Internship Report at the Mihajlo Pupin Institute

In this report, I will briefly present the results of my work during the internship at the Mihajlo Pupin Institute, specifically the part of the internship related to programming, i.e., the design and implementation of WebScraping software. By work results, I primarily mean the Python scripts created during my work at the Institute, the conceptual solution for the software, and the instructions on how to further develop the system. I would like to emphasize that working on this project was a very important and useful experience for me, both because I got acquainted with elements of web scraping in Python, and because I independently designed the entire system, and with the help of my mentor's suggestions and advice, brought it to a state where it is practically applicable, useful, and easy to use.

I had the opportunity to practically apply some university knowledge from courses such as object-oriented programming and to understand in practice what modularity and scalability mean, as well as object-oriented and algorithmic decomposition of a system. Currently, the system reads information from about 150 river stations in Serbia for 4 basins, possesses a global system operation history that records some issues occurring outside the readers themselves, keeps a history for each active reader in the system, and maintains a record of each activation (without history). Special emphasis during work on the system was placed on monitoring operation history, possible errors, and exceptions, all aimed at increasing the robustness of the software, as it will run autonomously on a machine. The system is very easily extendable, which will be outlined later in the report. The next improvements are also related to connecting with the database. The system is adapted for both Windows and Linux.

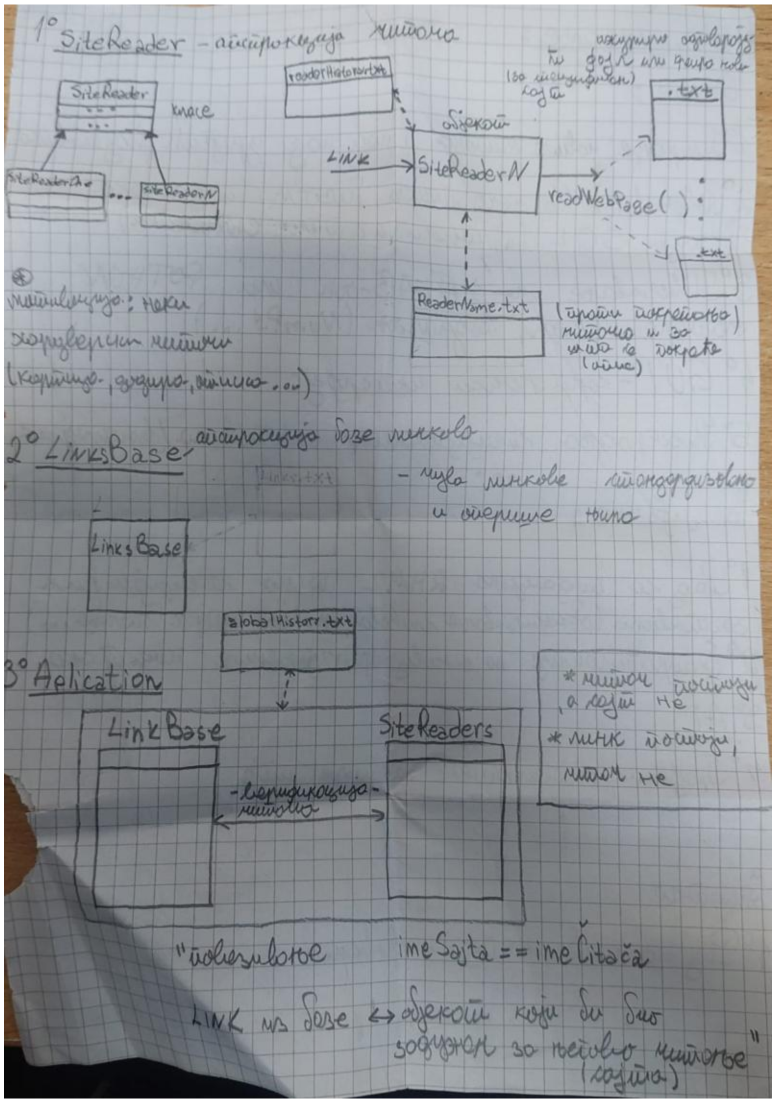
I am very grateful to my mentor, Đorđe Koprivica, who guided me through this part of the internship and introduced me to the Mihajlo Pupin Institute.

