Linnéuniversitetet
Kalmar Växjö

Report

Assignment 2

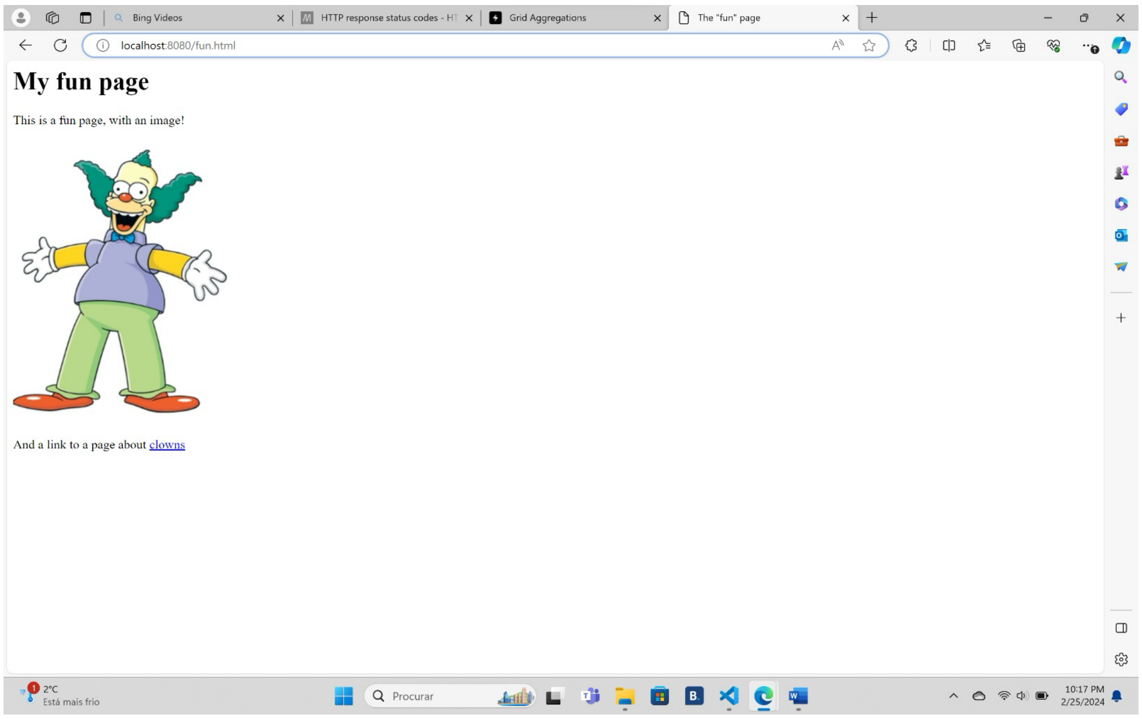
1DV701

Linnéuniversitetet
Kalmar Växjö

Authors: Martim Oliveira,
Melika Moayer
Semester: Spring 2023
Emails:
mo223tz
mm224ys

Table of Contents

1 Problem 1



My fun page

This is a fun page, with an image!

