Dokumentacja projektu

Notatnik z możliwością wysyłania maila

Zespół 1, Lab 7

Mateusz Sabat
Igor Siata
Jan Skarboń

Link do repozytorium na GitHub - https://github.com/JanRobertS/NPG_Projekt.git

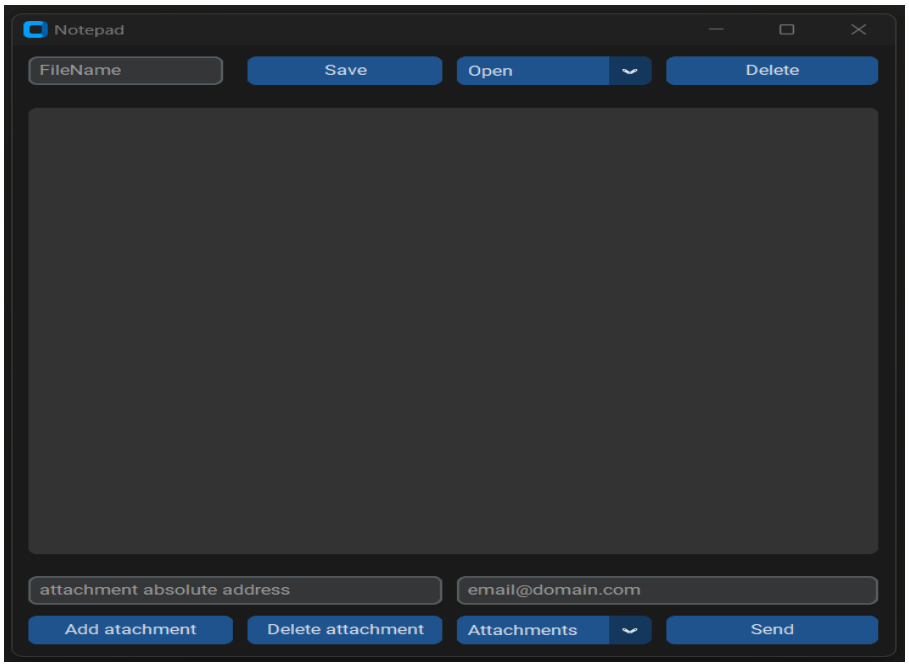
Głównym tematem projektu było zaprojektowanie i zaprogramowanie programu który jest notatnikiem z możliwością wysyłania maili. Dodatkowym zadaniem było wymyślenie i zaprogramowanie dodatkowej funkcjonalności programu. Program jest przystosowany pod różnorodnego odbiorcę, dlatego zastosowaliśmy anglojęzyczny interfejs.

Program posiada wymienione funkcjonalności:

1. Dodanie notatki z nazwą i opisem,
2. Wypisanie n notatek,
3. Zapis i wczytywanie notatek przy zamknięciu/otwarciu programu,
4. Wysłanie wybranych notatek mailem do podanego adresata,
5. Usuwanie i edytowanie notatek,
6. Dodawanie załączników do maili.