Intern: Rastko Lazarević, 3rd year at ETF, Signals and Systems module
Email: lazarevicc.r@gmail.com
Mentor: Đorđe Koprivica
Date: October(2024) - January(2025)

List of all files:

- source.py
- Interface.py
- Aplication.py
- LinksBase.py
- SiteReader.py
- RhmzReaderFunctions.py
- MyExceptions.py

Initial, conceptual system design.


1° SiteReader - апстракција читача
SiteReader
класе
SiteReaderOne
SiteReaderN
readerHistory.txt
ажурира одговарајући фајл или прави нови (за специфичан) сајт
објекат
LINK
SiteReaderN
readWebPage( )
.txt
.txt
ReaderName.txt
(прати покретања читача и за шта се покреће (опис))
2° LinksBase апстракција базе линкова
Links.txt
- чува линкове стандардизовано
LinksBase
globalHistory.txt
3° Aplication
LinkBase
SiteReaders
-верификација читача-
* читач постоји, а сајт не
* линк постоји, читач не
"повезивање imeSajta == imeČitača
LINK из базе ↔ објекат који би био задужен за његово читање (сајта)"

This is some basic, initial idea of the system. Here it is necessary to note very important parts of the system **Interface.py**, from which we draw many important pieces of information and which greatly facilitates handling the software, and **MyExceptions.py**, where the entire system operation history resides. I would also highlight the basic idea of how folders should be organized in which the results are written.

......→ Results → ReaderResult1 (RHMZ) → Folder1 →……→ data_files.txt

⋮ ⋮

⋮ FolderM →......→ data_files.txt

⋮ Reader1OperationTracking.txt

⋮ Reader1History.txt

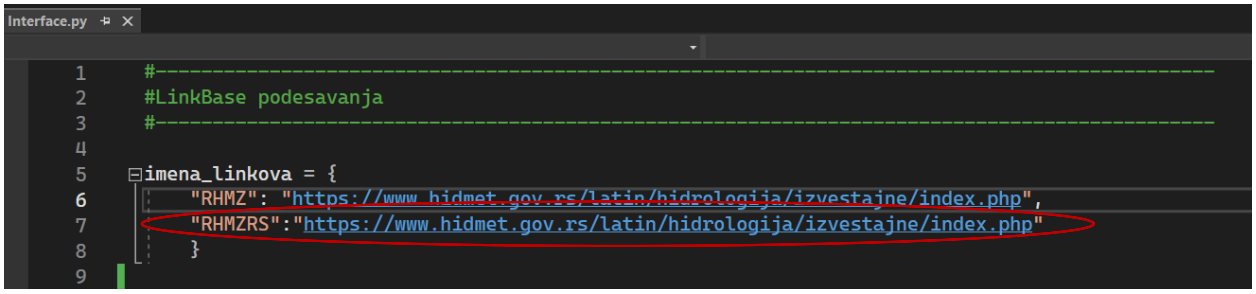
→ ReaderResultN (RHMZRS) → (like Reader1)

→ GlobalHistory

Instructions for Adding a New Reader to the System

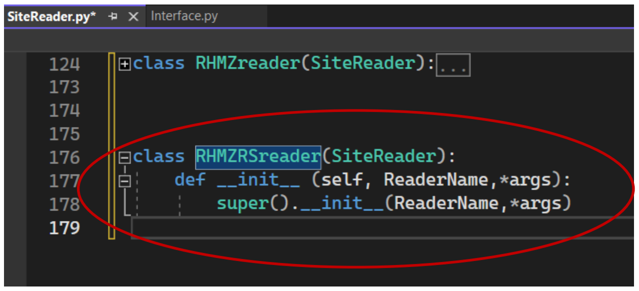
Adding a Reader for RHMZ.RS (illustrative example)

1. In **Interface.py**, add the website we want to read to the names_links dictionary, paying special attention to the key itself.



