



**POLITECNICO  
MILANO 1863**

GEOINFORMATICS PROJECT

**SIMILE Monitor Telegram bot**



# **SIMILE-Monitor**

Telegram bot

Author: Shengshen Li

Project teacher: Maria Antonella Brovelli

Project tutor: Daniele Oxoli

AY 2022/2023

## Table of contents

1. Description.....	3
1.1 Objective .....	3
1.2 Specification .....	3
1.3 Analysis of the specification .....	3
1.3.1 Architecture.....	3
2. Implementation .....	5
2.1 Redis .....	5
2.2 PostgreSQL .....	5
2.3 pyTelegramBotAPI .....	5
3. Test Cases.....	6
4. Conclusions.....	10

# 1. Description

## 1.1 Objective

Nowadays, lightweight, installation-free applications such as Telegram bots are highly favored by users. The author tries to develop a bot which users can help monitor the waters of Lake Como, Lake Lugano and Lake Maggiore, send photographs and other information about what they are observing (algae, foam, rubbish), forward the values of certain parameters (such as water transparency, temperature and pH), learn about lake environments and the organisms living there using a glossary, and find out about the environmental initiatives in programme in their area.

## 1.2 Specification

To access the Bot, the installation of Telegram is necessary and available on:

- Android
- IOS
- Desktop

Since it doesn't require installing any other specific application, it favors the contribution to sporadic users as well. The Bot can be used both in desktop and mobile versions, but it is meant for the second one, as it allows reporting real-time problems, that are sent and organized automatically in a database to be solved as soon as possible.

## 1.3 Analysis of the specification

### 1.3.1 Architecture

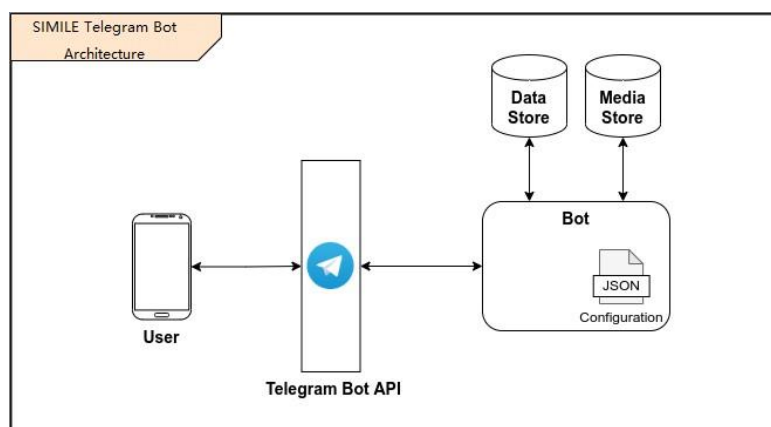


Fig.1 Architecture

It follows a basic, high-level architecture of the Bot.

All the communications between the end user and the Bot are disintermediated by the Telegram Bot API, while the Bot communicates directly with a database for data storage and, optionally, one for media file storage.

Under the hood, the Bot manages a finite-state machine received through the configuration. The states of the machine are the questions posed to the user, and the answers are what causes the transitions between states.

It follows a sequence diagram that shows the internal flow of the Bot when a user issues a new interaction with the /start command (remember that all the communications between the user and the Bot pass through the Telegram Bot API).