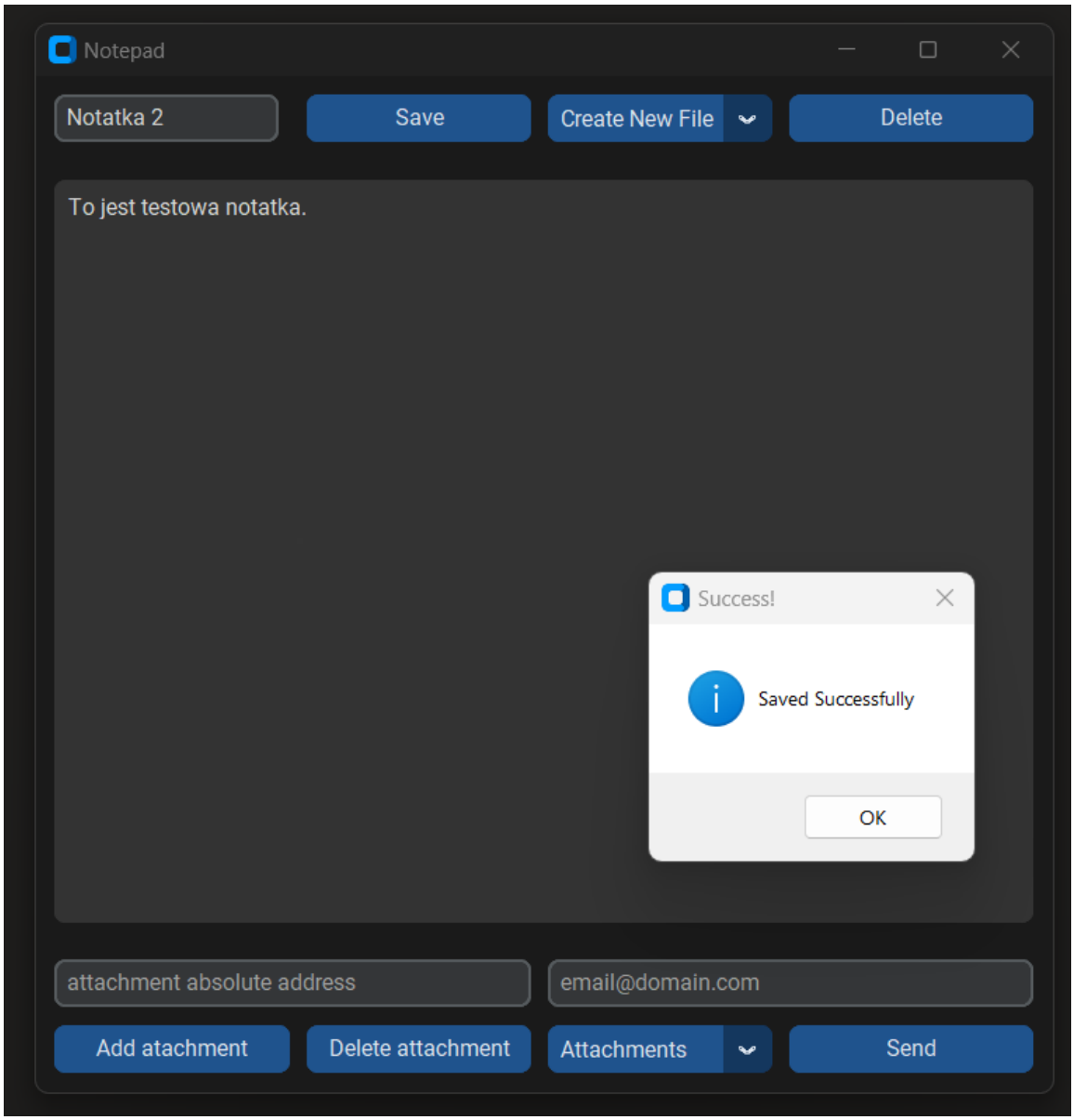
Działanie programu:

Po uruchomieniu programu uruchamia się okno Notepad, mamy możliwość zminimalizowania powiększenia i zamknięcia programu.

