Министерство Образования, Культуры и Исследований
Технический Университет Молдовы Департамент Программной Инженерии и Автоматики

ОТЧЕТ

Лабораторная работа №2

По предмету: **Программирование в сети**

Тема: Protocoalele postei electronice.

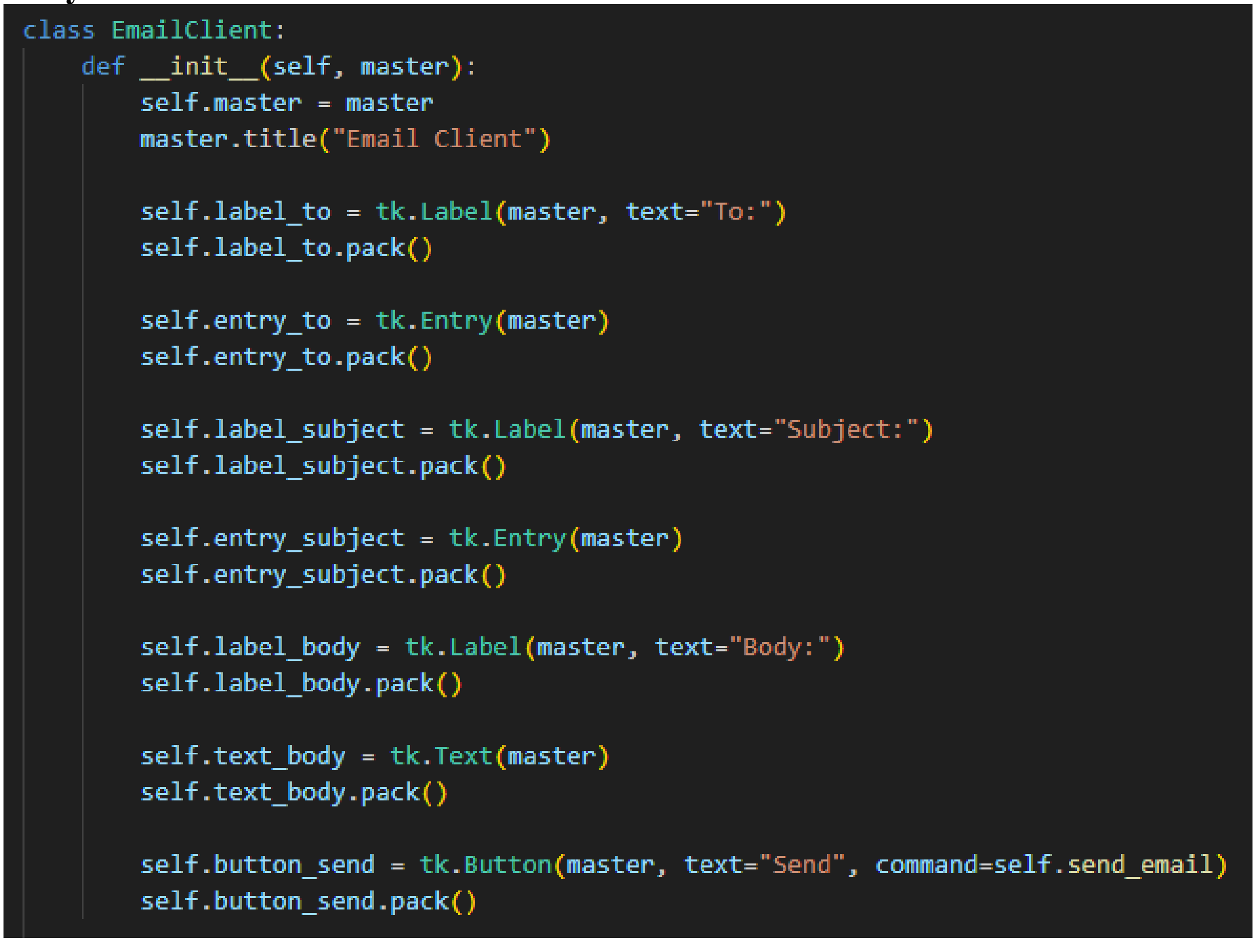
Выполнил студент гр.SI-212: Решетников Максим

Проверил: Лях Аркадий

Кишинев 2024 г.

