ПРИЛОЖЕНИЕ ДЛЯ ОТСЛЕЖИВАНИЯ ПРИСУТСТВИЯ НА ДИСТАНЦИОННОМ УРОКЕ

Подготовили ученики ГБОУ школы №1568
Винокуров Кирилл и Серёжин Илья
Под руководством
Головина Александра Дмитриевича

Актуальность

У онлайн-занятий главный минус – невозможность учителю получить точную и достоверную информацию о присутствии конкретного ученика на уроке.

Решение проблемы

Для решения этой проблемы мы решили создать приложение, которое сможет давать информацию учителю о том или ином ученике на уроке. Оно должно состоять из двух частей : сервер(учитель) и клиент(ученик).

Задачи проекта

Для успешного создания нашего проекта требуется обозначить задачи:

- *Отладить соединение двух компьютеров для дальнейшей работы.*
- *Реализовать функционал приложения-мессенджера, передающего сообщения от ученика к учителю.*
- *Создать простой, интуитивно-понятный для всех возрастов графический интерфейс приложения.*

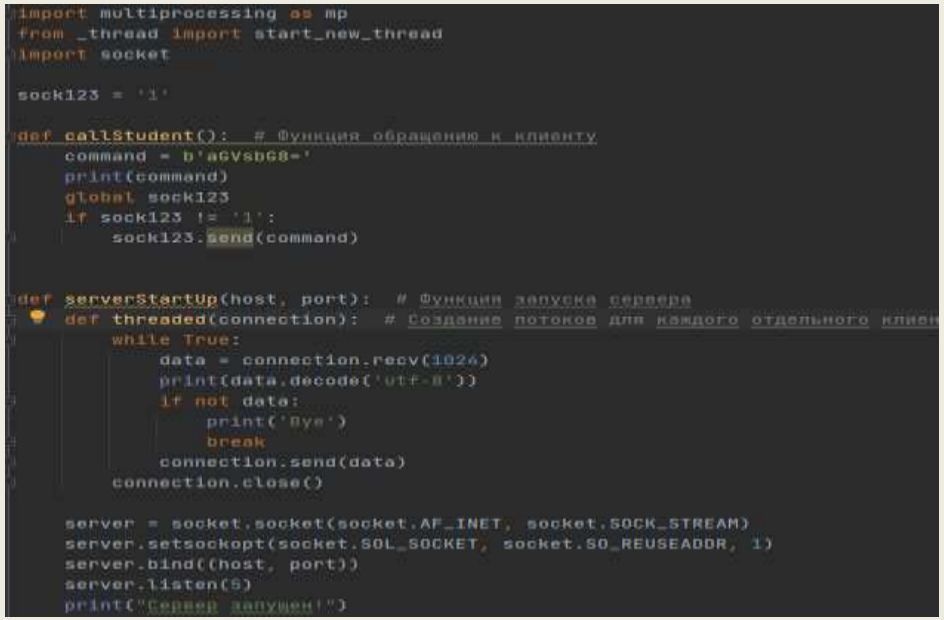
План выполнения работы

Чтобы работа получилась максимально качественной, требуется составить план по распределении времени на каждую задачу. В нашем случае он таков:

- Связь двух компьютеров в локальной сети (ноябрь 2021)
- Создание функциональной части приложения (декабрь 2021)
- Создание графического интерфейса приложения (январь 2022)
- Итоговая разработка и связь графического интерфейса с функциональными частями (февраль 2022)

