Comp 416
Fall 2020

Project 1: WeatNet

November 8, 2020

Group 10

Gül Sena Altıntaş

Okan Aslan

Eren Deniz Sanlıer

Table of Contents

Introduction

WeatNet implementation of ours has three main components as requested. The client-side application, the API and the server-side application. The API is responsible for the data transmission from OWM to the server-side application. The server-side application processes the received data and provides it to multiple connected clients through a web socket. The client's are required to successfully complete an auth challenge in order to get authorized by the server. Once they are authenticated, they are being provided with a token, and their auth information is stored as sessions. Each session is stored in a map, allowing multiple clients to be authorized simultaneously. The clients can start to perform queries. The clients and the server use different sockets for communication and data transfer, and all the data transfers were implemented as protocols.

Authentication

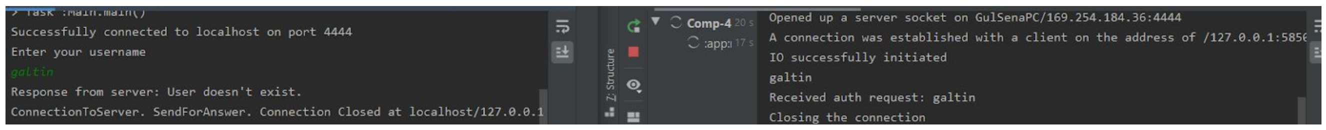
When the server establishes a connection, we delegate the authentication to Authentication class which checks whether the

1. Entered username is valid (ie. it exists in the database)
2. User provides the correct answers to challenge questions (server picks a maximum of 3 questions and shuffles the available questions every time)
3. User provides answers within the predefined timeout limit

If user passes all above tests, Authentication generates a token and a session is created for this user. Implementing sessions allows our system to store information related to the each active user and manage their sessions.

Sample Authentication Demonstrations

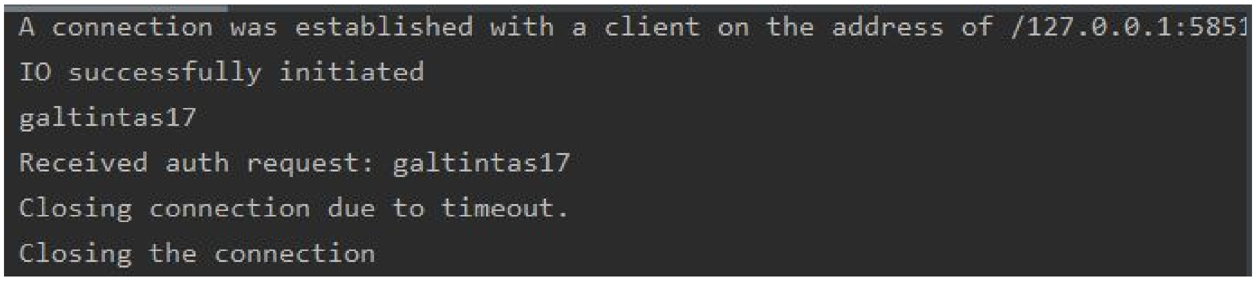
1. User doesn't exist (available users: galtintas17, esanlier16, oaslan16, icekin17, professor)



```
> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
galtin
Response from server: User doesn't exist.
ConnectionToServer. SendForAnswer. Connection Closed at localhost/127.0.0.1
```

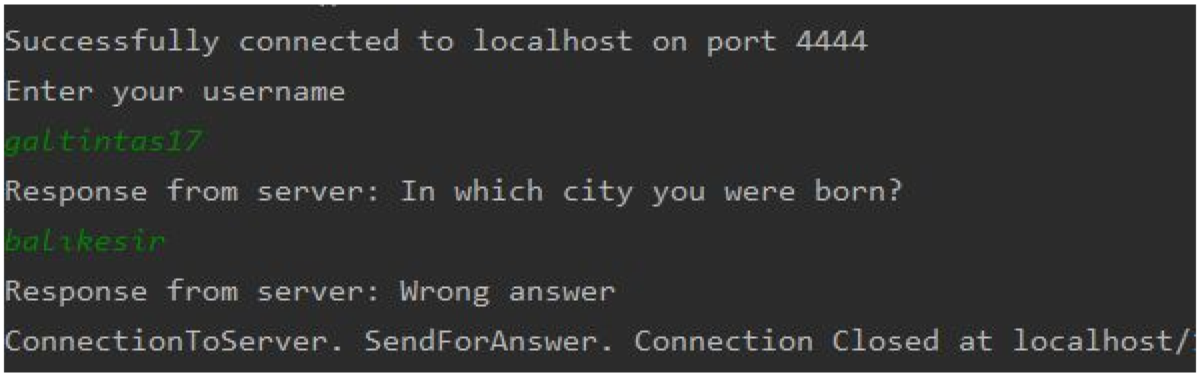
```
Opened up a server socket on GulSenaPC/169.254.184.36:4444
A connection was established with a client on the address of /127.0.0.1:5856
IO successfully initiated
galtin
Received auth request: galtin
Closing the connection
```