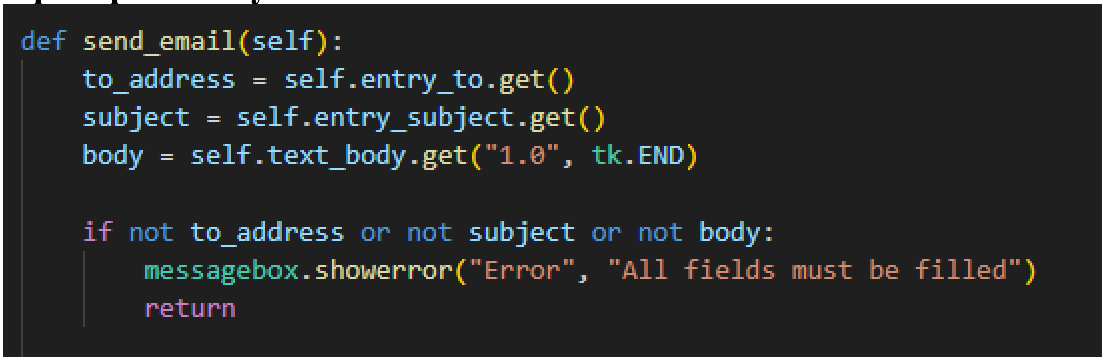
Ссылка на GitHub

Визуальная часть



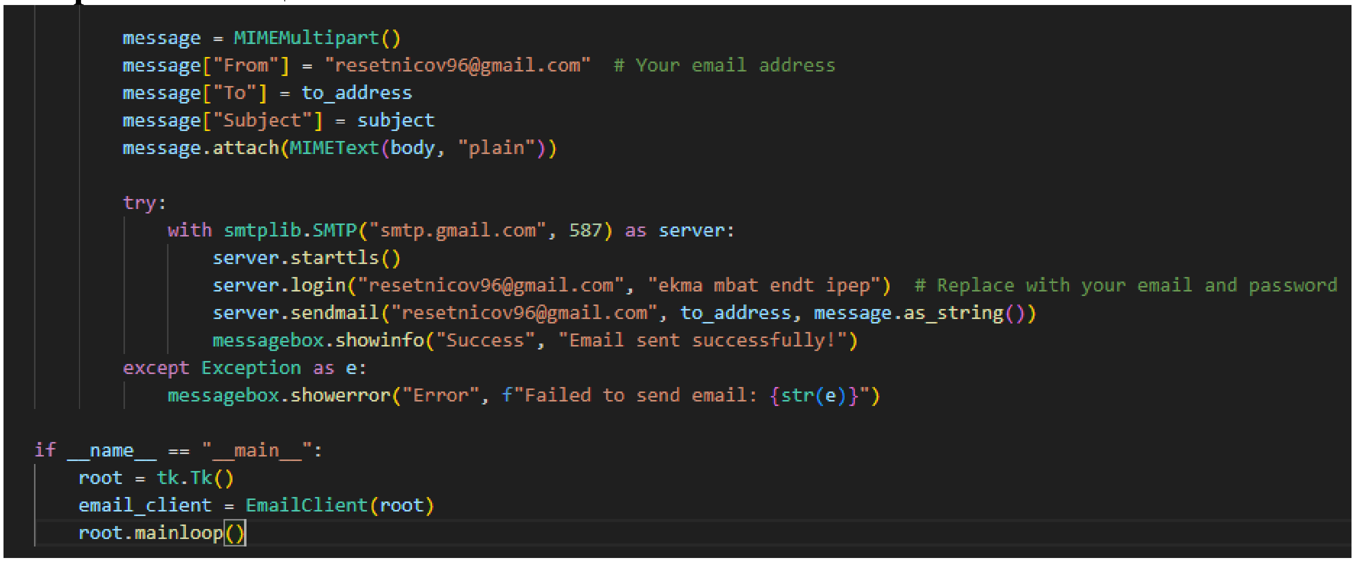
```python
class EmailClient:
    def __init__(self, master):
        self.master = master
        master.title("Email Client")

        self.label_to = tk.Label(master, text="To:")
        self.label_to.pack()

        self.entry_to = tk.Entry(master)
        self.entry_to.pack()

        self.label_subject = tk.Label(master, text="Subject:")
        self.label_subject.pack()

        self.entry_subject = tk.Entry(master)
        self.entry_subject.pack()

        self.label_body = tk.Label(master, text="Body:")
        self.label_body.pack()

        self.text_body = tk.Text(master)
        self.text_body.pack()

        self.button_send = tk.Button(master, text="Send", command=self.send_email)
        self.button_send.pack()
```

