WeatherApp: A Simple Qt GUI Application for Displaying Weather Information

Murat Kaliyev
Department of Computer Science
Nazarbayev University
Astana, Kazakhstan
m.kaliyev@nu.edu.kz

***Abstract*—In this paper I present a Qt-based GUI application called *WeatherApp* that retrieves weather information from *OpenWeatherMap* data source and display it in a user-friendly manner. This application fetches real-time weather information such as temperature, humidity, wind speed, and pressure and displays them on a simple graphical user interface (GUI). This project shows how to develop GUI with Qt and communicate with an API.**

***Index Terms*—Qt, C++, Weather Application, API, GUI Development**

I. Introduction

WeatherApp is a GUI that implemented with Qt 6 and C++. It prompts the user to input the name of a city and returns weather information. The app gets data from the OpenWeatherMap and shows primary weather parameters.

The key features include:

- Widget class for real-time weather data from OpenWeatherMap API;
- QT Widgets for a simple and easy-to-use interface.
- Showing weather icons corresponding to the current weather condition.
- Cross-platform support for Window and Linux.

This document explains the design, development, and key aspects of the implementation.

II. System Design and Implementation

The application consists of three main parts:

- **User Interface (UI)**: The frontend is built using Qt Widgets.
- **Network Handling**: API requests are managed using `QNetworkAccessManager`.
- **Data Processing**: JSON responses are parsed using `QJsonDocument`.

A. Application Workflow

1) The user enters a city name.
2) The application sends a request to OpenWeatherMap API.
3) The API returns a JSON response containing weather data.
4) The data is extracted and displayed in the UI.
5) The weather icon updates according to the weather condition.

III. Graphical User Interface (GUI)

The UI consists of:

- A text box for entering the city name.
- A button to fetch the weather data.
- Labels displaying the temperature, humidity, wind speed, and pressure.
- An icon representing the current weather condition.

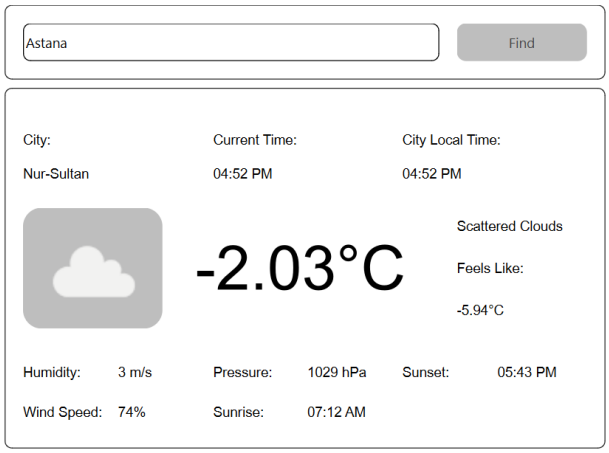

Fig. 1. User Interface of the WeatherApp.

A. Styling the UI

The appearance of the UI is customized using Qt stylesheets.

IV. Fetching Weather Data

The application communicates with OpenWeatherMap API using Qt's `QNetworkAccessManager`.

A. Sending an API Request

B. Processing JSON Response

C. Displaying Weather Icons

Weather icons are stored in a local directory to avoid network issues.

