ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ

ДЕПАРТАМЕНТ ИНФОРМАТИКИ

Ciobanu Stanislav

Отчет

по индивидуальной работе по дисциплине

„ПРОГРАММИРОВАНИЕ В PYTHON”

Руководители: Плешка Наталья (Лектор), Felicia Schiopu (Лаборант)

Автор: Ciobanu Stanislav

Кишинев, 2024

GitHub: https://github.com/Rengeka/PythonIndividualWork.git

Содержание:

(1) Задача:

Создать приложение - чат, в котором различные пользователи могли бы обмениваться сообщениями.

Реализовать взаимодействие клиент-сервер при помощи web-сокетов.

Создать базу данных, в которой будут храниться данные пользователей, чаты и сообщения.

(2) Краткое описание принципа работы:

Для реализации данного проекта потребуется создать два проекта. Один будет отвечать за работу сервера, второй за работу клиента. Для обеспечения безопасности, клиент будет посылать запросы на сервер, где они будут обрабатываться и, если это необходимо, будут отправляться

ответы с необходимыми данными. Таким образом доступ к базе данных будет только у сервера.

Для простоты, база данных будет реализована в виде json-файла.

(3) Реализация сервера:

Создадим json – базу данных со значениями – “users” и “chats”.

Каждый пользователь обладает значениями “ID”, “username”, “chats”, “password”, а каждый чат значениями “ID”, “users”, “messages”, где каждое из сообщений это пара значений “user”, “text”.

Устанавливаем библиотеки flask и flask_socketio и импортируем их в проект.

```
from flask import Flask, request
from flask_socketio import SocketIO, emit
import json
```

Создаём и запускаем сервер.

```
app = Flask(__name__)
app.config['SECRET_KEY'] = '1234567890'
socketio = SocketIO(app)
```

```python
if __name__ == '__main__':
    socketio.run(app, host='0.0.0.0', port=5000, allow_unsafe_werkzeug=True)
```

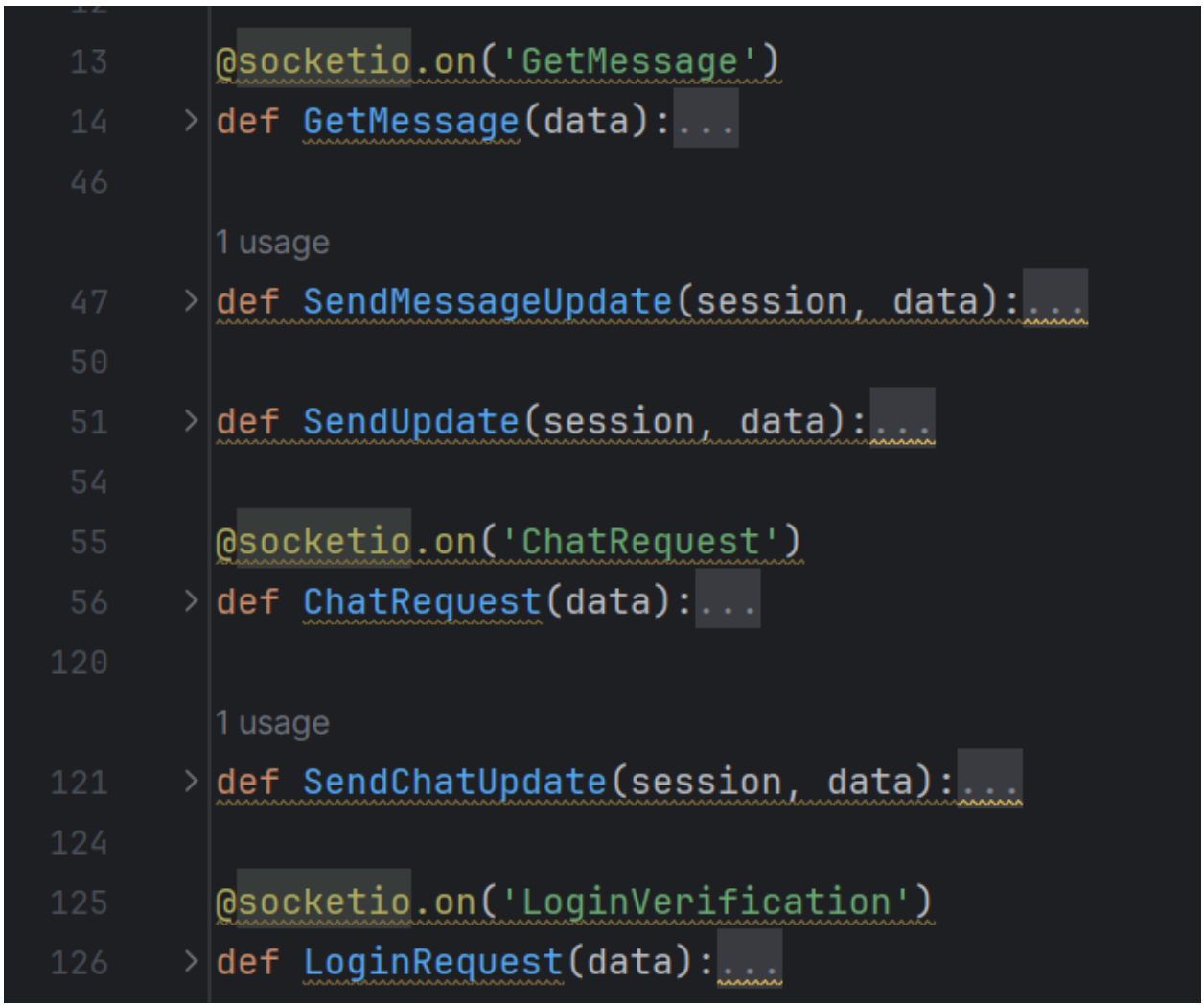
Создаём методы для получения данных из базы данных.

```python
def LoadUsers(filename):
    with open(filename, 'r') as file:
        data = json.load(file)
    return data['users']


1 usage
def LoadChats(filename):...


2 usages
def LoadDatabase(filename):...
```