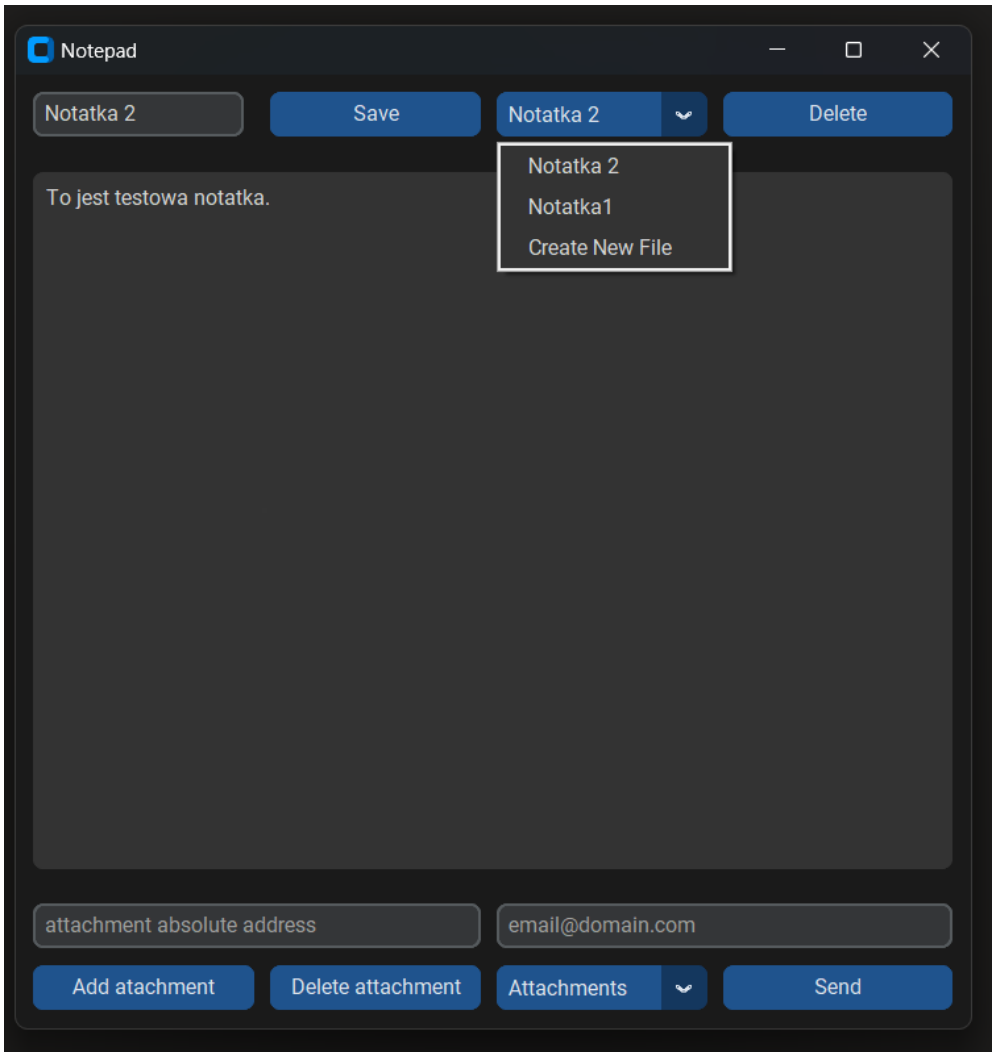


Miejsce z napisem ”FileName” służy do podawania nazwy notatki.

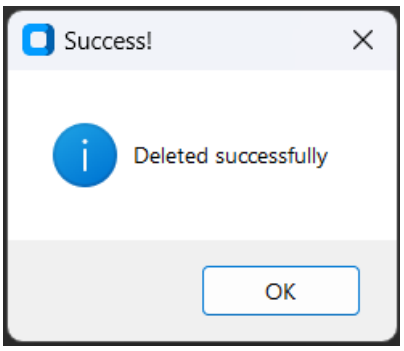
Opcja ”Save” zapisuje naszą notatkę do folderu ”notes”, jeżeli takowy nie istnieje, zostanie on dodany automatycznie podczas zapisywania notatki. W folderze ”notes” są zapisywane wszystkie notatki które zapiszemy. Po zapisaniu notatki pojawi nam się okno z napisem ”Success! Save successfully.”, po naciśnięciu ”OK”, program wraca do naszego notatnika.



Opcja ”Open” służy do otwierania zapisanych plików. Po kliknięciu w opcje ”Open” pokazują nam się wszystkie notatki które mamy zapisane, po kliknięciu w daną notatkę, wyświetli się ona w naszym programie. Jeżeli nie mamy żadnej notatki zapisanej, pojawi nam się tylko jedna opcja ”Create New File”, jeżeli klikniemy w nią będzie to pusty szablon do stworzenia nowej notatki.



Opcja ”Delete” służy do usuwania pliku w którym się znajdujemy. Po kliknięciu w przycisk ”Delete” pojawi nam się okno z komunikatem ”Success! Deleted successfully.”, po naciśnięciu ”OK” powrócimy do programu notesu.



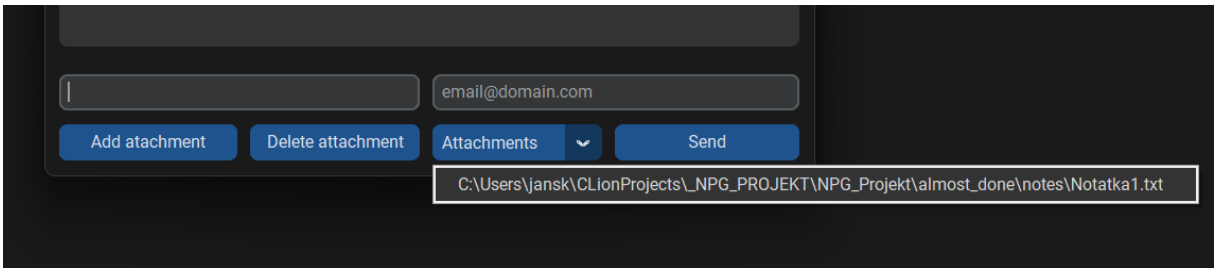
Miejsce na samym środku służy do tworzenia treści notatki. Po kliknięciu kursorem na to miejsce możemy zacząć pisać treść naszej notatki.

