Weather CLI app

Name: Aikaterini Karanasiou

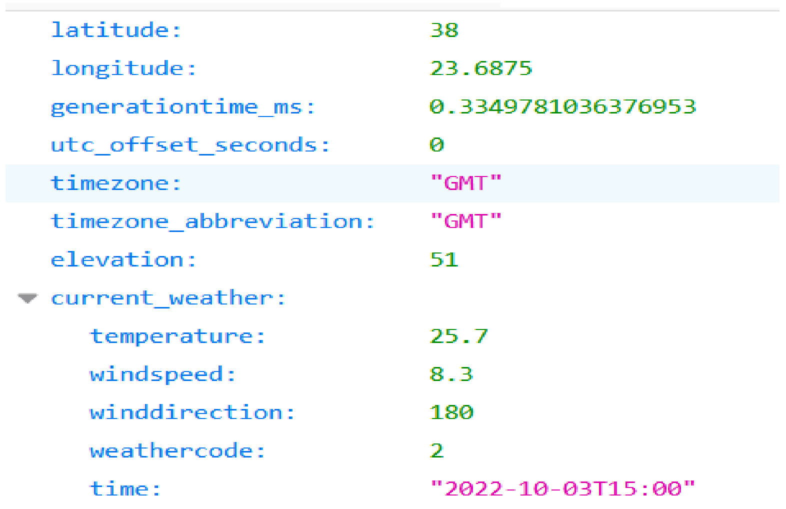
Open-Meteo

Link: https://open-meteo.com/en

➢ Weather Forecast API

https://api.open-meteo.com/v1/forecast?latitude=37.9792&longitude=23.7166

current_weather is TRUE



```
latitude:                  38
longitude:                 23.6875
generationtime_ms:         0.3349781036376953
utc_offset_seconds:        0
timezone:                  "GMT"
timezone_abbreviation:     "GMT"
elevation:                 51
▼ current_weather:
    temperature:           25.7
    windspeed:             8.3
    winddirection:         180
    weathercode:           2
    time:                  "2022-10-03T15:00"
```