Создаём глобальный список для отслеживания активных сессий.

```python
activeSessions = []
```

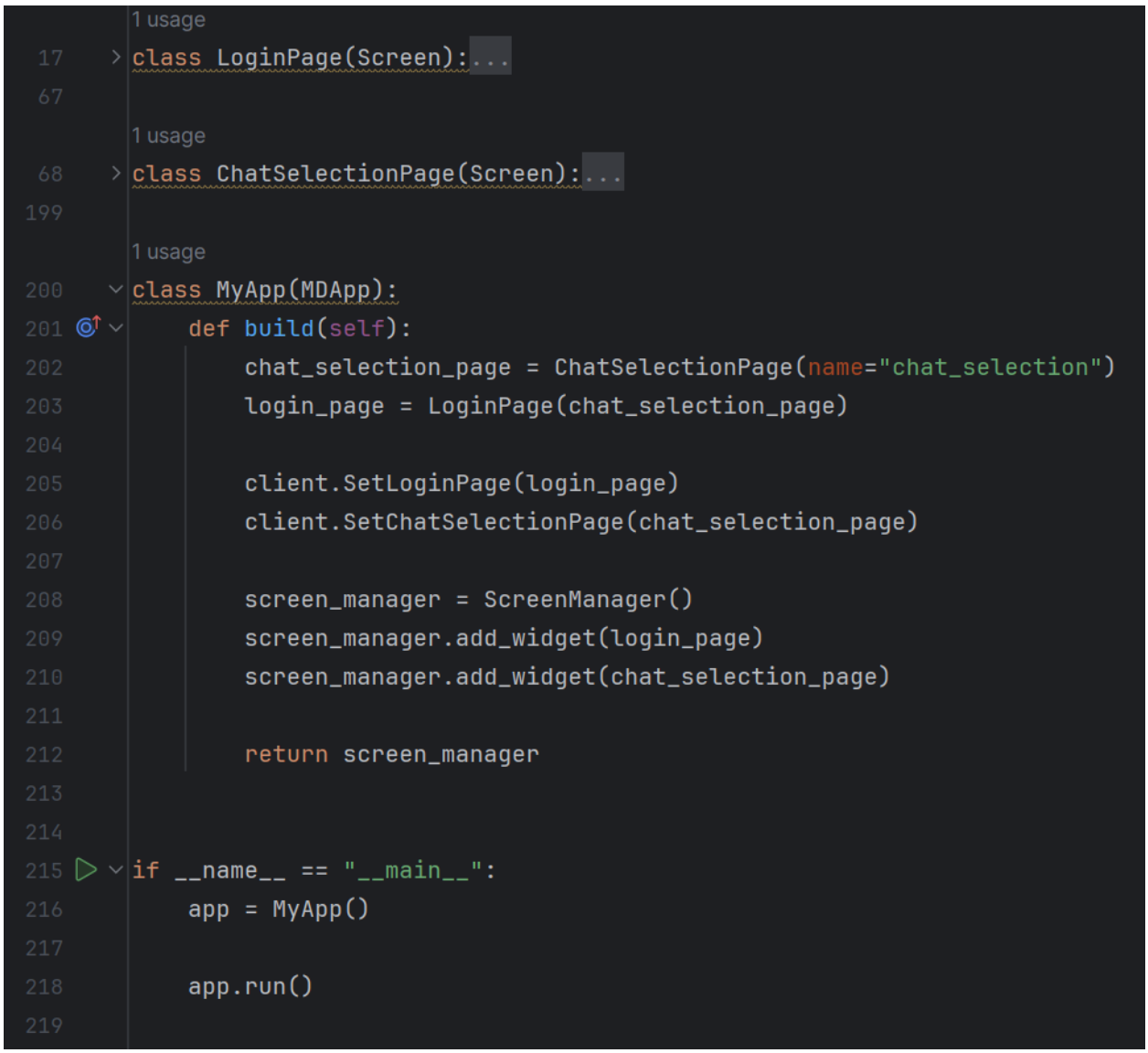
Создаём функционал для принятия и отправки запросов клиенту.