Miejsce z napisem ”attachment absolute address” służy do podania adresu pliku który chcemy załączyć do maila. Możemy w to miejsce napisać adres naszego pliku, zostanie on dodany i podczas wysyłania maila wysłany.

Opcja ”Add atchment” służy do dodania załącznika, którego adres podaliśmy w miejscu ”attachment absolute address”.

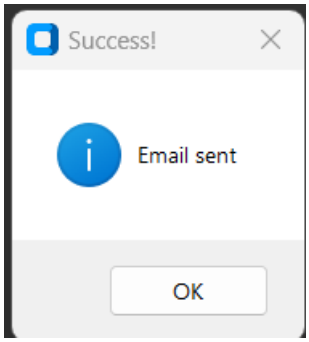
Opcja ”Delete attchment” służy do usuwania konkretnego załącznika.

Opcja ”Attachments” służy do sprawdzania jakie załączniki są dołączone do naszego maila z notatką.

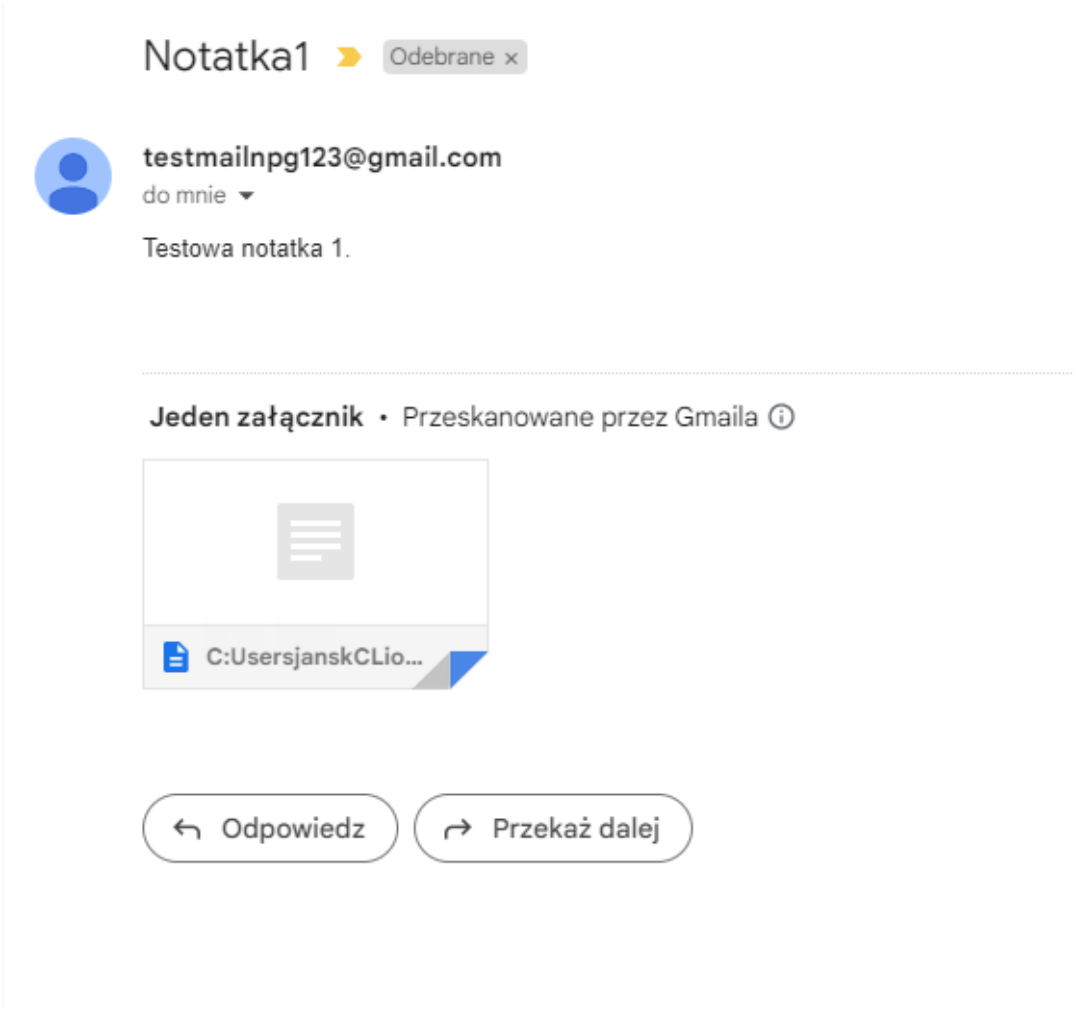


Miejsce z napisem ” email@domain.com” służy do podania maila na który zostanie przesłana notatka z załącznikiem.

