МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Университет «Дубна»
(государственный университет Дубна)

Институт системного анализа и управления

ДНЕВНИК
учебной практики

Технологии разработки приложений в области профессиональной деятельности
наименование практики

Студент _____**Осипенко Ольга Павловна**_____ /_____/
Ф.И.О. подпись

Группа № ____**2254** ___, __ курс, ____*очная*__ форма обучения
очная, очно-заочная, заочная

Направление подготовки (специальность, профессия): 09.03.04 – Программная инженерия_____

Место прохождения практики: _____*ФГБОУ ВО «Университет «Дубна»*_____
полное наименование организации

Руководители практики:

от университета **доцент Думбрайс Крыстина Ольгертовна**
должность, Ф.И.О.

от организации _____-/-_____
должность, Ф.И.О.

Сроки прохождения практики ____24.06.2024 г. – 07.07.2024 г._____

Дубна, 2024 г.

Дата	Выполняемая работа	Кол-во часов	Отметка о выполнении	Подпись непосредственного руководителя по месту прохождения практики
24.06.2024 – 27.06.2024	Постановка задачи на практику, изучение исходных данных задачи	10	Выполнено	Думбрайс Крыстина Ольгертовна
28.06.2024 – 04.07.2024	Поэтапная работа над задачей	88	Выполнено	Думбрайс Крыстина Ольгертовна
05.07.2024 – 07.07.2024	Оформление отчета	10	Выполнено	Думбрайс Крыстина Ольгертовна
	Итого по учебному плану	108		

Руководители практики:

от университета ___**Думбрайс Крыстина Ольгертовна**____ /_____/
должность, Ф.И.О. подпись

от организации _____-/-_____ /_____/
М.П. (при наличии), должность, Ф.И.О. подпись

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Университет «Дубна»
(государственный университет Дубна)

Институт системного анализа и управления
Кафедра распределенных информационно-вычислительных систем

«Утверждаю»
Заведующий кафедрой
_____ проф. Кореньков В.В.

Отчет по учебной практике (Технологии разработки приложений в области профессиональной деятельности)

Определение местонахождения клиента по IP-адресу

Студент-практикант **Осипенко Ольга Павовна**

Группа студента **2254** Направление: 09.03.04 – Программная инженерия

Место прохождения практики ФГБОУ ВО «Университет «Дубна»

Руководитель от кафедры **Думбрайс Крыстина Ольгертовна**
Консультант от кафедры _____
Руководитель/Консультант от предприятия _____

Рекомендуемая оценка _____
(оценка) (подпись руководителя от кафедры)

Рекомендуемая оценка _____
(оценка) (подпись руководителя/консультанта от предприятия)

Дата представления отчета «_6_» __июля_ 2024 г.

Студент-практикант _____**Осипенко О.П**_____
(подпись)

Введение

Для решения ряда задач, таких как геотаргетинг, предотвращение мошенничества, установка даты и времени на устройстве, аналитика и статистика, сетевая оптимизация, часто бывает нужно определение местонахождение клиента по сетевому адресу. Существует разные способы сделать это, и в данной курсовой работе был использован специальный сервис с готовым *API – DaData.*

Сервис позволяет на основе *IP*-адреса определить местоположение клиента с точностью до города, а также сопутствующую информацию – почтовый индекс, регион, тип региона, округ и т.п. Сутью курсовой работы будет создание двухстраничного приложения для демонстрации работы данного *API.*

Для написания работы был выбран язык программирования *Python* по причине личного желания получить больше опыта в использовании этого языка и его библиотек. Для реализации пользовательского интерфейса выбрана библиотека *PyQt6* ввиду больших возможностей для кастомизации приложения по сравнению с аналогами.

Постановка задачи

Название

Определение местонахождения клиента по *IP*-адресу

Цель

Создание приложения на языке *Python* с использованием библиотеки *PyQt6*, которое будет на основе введенного *IP*-адреса определять местонахождение с точностью до города. Приложение должно уметь самостоятельно определять адрес в сети компьютера, на котором производится запуск.

