Programming II - Personal Data Server

Pascal Bobbià, Luca Cereghetti, Matteo Rampazzo, Dario Vicedomini

20 June 2024

Contents

1 Introduction **1**

2 Process **1**

2.1 Setting up the server 1

2.2 Python and crontab test 1

2.3 Setting up mysql 1

2.4 Setting up the database 1

2.5 Setting up the website 2

2.6 Setting up the crontab 2

3 Conclusions **2**

4 Bibliography **3**

1 Introduction

Our project aims to set up a personal virtual server that stores real-time prices of the SMI and its components in a database and to create a website that displays the collected data.

2 Process

2.1 Setting up the server

We acquired a 2.5 GB KVM VPS. After logging in with the provided credentials via the IP address, we created a folder `data` using the command `mkdir data`. We then installed MySQL using the command
`apt install mysql-client-core-8.0`.

2.2 Python and crontab test

To test the server functionality, we used the Python code `fxday.py` from the lesson, editing it with `nano` and placing it in the `data` directory. We ran the Python program several times using the command `python3` to ensure it worked correctly. Afterward, we set up a crontab to execute this program twice daily. Observing the crontab's success, we proceeded to create a program to fetch data and insert it into a database.

2.3 Setting up mysql

We initially encountered issues connecting to MySQL with the actual credentials. To resolve this, we changed the credentials and created the database `mydatabase` along with the table `company_prices`.

2.4 Setting up the database

We developed a Python program, `fetch_quotes.py` situated in the main directory, which downloads real-time data for the SMI and its components and writes it into the `mydatabase` database under the `company_prices` table. This program was based on `quotesUpdate.py` from class, which we modified to work within the database. We set up a crontab to run this program every half hour and verified that it correctly wrote data to the appropriate table in the database.

2.5 Setting up the website

We created a Python program, `app.py` situated in the main directory, to develop the dashboard using the `dash` library, similar to what we did in class. The crucial step was connecting the program to MySQL to extract necessary data from the tables. The program then organized the data and generated plots using `plotly.express`. For the website template, we linked company names to their respective Yahoo Finance tickers to facilitate navigation. We designed the main page to display the SMI graph with a link to another page showing SMI components categorized by industry. The url is `http://107.173.226.145:8050/`.

2.6 Setting up the crontab

The final crontab runs `fetch_quotes.py` as previously described, with the addition of `run_dash_app.sh` and `fetch_quotes.sh` to run the website and to start the process. We relied on ChatGPT (OpenAI, 2024) for setting up the crontab to run `run_dash_app.sh`.

Later on we modified the crontab to restart the `run_dash_app.sh` every minute to refresh the graph on the website. Additionally, we changed the frequency of the `fetch_quotes.py` to run every 5 minutes.

We also computed two additional scripts, `restart_dash_app.sh` and `start_dashboard.sh` situated both in the main directory, that work in tandem to make sure that the process starts after there is a reboot of the VPS, so the `fetch_quotes.py` keeps running and the website keeps updating.

3 Conclusions

We successfully acquired and set up a Virtual Private Server (VPS). We then developed a program that, thanks to the set up of a crontab, continuously downloads data on the SMI and its components from Yahoo Finance, storing this information in an SQL database. Additionally, we created a website that fetches the data from the database and visually represents it through graphs.

4 Bibliography

OpenAI, ChatGPT 4.0, 2024-06-10, Prompts: How can I permanently run the dashboard utilising the crontab?

All the python programs and scripts were written by us and optimized with the help of ChatGPT.