Open-Meteo

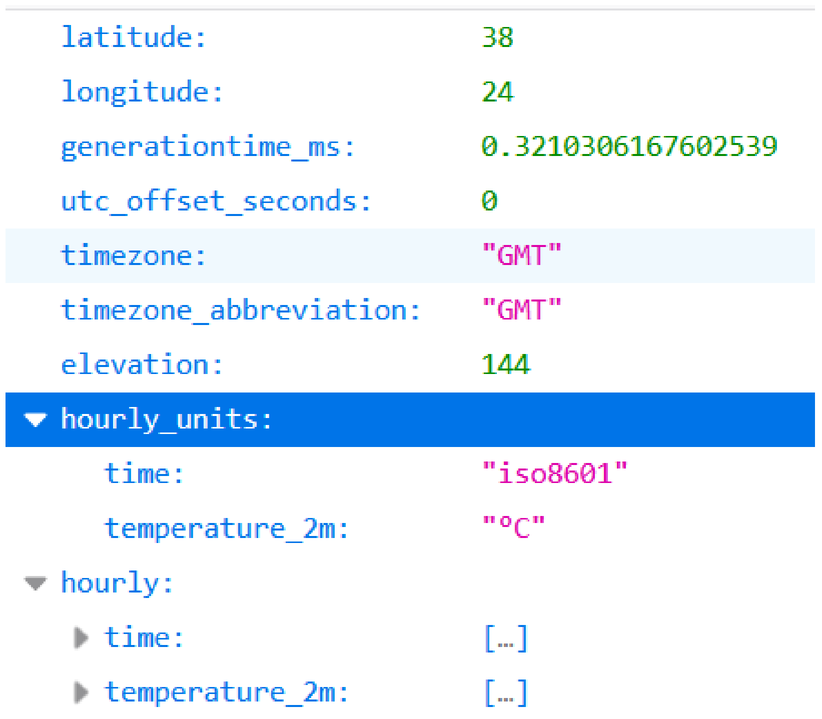
- Historical Weather API

https://archive-api.open-meteo.com/v1/era5?latitude=37.9792&longitude=23.7166

start_date is 2021-01-01

end_date is 2022-07-13

hourly is temperature_2m



```
latitude:                   38
longitude:                  24
generationtime_ms:          0.3210306167602539
utc_offset_seconds:         0
timezone:                   "GMT"
timezone_abbreviation:      "GMT"
elevation:                  144
▼ hourly_units:
    time:                   "iso8601"
    temperature_2m:         "°C"
▼ hourly:
  ▶ time:                   […]
  ▶ temperature_2m:         […]
```

Process Description - Storing Weather Data

WeatherData:

- ➢ When asio timer is expired (every 24 hours), process updates the tables on DB.

ConnectionWeatherAPI:

- ➢ Functions for sending HTTP requests, receiving and parsing HTTP responses.

Process Description - Weather CLI

CC (WeatherCLI):

- ➢ It executes a specific action according to the user’s input.

Command:

- ➢ It is a base class and its function is implemented by DISPLAY, COMPARE and AVERAGE sub-classes.

ControlCommands:

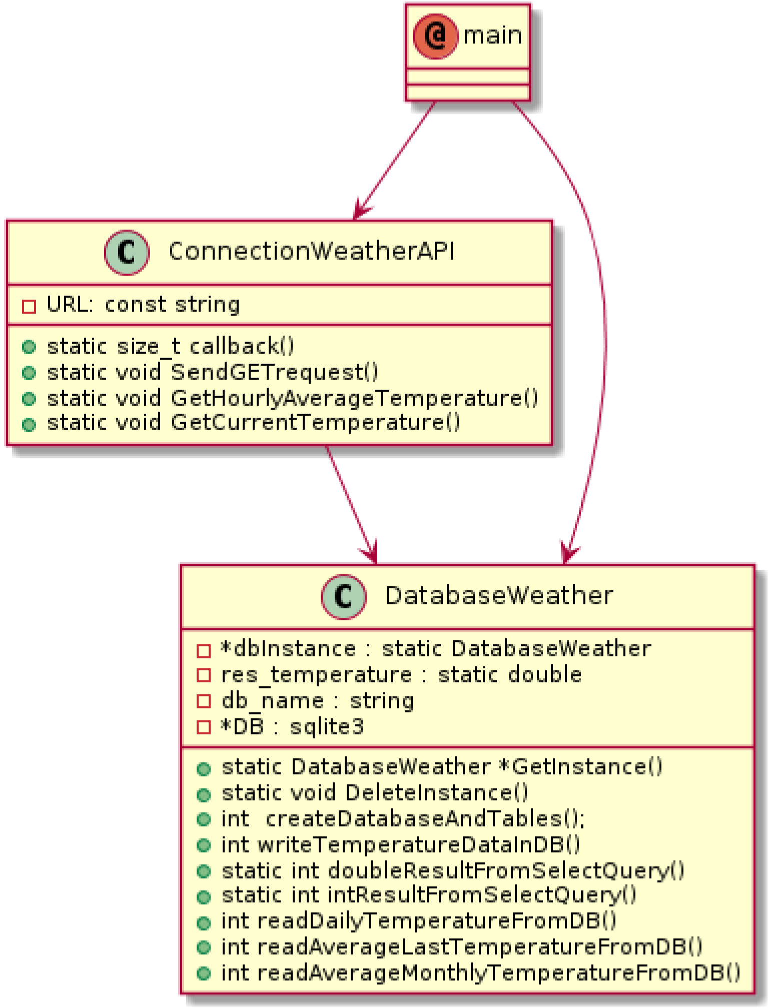
- ➢ It takes as input a command object in order to execute a specific action.