2. Timeout



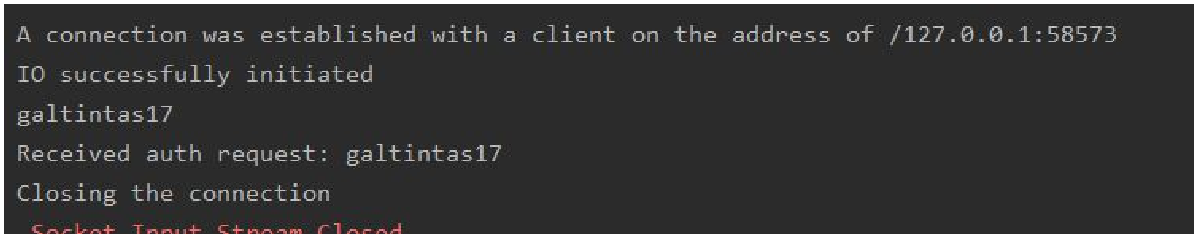
```
A connection was established with a client on the address of /127.0.0.1:5851
IO successfully initiated
galtintas17
Received auth request: galtintas17
Closing connection due to timeout.
Closing the connection
```

3. Wrong answer



```
Successfully connected to localhost on port 4444
Enter your username
galtintas17
Response from server: In which city you were born?
balıkesir
Response from server: Wrong answer
ConnectionToServer. SendForAnswer. Connection Closed at localhost/
```

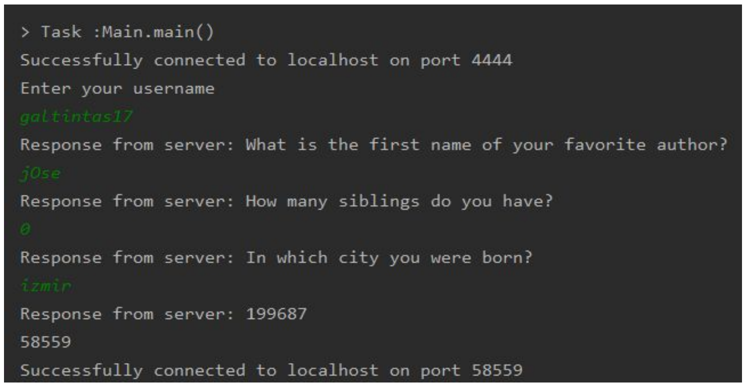
Client side



```
A connection was established with a client on the address of /127.0.0.1:58573
IO successfully initiated
galtintas17
Received auth request: galtintas17
Closing the connection
 Socket Input Stream Closed
```

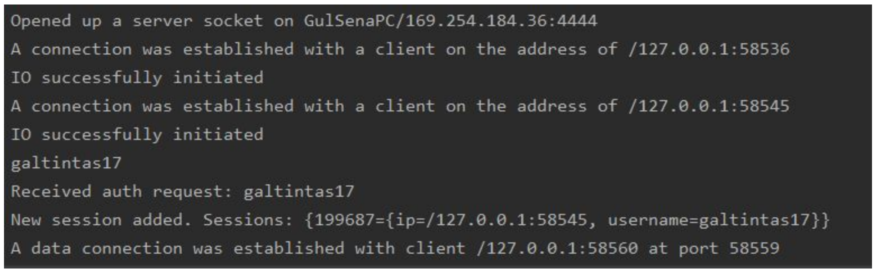
Server side

4. Successful connection returns first the token and then the randomly assigned data port



```
> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
galtintas17
Response from server: What is the first name of your favorite author?
jOse
Response from server: How many siblings do you have?
0
Response from server: In which city you were born?
izmir
Response from server: 199687
58559
Successfully connected to localhost on port 58559
```

Client side

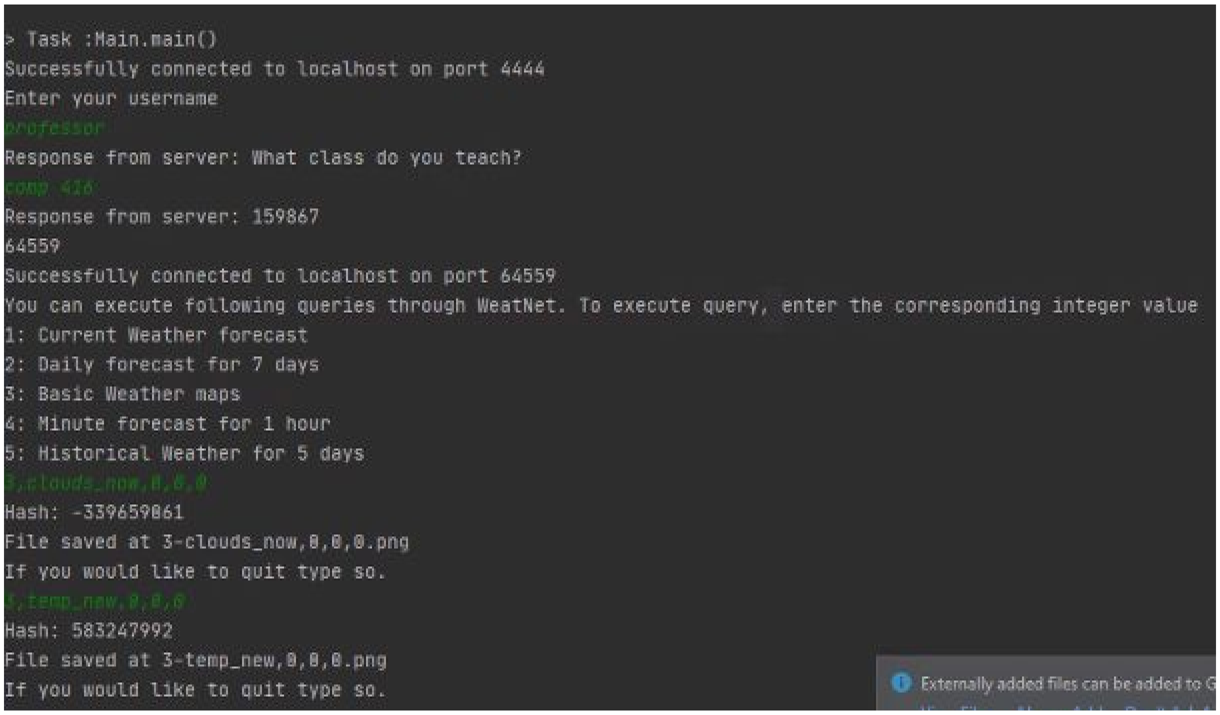


```
Opened up a server socket on GulSenaPC/169.254.184.36:4444
A connection was established with a client on the address of /127.0.0.1:58536
IO successfully initiated
A connection was established with a client on the address of /127.0.0.1:58545
IO successfully initiated
galtintas17
Received auth request: galtintas17
New session added. Sessions: {199687={ip=/127.0.0.1:58545, username=galtintas17}}
A data connection was established with client /127.0.0.1:58560 at port 58559
```