(4) Реализация сервера:

Разделим клиент на два модуля. В одном будем хранить всё взаимодействие с UI, а в другом (Client) непосредственно методы для реализации сетевого соединения.

Для UI клиента воспользуемся библиотекой для создания desktop-приложений kivy.

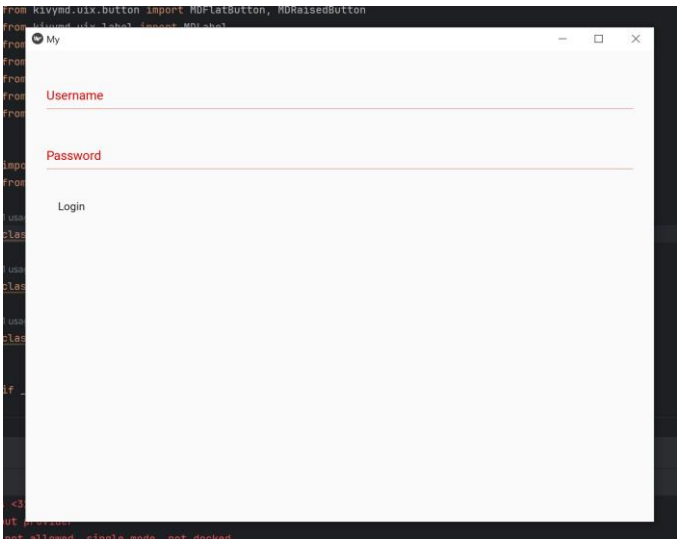


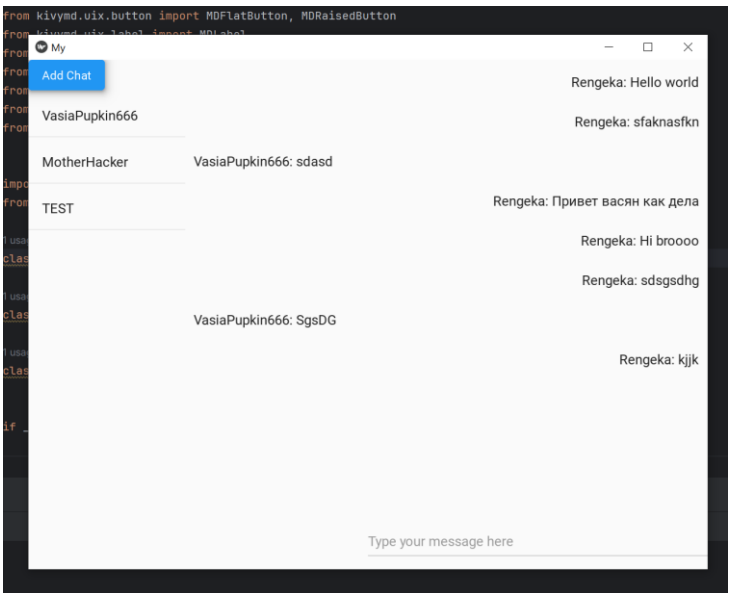
Создадим классы LoginPage и ChatSelectionPage. В этих классах при помощи библиотеки реализуем необходимый UI, который при взаимодействии с пользователем, будет обращаться к модулю Client и взаимодействовать с сервером.