Задачи

- Разработать классы:
    - Создать классы для представления графического интерфейса приложения, состоящего из 2 окон.
    - Создать класс-оболочку для *IP*-адреса, в котором будут основные действия над введенным *IP*
- Изучить информацию
    - Изучить библиотеку *PyQt6* и ее инструменты для создания графического интерфейса приложения.
    - Ознакомиться в предоставленным *API – DaData*, с помощью которого и будет определено местонахождение клиента.
    - Найти сервис, который будет самостоятельно определять *IP*-адрес компьютера, на котором запущено приложение.
- Разработать функционал обработки данных
    - Реализовать метод валидации введенного текста.
    - Реализовать метод перевода полученного из *API* значения в тип, удобный для вывода в интерфейс.
    - Реализовать методы использования инструментов автозаполнения *IP*-адреса и *DaData*.
- Разработать механизм визуализации:
    - Обеспечить ввод и вывод информации, понятный пользователю
    - Создать маркеры успешной и неуспещной валидации данных.
    - Создать события переключения окон приложения между собой
    - Ограничить возможность «спама» при отправлении запросов и долгом ожидании обработки.

Исходные данные

Сервис *DaData* с *API*, библиотека *PyQt6*, библиотека *urllib.request*

Результат

Приложение на языке *Python*, реализующее выполнение всех задач из раздела «Задачи».

Критерии оценки результата

Корректность работы методов:

- При введении некорректного адреса пользователю это отображается и предлагается ввести заново.
- Местоположение определяется правильно или близко к правильному. При невозможности определения выводится соответствующая ошибка.
- *JSON*-документ обрабатывается точно без утери информации.

Качество визуализации:

- Интерфейс удобен и прост в использовании..
- Валидация визуализирована.
- При долгом ожидании ответа от сервера кнопка блокируется, чтобы не было возможности перегрузить сервер и получить от него бан.

Теоретическая часть

Для понимания проекта необходимо иметь представление о следующих понятиях:

IP-адрес — это адрес устройства в сети. Для подключения устройства непосредственно в сеть Интернет требуется внешний *IP*-адрес, поэтому точность будет только до города.

API (Application Programming Interface) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

