co> - urna koncentracija ogljikovega monoksida v mg/m³
<o3> - urna koncentracija ozona v µg/m³
<no2> - urna koncentracija dušikovega dioksida v µg/m³
<pm10> - urna koncentracija delcev pm10 v µg/m³
<pm2.5> - urna koncentracija delcev pm2,5 v µg/m³
<benzen> - urna koncentracija benzena (C_6H_6) v µg/m³

Slika 3.1: Prikaz ARSO-vega opisa strukture .xml datoteke

```
▼<arsopodatki verzija="1.3">
   <vir>Agencija RS za okolje</vir>
   <predlagan_zajem>48 minut čez polno uro</predlagan_zajem>
   <predlagan_zajem_perioda>60 min</predlagan_zajem_perioda>
   <datum_priprave>2022-06-05 20:35</datum_priprave>
  ▼<postaja sifra="E21" ge_dolzina="14.517454" ge_sirina="46.065851" nadm_visina="299">
     <merilno_mesto>LJ Bežigrad</merilno_mesto>
     <datum_od>2022-06-05 19:00</datum_od>
     <datum_do>2022-06-05 20:00</datum_do>
     <o3>111</o3>
     <no2>6</no2>
     <pm10>33</pm10>
     <pm2.5>13</pm2.5>
     <benzen>0.1</benzen>
   </postaja>
  ▼<postaja sifra="E405" ge_dolzina="14.491849" ge_sirina="46.072399" nadm_visina="305">
     <merilno_mesto>LJ Celovška</merilno_mesto>
     <datum_od>2022-06-05 19:00</datum_od>
     <datum_do>2022-06-05 20:00</datum_do>
     <no2>18</no2>
     <pm10>36</pm10>
     <pm2.5>14</pm2.5>
   </postaja>
  ▼<postaja sifra="E404" ge_dolzina="14.494001" ge_sirina="46.037791" nadm_visina="293">
     <merilno_mesto>LJ Vič</merilno_mesto>
     <datum_od>2022-06-05 19:00</datum_od>
     <datum_do>2022-06-05 20:00</datum_do>
     <pm10>42</pm10>
     <pm2.5>13</pm2.5>
   </postaja>
  ▼<postaja sifra="E417" ge_dolzina="14.366963" ge_sirina="46.242115" nadm_visina="388">
     <merilno_mesto>Kranj</merilno_mesto>
     <datum_od>2022-06-05 19:00</datum_od>
     <datum_do>2022-06-05 20:00</datum_do>
     <pm10>41</pm10>
     <pm2.5>14</pm2.5>
   </postaja>
  ▼<postaja sifra="E22" ge_dolzina="15.656191" ge_sirina="46.559202" nadm_visina="270">
     <merilno_mesto>MB Titova</merilno_mesto>
     <datum_od>2022-06-05 19:00</datum_od>
     <datum_do>2022-06-05 20:00</datum_do>
     <no2>31</no2>
     <pm10>20</pm10>
     <pm2.5>13</pm2.5>
     <benzen>0.7</benzen>
   </postaja>
```

Slika 3.2: Izgled .xml datoteke, pridobljen s spletne strani ARSO

Zgoraj opisanim podatkom smo nato priredili našo podatkovno bazo v MongoDB, da smo omogočili shranjevanje vseh podatkov, dodajanje lastnih podatkov, ter podatkov povzetih iz drugih virov. V podatkovni bazi tako shranjujemo podatke o imenu in regiji merilne naprave, geografske podatke, vir podatkov, njihovo zanesljivost, koncentracije delcev in časovne podatke. Vsi omenjeni podatki so prikazani v kodi spodaj, kot so bili definirani pri implementaciji. Izgled podatkov v podatkovni bazi je viden na sliki Slika 3.3.

```
MODEL:
air pollution:
    id (automatically selected),
    name (name of the sensor taking the data),
    region (the region in which the sensor is situated in),
    coordinates {
        longitude,
        latitude,
        altitude
    } (the exact positional coordinates of the sensor),
    source (source of the data - how we are connecting to the sensor (can be from arso, our
own, or made up)),
    reliability (min: 0, max: 100 - a made-up source or a faulty sensor would have lower
reliability, gov data would have higher),
    date_time (date and time of when this entry was created),
    concentrations {
        PM10, (particulates with a diameter of 10 micrometers or less)
        PM2_5, (particulates with a diameter of 2,5 micrometers or less)
        SO2, (sulfur dioxide)
        CO, (carbon monoxide)
        O3, (ozone)
        NO2, (nitrogen dioxide)
        C6H6, (benzene)
    } (the actual important part: all the data on the different parts we collect, if sensor
cannot measure null will be written)
    date_time_of_measurement {
        measuring_start,
        measuring_end,
        length (can be hourly (1) or daily (24))
    } (the time at which this data was started to be taken and at which point it ended)
```


```
_id: ObjectId('6277dafe94a0b4e43ab49dea')
name: "CE bolnica"
region: "Celje"
geo: Object
  type: "Point"
  coordinates: Array
    0: 15.267305
    1: 46.234818
  _id: ObjectId('6277dafe94a0b4e43ab49deb')
source: "arso"
reliability: 100
concentrations: Object
  PM10: 23
  PM2_5: 16
  SO2: 3
  CO: null
  O3: 54
  NO2: 9
  C6H6: null
  _id: ObjectId('6277dafe94a0b4e43ab49dec')
date_time_of_measurement: Object
  measuring_start: 2022-05-08T08:00:00.000+00:00
  measuring_end: 2022-05-08T09:00:00.000+00:00
  _id: ObjectId('6277dafe94a0b4e43ab49ded')
dateTime: 2022-05-08T15:00:14.752+00:00
__v: 0
```