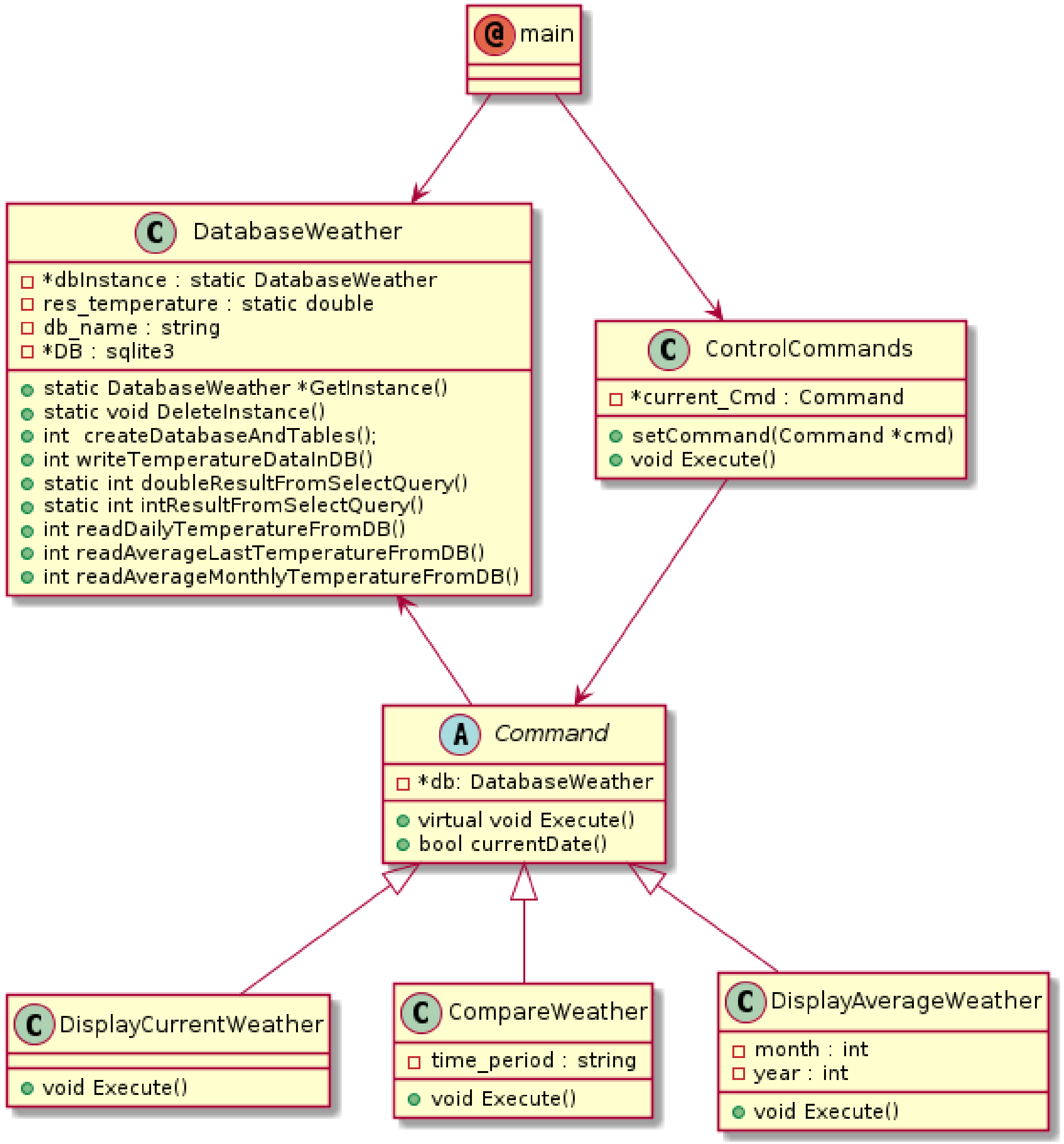
Processes Description

DatabaseWeather:

- It includes functions for (1) creating tables (2) writing on DB, (3) reading from DB and (4) storing data of specific SQL queries.