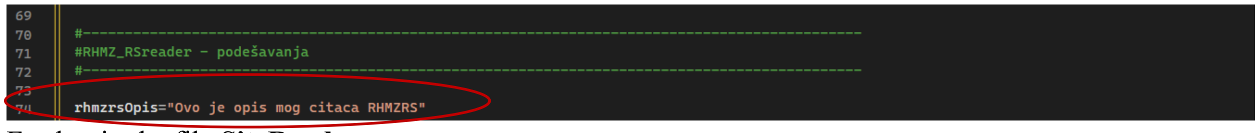
```
#-----------------------------------------------------------------------------------------------
#LinkBase podesavanja
#-----------------------------------------------------------------------------------------------

imena_linkova = {
    "RHMZ": "https://www.hidmet.gov.rs/latin/hidrologija/izvestajne/index.php",
    "RHMZRS":"https://www.hidmet.gov.rs/latin/hidrologija/izvestajne/index.php"
    }
```

2. In **SiteReader.py**, create the RHMZRSreader class, ensuring that the constructor of the base class is also called.



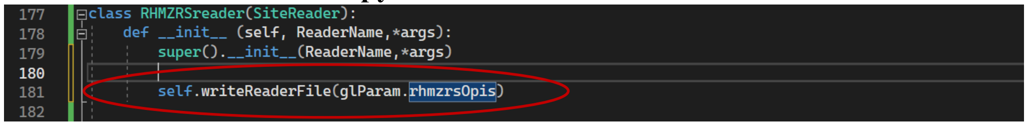
```
class RHMZreader(SiteReader):...


class RHMZRSreader(SiteReader):
    def __init__ (self, ReaderName,*args):
        super().__init__(ReaderName,*args)
```

3. **When creating a new reader, all necessary descriptions for it can be placed in Interface.py.**



```
#-----------------------------------------------------------------------------
#RHMZ_RSreader - podešavanja
#-----------------------------------------------------------------------------

rhmzrsOpis="Ovo je opis mog citaca RHMZRS"
```

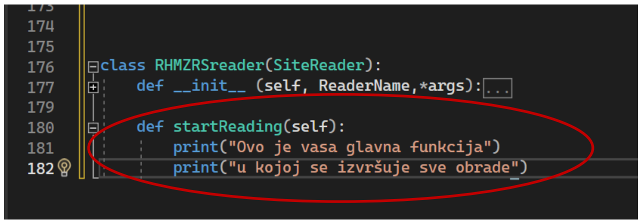
Further in the file **SiteReader.py**



```
class RHMZRSreader(SiteReader):
    def __init__ (self, ReaderName,*args):
        super().__init__(ReaderName,*args)

        self.writeReaderFile(glParam.rhmzrsOpis)
```

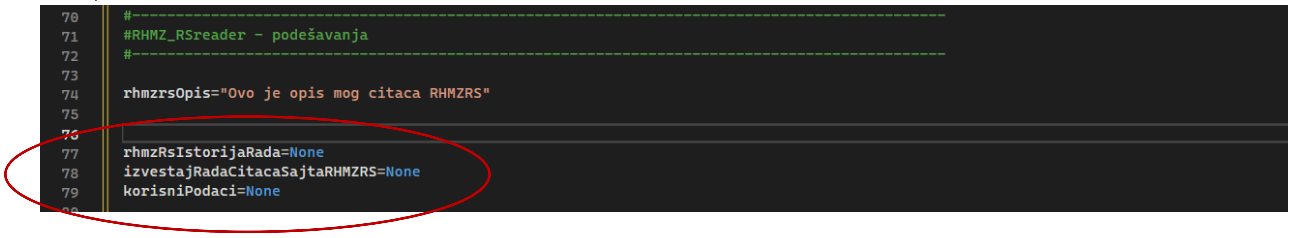
In addition, new fields can be formed, depending on the needs.