Slika 3.3: Prikaz podatkov v podatkovni bazi MongoDB

3.2 PROGRAM ZA PRIDOBIVANJE PODATKOV

Kodi za razčlenjevanje in generacijo sta bili, kot omenjeno pripravljeni v programskem jeziku Python, za lažjo implementacijo in razumljivejšo kodo. Knjižnice, uporabljene pri implementaciji so: random (generacija naključnih števil), sys (omogoča sistemske ukaze), json (knjižnica za delo z jezikom json), requests (knjižnica za pošiljanje ukazov) in xml (knjižnica za razbijanje .xml datotek).

Programska koda najprej kliče ARSO spletno stran, iz katere pridobi .xml datoteko. Vsebovani podatki se nato, s pomočjo razčlenjevanja vidnega na sliki Slika 3.4 pretvorijo iz njihovega v naš format. Vse informacije se shranjujejo v slovar (spremenljivka a), po njihovi obdelavi pa jih dodamo v polje slovarjev (spremenljivka x). Ko se vsi podatki uspešno obdelajo in so dodani v polje, se dodajo podatki, ki niso pridobljeni iz datoteke.

```
if(source=="arso"):
    if (child.text == None and child.attrib == {}):
        print("NAPAKA1")
        exitcode = 1
    elif (child.text == None and child.attrib != {}):
        #print("child tag: ", child.tag, "child attrib: ", child.attrib)
        a["longtitude"] = child.attrib["ge_dolzina"]                #ge dolzina
        a["latitude"] = child.attrib["ge_sirina"]                   #ge sirina
    elif (child.attrib == {} and child.text != None):
        if(child.tag=="vir"):
            continue
        elif(child.tag=="predlagan_zajem"):
            continue
        elif (child.tag == "predlagan_zajem_perioda"):
            continue
        elif (child.tag == "datum_priprave"):
            continue
        else:
            print("child tag: ", child.tag, "child text: ", child.text)
            print("NAPAKA2")
            exitcode = 2
            break
    else:
        a["longtitude"] = child.attrib[1]               # ge dolzina
        a["latitude"] = child.attrib[2]                 # ge sirina
        #print(child.tag, child.attrib, child.text)
```

Slika 3.4: Razčlenjevanje podatkov, iskanje geografskih informacij

Vsi omenjeni podatki se nato en po en s klici API, vidnimi na sliki Slika 3.5, pošljejo v podatkovno bazo. V kolikor se med delom zgodi napaka, se že vneseni podatki ne brišejo, ampak se ohranijo v podatkovni bazi. Po končanem delu program vrne eno izmed pred nastavljenih kod z informacijo o zaključku, ki programerju v primeru napake pomaga ugotoviti, kateri del kode se ni izvedel pravilno. Možne napake in njihovi opisi so vidni na sliki Slika 3.6.

```
    # Priprava json in klic API
    try:
        body = json.dumps(i)
        headers = {'Content-Type': 'application/json',
                   'Authorization': 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjp7InVz

        response = requests.post(target, headers=headers, data=body, verify=False)
        #print(response.json())
        #print(json.loads(response.json()))
        if(json.dumps(response.json()).find("validation failed")!=-1):
            print("NAPAKA V APIju")
            exitcode = 7
            print(response.json())
            print(i)
    except Exception as e:
        print("NAPAKA POŠILJANJE V API")
        print(i)
        print(e)
        exitcode = 8
        break

print(x)
sys.exit(exitcode)
```

Slika 3.5: Klic API in končni izhod programa

```
# USPEH
0 - uspesno izveden program

# NEMOGOČI NEUSPEHI
1 - Nemogoča napaka v child (razen če (kar bodo prej ali slej) updatajo strukturo podatkov)
2 - Nemogoča napaka v child (razen če (kar bodo prej ali slej) updatajo strukturo podatkov)
3 - Nemogoča napaka v child2 (razen če (kar bodo prej ali slej) updatajo strukturo podatkov)
4 - Nemogoča napaka v child2 (razen če (kar bodo prej ali slej) updatajo strukturo podatkov)
5 - Nemogoča napaka v child2 (razen če (kar bodo prej ali slej) updatajo strukturo podatkov)

# POMEMBNI NEUSPEHI
6 - Podatki niso prišli iz poznanega vira
7 - Napaka se je zgodila na APIjevi strani (npr. model je zavrnil zaradi napačnega tipa)
8 - Napaka pri pošiljanju APIju (npr. ni konekcije)
```