Проверка на пустые поля



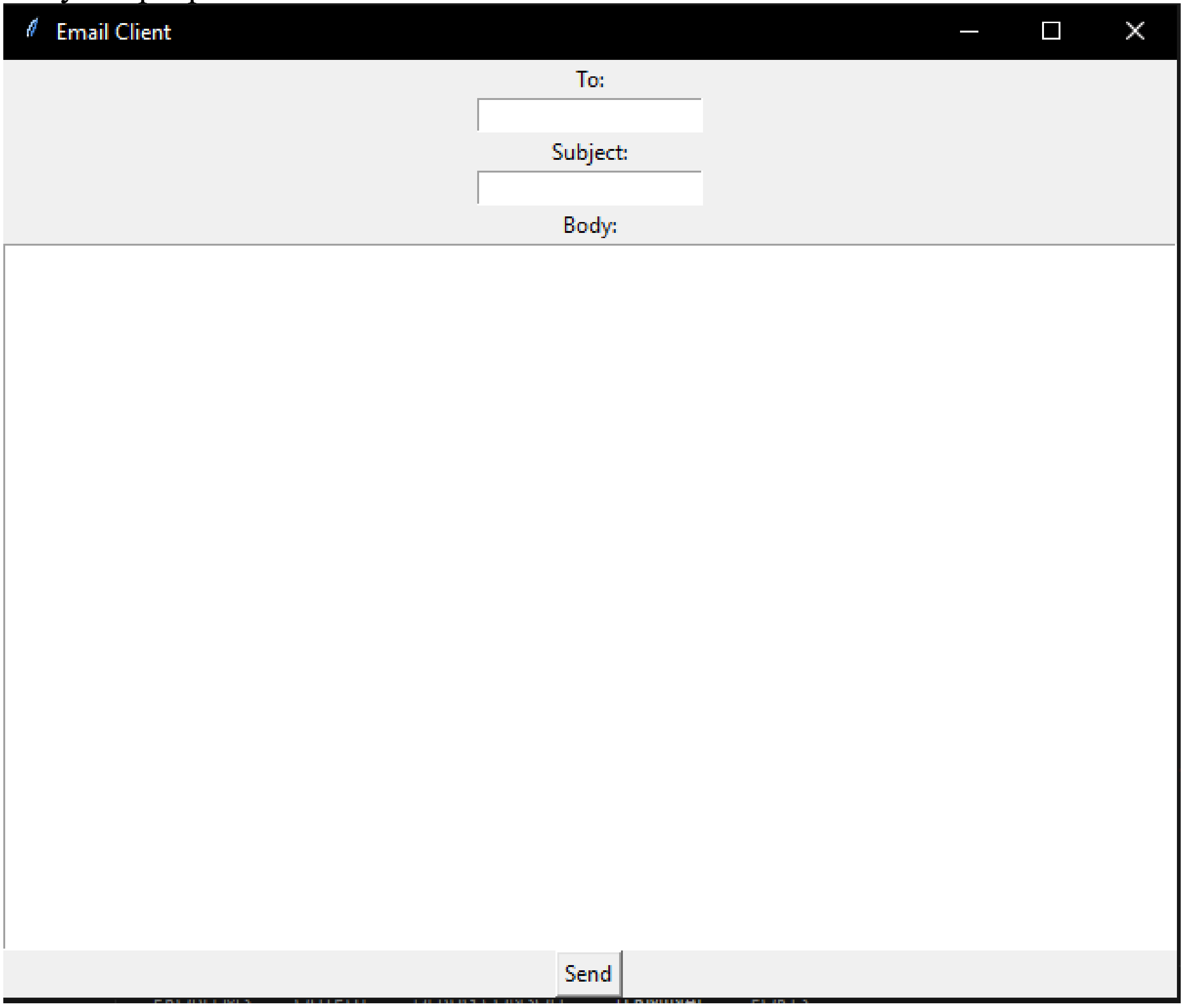
```python
def send_email(self):
    to_address = self.entry_to.get()
    subject = self.entry_subject.get()
    body = self.text_body.get("1.0", tk.END)

    if not to_address or not subject or not body:
        messagebox.showerror("Error", "All fields must be filled")
        return
```