And a link to a page about clowns

1.1 Discussion

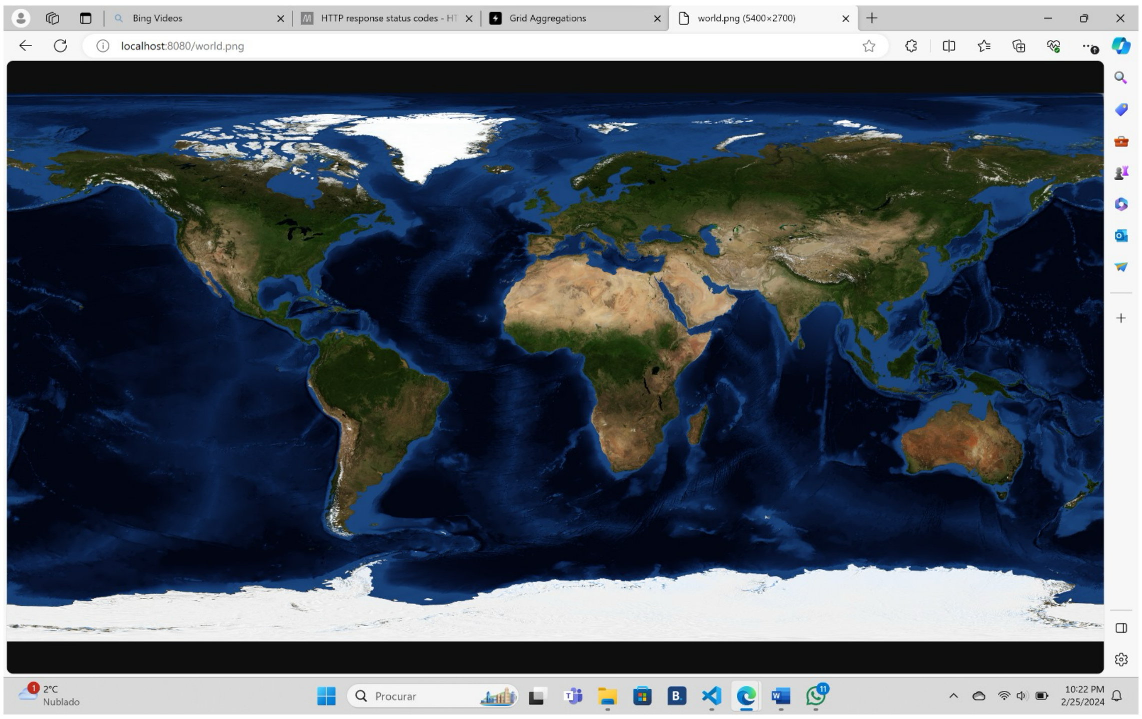
For Problem 1, the task was to implement a simple web server that serves HTML and PNG files from a local directory using the GET method in the HTTP protocol. The server should accept multiple client connections on a specified port and serve the requested files while ensuring safety against escaping the root directory.

For the HTTP Server, we implemented a Java program that utilizes the ServerSocket class to listen for incoming connections on a specified port. Afterwards, GET requests are parsed to extract the requested file path. We serve HTML and PNG files by reading them from the local directory and sending them as HTTP responses. When a directory is requested, we check for an index.html file within that directory. If found, we serve the index file; otherwise, we respond with a 404 error. To prevent users from accessing files outside the root directory, we validate file paths to ensure they are within the designated public directory.

Regarding exception handling, we have implemented several strategies to handle potential errors. Firstly, we use try-with-resources statements to automatically close resources like ServerSocket, BufferedReader, and DataOutputStream, ensuring proper resource management and avoiding resource leaks. Additionally, we catch IOExceptions that may occur during socket operations, file reading, and writing processes. For instance, in the serveFile method, we catch FileNotFoundException and IOException separately to handle missing files and other IO errors appropriately.

The program accepts two command-line arguments: the port number and the relative path to the public directory, ensuring flexibility and configurability.

2 Problem 2



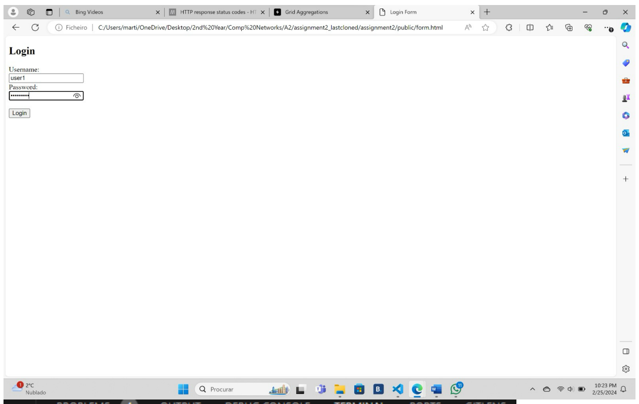
2.1 Discussion

In Problem 2, we extended the web server to support additional HTTP response types: 302 (Found), 404 (Not Found), and 500 (Internal Server Error).

We implemented support for each response code by generating the appropriate response headers and messages. to ensure correctness, we tested each response code by simulating specific conditions, such as accessing nonexistent resources or inducing server errors.

- 302 Redirect Response:
We hardcoded a specific URL to redirect to in the sendErrorResponse method, which correctly implements a 302 response.
Testing: We can trigger this response by accessing the /redirect endpoint.
- 404 Not Found Response:
We implemented a 404 response in the sendErrorResponse method when the requested resource does not exist.
Testing: Accessing a non-existent resource triggers this response.
- 500 Internal Server Error Response:
We implemented a 500 response in the sendErrorResponse method for internal server errors.
Testing: Causing an internal server error (e.g., accessing an incorrect URL) should trigger this response.