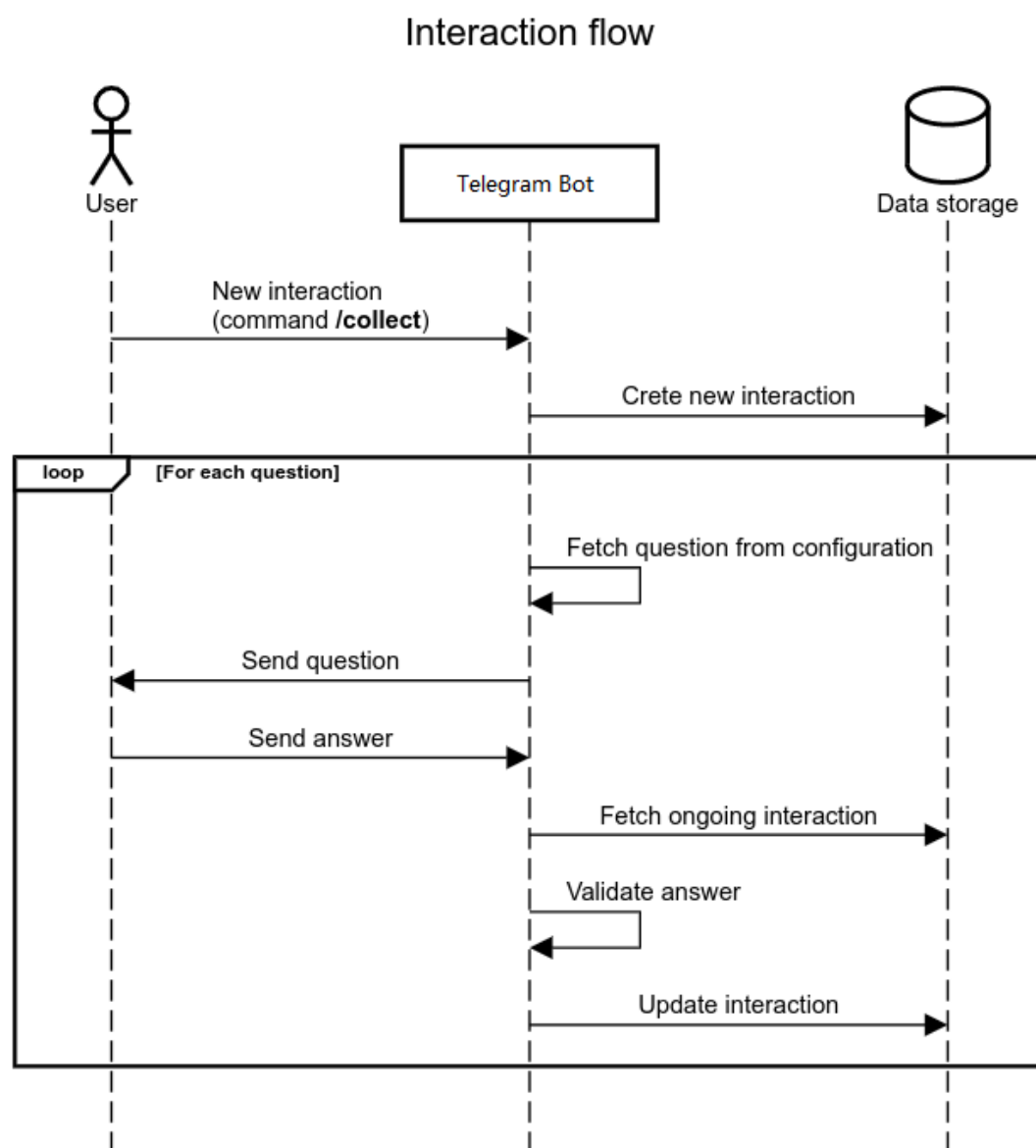


Fig.2 Interaction flow

## 2. Implementation

We developed this telegram bot using the Python programming language and the pyTelegramBotAPI library, along with Redis and PostgreSQL as our technology stack. Additionally, we utilized Leaflet to create a simple visualization interface for viewing user-uploaded data.

With the Telebot package, we can develop the bot efficiently using concise annotations. Combining Redis and PostgreSQL offers better performance and scalability. Redis is fast for read/write operations and supports various data structures. PostgreSQL provides transaction support and complex queries. By using both, we can optimize performance and handle different types of data efficiently, especially in application scenarios like Telegram bots, where frequent data read/write operations are required along with the need for permanent storage of user-input data.

### 2.1 Redis

We use Redis, a popular non-relational database, to temporarily store incomplete input information. Redis can be downloaded from [this website](http://download.redis.io/releases/redis-6.0.8.tar.gz) and installed by.

```
$ wget http://download.redis.io/releases/redis-6.0.8.tar.gz
$ tar -xvf redis-6.0.8.tar.gz
$ cd redis-6.0.8
$ make
```

### 2.2 PostgreSQL

PostgreSQL is a free object-relational database server (ORDBMS) that is distributed under a flexible BSD license.