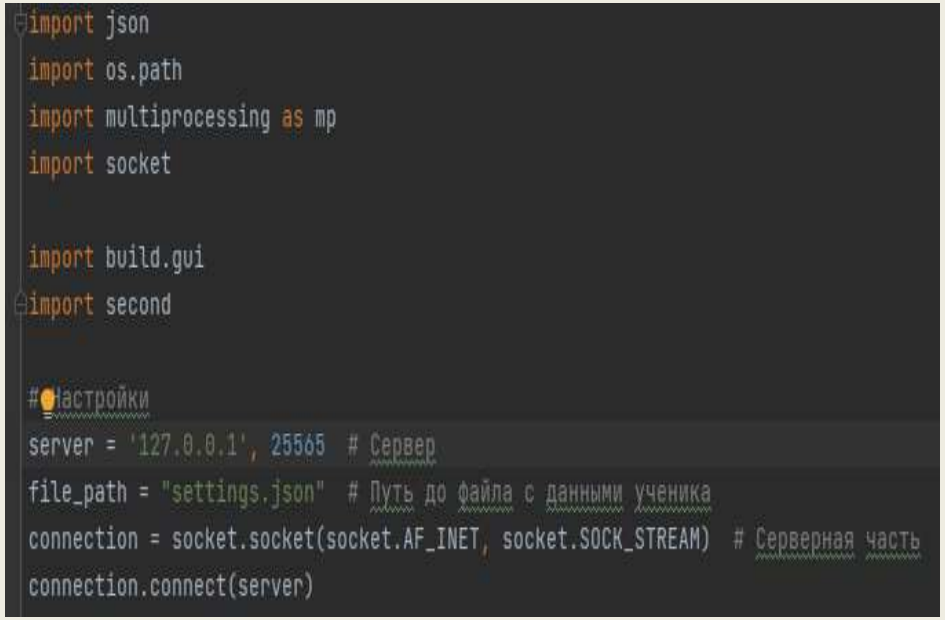
Сервер и Клиент

Для написания кода программы был использован язык программирования Python. Для связи нескольких компьютеров мы воспользовались библиотекой Socket.



```
import multiprocessing as mp
from _thread import start_new_thread
import socket

sock123 = '1'

def callStudent():  # Функция обращению к клиенту
    command = b'aGVsbG8='
    print(command)
    global sock123
    if sock123 != '1':
        sock123.send(command)


def serverStartUp(host, port):  # Функция запуска сервера
    def threaded(connection):  # Создание потоков для каждого отдельного клиен
        while True:
            data = connection.recv(1024)
            print(data.decode('utf-8'))
            if not data:
                print('Bye')
                break
            connection.send(data)
        connection.close()

    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server.bind((host, port))
    server.listen(5)
    print("Сервер запущен!")
```

Сервер



```
import json
import os.path
import multiprocessing as mp
import socket

import build.gui
import second

#Настройки
server = '127.0.0.1', 25565  # Сервер
file_path = "settings.json"  # Путь до файла с данными ученика
connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Серверная часть
connection.connect(server)
```

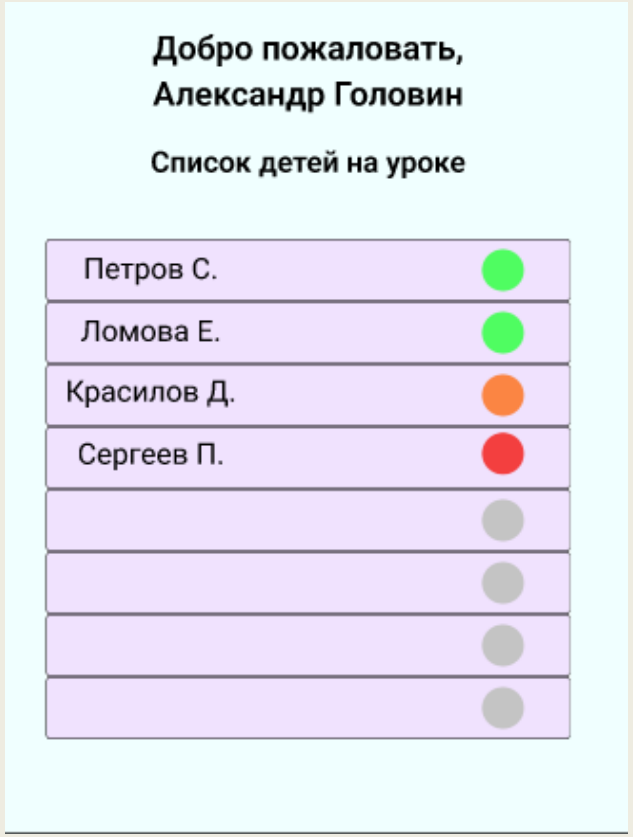
Клиент

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС СПОСОБ СОЗДАНИЯ

Для создания понятного и простого дизайна нашего приложения было решено использовать встроенную библиотеку Python под названием Tkinter.