Server side

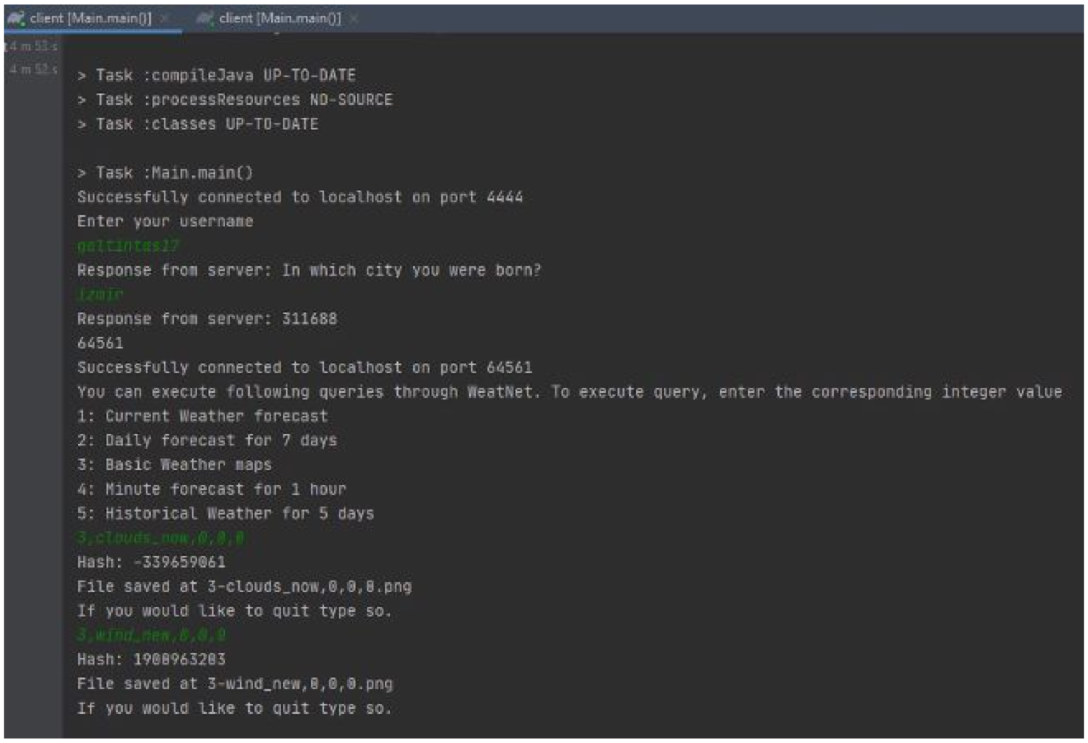
Multiple Client Support

Our network is able to support multiple clients at the same time, each client has a different data socket assigned to them. The sample execution can be seen below:
Client 1:



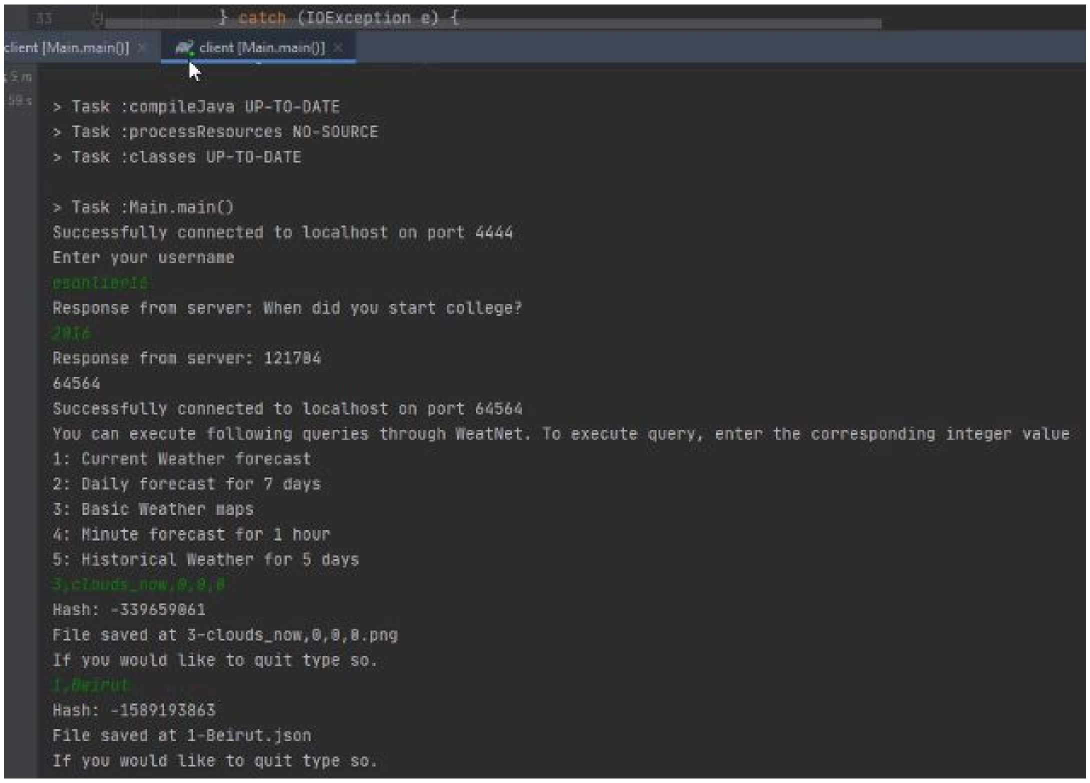
```
> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
professor
Response from server: What class do you teach?
comp 416
Response from server: 159867
64559
Successfully connected to localhost on port 64559
You can execute following queries through WeatNet. To execute query, enter the corresponding integer value
1: Current Weather forecast
2: Daily forecast for 7 days
3: Basic Weather maps
4: Minute forecast for 1 hour
5: Historical Weather for 5 days
3,clouds_now,0,0,0
Hash: -339659061
File saved at 3-clouds_now,0,0,0.png
If you would like to quit type so.
3,temp_new,0,0,0
Hash: 583247992
File saved at 3-temp_new,0,0,0.png
If you would like to quit type so.
```