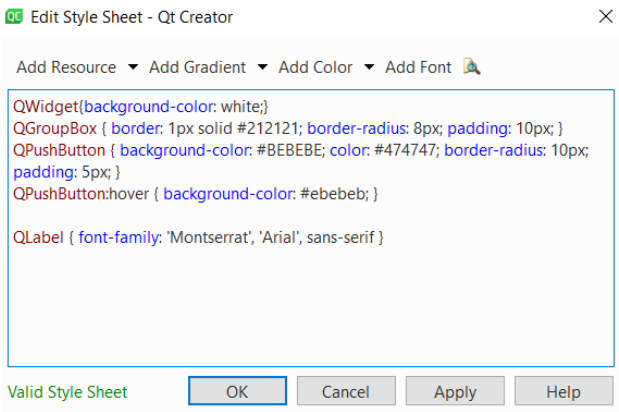

Fig. 2. Qt Stylesheet for UI Elements

```cpp
void MainWindow::fetchWeather() {
    if (API_KEY.isEmpty()) {
        ui->labelCityName->setText("Error: API Key not loaded!");
        return;
    }

    QString city = ui->lineEditCity->text().trimmed();
    if (city.isEmpty()) {
        ui->labelCityName->setText("Please enter a city name.");
        return;
    }

    QString url = "https://api.openweathermap.org/data/2.5/weather?q=" + city +
                  "&appid=" + API_KEY + "&units=metric";

    qDebug() << "Requesting URL:" << url;

    QNetworkRequest request((QUrl(url)));
    QNetworkReply *reply = manager->get(request);

    connect(reply, &QNetworkReply::finished, this, &MainWindow::handleResponse);
}
```

Fig. 3. Sending a Request to OpenWeatherMap API

```cpp
if (reply->error() == QNetworkReply::NoError) {
    QByteArray response = reply->readAll();
    qDebug() << "API Response:" << response;

    QJsonDocument jsonDoc = QJsonDocument::fromJson(response);
    QJsonObject jsonObj = jsonDoc.object();

    QString cityName = jsonObj["name"].toString();
    double temp = jsonObj["main"].toObject()["temp"].toDouble();
    double feelsLike = jsonObj["main"].toObject()["feels_like"].toDouble();
    int humidity = jsonObj["main"].toObject()["humidity"].toInt();
    double windSpeed = jsonObj["wind"].toObject()["speed"].toDouble();
    int pressure = jsonObj["main"].toObject()["pressure"].toInt();
    QString desc = jsonObj["weather"].toArray()[0].toObject()["description"].toString();
    QString iconCode = jsonObj["weather"].toArray()[0].toObject()["icon"].toString();

    QDateTime sunriseTime = QDateTime::fromSecsSinceEpoch(jsonObj["sys"].toObject()["sunr
    QDateTime sunsetTime = QDateTime::fromSecsSinceEpoch(jsonObj["sys"].toObject()["sunse

    int timezoneOffset = jsonObj["timezone"].toInt();
    QDateTime utcNow = QDateTime::currentDateTimeUtc();
    QDateTime cityTime = utcNow.addSecs(timezoneOffset);

    QDateTime userLocalTime = QDateTime::currentDateTime();
```

Fig. 4. Parsing JSON Response from API

```cpp
void MainWindow::loadWeatherIcon(QString iconCode) {
    QString iconPath = QCoreApplication::applicationDirPath() + "/icons/" + iconCode + "@2x.png";
    qDebug() << "Loading Weather Icon from:" << iconPath;

    QPixmap pixmap(iconPath);
    if (pixmap.isNull()) {
        qDebug() << "Error: Failed to load icon from local directory!";
    } else {
        qDebug() << "Weather icon loaded successfully!";
        ui->labelIcon->setPixmap(pixmap.scaled(300, 300, Qt::KeepAspectRatio, Qt::SmoothTransformation));
    }
}
```

Fig. 5. Loading Weather Icons from Local Directory

V. Challenges and Solutions

During development, some issues were encountered:

- **API Key Security**
  - **Problem**: Having your API key hard-coded in your code
  - **Solution**: The key is stored in a **.env** file instead.
- **SSL Errors with Icons**
  - **Problem**: Application unable to get icons due to SSL issues.
  - **Solution**: Weather Icons are downloaded.
- **UI Update Issues**
  - **Problem**: UI elements were not updating properly after fetching data.
  - **Solution**: The `update()` function was used to refresh labels.

VI. Conclusion and Future Improvements

WeatherApp uses Qt to send and receive Weather data It showcases the GUI side of Qt and its ability to deal with APIs.

Possible improvements include:

- Adding a **5-day forecast feature**.
- Using **GPS location** to detect the user's city automatically.
- Improving error handling for network failures.

The project is a simple yet effective example of Qt GUI development.

VII. Project Repository

The complete source code is available on GitHub:

https://github.com/Nocs782/Weather-soft-Qt/tree/main

References

[1] Qt Documentation, "Qt Network Module," Available: https://doc.qt.io/qt-6/qtnetwork-module.html
[2] OpenWeatherMap API Documentation, Available: https://openweathermap.org/api