Slika 3.6: Možni izhodi programa, 0 predstavlja uspešno izveden program, vsi ostali pa napako med izvajanjem

3.3 PROGRAM ZA GENERACIJO PODATKOV

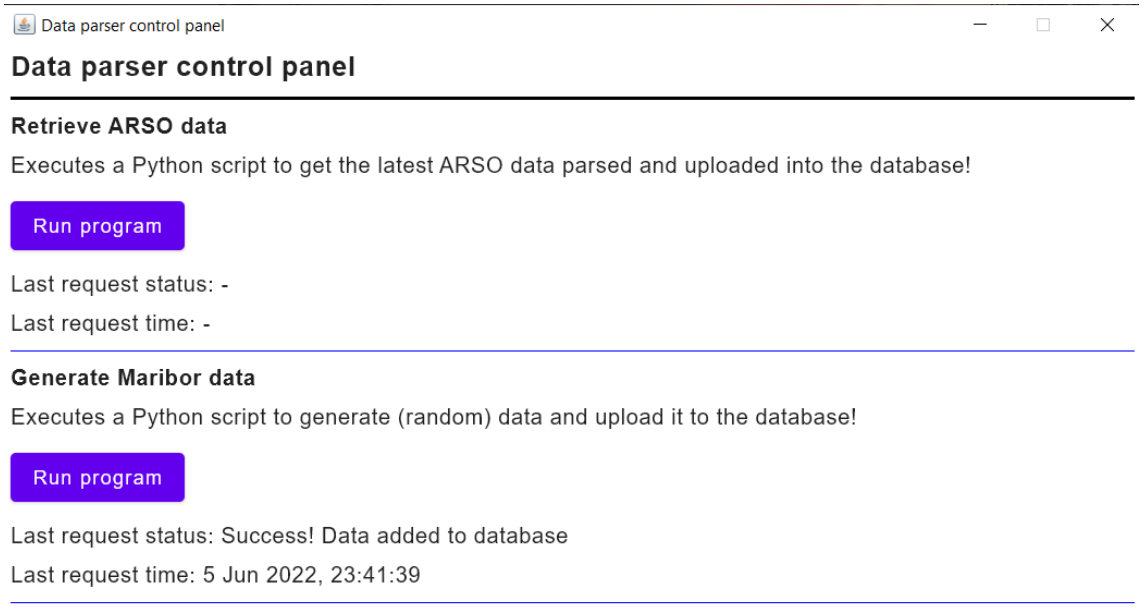
Programska koda, v nasprotju z predhodnim programom podatkov ne pridobiva iz uradnih virov, temveč jih le generira (Slika 3.7) na podlagi pred nastavljenih privzetih vrednosti. Program tako generira 3 različne izmišljene postaje iz okolice Maribora, jim pripiše generirane vrednosti, nastavi nizko zaupnost podatkov in jih nato, s pomočjo API klicev pošlje v podatkovno bazo. Tudi ta program ima implementirane kode napake za lažje iskanje in identifikacijo napake, v primeru nedelovanja.

```
PM10 = [random.randrange(5,30), random.randrange(5,30), random.randrange(5,30)]
PM2_5 = [random.randrange(5,30), random.randrange(5,30), random.randrange(5,30)]
SO2 = [random.randrange(1,7), random.randrange(1,7), random.randrange(1,7)]
O3 = [random.randrange(50,100), random.randrange(50,100), random.randrange(50,100)]
NO2 = [random.randrange(5,10), random.randrange(5,10), random.randrange(5,10)]
```

Slika 3.7: Naključna generacija podatkov

3.4 GRAFIČNI PROGRAMSKI VMESNIK

V nasprotju s programoma za delo s podatki in povezavo z API-jem pa je bil grafični vmesnik implementiran v jeziku Kotlin s pomočjo ogrodja Compose Multiplatform. Omenjeno orodje nam je omogočilo preprostejšo izgradnjo osnovnega in vizualno prijaznega uporabniškega vmesnika, namenjenega lažjemu delo s programoma. Kot vidno na sliki Slika 3.8, nam program omogoča zagon Python skript z pritiskom na gumb, izpis informacij o stanju in času zadnjega vnosa, opozori pa nas tudi v primeru napak. Program pa ima implementirana tudi pozivna okna (Slika 3.9), ki v določenih primerih uporabnika še dodatno opozarjajo in prosijo za potrditev.



Slika 3.8: Glavno okno grafičnega vmesnika programa za nadzor skript

Slika 3.9: Pozivno okno uporabniku, namenjeno potrditvi zapiranja programa