Client 2:



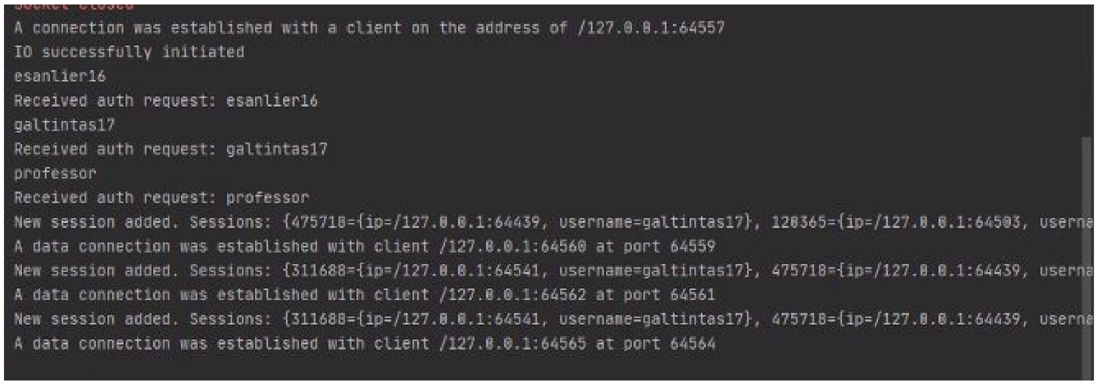
```
client [Main.main()]    client [Main.main()]

> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE

> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
goltintas17
Response from server: In which city you were born?
izmir
Response from server: 311688
64561
Successfully connected to localhost on port 64561
You can execute following queries through WeatNet. To execute query, enter the corresponding integer value
1: Current Weather forecast
2: Daily forecast for 7 days
3: Basic Weather maps
4: Minute forecast for 1 hour
5: Historical Weather for 5 days
3,clouds_new,0,0,0
Hash: -339659061
File saved at 3-clouds_now,0,0,0.png
If you would like to quit type so.
3,wind_new,0,0,0
Hash: 1908963203
File saved at 3-wind_new,0,0,0.png
If you would like to quit type so.
```

Client 3:



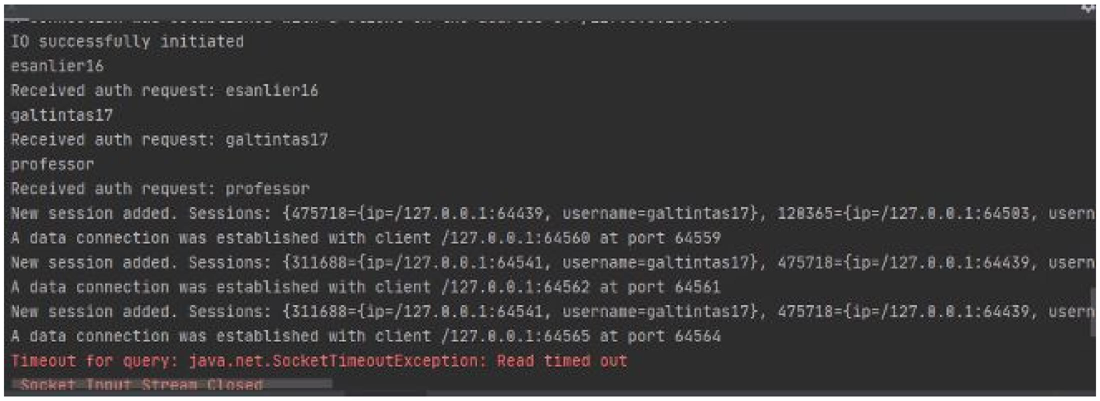
```
} catch (IOException e) {
client [Main.main()]    client [Main.main()]

> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE

> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
esonlier16
Response from server: When did you start college?
2016
Response from server: 121784
64564
Successfully connected to localhost on port 64564
You can execute following queries through WeatNet. To execute query, enter the corresponding integer value
1: Current Weather forecast
2: Daily forecast for 7 days
3: Basic Weather maps
4: Minute forecast for 1 hour
5: Historical Weather for 5 days
3,clouds_now,0,0,0
Hash: -339659061
File saved at 3-clouds_now,0,0,0.png
If you would like to quit type so.
1,Beirut
Hash: -1589193863
File saved at 1-Beirut.json
If you would like to quit type so.
```

