The PR labs will be interesting because you will write programs to network with your colleagues (e.g. query your friend's HTTP server using your own client, host an online game server and play with your friends, etc.)

Rules:

- You must use Docker Compose
- You must use Python

How to present

Upload your project to GitHub and send me the URL in an email with the subject "PR lab 1" (artiom.balan@isa.utm.md). If you will be presenting later than the deadline, include the last commit hash so I can verify the time of the commit.

Include a short report in the repository (markdown or a link to a Google doc). The report will be like a demo but with screenshots, so just include screenshots of everything you would need to show in the demo, each described with a short sentence: the commands you run, the outputs you see (browser, terminal). The report should prove that you satisfied all the requirements of the lab.

You will present to me during seminars, defend your code, and answer a few theoretical questions. I will give priority to those who sent me an email first.

Docker

You must use Docker Compose for all your laboratory works so that I can run them on my computer.

The official Docker 101 tutorial is a good resource.

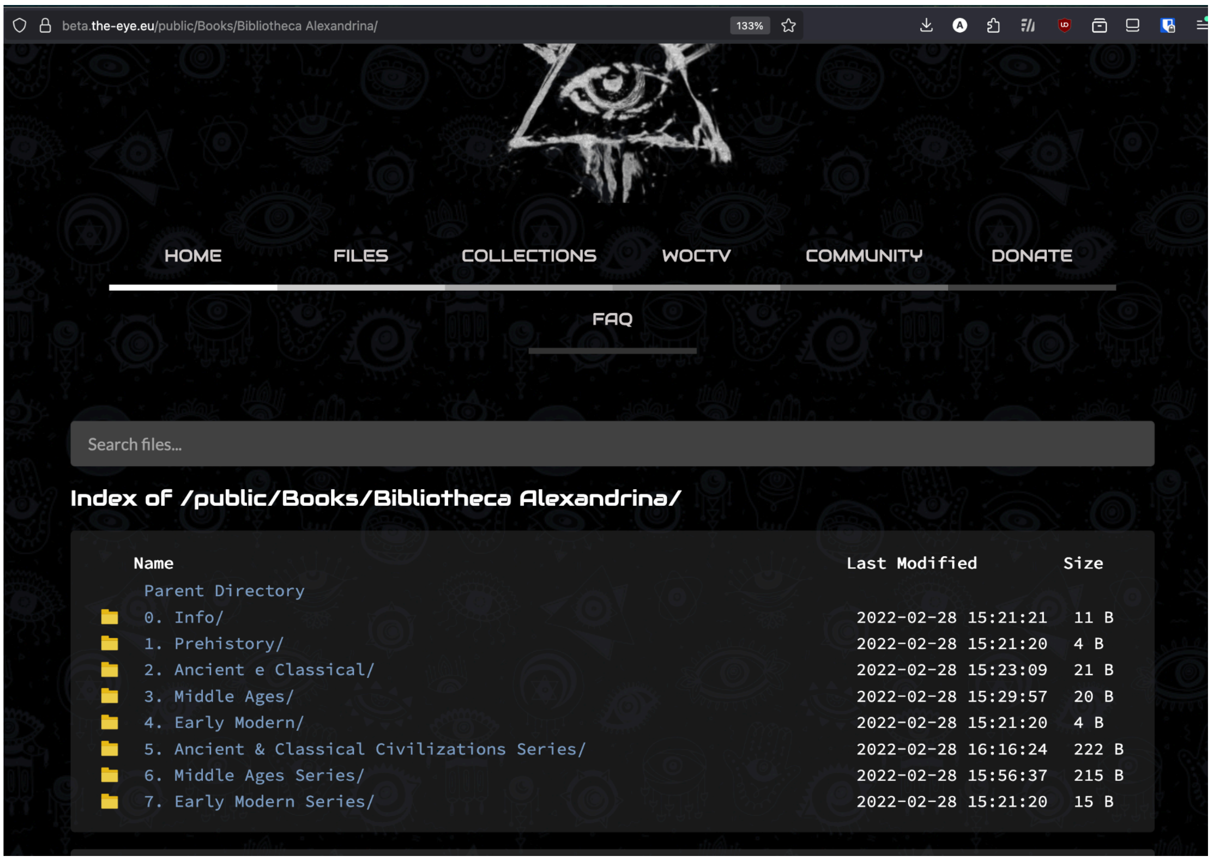
Lab 1: HTTP file server with TCP sockets

The questions will be based on these resources:

- Chapter 2 from "Computer Networking 8th edition" (you can skip 2.3-2.6)
- This Python HOWTO article
- (Optional) MIT class reading on Sockets & Networking

You can practice on the questions from the end of the chapter.

