# Yonder Junior DevOps

# -Technical Assessment-

Bichilie Tudor Ovidiu

Contents

[Start Docker Container 3](#_Toc160922934)

[Requests 3](#_Toc160922935)

[Data Filtering 3](#_Toc160922936)

[Excel Generator 4](#_Toc160922937)

[Application 5](#_Toc160922938)

[Theoretical Part 6](#_Toc160922939)

# Start Docker Container

To start the Docker container, I ran the command:

“docker run -p 30000:8080 IMAGE\_NAME”

Break down:

* Docker run: for initializing the image and running it in a container
* -p 30000:8080: “p” for specifying that it will receive a port, “30000” is the host port and “8080” is the port within docker that runs the image
* IMAGE\_NAME: replaced with the actual image name to specify what to run

Now the endpoints can be accessed on <http://localhost:30000>

* Mention:

I included a requirements.txt file where I listed the pip external packages used, for easier virtual environment configuration

# Requests

First things first, I added the python package “requests” to my virtual environment for easier endpoint access.

After that I created a class named “DataExtractor” that has an “extract” method which makes a GET method on the endpoint provided.

Also, I included parameters for the GET method to be able to specify how many data points to be consumed.

The request is surrounded by a try-except block and a verification of the status code returned for making it easier in case of debugging.

# Data Filtering

This functionality is present in the same class with the requests, in “DataExtractor”.

In this step I had to decide between 2 methods:

* Using filter()
* Using list comprehension

Analyzing the problem, I found it is better to use list comprehension, despite the fact that it is less versatile, because it is not necessarily to make complicated filtering on the data and the comprehension is faster as well as more readable.

There are 3 methods associated to the 3 tasks provided in the PDF file and also I added a custom decorator for simpler future access to the methods, those will be discussed with more detail in the Application breakdown.

Mention:

I used the built-in library “datetime” for date handling in the second task.

# Excel Generator

I created a class named “ExcelCreator”.

External packages used:

* Pandas
* Xlsxwriter

The class initializes a “pandas” data frame and operates with it.

As far as I am concerned, the packages are more or less plug and play, so I added some methods for styling the excel file and to make it more readable.

The class contains:

* Method for adjusting the column length to fit the entire string in a shell
* Method for creating borders for shells based on what they describe
* Method for generating the final format file
* Fail switch in case of no data is provided or an incorrect file name is given

Most importantly I think is how the package “pandas” uses another package “xlsxwriter” as it’s engine for generating the excel file.

# Application

This part contains a class with the same name, “Application”, which has the purpose of starting the processes and running them in the console. In short, to manage all the things we have discussed so far.

The class initializes the data extractor on the URL and ENDPOINT specified and consumes by the method extract(COUNT) the number of data points specified in the PDF.

Most importantly, it contains a static method called build\_options() that analyses the class DataExtractor and retrieves the method within that class. After that step it filters from the method, the ones that contain the custom decorator mentioned earlier and organizes them into a python dictionary for sequential access and display.

The decorator contains 2 fields:

* The method number
* The method name/descriptor

When listing the methods, they are accessed by the number provided, for being ordered (1, 2, 3…), and it is printed in the console the method descriptor.

This makes it a lot more developer friendly for future improvements and updates, because it lets you write down only the method in the class DataExtractor and it will be automatically uploaded to the application and ready to use. The only thing is to add above the method the custom decorator and specify in it the method number as well as the descriptor.

I included another static method, “print\_options()”, for printing the things I mentioned in the previous paragraph with some cleanup in the standard output by calling the os.system(‘cls’) command.

The loop() method is the one that runs the application, it sums up all the methods and give them logic and continuity, as well as stability.

It functions based on a while loop that awaits an operator introduced by the user via keyboard, after that it knows what method to call and prints the steps it has taken so far.

When one iteration is finished, it sleeps for 1 second to let the user acknowledge the fact that the program has finished it’s work and is ready to do it once again.

# Theoretical Part

Q1:

Queue: this data structure is verry useful when organizing data and when setting priorities, for example a thread management system where each thread has its priority and in another case the BlockingQueue which is a thread safe data structure

Balanced Binary Search Tree: this data structures are by far the most efficient of all in case of searching. One often used example is the HashMap. It operates behind on a Red Black Tree for efficiency proofing.

Q2:

For <https://tss-yonder.com/>:

IPv4: 104.26.0.62

The browser needs to find the IP for establishing a connection to the web server. It finds the correct IP via DNS resolution mechanisms.

Q3:

FTP (File Transfer Protocol): this protocol is used for file sharing.

HTTP (Hyper Text Transfer Protocol): web servers transfer their data via this protocol, ex. the application I used to solve this technical assessment made with Java SpringBoot.

Q4:

First things first, I will write my application web server, to ensure the functionality and after I will build the frontend part for people to be able to use it.

Then, I will design a docker image to ensure that my application has everything it needs to work properly.

After those steps I will send my application to be hosted by a hosting provider to make it accessible by a public domain.

Q5:

By the fact that my application is famous at the moment, it is most likely that the problems are caused by a large number of people accessing it at the same time.

The hard approach in resolving this problem is to maximize the efficiency and lower the data transferred between the server and the clients, so it can support a larger number of clients connected simultaneously. But this solution is hard to accomplish and involves sophisticated algorithms. It is also a hard thing to find where to start as our team is small in number.

The best approach is to implement a new technology, as we expect a large number of users, Kubernetes, which purpose is exactly dealing with a large amount of clients. It works by scaling the number of nodes with the number of users so both, the clients and the server, are satisfied. It is also far easier to implement and it is accomplishable in a small time with a limited number of team members.

Q6:

The way to achieve this is trough encrypted transfers and end to end encryption so not even I, the owner of the app, am not able to see random messages between users. This can be done by implementing strong encryption algorithms and generating encryption keys.

Also, since we are a small but famous team, for extra securing the application we will need to change the host provider to one that can offer us the HTTPS encryption. In that way all data will be safely transferred.

Q7:

Cookies are small data that are stored on the local machine. They are transferred between the web server and the browser and are used for several actions such as authentication.

A cookie value from <https://tss-yonder.com/>: GS1.1.1710024949.3.0.1710024949.0.0.0

I think that the purpose of this cookie is to remember the user preferences to provide a better experience on the website.

Q8:

In Python (a programming language that I use verry often) to create a child process is used the package “multiprocessing” and the syntax is similar to the thread one. Define a function to be executed in the child process and in the main process you create an object type Process and assign to it that function signature and write the name of that child process followed by “.start()”

Also in C (a programming language that I admire) to create a child process is used the fork() function. It returns a value and based on that you determine if you are executing the parent process or the child process.

The child processes can be in states:

* Start
* Running
* Blocked
* Terminated

Based on what it encounters or in which point of its execution is located.

Q9:

I can find the PID of my application in Task Manager

The problem is most probably caused by not enough verified POST method that in some cases it returns an error code so I will inspect the console output for the problem in debug mode and double verify the POST methods in the direction where the console output is directing me as well as introducing some special cases handlers for those methods.

Q10:

Despite the reliability and the speed offered by a NoSQL Database, I will most likely use a Relational Database Management System because a chat application needs to have its users well defined and organized by themselves and also by the conversation they are in. To mention that they can create group chats too that need to be stored, so the data must not be lost.

Also, a Relational Database gives developers more space for further features to be added, as example a mini social media platform integrated in the chat application that supports stories system.