4. You are redefining **the startReading(self) function** (and other abstract functions, if any) from the base class! This is **the most important step**, in it all the processing of the **RHMZRSreader reader must be done.**



```
class RHMZRSreader(SiteReader):
    def __init__ (self, ReaderName,*args):...

    def startReading(self):
        print("Ovo je vasa glavna funkcija")
        print("u kojoj se izvršuje sve obrade")
```

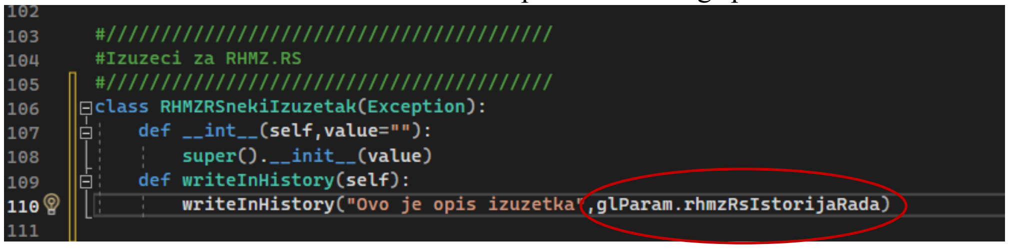
5. If necessary, some additional functions can also be overridden if useful; new functions can also be added as needed, or if there are many such functions, a new file with functions can be created, as is the case with **RhmzReaderFunctions.py** for RHMZreader.
6. All new file paths and folder paths used to store data from RHMZRSreader need to be recorded in **Interface.py** and imported from there. It is necessary to provide the path for the RHMZRSreader operation history file/folder, the path for monitoring the reader's operation, and the path where the collected data will be stored.
   First, we define them.



```
#-------------------------------------------------------------------------------------------
#RHMZ_RSreader - podešavanja
#-------------------------------------------------------------------------------------------

rhmzrsOpis="Ovo je opis mog citaca RHMZRS"


rhmzRsIstorijaRada=None
izvestajRadaCitacaSajtaRHMZRS=None
korisniPodaci=None
```

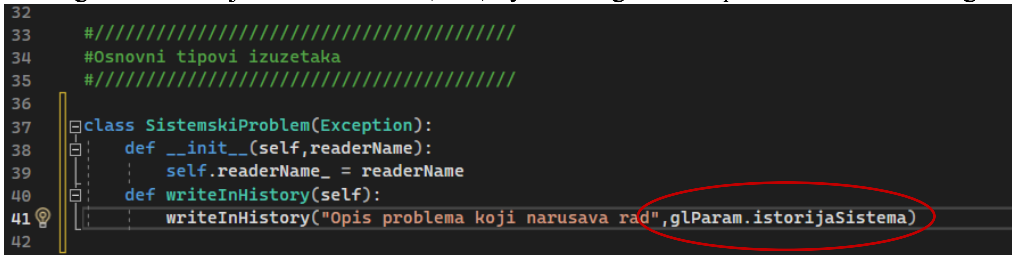
   Then, we also add them to the initializeAndSet() function in **Interface.py**, as has already been done. This is done for initialization depending on the system we are working on.

7. Working with exceptions is closely linked to writing to the history file. We provide an example of how to define a new exception if necessary. The goal is for the exception description to be recorded in the file that tracks the reader's operation and logs possible issues.



```
#/////////////////////////////////////////
#Izuzeci za RHMZ.RS
#/////////////////////////////////////////
class RHMZRSnekiIzuzetak(Exception):
    def __int__(self,value=""):
        super().__init__(value)
    def writeInHistory(self):
        writeInHistory("Ovo je opis izuzetka",glParam.rhmzRsIstorijaRada)
```

8. If, during further system expansion, problems or specific situations affecting the entire system are observed, they should be recorded in the global system operation history. This can be done by writing to the IstorijaSistema.txt file, i.e., by creating an exception in the following way.