Opcja ”Send” służy do wysyłania wybranej notatki z załącznikiem na maila podanego wcześniej. Po udanym wysłaniu notatki dostajemy komunikat ”Success! Email sent”.



Notatka wysłana przychodzi na maila którego wpisaliśmy, jako Temat jest wysyłana nazwa notatki, jako treść treść notatki i dodatkowo załączniki które dodaliśmy. Notatka wysyłana jest z maila testmailnpg123@gmail.com, który jest specjalnie dostosowany by oprogramowanie python mogło wysyłać maile.



Używane narzędzia:

Do zaprogramowania używaliśmy PyCharm2022.3.3.

Do komunikacji między zespołem używaliśmy grupy stworzonej na Aplikacji firmy Facebook „Messenger”.

Do kontroli wersji programu używaliśmy aplikacji Git.

Do wyznaczanie zadań i kontrolowania postępów używaliśmy aplikacji GitHub.

Biblioteki:

Do wysyłania emaili użyliśmy bibliotek:

encoders, estmplib, MIMEBase, MIMEMultipart, MIMEText.

Do interfejsu graficznego przysłużyły się biblioteki:

customtkintera, tkintera.

Do obsługi plików posłużyła biblioteka:

os.

Jan Robert Skarboń:

Byłem odpowiedzialny za stworzenie repozytorium na GitHub-ie, stworzyłem zadania które musieliśmy zrobić i po rozmowie z wszystkimi członkami zespołu rozdzieliłem zadanie które poszczególna osoba musi zrobić. Moim pierwszym zadanie było stworzenie notatnika z możliwością pisania notatek z nazwą i opisem. Na początku napisałem program który działał w konsoli aplikacji PyCharm. Później program został zmieniony poprzez dodanie interfejsu graficznego i dopasowaniu programu do wysyłania emaili. Zainicjalizowałem także repozytorium git i dodałem do niego początkowy plik main.py na którym później wszyscy pracowali. Wymyśliłem dodatkową funkcję dodawania stopki do wysyłanego emaila automatycznie, napisanie tej funkcji okazało się dosyć łatwe. Wystarczyła jedna linijka kodu, zapisana w odpowiednim miejscu.

```
# dodanie stopki i notatki
# em.set_content(f"{text}\n\nPozdrawiam,\nJan Robert Skarbon")
```

Ostatecznie uznaliśmy, że wykorzystamy pomysł od Igora, który wymyślił dodawanie załączników do emaila. Byłem też zaangażowany w sprawdzanie i testowanie kodu. Po każdorazowym zmienianiu kodu i dodawaniu funkcji przez kolegów z zespołu, sprawdzałem poprawność i możliwe błędy kodu. Ostatnim moim zadanie było napisanie dokumentacji.

Mateusz Sabat:

W naszym projekcie ”Notatnik z możliwością wysyłania maili” byłem odpowiedzialny za napisanie funkcji dodawania n notatek, które zaprogramowałem na samym początku programu. Na początku był to trochę inny program gdyż był on bez możliwości wysyłania maili i bez interfejsu graficznego, miał on funkcje pisania notatek w konsoli, dodałem funkcje dodawania n notatek. Byłem też odpowiedzialny za funkcje usuwania i edytowania notatek. Napisałem ten

kod:

```
def DeleteFile(self):
    # delete note from folder
    path = os.path.join(self.notes_dir, self.entry_name.get() + ".txt")
    if os.path.exists(path):
        os.remove(path)
    else:
        tkinter.messagebox.showerror(title="Error", message="Wrong file name")
        return
    # delete note from list of notes
    self.open_file.configure(values=SavedNotes(self.notes_dir))
    # clear view
    self.open_file.set("Open")
    self.textbox.delete("0.0", "end")
    self.entry_name.delete("0", "end")
    tkinter.messagebox.showinfo(title="Success!", message="Deleted successfully")
```