Создадим необходимые методы, которые будут отправлять запросы на сервер и, получив данные, подгружать в UI.

```python
import socketio
from kivy.clock import Clock

sio = socketio.Client()

2 usages
def SetLoginPage(value):...
2 usages
def SetChatSelectionPage(value):...

@sio.event
def TEST(data):
    print(data)

"""..."""

@sio.event
def GetChats(data):...

"""..."""

@sio.event
def GetUpdate(data):...
2 usages
def StartChat(user, login):...
2 usages
def LoginRequest(login, password):
    print("LOGIN REQUEST")
    sio.emit( event: 'LoginVerification',  data: {'login': login, 'password': password, 'sessionID': sio.sid})
@sio.event
def VerifyLogin(response):
    print(response)
    loginPage.check_login_response_wrapper(response)
2 usages
def SendMessage(ID, text, user):...
@sio.event()
def GetMessages(data):...


sio.connect('http://127.0.0.1:5000')
```

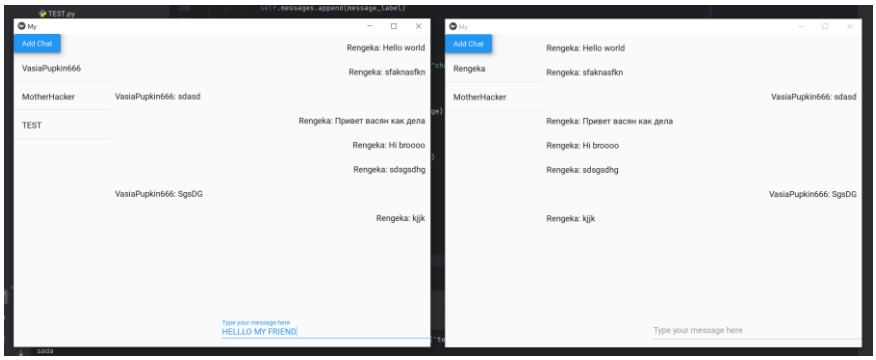
Результат работы клиента:

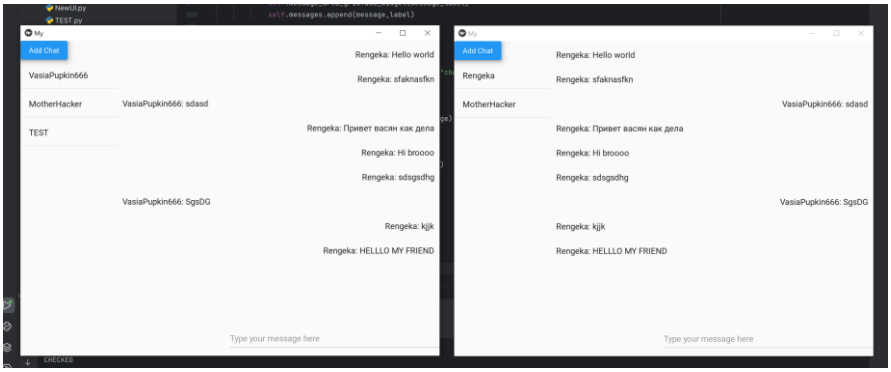




(5) Конечный результат:

Конечная реализация проекта позволяет пользователям в реальном времени обмениваться сообщениями. Благодаря тому, что сервер хранит у себя адреса активных сессий (клиентов), он может посылать обновления всем необходимым пользователям (Даже если один пользователь авторизирован в двух клиентах).





(6) Дальнейшее развитие проекта:

По причине неопытности и первоначального незнания технологий, которые тут используются, в базовой проектировке проекта было допущено несколько неприятных ошибок, что мешают дальнейшей комфортной работе над проектом. Поэтому в качестве первого шага по развитию проекта, я бы провёл рефакторинг кода, удаление ненужных частей и более грамотно продумал бы дальнейшую архитектуру.

Далее я бы реализовал функционал по созданию новых чатов и пользователей т.к в текущем виде это возможно только при помощи прямого редактирования базы данных.

Так же было бы интересно реализовать возможность отправки аудио сообщений и изображений, а так-же некоторые косметические изменения интерфейса (Например аватары пользователей).

(7) Выводы:

Не смотря на ряд трудностей и ошибок, с которыми я столкнулся, я доволен проделанной работой. Я изучил новые технологии и библиотеки (kivy, flask, web-сокеты), создал функциональный проект, гораздо лучше познакомился с работой с json-файлами и словарями и научился обеспечивать двустороннее взаимодействие сервера с клиентами.