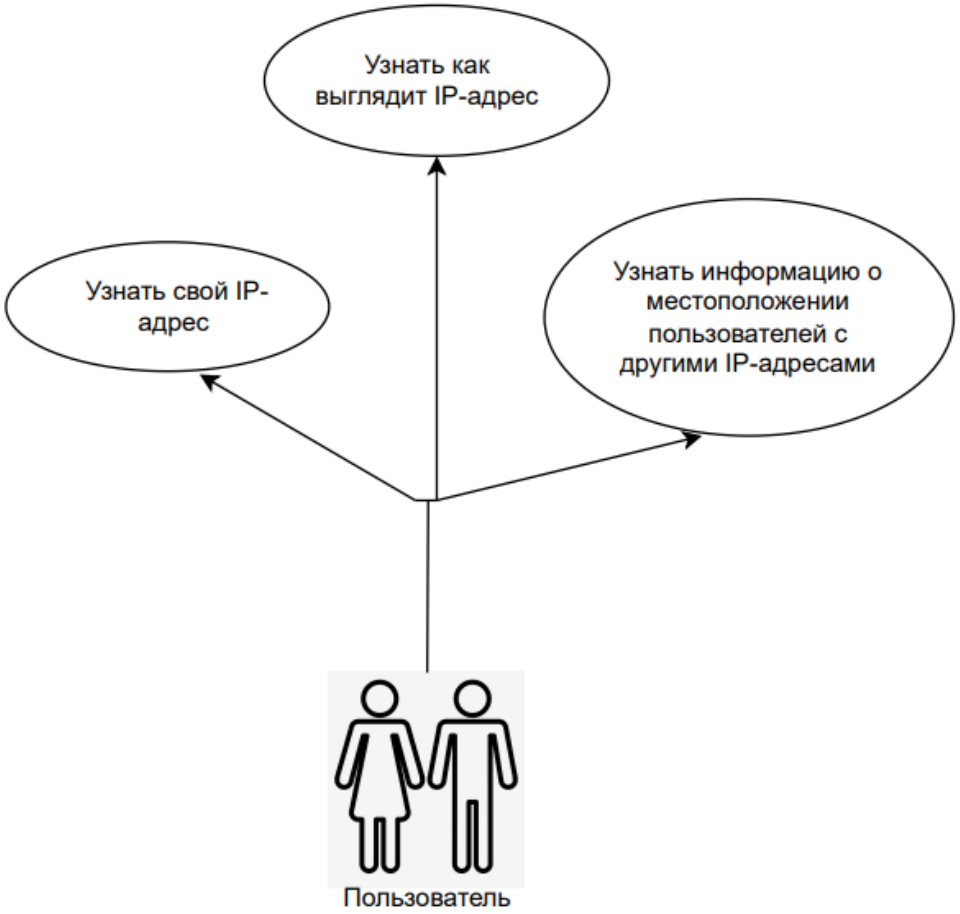


Схема 1.

Практическая часть

Описание продукта

Продукт представляет из себя двухстраничное приложение, написанное на языке Python. После запуска открывается данное окно (рис.1). Пользователь должен ввести *IP*-адрес в текстовое поле.

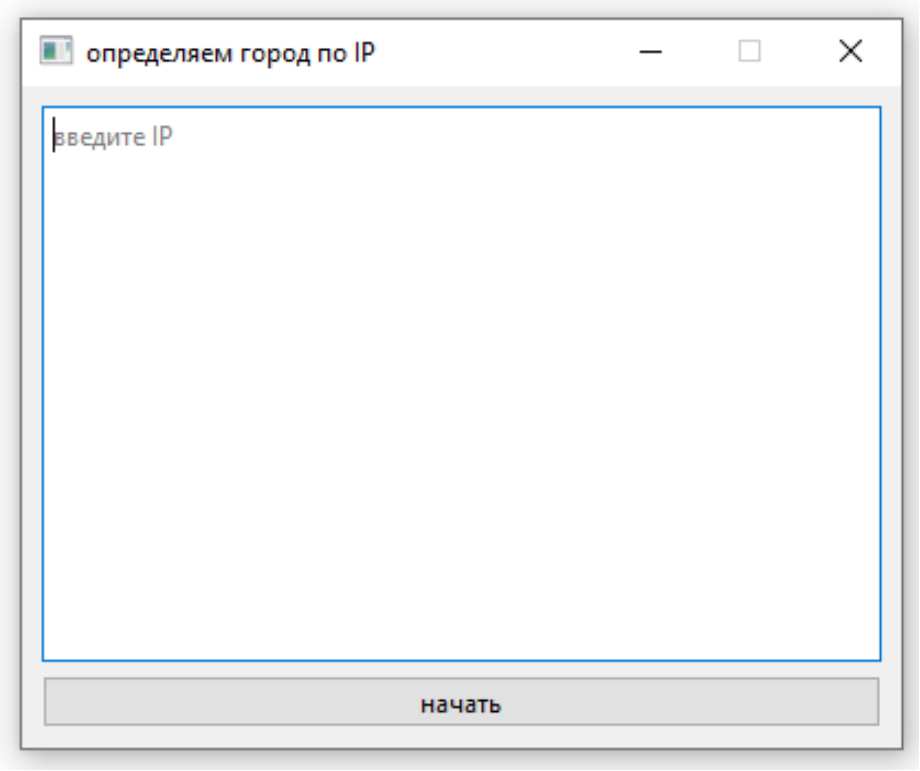


Рис 1.

При вводе набора символов, не являющихся адресом, возникает предупредение (рис.2)