You will develop a simple HTTP file server like Python's http.server. You will use it to build a website that lets your friends browse your collection of PDFs, like this one:



Index of /public/Books/Bibliotheca Alexandrina/

```
Name                                                    Last Modified        Size
 Parent Directory
 0. Info/                                               2022-02-28 15:21:21  11 B
 1. Prehistory/                                         2022-02-28 15:21:20  4 B
 2. Ancient e Classical/                                2022-02-28 15:23:09  21 B
 3. Middle Ages/                                        2022-02-28 15:29:57  20 B
 4. Early Modern/                                       2022-02-28 15:21:20  4 B
 5. Ancient & Classical Civilizations Series/           2022-02-28 16:16:24  222 B
 6. Middle Ages Series/                                 2022-02-28 15:56:37  215 B
 7. Early Modern Series/                                2022-02-28 15:21:20  15 B
```

It will handle one HTTP request at a time. The server program should take the directory to be served as a command-line argument.

Your web server should accept and parse the HTTP request, read the requested file from the directory, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server (or has an unknown extension), the server should send an HTTP “404 Not Found” message back to the client. Update: your server should handle HTML, PNG and PDF file types.

Prepare a directory with content to be served. At the very least, it should contain an HTML file, a PNG image and a few PDF files. Reference the image in the HTML file (using <img>).

To get a 10, you need to do the following tasks as well:

1. Implement an HTTP client for your server - **2 points**

   Update: It will be a python script that takes as command-line arguments a URL and a directory to save files in, and acts depending on the file type:

   - HTML (page, directory listing): print the body of the response as-is
   - PNG, PDF: save the file in the specified directory

   Use the following format for the command arguments:
   client.py server_host server_port url_path directory

2. Make the server work with nested directories - **2 points**:

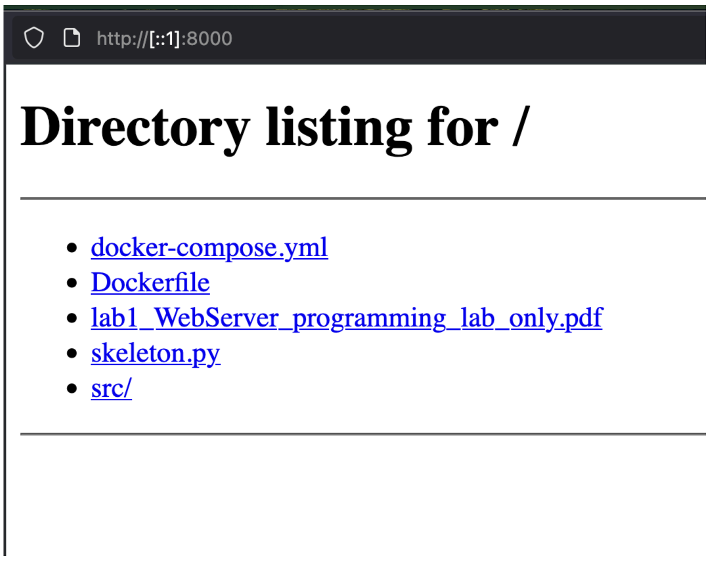
If a directory path is requested, respond with a directory listing (a generated HTML page that displays the contents of the directory, with hyperlinks).

Create a subdirectory in your content directory with a few PDF/PNG files inside it.

3. Browse the books on your friend's server (local network), have fun - **1 point**

   You can use your client to download a book from your friend's server right into your own website directory.

What the directory listing can look like (python's http.server response shown here):



http://[::1]:8000

Directory listing for /

- docker-compose.yml
- Dockerfile
- lab1_WebServer_programming_lab_only.pdf
- skeleton.py
- src/

Report contents:

- The contents of your source directory
- The docker compose file (and Dockerfile if you use one)
- show how you start the container
- Command that runs the server inside the container, with a directory as an argument
- The contents of the served directory
- Requests of 4 files in the browser: inexistent file (404), HTML file with image, PDF file, PNG file
- (If you made the client) How you run the client, show output and saved files
- (If you made directory listing) The directory listing generated page, subdirectory
- (If you browsed your friend's server) Describe your network setup, how you found their IP, screenshots of the contents of their server, screenshots of requests to their server (using your own client if you implemented it)