Отправка сообщения



```
        message = MIMEMultipart()
        message["From"] = "resetnicov96@gmail.com"  # Your email address
        message["To"] = to_address
        message["Subject"] = subject
        message.attach(MIMEText(body, "plain"))

        try:
            with smtplib.SMTP("smtp.gmail.com", 587) as server:
                server.starttls()
                server.login("resetnicov96@gmail.com", "ekma mbat endt ipep")  # Replace with your email and password
                server.sendmail("resetnicov96@gmail.com", to_address, message.as_string())
                messagebox.showinfo("Success", "Email sent successfully!")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to send email: {str(e)}")

if __name__ == "__main__":
    root = tk.Tk()
    email_client = EmailClient(root)
    root.mainloop()
```

Визуал программы



Прочтение сообщения

```
    def read_email_thread(self):
        try:
            mail = imaplib.IMAP4_SSL("imap.gmail.com", 993)
            mail.login("resetnicov96@gmail.com", "ekma mbat endt ipep")  # 
Replace with your email and password
            mail.select("inbox")

            result, data = mail.search(None, "ALL")
            message_ids = data[0].split()

            if not message_ids:
                self.message_display.delete(1.0, tk.END)
                self.message_display.insert(tk.END, "No emails in the inbox.")
                return

            if self.current_message_index >= len(message_ids):
                self.current_message_index = 0

            message_id = message_ids[self.current_message_index]
            _, msg_data = mail.fetch(message_id, "(RFC822)")
            raw_email = msg_data[0][1]
            msg = email.message_from_bytes(raw_email)

            subject, encoding = email.header.decode_header(msg["Subject"])[0]
            if isinstance(subject, bytes):
                subject = subject.decode(encoding or "utf-8")

            from_, encoding = email.header.decode_header(msg["From"])[0]
            if isinstance(from_, bytes):
                from_ = from_.decode(encoding or "utf-8")

            body = ""
            if msg.is_multipart():
                for part in msg.walk():
                    if part.get_content_type() == "text/plain":
                        body = part.get_payload(decode=True)
                        break
            else:
                body = msg.get_payload(decode=True)

            if isinstance(body, bytes):
                body = body.decode(encoding or "utf-8")

            self.message_display.delete(1.0, tk.END)
            self.message_display.insert(tk.END, f"Subject: {subject}\n")
            self.message_display.insert(tk.END, f"From: {from_}\n\n")
            self.message_display.insert(tk.END, f"{body}\n\n")
            mail.close()
            mail.logout()
        except Exception as e:
            self.message_display.delete(1.0, tk.END)
            self.message_display.insert(tk.END, f"Failed to read emails:
{str(e)}\n")

    def show_next_message(self):
        self.current_message_index += 1
        self.read_email()

    def show_previous_message(self):
        if self.current_message_index > 0:
            self.current_message_index -= 1
            self.read_email()
```