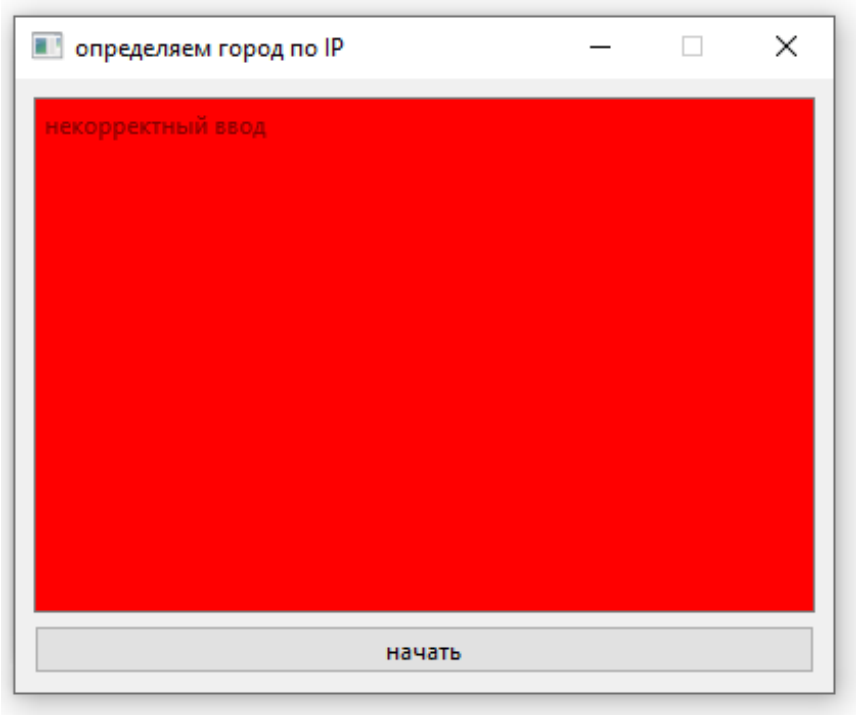


Рис 2.

Также есть опция пустого ввода. В таком случае *IP*-адрес заполнится автоматически. Или же есть ручной ввод. Введем любой *IP*-адрес (рис.3). Вывод будет следующим (рис.4).

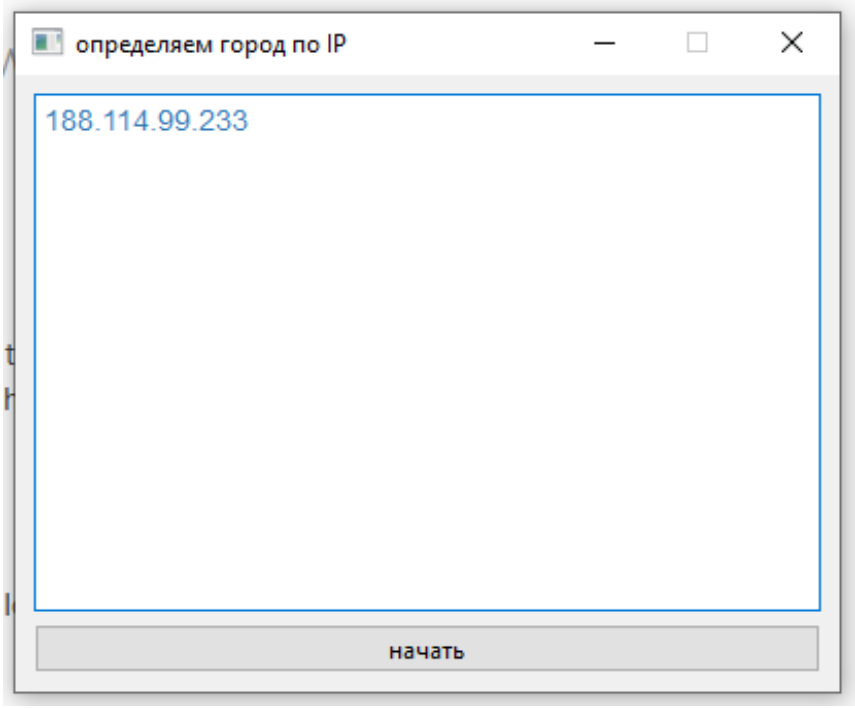


Рис 3.