Structure of Processes


main
ConnectionWeatherAPI
URL: const string
static size_t callback()
static void SendGETrequest()
static void GetHourlyAverageTemperature()
static void GetCurrentTemperature()
DatabaseWeather
*dbInstance : static DatabaseWeather
res_temperature : static double
db_name : string
*DB : sqlite3
static DatabaseWeather *GetInstance()
static void DeleteInstance()
int createDatabaseAndTables();
int writeTemperatureDataInDB()
static int doubleResultFromSelectQuery()
static int intResultFromSelectQuery()
int readDailyTemperatureFromDB()
int readAverageLastTemperatureFromDB()
int readAverageMonthlyTemperatureFromDB()


main
DatabaseWeather
*dbInstance : static DatabaseWeather
res_temperature : static double
db_name : string
*DB : sqlite3
static DatabaseWeather *GetInstance()
static void DeleteInstance()
int createDatabaseAndTables();
int writeTemperatureDataInDB()
static int doubleResultFromSelectQuery()
static int intResultFromSelectQuery()
int readDailyTemperatureFromDB()
int readAverageLastTemperatureFromDB()
int readAverageMonthlyTemperatureFromDB()
ControlCommands
*current_Cmd : Command
setCommand(Command *cmd)
void Execute()
Command
*db: DatabaseWeather
virtual void Execute()
bool currentDate()
DisplayCurrentWeather
void Execute()
CompareWeather
time_period : string
void Execute()
DisplayAverageWeather
month : int
year : int
void Execute()

Write-Ahead Logging

Links:

https://sqlite.org/wal.html

https://sqlite.org/walformat.html

- The original content is mainting on the database file and the new content is stored initially on the WAL file. SQLite transfers all the transactions from WAL to the original database file automatically. (checkpoint)
- Threshold of 1000 pages
- Reading from the original database and writing on the WAL file.
- When the WAL is enabled:
  - Main database file (.db)
  - WAL file (.db-wal)
  - WAL-index (.db-shm): According to the SQLite documentation, it is the *"shared memory for coordinating access to the database and a cache for quickly locating frame within the wal file"*

Write-Ahead Logging

- What I observed:
  - Before I set journal mode equal to WAL, all read requests were failing during parallel read-write requests
  - When I set journal mode equal to WAL, few read requests are failing (during parallel read-write requests)

- 1st Issue:
  - According to the SQLite documentation, processes with a different root directory will use different shared memory areas (it is possible to use different WAL-index files), that leads to database corruption

- 1st Possible Solution
  - According to the SQLite documentation, when exclusive locking mode is set, SQLite uses heap memory instead the memory-mapped shm file (WAL-index files).

- 2nd Issue:
  - I did not manage to enable the exclusive locking mode with C++ code.
  - It is enabled only manually.

- 2nd Possible Solution
  - Use of Boost shared memory

Improvements

- ***ConnectionWeatherAPI*** *class*: use of namespace instead of class
- ***currentDate()*** *funciton:* use of one source file
- ***WeatherData*** process: should check if DB exists
- Updates on the ***WeatherData*** process makefile:

```
WeatherData: …..
    …..
    mkdir ../../Database
clean_db:
    rm ../../Database/WeatherDatabase.db*
```

Corrections

- ***DatabaseWeather*** *class:*

cout << endl << resName[0] << " = " << ((argv[0]) ? argv[0] : "NULL") << endl → before if (argc == 1 && argv[0]!=nullptr)

- ***ConnectionWeatherAPI*** *class:*
  - *remove unused variable*
  - *SendGETrequest(CURL* curl, string url, const string &receivedData, long &result_httpCode, CURLcode &result_curlCode)*
  - *Addition of curl_easy_cleanup(curl) on the GetCurrentTemperature() function*