Server:



```
Socket closed
A connection was established with a client on the address of /127.0.0.1:64557
IO successfully initiated
esanlier16
Received auth request: esanlier16
galtintas17
Received auth request: galtintas17
professor
Received auth request: professor
New session added. Sessions: {475718={ip=/127.0.0.1:64439, username=galtintas17}, 120365={ip=/127.0.0.1:64503, userna
A data connection was established with client /127.0.0.1:64560 at port 64559
New session added. Sessions: {311688={ip=/127.0.0.1:64541, username=galtintas17}, 475718={ip=/127.0.0.1:64439, userna
A data connection was established with client /127.0.0.1:64562 at port 64561
New session added. Sessions: {311688={ip=/127.0.0.1:64541, username=galtintas17}, 475718={ip=/127.0.0.1:64439, userna
A data connection was established with client /127.0.0.1:64565 at port 64564
```

Server timeout:



```
IO successfully initiated
esanlier16
Received auth request: esanlier16
galtintas17
Received auth request: galtintas17
professor
Received auth request: professor
New session added. Sessions: {475718={ip=/127.0.0.1:64439, username=galtintas17}, 120365={ip=/127.0.0.1:64503, usern
A data connection was established with client /127.0.0.1:64560 at port 64559
New session added. Sessions: {311688={ip=/127.0.0.1:64541, username=galtintas17}, 475718={ip=/127.0.0.1:64439, usern
A data connection was established with client /127.0.0.1:64562 at port 64561
New session added. Sessions: {311688={ip=/127.0.0.1:64541, username=galtintas17}, 475718={ip=/127.0.0.1:64439, usern
A data connection was established with client /127.0.0.1:64565 at port 64564
Timeout for query: java.net.SocketTimeoutException: Read timed out
Socket Input Stream Closed
```

Protocols

TCP

Data transfer between the Client and Server is implemented according to the TCP protocol specified in the PDF. Both the client side and the server side have a TCP class that handles the appropriate writing and reading of the messages.

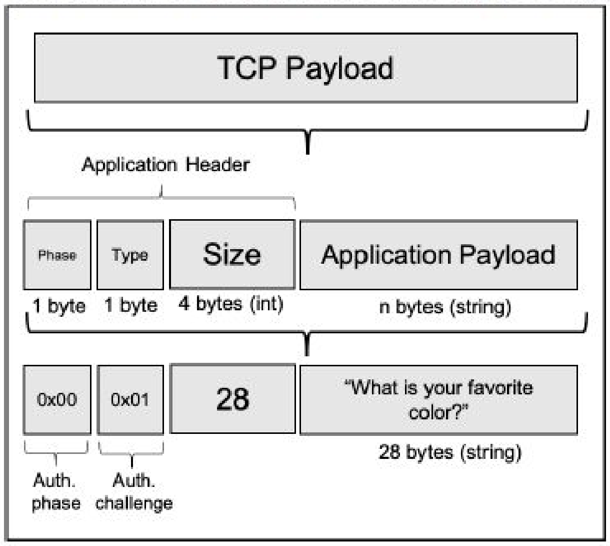


Fig. 3. Deconstructing the TCP data.

Fig1. TCP Payload as demonstrated in the project pdf

Query and File Transfer Protocol

Query

File transfer protocol is implemented similar to TCP. First byte is **1** meaning that the system is executing the query phase. Then we add the token of the user, then the next byte represents the query type (specified in page 1, ranging from 1 to 5). Then we specify the size of the query text and finally send the query text as bytes.

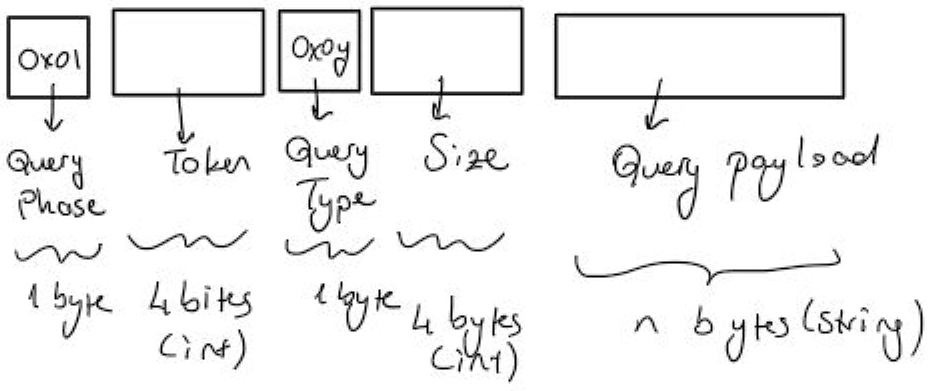


Fig2. Deconstructing TCP-Query Protocol

Data Transfer

We transfer bytes corresponding to the file and its size through the data socket and the hash code of the file (bytes of the file) through the communication channel. The client then accepts the bytes from the data socket, compares the hash value of the received file and the hash value received from the server. In this way the completeness of the received file can be assured.