Листинг

```python
import smtplib
import imaplib
import email
import tkinter as tk
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import threading

class EmailClient:
    def __init__(self, master):
        self.master = master
        master.title("Email Client")

        self.label_to = tk.Label(master, text="To:")
        self.label_to.pack()

        self.entry_to = tk.Entry(master)
        self.entry_to.pack()

        self.label_subject = tk.Label(master, text="Subject:")
        self.label_subject.pack()

        self.entry_subject = tk.Entry(master)
        self.entry_subject.pack()

        self.label_body = tk.Label(master, text="Body:")
        self.label_body.pack()

        self.text_body = tk.Text(master)
        self.text_body.pack()

        self.button_send = tk.Button(master, text="Send",
command=self.send_email)
        self.button_send.pack()

        self.button_read = tk.Button(master, text="Read",
command=self.read_email)
        self.button_read.pack()

        self.message_display = tk.Text(master, height=10, width=50)
        self.message_display.pack()

        self.current_message_index = 0

        self.button_next = tk.Button(master, text="Next",
command=self.show_next_message)
        self.button_next.pack()
        self.button_previous = tk.Button(master, text="Previous",
command=self.show_previous_message)
        self.button_previous.pack()

    def send_email(self):
        to_address = self.entry_to.get()
        subject = self.entry_subject.get()
        body = self.text_body.get("1.0", tk.END)
        if not to_address or not subject or not body:
            messagebox.showerror("Error", "All fields must be filled")
            return

        message = MIMEMultipart()
        message["From"] = "resetnicov96@gmail.com"  # Your email address
        message["To"] = to_address
        message["Subject"] = subject
        message.attach(MIMEText(body, "plain"))

        try:
            with smtplib.SMTP("smtp.gmail.com", 587) as server:
                server.starttls()
                server.login("resetnicov96@gmail.com", "ekma mbat endt ipep")  #
Replace with your email and password
                server.sendmail("resetnicov96@gmail.com", to_address,
message.as_string())
                messagebox.showinfo("Success", "Email sent successfully!")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to send email: {str(e)}")

    def read_email(self):
        threading.Thread(target=self.read_email_thread).start()

    def read_email_thread(self):
        try:
            mail = imaplib.IMAP4_SSL("imap.gmail.com", 993)
            mail.login("resetnicov96@gmail.com", "ekma mbat endt ipep")  #
Replace with your email and password
            mail.select("inbox")

            result, data = mail.search(None, "ALL")
            message_ids = data[0].split()

            if not message_ids:
                self.message_display.delete(1.0, tk.END)
                self.message_display.insert(tk.END, "No emails in the inbox.")
                return

            if self.current_message_index >= len(message_ids):
                self.current_message_index = 0

            message_id = message_ids[self.current_message_index]
```

```python
            _, msg_data = mail.fetch(message_id, "(RFC822)")
            raw_email = msg_data[0][1]
            msg = email.message_from_bytes(raw_email)

            subject, encoding = email.header.decode_header(msg["Subject"])[0]
            if isinstance(subject, bytes):
                subject = subject.decode(encoding or "utf-8")

            from_, encoding = email.header.decode_header(msg["From"])[0]
            if isinstance(from_, bytes):
                from_ = from_.decode(encoding or "utf-8")

            body = ""
            if msg.is_multipart():
                for part in msg.walk():
                    if part.get_content_type() == "text/plain":
                        body = part.get_payload(decode=True)
                        break
            else:
                body = msg.get_payload(decode=True)

            if isinstance(body, bytes):
                body = body.decode(encoding or "utf-8")

            self.message_display.delete(1.0, tk.END)
            self.message_display.insert(tk.END, f"Subject: {subject}\n")
            self.message_display.insert(tk.END, f"From: {from_}\n\n")
            self.message_display.insert(tk.END, f"{body}\n\n")

            mail.close()
            mail.logout()
        except Exception as e:
            self.message_display.delete(1.0, tk.END)
            self.message_display.insert(tk.END, f"Failed to read emails:
{str(e)}\n")

    def show_next_message(self):
        self.current_message_index += 1
        self.read_email()

    def show_previous_message(self):
        if self.current_message_index > 0:
            self.current_message_index -= 1
            self.read_email()

if __name__ == "__main__":
    root = tk.Tk()
    email_client = EmailClient(root)
    root.mainloop()
```

Вывод:

В ходе выполнения работы мы научились работать с протоколами SMTP, POP3, IMAP.

Была написана программа, выполняющая все поставленные условия