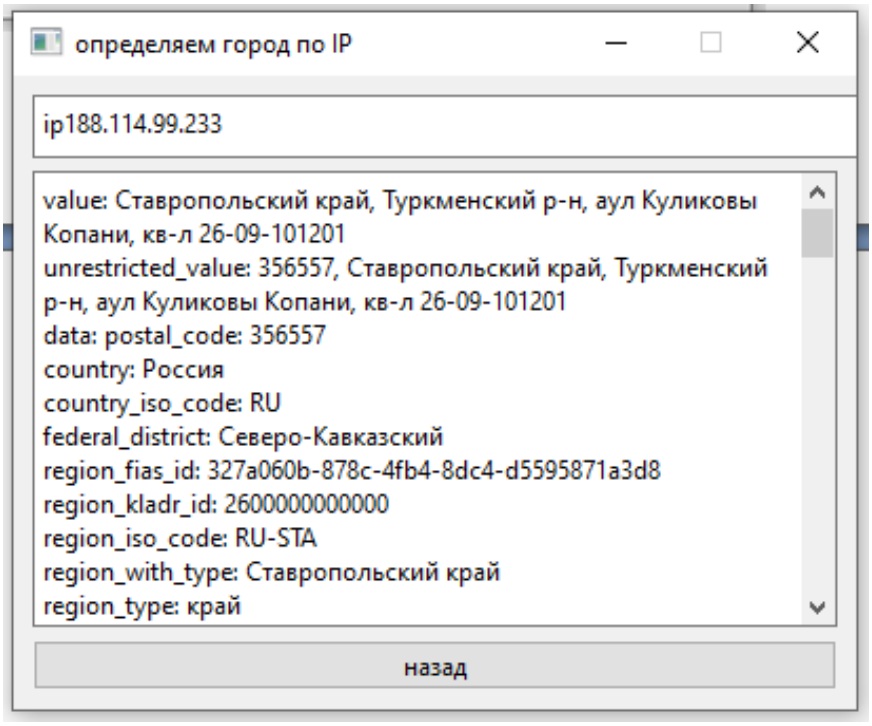


Рис 4.

При пустом вводе вывод будет следующим, то есть программа самостоятельно введет адрес устройства, с которого произведен вход (рис.5):

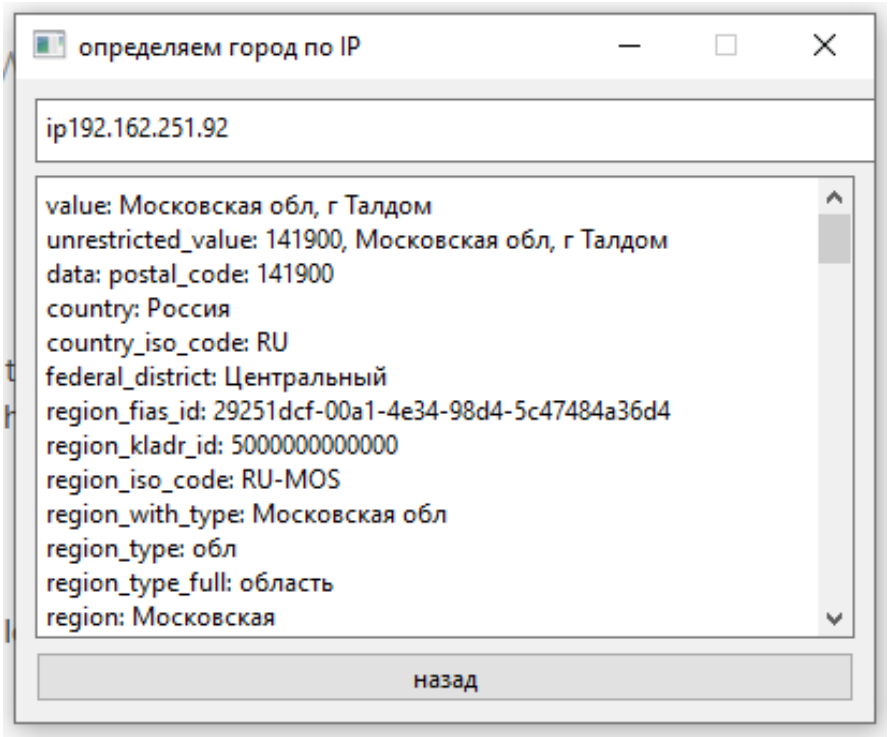


Рис 5.