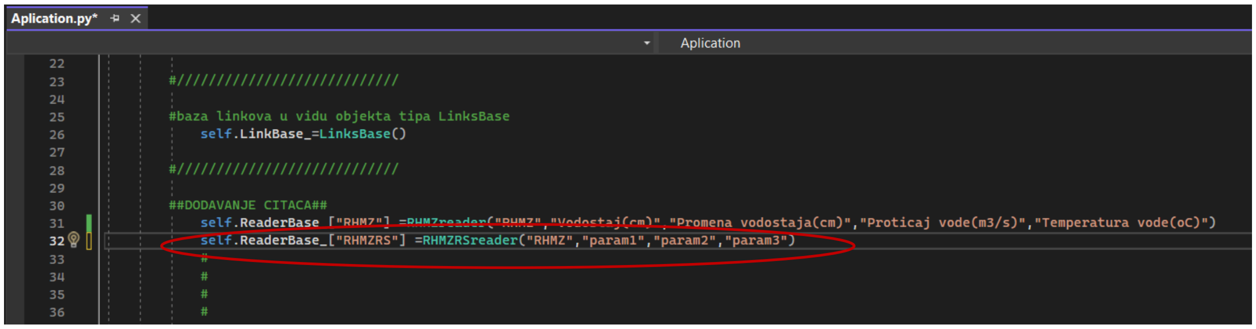


```
#////////////////////////////////////////
#Osnovni tipovi izuzetaka
#////////////////////////////////////////

class SistemskiProblem(Exception):
    def __init__(self,readerName):
        self.readerName_ = readerName
    def writeInHistory(self):
        writeInHistory("Opis problema koji narusava rad",glParam.istorijaSistema)
```

These are just some of the ways an exception can be defined, which were useful to me to achieve the desired result. It does not necessarily have to be done this way.

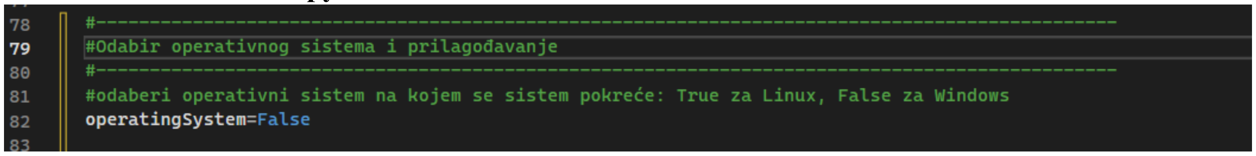
The important thing is that everything is written to files using the writeInHistory(description, filepath) function.

The remaining step is to integrate the reader into the system operation. This is done as follows.



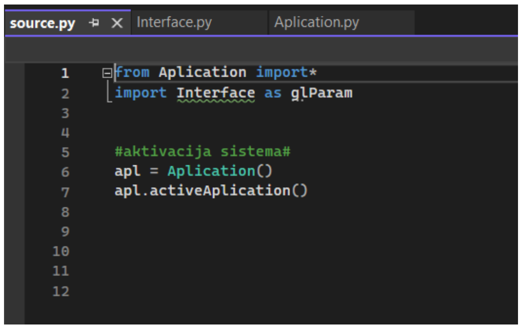
```
    #////////////////////////////

    #baza linkova u vidu objekta tipa LinksBase
        self.LinkBase_=LinksBase()

    #////////////////////////////

    ##DODAVANJE CITACA##
        self.ReaderBase_["RHMZ"] =RHMZreader("RHMZ","Vodostaj(cm)","Promena vodostaja(cm)","Proticaj vode(m3/s)","Temperatura vode(oC)")
        self.ReaderBase_["RHMZRS"] =RHMZRSreader("RHMZ","param1","param2","param3")
        #
        #
        #
        #
```

Here, be very careful!!!!
The key under which we add the new reader must match the link key from point 1.
With this, the system is ready to operate. It is still necessary to select the operating system before starting.
This is done in **Interface.py**.



```
#----------------------------------------------------------------------------------------------
#Odabir operativnog sistema i prilagođavanje
#----------------------------------------------------------------------------------------------
#odaberi operativni sistem na kojem se sistem pokreće: True za Linux, False za Windows
operatingSystem=False
```

The system is launched from the **source.py** script.



```
from Aplication import*
import Interface as glParam


#aktivacija sistema#
apl = Aplication()
apl.activeAplication()
```