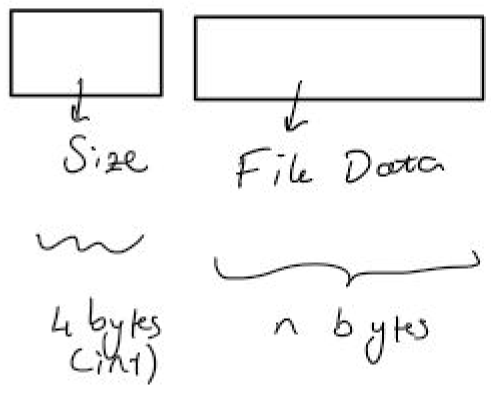
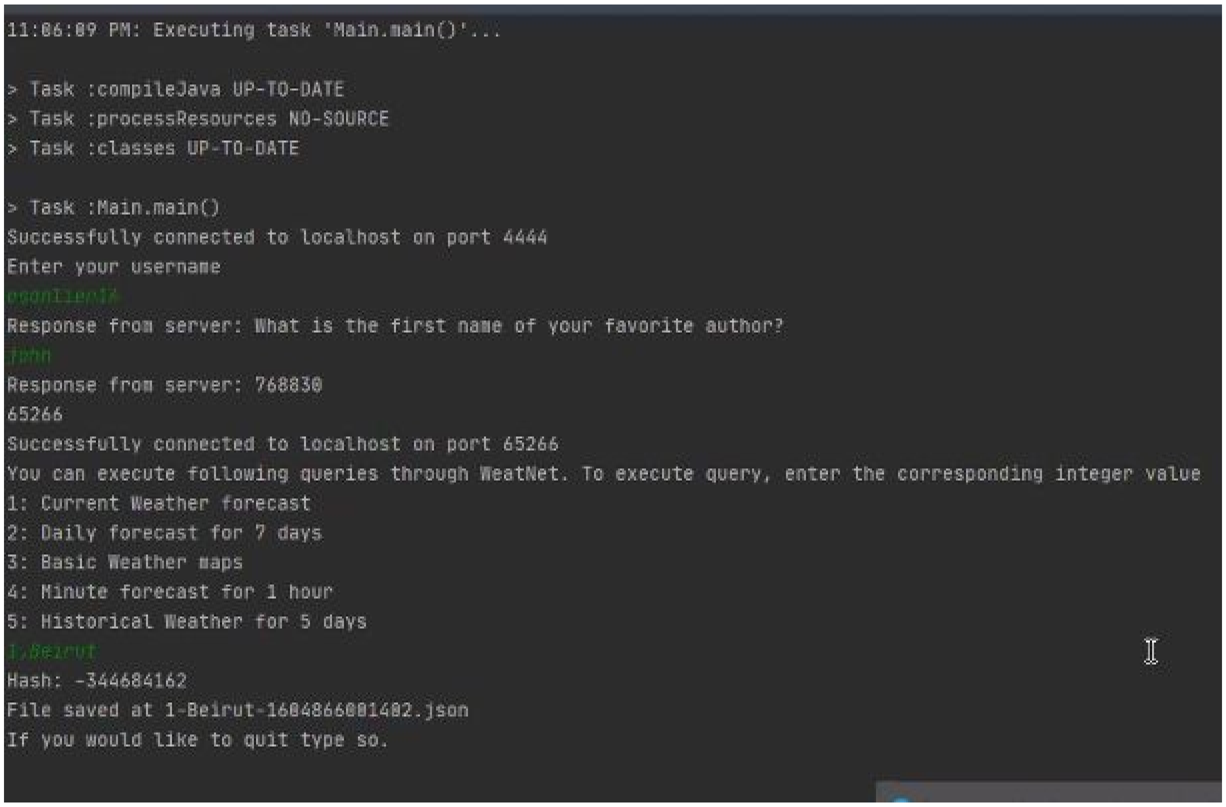


Fig3. Deconstructing DataTransfer Protocol

Adding timestamp to the files

Server side sends the current time millisecond as long and the files in the client side are



```
11:06:09 PM: Executing task 'Main.main()'...

> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE

> Task :Main.main()
Successfully connected to localhost on port 4444
Enter your username
cgonlientX
Response from server: What is the first name of your favorite author?
john
Response from server: 768830
65266
Successfully connected to localhost on port 65266
You can execute following queries through WeatNet. To execute query, enter the corresponding integer value
1: Current Weather forecast
2: Daily forecast for 7 days
3: Basic Weather maps
4: Minute forecast for 1 hour
5: Historical Weather for 5 days
1,Beirut
Hash: -344684162
File saved at 1-Beirut-1604866001402.json
If you would like to quit type so.
```

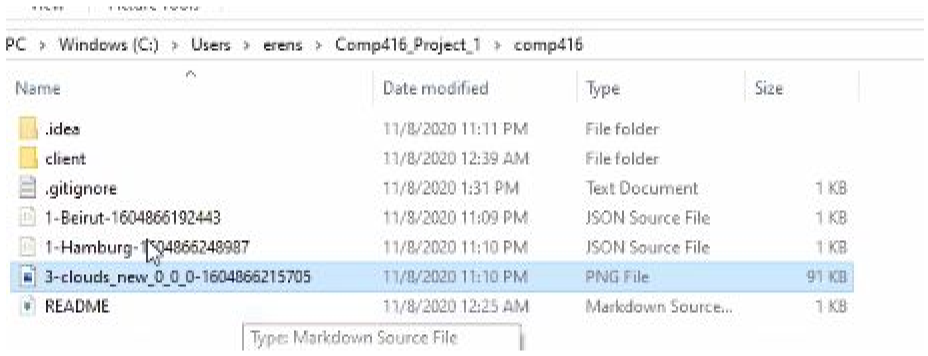
Re-requesting file transfer

Example:

```
3,clouds_new,0,0,0
Hash: -1
File saved at 3-clouds_new_0_0_0.png
Hash values mismatch, re-requesting file transfer.
1,London
File saved at 3-clouds_new,0,0,0.png
If you would like to quit type so.
Hash: -1930173641
File saved at 1-London.json
Hash values mismatch, re-requesting file transfer.
File saved at 1-London.json
If you would like to quit type so.
```