В некоторых случаях сервис не может определить город, и приложение обработает этот случай (рис.6)

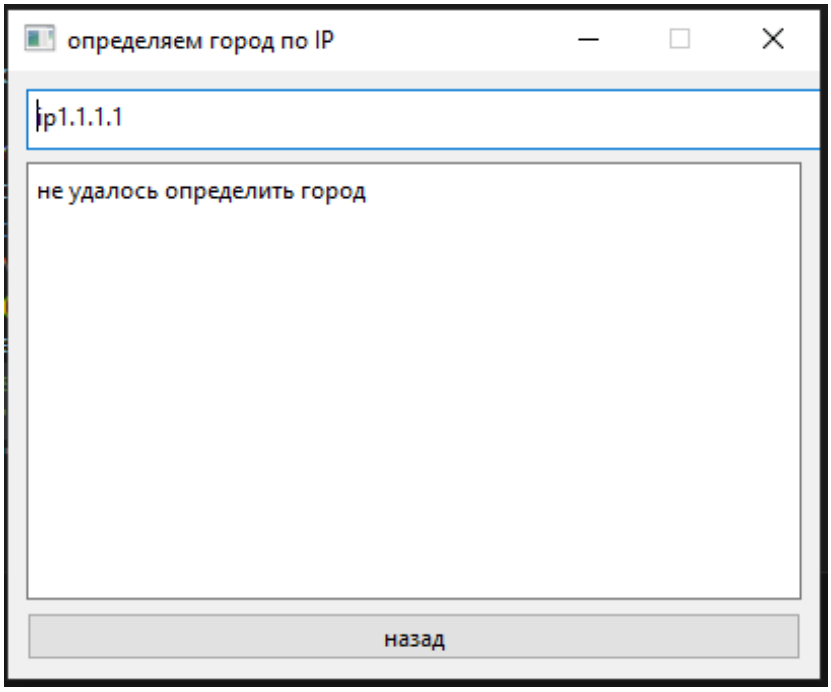


Рис 6.

Программная реализация

Код написан с подходом ООП.

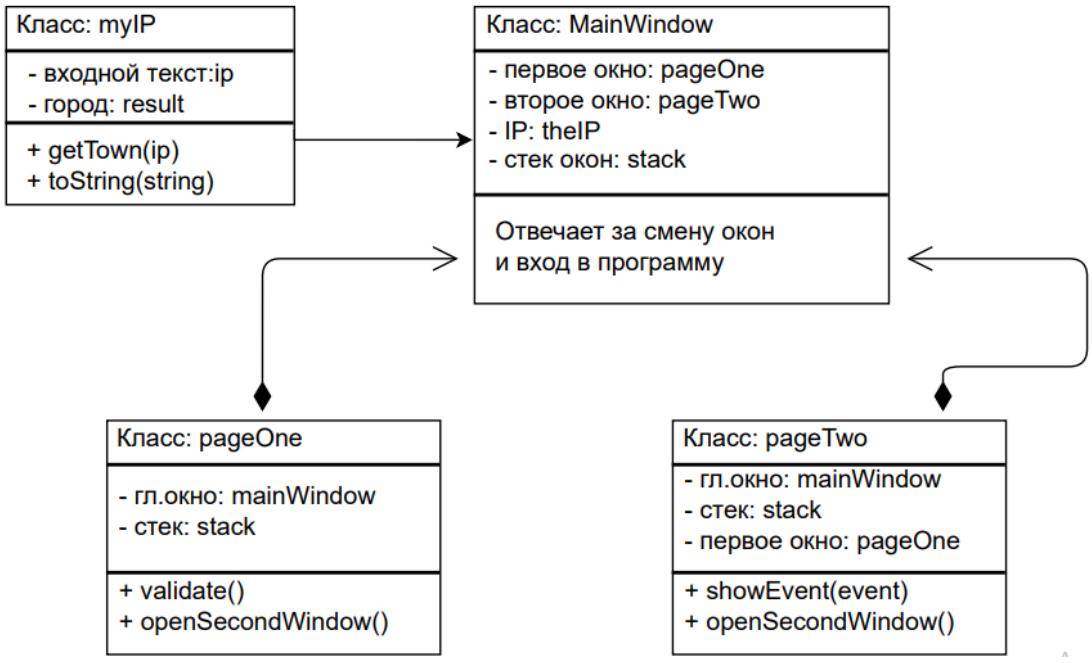


Схема 2.

