УДК 004

Создание приложения «погода» для компьютера на python

Вавилов Егор Дмитриевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается возможность создания приложения по прогнозу погоды для компьютера. Приложение будет написано на языке программирования Python. Практическим результатом является компьютерное приложение.
Ключевые слова: python, приложение, погода

Building a weather application for a computer in python

Vavilov Yegor Dmitrievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes how to create a weather forecast application for your computer. The application will be written in the Python programming language. The bottom line is a computer application.
Keywords: python, application, weather

Любой человек всегда желает одеваться по погоде. Не особо приятно, когда легко одевшись, выходишь на улицу, а там идет дождь и ветрено. Существует множество программ, телепередач, приложений, которые связываются с метеорологическими станциями и выводят данные о погоде.

В данной статье рассматривается возможность создания компьютерного приложения о погоде.

Е.В. Корныхин, А.В Хорошилова, рассмотрели подход к описанию и верификации структурных ограничений на архитектурные модели, в основе которого лежит переиспользование возможностей языка программирования Python [1]. Н.Б. Дерябин, В.Г. Соколов, Д.Д. Жданов, М.С Копылов. рассмотрели вопросы по работе с языком python в направлениях оптического моделирования и визуализации [2]. А.А.Зуенко, А.А Алмаматов в своей работе рассмотрели различные особенности подходов решения задач с ограничениями [3]. А.А. Гладких разработал расширение в системах автоматизированного проектирования с использованием языка python [4].

Выполнение запросов к API может занять некоторое время. Если это выполняется в основном цикле приложения, то это приведет к зависанию

приложения в ожидании данных. Чтобы избежать этого, выполняем все запросы в отдельных рабочих потоках,

Для начала сделаем «воркер», который будет собирать текущую погоду и прогноз и возвращать их в основной поток для обновления пользовательского интерфейса. Сначала определяем ряд настраиваемых сигналов, которые может испускать «воркер». К ним относятся готовый общий сигнал для выполнения работником, ошибка, которая выдает «Exception» сообщение в случае возникновения ошибки, и результат, который возвращает результат вызова API. Данные возвращаются в виде двух отдельных «dict» объектов, один из которых представляет текущую погоду, а другой – прогноз (рис.1).

```
:::python
class WorkerSignals(QObject):
    '''
    Defines the signals available from a running worker thread.
    '''
    finished = pyqtSignal()
    error = pyqtSignal(str)
    result = pyqtSignal(dict, dict)
```

Рисунок 1 - worker

Далее напишем «WeatherWorker», который обрабатывает фактические запросы к API. Он инициализируется одним параметром, «location» который указывает местоположение, для которого «воркер» будет получать данные о погоде. Каждый «воркер» выполняет два запроса, один для погоды, а другой для прогноза, получая строки JSON от OpenWeatherMap.org. Затем они распаковываются в «dict» объекты и выводятся с помощью «.result» сигнала (рис.2).

```python
:::python
class WeatherWorker(QRunnable):
    '''
    Worker thread for weather updates.
    '''
    signals = WorkerSignals()
    is_interrupted = False

    def __init__(self, location):
        super(WeatherWorker, self).__init__()
        self.location = location

    @pyqtSlot()
    def run(self):
        try:
            params = dict(
                q=self.location,
                appid=OPENWEATHERMAP_API_KEY
            )

            url = 'http://api.openweathermap.org/data/2.5/weather?%s&units=metric'
            r = requests.get(url)
            weather = json.loads(r.text)
```

Рисунок 2 – WeatherWorker

Пользовательский интерфейс приложения был создан с помощью «Qt Designer» и сохранен в виде «.ui» файла. Он был преобразован в импортируемый файл Python с использованием «pyuic5». С макетом главного окна, определенным в «Qt Designer». Чтобы создать главное окно, просто создаем подкласс «Ui_MainWindow» и вызываем его «self.setupUi» как обычно. Для того, чтобы инициировать запрос на данные о погоде с помощью кнопки связываем этот сигнал нашего пользовательского «update_weather» слота (рис.3).

```python
:::python
class MainWindow(QMainWindow, Ui_MainWindow):

    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.pushButton.pressed.connect(self.update_weather)
        self.threadpool = QThreadPool()
        self.show()
```

Рисунок 3 - интерфейс

Далее сделаем обработку результата. «dict» объекты, возвращают сигнал, через «воркер». Этот сигнал подключен к пользовательскому слоту «weather_result», который получает два «dict» объекта. Этот метод отвечает за обновление пользовательского интерфейса возвращаемым результатом, отображая как числовые данные, так и обновляя значки погоды. Результаты погоды обновляются в пользовательском интерфейсе с «setText» помощью

определенного «QLabels», форматированного до десятичных знаков, где это необходимо (рис.4).

```python
:::python
    def weather_result(self, weather, forecasts):
        self.latitudeLabel.setText("%.2f °" % weather['coord']['lat'])
        self.longitudeLabel.setText("%.2f °" % weather['coord']['lon'])

        self.windLabel.setText("%.2f m/s" % weather['wind']['speed'])

        self.temperatureLabel.setText("%.1f °C" % weather['main']['temp'])
        self.pressureLabel.setText("%d" % weather['main']['pressure'])
        self.humidityLabel.setText("%d" % weather['main']['humidity'])

        self.weatherLabel.setText("%s (%s)" % (
            weather['weather'][0]['main'],
            weather['weather'][0]['description']
        )
```

Рисунок 4 – обработка результата

Сначала устанавливаем значок текущей погоды из «weatherdict», затем перебираем первые 5 из предоставленных прогнозов. Прогноз иконки, время и температура иконки были определены в «Qt Designer» с именами «forecastIcon<n>», «forecastTime<n>» и «forecastTemp<n>», и делает просто итерацию над ними и получает их, используя «getattrc» текущим индексом итерации (рис.5).

```
f.set_weather_icon(self.weatherIcon, weather['weather'])

 n, forecast in enumerate(forecasts['list'][:5], 1):
 getattr(self, 'forecastTime%d' % n).setText(from_ts_to_time_of_day(forecast['dt']
 self.set_weather_icon(getattr(self, 'forecastIcon%d' % n), forecast['weather'])
 getattr(self, 'forecastTemp%d' % n).setText("%.1f °C" % forecast['main']['temp'])
```

Рисунок 5 – установка иконок

Теперь осталось запустить данный код и проверить работоспособность данного приложения (рис.6).

Рисунок 6 – работа приложения

После запуска приложения открывается окно, где расписаны данные о погоде на некоторый временной период.

В данной статье была рассмотрена возможность создания компьютерного приложения о погоде на языке Python.

Библиографический список

1. Корныхин Е.В., Хорошилова А.В Использование языка программирования python для описания ограничений на архитектурные модели// Автоматика. Вычислительная техника. 2014. №6. С. 27-29
2. Дерябин Н.Б., Соколов В.Г., Жданов Д.Д., Копылов М.С. Автоматизация генерации серий реалистичных изображений с использованием языка сценариев python // Современные научные исследования и инновации. 2015. № 7-5 (43). С. 132-136.
3. Зуенко А.А., Алмаматов А.А. Программирование в ограничениях на языке python с применением структур и алгоритмов алгебры кортежей// Труды Международного симпозиума «Надежность и качество». 2016. №7. С. 14-20.
4. Гладких А.А. Использование языка программирования python для разработки расширений систем автоматизированного проектирования // Автоматика. Вычислительная техника. 2019. №1. С. 28-30