Saving files to the client side

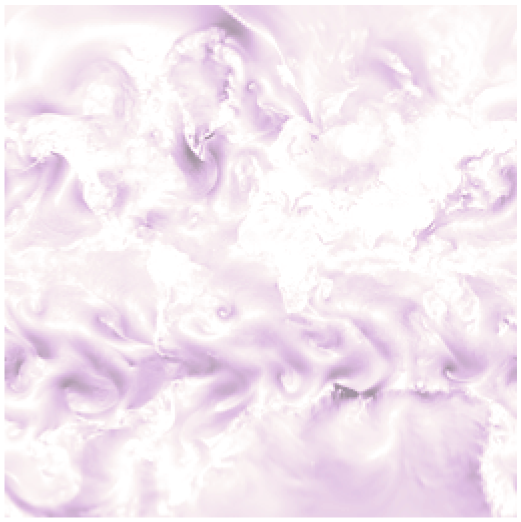
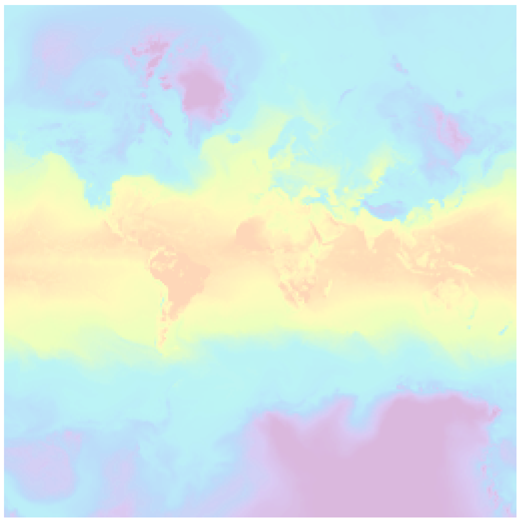
Files are saved according to the format in the client side



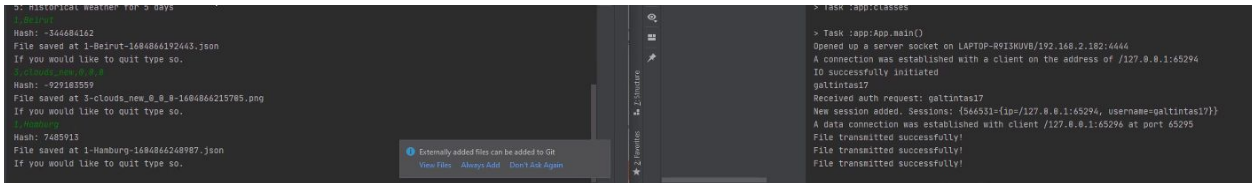
PC > Windows (C:) > Users > erens > Comp416_Project_1 > comp416

Name	Date modified	Type	Size
.idea	11/8/2020 11:11 PM	File folder	
client	11/8/2020 12:39 AM	File folder	
.gitignore	11/8/2020 1:31 PM	Text Document	1 KB
1-Beirut-1604866192443	11/8/2020 11:09 PM	JSON Source File	1 KB
1-Hamburg-1604866248987	11/8/2020 11:10 PM	JSON Source File	1 KB
3-clouds_new_0_0_0-1604866215705	11/8/2020 11:10 PM	PNG File	91 KB
README	11/8/2020 12:25 AM	Markdown Source...	1 KB

Type: Markdown Source File

 wind_new at 0,0,0	 temp_new at 0,0,0

Confirmation message at the server end:



```
5: Historical weather for 5 days
1,Beirut
Hash: -344684162
File saved at 1-Beirut-1604866192443.json
If you would like to quit type so.
3,clouds_new,0,0,0
Hash: -929103559
File saved at 3-clouds_new_0_0_0-1604866215705.png
If you would like to quit type so.
1,Hamburg
Hash: 7485913
File saved at 1-Hamburg-1604866248987.json
If you would like to quit type so.
```

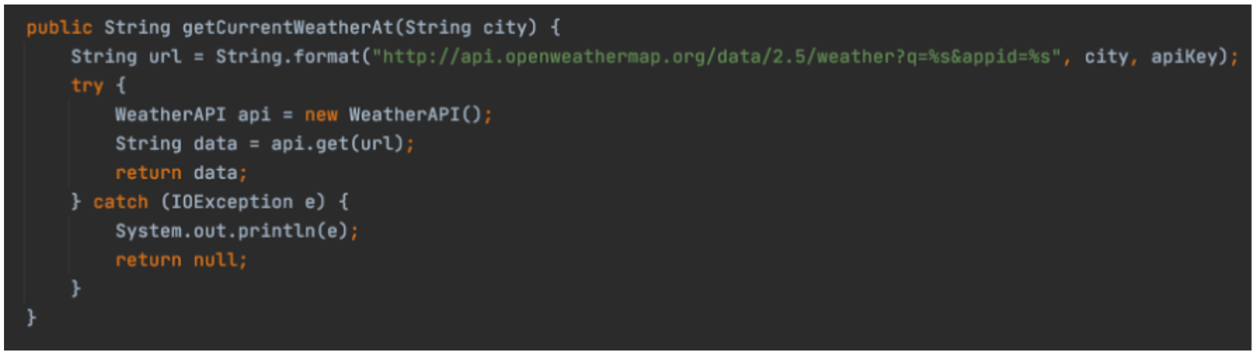
```
> Task :app:classes

> Task :app:App.main()
Opened up a server socket on LAPTOP-R9I3KUVB/192.168.2.182:4444
A connection was established with a client on the address of /127.0.0.1:65294
IO successfully initiated
galtintas17
Received auth request: galtintas17
New session added. Sessions: {566531={ip=/127.0.0.1:65294, username=galtintas17}}
A data connection was established with client /127.0.0.1:65296 at port 65295
File transmitted successfully!
File transmitted successfully!
File transmitted successfully!
```

OpenWeatherMap API

In order to use the API, we created a free account and received a token. The token is sent with the request body to authorize our request. We used the OkHttp framework to send requests to the server. We hide sensitive information in the .env file using dotenv framework.