Описание кода

```
class myIP():
    def __init__(self,text):
        self.ip = text
        self.getTown(self.ip)

    def getTown(self, ip):
        token = "c3579b2fe45f07d558304c3e678a0eb9b080b05d"
        dadata = Dadata(token)
        res = dadata.iplocate(self.ip)
        if res == None:
            self.result = "не удалось определить город"
            return
        print(type(res))
        self.result = self.toString(res)

    def toString(self, dictionary):
        st = ''
        for i in dictionary.keys():
            value = dictionary[i]
            if value == None:
                value = "None"
            if isinstance(value, dict):
                value = self.toString(value)
            st += i + ': ' + value + '\n'
        return st
```

Рис 7.

Центральный класс проекта – класс *IP*-адресов. Использование *API* находится в методе *getTown()*. Метод *toString()* реализован по причине того, что ответ сервера приходит в виде словаря (*JSON* – документ), а это недостаточно гибкий тип, чтобы вывести его.

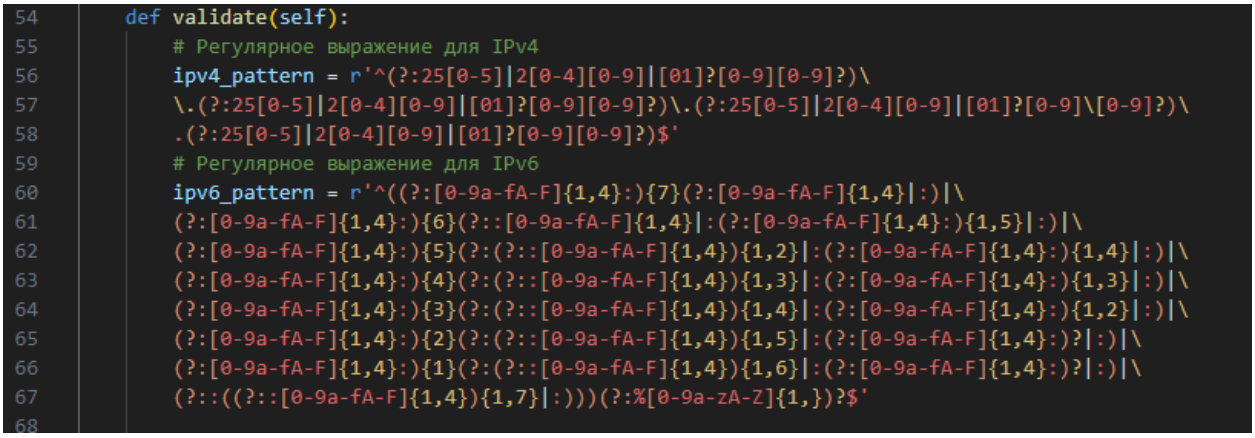
```
class PageOne(QWidget):

    def __init__(self, stack, mainWindow):
        super().__init__()
        self.stack = stack
        self.initUI()
        self.mainWindow = mainWindow

    def initUI(self):
        self.setFixedSize(QSize(400, 300))
        layout = QVBoxLayout()
        self.textEdit = QTextEdit()
        self.textEdit.setPlaceholderText('введите IP')
        self.button = QPushButton('начать')
        self.button.clicked.connect(self.openSecondWindow)
        layout.addWidget(self.textEdit)
        layout.addWidget(self.button)
        self.setLayout(layout)
```

Рис 8.