PostgreSQL can be easily installed through the code below.

```
$ yum install -y postgresql13-server
```

### 2.3 pyTelegramBotAPI

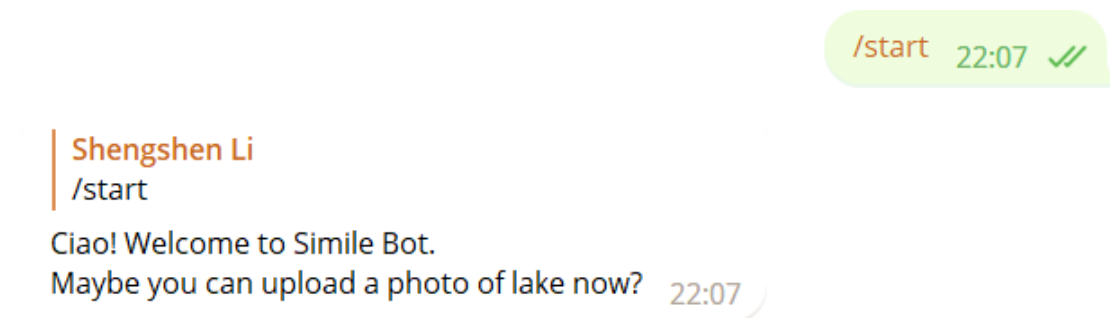
pyTelegramBotAPI is a bot framework for Telegram Bot API. This package provides the best of its kind API for command routing, inline query requests and keyboards, as well as callbacks.

The package can be installed by pip.

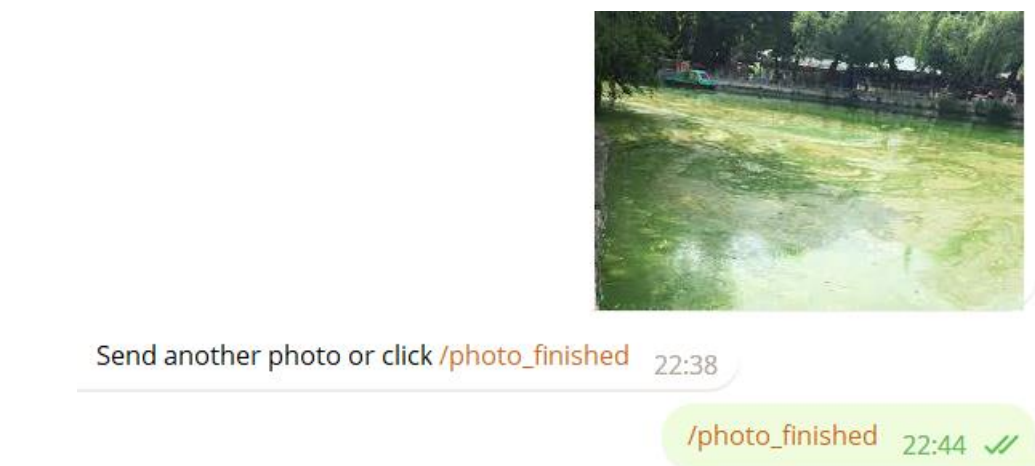
```
$ pip install pyTelegramBotAPI
```

### 3. Test Cases

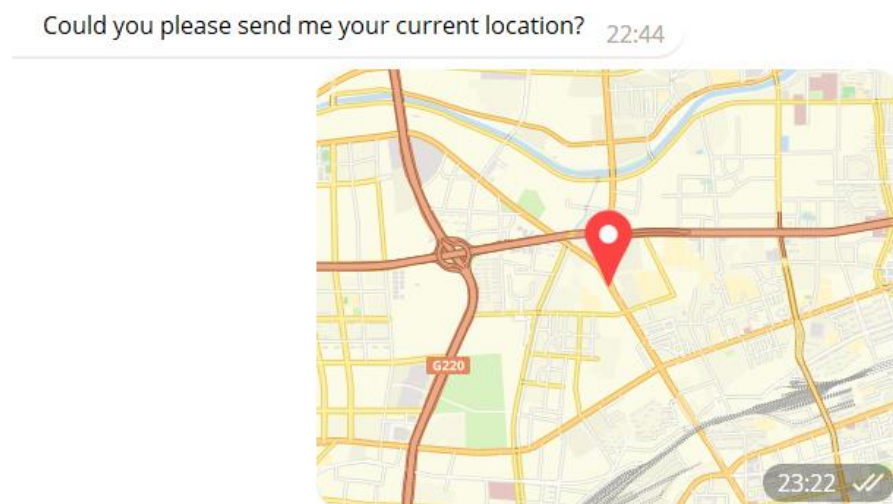
In the bot, registration is not required, and there is only one user group. After the "/start" command is issued, all users will automatically have a key-value pair created in Redis to store their relevant data.