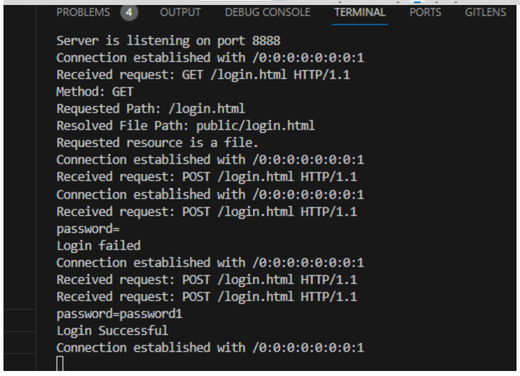
3 Problem 3


C:/Users/marti/OneDrive/Desktop/2nd%20Year/Comp%20Networks/A2/assignment2_lastcloned/assignment2/public/form.html

Login

Username:
user1
Password:
••••••••

Login



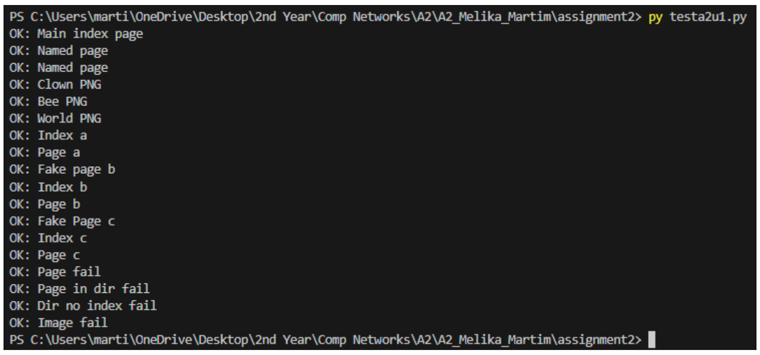
```
Server is listening on port 8888
Connection established with /0:0:0:0:0:0:0:1
Received request: GET /login.html HTTP/1.1
Method: GET
Requested Path: /login.html
Resolved File Path: public/login.html
Requested resource is a file.
Connection established with /0:0:0:0:0:0:0:1
Received request: POST /login.html HTTP/1.1
Connection established with /0:0:0:0:0:0:0:1
Received request: POST /login.html HTTP/1.1
password=
Login failed
Connection established with /0:0:0:0:0:0:0:1
Received request: POST /login.html HTTP/1.1
Received request: POST /login.html HTTP/1.1
password=password1
Login Successful
Connection established with /0:0:0:0:0:0:0:1
```

3.1 Discussion

Problem 3 involved configuring the web server to handle a login page using the HTTP POST method and implementing support for uploading PNG images.

For the login page, We created an HTML login form with fields for username and password, along with a POST action to send the credentials to the server.
After receiving login credentials via POST request, the server verifies them against stored credentials. It is supposed to respond with either 200 OK or 401 Unauthorized. Users can upload PNG images via a dedicated webpage, and the server saves them in the designated directory.

4 Python Test



```
PS C:\Users\marti\OneDrive\Desktop\2nd Year\Comp Networks\A2\A2_Melika_Martim\assignment2> py testa2u1.py
OK: Main index page
OK: Named page
OK: Named page
OK: Clown PNG
OK: Bee PNG
OK: World PNG
OK: Index a
OK: Page a
OK: Fake page b
OK: Index b
OK: Page b
OK: Fake Page c
OK: Index c
OK: Page c
OK: Page fail
OK: Page in dir fail
OK: Dir no index fail
OK: Image fail
PS C:\Users\marti\OneDrive\Desktop\2nd Year\Comp Networks\A2\A2_Melika_Martim\assignment2>
```

Finally the above screenshot indicates that our program has successfully passed the python test as well.

5 Contribution of each group member:

Martim Oliveira: 50%, Melika Moayer: 50%
We mostly met up at the library/ Zoom meetings and worked on the code together, trying to come up with solutions to solve the encountered problems. Finally, we both wrote the report after holding relative discussions.

6 Refernces:

1. MozDevNet, "HTTP response status codes - HTTP: MDN," MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#redirection_messages (accessed Feb. 24, 2024).

2. "Build a simple web server with Java," YouTube, https://www.youtube.com/watch?v=FqufxoA4m70 (accessed Feb. 25, 2024).

3. U. Mana, "A simple HTTP server in Java - DZone," dzone.com, https://dzone.com/articles/simple-http-server-in-java (accessed Feb. 25, 2024).