Один из классов окон. Его создание и инициализация компонентов.



```
    def validate(self):
        # Регулярное выражение для IPv4
        ipv4_pattern = r'^(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\
        \.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9]\[0-9]?)\
        .(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$'
        # Регулярное выражение для IPv6
        ipv6_pattern = r'^((?:[0-9a-fA-F]{1,4}:){7}(?:[0-9a-fA-F]{1,4}|:)|\
        (?:[0-9a-fA-F]{1,4}:){6}(?::[0-9a-fA-F]{1,4}|:(?:[0-9a-fA-F]{1,4}:){1,5}|:)|\
        (?:[0-9a-fA-F]{1,4}:){5}(?:(?::[0-9a-fA-F]{1,4}){1,2}|:(?:[0-9a-fA-F]{1,4}:){1,4}|:)|\
        (?:[0-9a-fA-F]{1,4}:){4}(?:(?::[0-9a-fA-F]{1,4}){1,3}|:(?:[0-9a-fA-F]{1,4}:){1,3}|:)|\
        (?:[0-9a-fA-F]{1,4}:){3}(?:(?::[0-9a-fA-F]{1,4}){1,4}|:(?:[0-9a-fA-F]{1,4}:){1,2}|:)|\
        (?:[0-9a-fA-F]{1,4}:){2}(?:(?::[0-9a-fA-F]{1,4}){1,5}|:(?:[0-9a-fA-F]{1,4}:)?|:)|\
        (?:[0-9a-fA-F]{1,4}:){1}(?:(?::[0-9a-fA-F]{1,4}){1,6}|:(?:[0-9a-fA-F]{1,4}:)?|:)|\
        (?::((?::[0-9a-fA-F]{1,4}){1,7}|:)))(?:%[0-9a-zA-Z]{1,})?$'
```

Рис 9.

Метод *validate()* находится в классе *pageOne*. Валидация реализована с помощью регулярный выражений. Она сделана для обеих версий *IP*-адресов, доступных сейчас. На рис.9 показана первая часть метода.

```
if text == '':
    mainWindow.theIP = myIP(urllib.request.urlopen('https://ident.me').read().decode('utf8'))
    print(mainWindow.theIP)
    return True
elif re.match(ipv4_pattern, text):
    mainWindow.theIP = myIP(text)
    return True
elif re.match(ipv6_pattern, text):
    mainWindow.theIP = myIP(text)
    return True
else:
    return False
```

Рис 10.

Вторая часть метода *validate()*, здесь интерес представляет определение *IP*-адреса компьютера, с которого производится запуск программы.

```
def openSecondWindow(self):

    valid = self.validate()
    if not valid:
        palette = self.textEdit.palette()
        palette.setColor(QPalette.ColorRole.Base, QColor('red'))
        self.textEdit.setPalette(palette)
        self.textEdit.clear()
        self.textEdit.setPlaceholderText('некорректный ввод')
        return

    self.stack.setCurrentIndex(1)
```

Рис 11.

Метод переключения страниц окрашивания текстового поля в красный цвет при некорректном вводе. Здесь используется стек окон.

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.stack = QStackedWidget()
        self.setCentralWidget(self.stack)
        self.theIP = myIP('')
        self.initUI()

    def initUI(self):
        self.setFixedSize(QSize(400, 300))
        page_one = PageOne(self.stack, self)
        page_two = PageTwo(self.stack, self, page_one)
        self.stack.addWidget(page_one)
        self.stack.addWidget(page_two)
        self.setWindowTitle('определяем город по IP')
        self.show()

if __name__ == '__main__':
    app = QApplication([])
    mainWindow = MainWindow()
    app.exec()
```

Рис 12.

Класс главного окна, который агрегирует остальные окна, и точка входа в программу.

Заключение

В ходе работы было создано приложение, использующее различные инструменты обработки данных, в данном случае, *IP*-адресов.

Получены новые знания в области сетей, библиотек языка *Python, API* и возможностей сторонних ресурсов. Были успешно использованы регулярные выражения.

Выявлено, что сервис определения местоположения *DaData* не всегда может определить нахождение клиента по *IP* – адресу. Это может быть с тем, что адрес никем не используется или же является зарезервированным.

В целом, работа проведена успешно, т.к. продукт исправен и может быть использован.

.

Источники

1. API: город по IP-адресу (dadata.ru)
2. PyQt6 — полное руководство для новичков / Хабр (habr.com)
3. Installation - pip documentation v24.2.dev0 (pypa.io)
4. PyQt4 — Управление расположением виджетов / Хабр (habr.com)
5. PyQt6 Tutorial - Create TextBox in PyQt6 - Codeloop
6. ТОП-8 видов эмоционального насилия (b17.ru)
7. Find IP address and location using python | by Allwin Raju | Medium
8. Использование диаграммы классов UML при проектировании и документировании программного обеспечения / Хабр (habr.com)