Users are allowed to upload a minimum of 1 photo and a maximum of 3 photos. However, uploading 0 photos or more than 3 photos is not permitted.



Then users are asked to send its position by sidebar tool.



The bot will automatically calculate the weather situation of user's position and store them into redis.



The weather condition is 3, temperature is 24.81 celsius degree and wind is 2.03

23:22

Now you have successfully uploaded the photos and location. It is time to add details and measurements. Click [/details](#) to see the list. If you have no detail want to submit, click [/submit](#)

23:22

By sending `"/details"` command after finishing those previous steps, the user can add specific details of the observation.

[/details](#) 23:26 ✓✓

Click the detail you want to describe.

[/algae](#) [/foam](#) [/oilStains](#) [/litters](#) [/odours](#) [/drains](#) [/fish](#) [/birds](#)  
[/molluscs](#) [/crustaceans](#) [/turtles](#)

23:26

We will focus on `"/algae"` as an example, the rest of these commands get similar interaction modes and interfaces.

You want to talk about algae! What is the extension of the algae?

23:29

Menu



| Write a message...



<5 sq.m

5-20 sq.m

>20 sq.m




The bot will send you a multiple-choice option, and you can select the correct option by single-click in order to send it back to the bot.

After we select one, the bot will record the answer in redis and move to the next question.

You want to talk about algae! What is the extension of the algae? 23:29

5-20 sq.m 23:32 ✓✓

Please choose the looking of the algae. 23:32

Menu  Write a message...  

Scattered Compact Grouped

Surface stripes

After finishing all the information we need to provide, the “/algae” part will be recorded. The user can decide to continue to input new information by “/details” or submit by “/submit”.