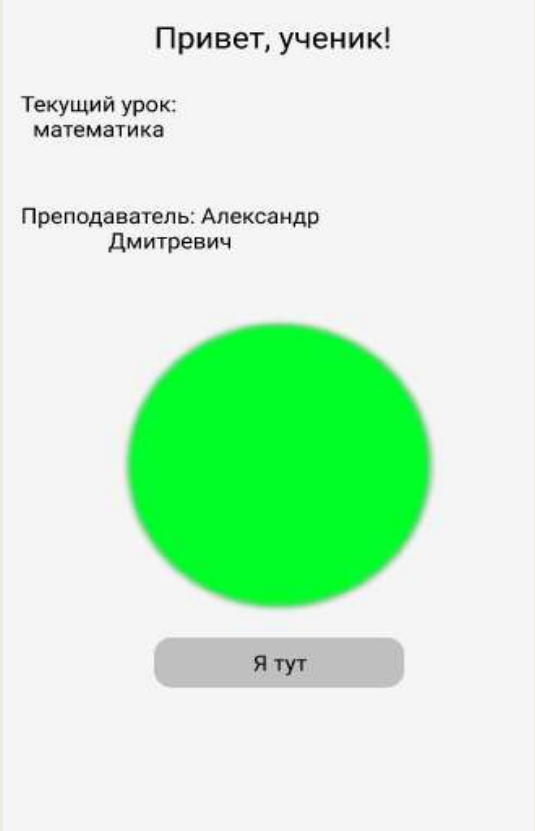
Графический интерфейс
Сервер

```
window = Tk()

window.geometry("200x300")
window.configure(bg = "#F0FFFF")


canvas = Canvas(
    window,
    bg = "#F0FFFF",
    height = 300,
    width = 200,
    bd = 0,
    highlightthickness = 0,
    relief = "ridge"
)

canvas.place(x = 0, y = 0)
canvas.create_text(
    30.0,
    66.0,
    anchor="nw",
    text="Список детей на уроке",
    fill="#000000",
    font=("RobotoRoman SemiBold", 12 * -1)
)

canvas.create_text(
    21.0,
    12.0,
    anchor="nw",
    text="Добро пожаловать Александр Дмитриевич",
    fill="#000000",
    font=("RobotoRoman SemiBold", 12 * -1)
)
```



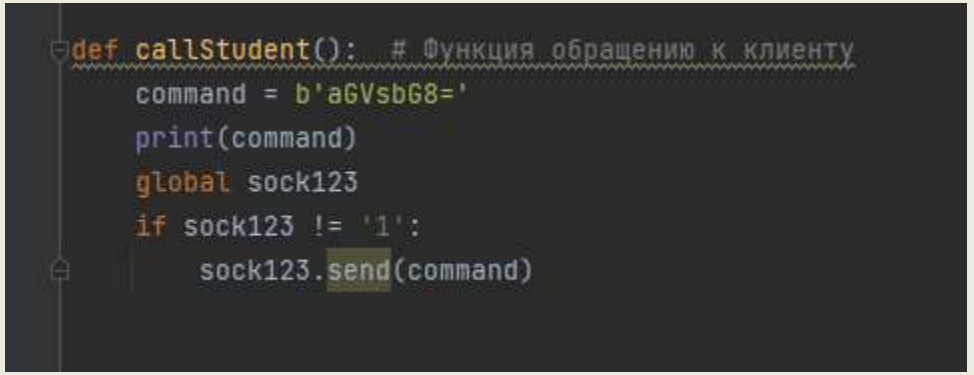
Графический интерфейс
Клиент

```
window = Tk()

window.geometry("300x500")
window.configure(bg="#F5F5F5")

canvas = Canvas(
    window,
    bg="#F5F5F5",
    height=500,
    width=300,
    bd=0,
    highlightthickness=0,
    relief="ridge"
)

canvas.place(x=0, y=0)
canvas.create_text(
    86.0,
    13.0,
    anchor="nw",
    text="Привет, " + studentName + "!",
    fill="#000000",
    font=("Roboto", 18 * -1)
)
```



Функционал приложения

В конечном виде программа получила возможность отмечать ученика, то есть при команде сервера, клиент воспроизводит звук на компьютере ученика и уведомляет его о необходимости ответить учителю. За это отвечает функция callStudent.



Результат работы

Ученик нажимает кнопку

Учитель получает информацию о присутствии конкретного ученика на уроке