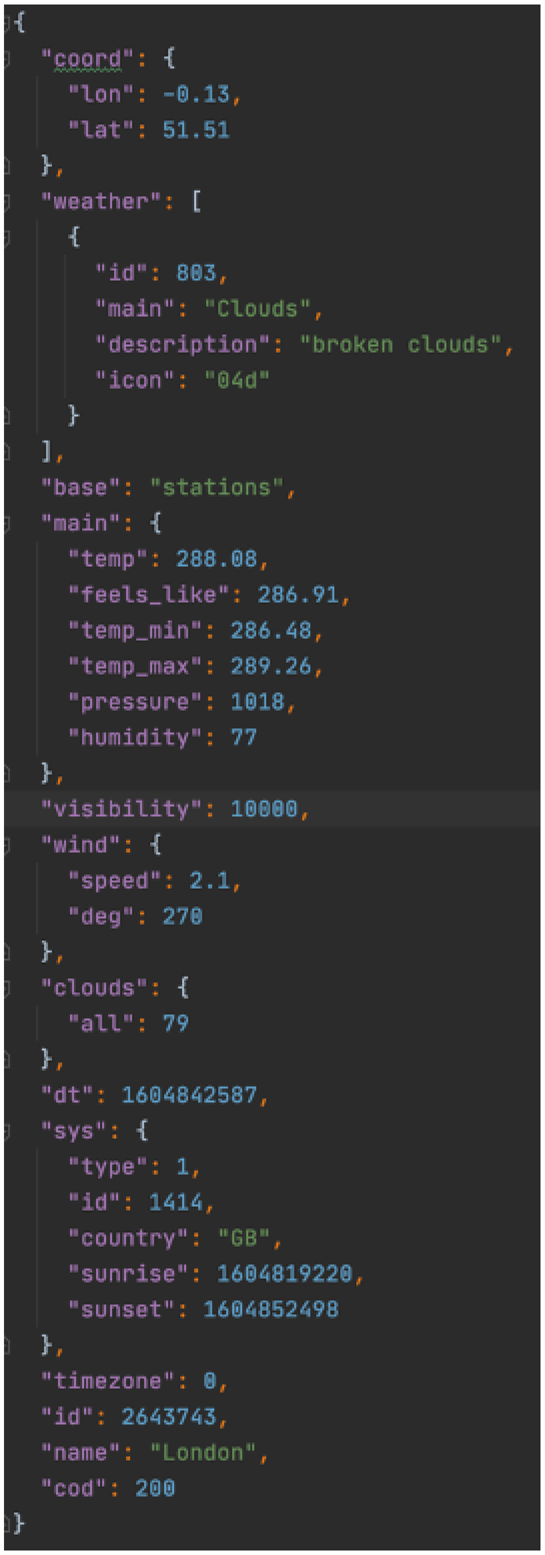
API integration part consists of 4 files: API.java, DateUtils.java, FileUtils.java, and WheatherAPI.java. WheatherAPI.java is used to send server requests to API and handle connection errors. FileUtils.java is used to save both JSON and PNG files to the current working directory. DateUtils.java helped to add timestamp to files and test API. API.java encapsulates other files and exports A functions for each API request.

Files downloaded from API named according to request endpoint and parameter. We also added a timestamp to each file.



```java
public String getCurrentWeatherAt(String city) {
    String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city, apiKey);
    try {
        WeatherAPI api = new WeatherAPI();
        String data = api.get(url);
        return data;
    } catch (IOException e) {
        System.out.println(e);
        return null;
    }
}
```

Figure 1: API request structure for current weather.



```
{
  "coord": {
    "lon": -0.13,
    "lat": 51.51
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 288.08,
    "feels_like": 286.91,
    "temp_min": 286.48,
    "temp_max": 289.26,
    "pressure": 1018,
    "humidity": 77
  },
  "visibility": 10000,
  "wind": {
    "speed": 2.1,
    "deg": 270
  },
  "clouds": {
    "all": 79
  },
  "dt": 1604842587,
  "sys": {
    "type": 1,
    "id": 1414,
    "country": "GB",
    "sunrise": 1604819220,
    "sunset": 1604852498
  },
  "timezone": 0,
  "id": 2643743,
  "name": "London",
  "cod": 200
}
```

Figure 2: Example API response for current weather at London.

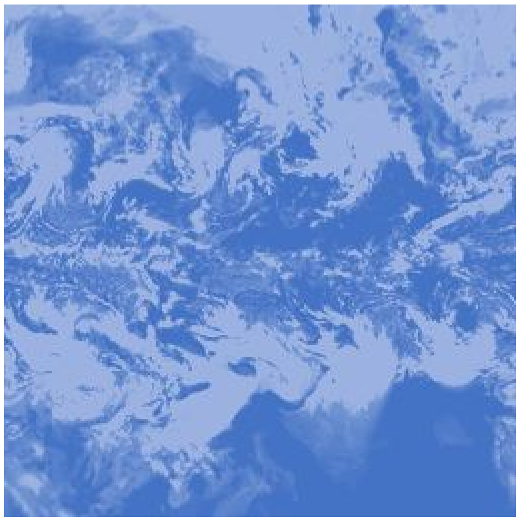


Figure 3: Example API response for basic Weather Map Data with clouds layer.