Compact 23:33 ✓✓

Please choose the colour of the algae. 23:33

blue 23:33 ✓✓

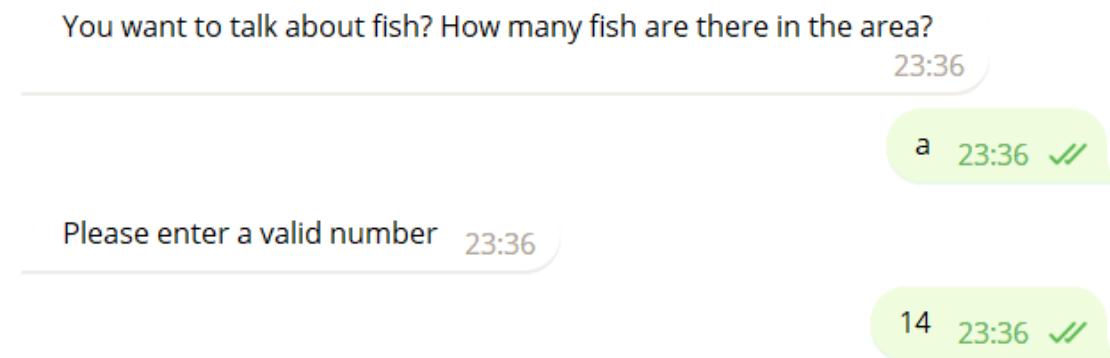
And is the algae iridescent? 23:33

Yes 23:33 ✓✓

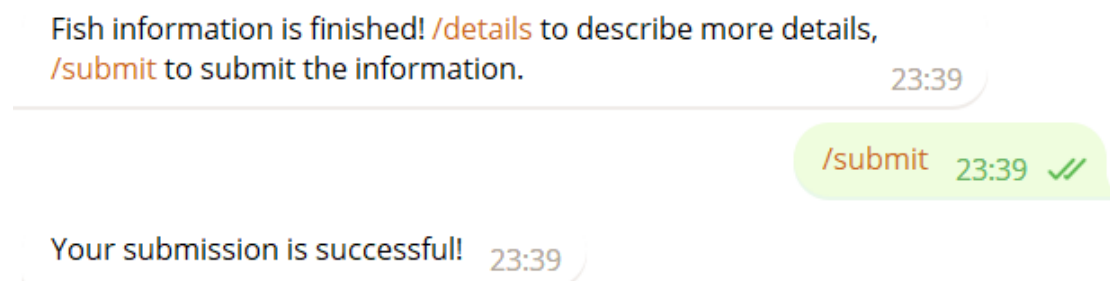
Algae information is finished! /details to describe more details,  
/submit to submit the information. 23:33



If the user needs to manually input an answer, the bot will also validate the format to ensure the correctness of the result.



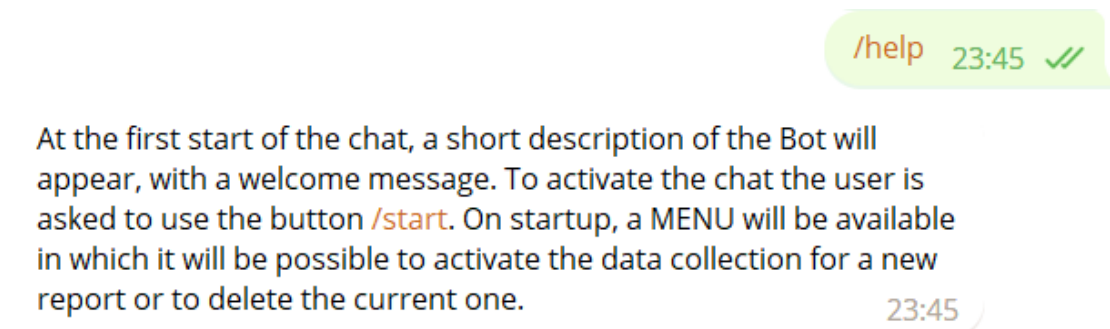
After finishing all the details, the user can submit by “/submit”.



Beside the standard operation procedure, the user can use commands from the menu.

- S **/start** Start the bot
- S **/continue** Continue on your previous work
- S **/detail** Provide the details of the lake after you provided the necessary i...
- S **/help** Provide guide of using this bot

By sending “/continue”, the use can continue inputing details from previous uncompleted report. And “/help” can give the users some useful tips about this bot.



## 4. Conclusions

In conclusion, the "SIMILE-Monitor" telegram bot offers a simple and user-friendly solution for individuals who are passionate about environmental conservation and wish to contribute to the well-being of the lakes. This tool has successfully transformed lake monitoring into an accessible and engaging activity that can be performed by anyone.

The bot's intuitive interface and streamlined yet parallel process allow users to actively participate in monitoring the lakes within just a few minutes. Users can easily contribute valuable information by submitting photographs and observations of algae, foam, and rubbish and forward important parameters such as water transparency, temperature, and pH.

But this tool also has its limitations:

1. **Data Credibility Limitations:** The bot relies on users actively uploading photos and providing observational data. This means that the quality and quantity of data depend on user participation. If user engagement is low, data collection may be limited.
2. **Data Coverage:** The bot can only collect data that users upload, which may not provide comprehensive coverage of lake pollution. This means that the bot cannot provide a complete picture of the entire lake's condition, but only fragments based on user-uploaded data.
3. **Scale Limitations:** Due to the inherent nature of telegram bot, it may not be able to handle large-scale data uploads or complex data analysis and processing requirements.

To overcome these limitations, it is advisable to combine the bot with other data collection methods and channels. This could involve collaborating with traditional monitoring agencies, utilizing sensors and automated devices for data collection, or encouraging broader user participation to improve data quality and coverage through methods such as citizen science.