Jest to kod odpowiedzialny za usuwanie notatki. Po usunięciu notatki z folderu ”notes”, cały interfejs wraca do stanu startowego aplikacji, czyli musiałem sprawić by został wyczyszczony widok główny notatki. Stworzyłem za pomocą biblioteki customtkintera i tkintera interfejs graficzny. Podczas tworzenia go uczyłem się tego z poradników na platformie YouTube, gdzie można znaleźć dużo przydatnych poradników do programowania. Poniżej zamieszczam screen kodu interfejsu graficznego.

```
    super().__init__()

    # configure window
    self.title("Notepad")
    self.geometry(f"{580}x{580}")

    # configure grid layout (4x3)
    # noinspection PyTypeChecker
    self.grid_columnconfigure((0, 1, 2), weight=1)
    # noinspection PyTypeChecker
    self.grid_rowconfigure((1, 2, 3), weight=1)

    # create textbox
    self.textbox = customtkinter.CTkTextbox(self)
    self.textbox.grid(row=1, column=0, rowspan=3, columnspan=4, padx=(10, 10), pady=(10, 10), sticky="nsew")

    # create entry for file name
    self.entry_name = customtkinter.CTkEntry(self, placeholder_text="FileName")
    self.entry_name.grid(row=0, column=0, columnspan=1, padx=(10, 10), pady=(10, 10), sticky="nsew")

    # create entry for email
    self.entry_email = customtkinter.CTkEntry(self, placeholder_text="email@domain.com")
    self.entry_email.grid(row=4, column=2, columnspan=2, padx=(5, 10), pady=(10, 5), sticky="nsew")
```

Na początku kodu mamy wymiary i konfigurację okna notepadu, później dodajemy miejsce na tekst, miejsce z nazwą pliku i miejsce na wpisanie emaila.

Igor Siata:

Moim zadaniem w projekcie było zaprogramowanie zapisu i wczytywania notatek przy zamknięciu/otwarciu programu. Udało mi się to zaprojektować, dzięki czemu możemy swobodnie zapisywać i usuwać notatki.

```
def OpenFile(self, choice):
    # clears view for new file
    if choice == "Create New File":
        self.textbox.delete("0.0", "end")
        self.entry_name.delete("0", "end")
        self.entry_name.configure(placeholder_text="FileName")
        self.entry_name.insert("0", choice)
        return
    # clear view
    self.textbox.delete("0.0", "end")
    self.entry_name.delete("0", "end")
    # open file
    with open(os.path.join(self.notes_dir, choice + ".txt")) as f:
        note_text = f.read()
    self.textbox.insert("0.0", note_text)
    self.entry_name.insert("0", choice)
```

W tym fragmęcie kodu tworzymy funkcję ”Open” która owiera zapisany plik.

```
def SaveFile(self):
    # error if file name is empty
    if self.entry_name.get() == "":
        tkinter.messagebox.showerror(title="Error", message="Empty file name")
        return
    # get content
    note_name = self.entry_name.get()
    note_text = self.textbox.get("0.0", "end")
    # save note
    note_path = os.path.join(self.notes_dir, note_name + ".txt")  # utwórz ścieżkę do pliku notatki
    with open(note_path, "w") as f:
        f.write(note_text)  # zapisz notatkę do pliku
    self.open_file.configure(values=SavedNotes(self.notes_dir))
    tkinter.messagebox.showinfo(title="Success!", message="Saved Successfully")
```

Tutaj pokazane jest stworzenie funkcji zapisywania notatki. Opisałem tutaj za pomocą komentarzy zaczynających się od znaku ”#”, potrzebne fragmenty kodu, żeby w późniejszym czasie kiedy chcielibyśmy poprawić coś lub dopisać jakiś fragment kodu, żeby orientować się co gdzie się znajduje.

Kolejną funkcją którą dodałem do programu było wysłanie wybranych notatek mailem do podanego adresata. Udało mi się to zrobić dzięki poradnikom na Internecie. Dodatkowo trzeba było skonfigurować emaila.

```python
def send_note(title, text, email_reciver):
    # dane do wysyłania emaila
    email_sender = 'testmailnpg123@gmail.com'
    email_password = 'mquoianwvzurwhyj'

    em = MIMEMultipart()
    em['From'] = email_sender
    em['To'] = email_reciver
    em['Subject'] = title

    em.attach(MIMEText(text, 'plain'))

    # dodanie stopki i notatki
    # em.set_content(f"{text}\n\nPozdrawiam,\nJan Robert Skarbon")

    # adding attachment
    for attachment_path in attachments:
        attachment = open(attachment_path, "rb")
        p = MIMEBase('application', 'octet-stream')
        p.set_payload(attachment.read())
        encoders.encode_base64(p)
        p.add_header('Content-Disposition', "attachment; filename= %s" % attachment_path)
        em.attach(p)
```

```python
# wysyła email
session = smtplib.SMTP('smtp.gmail.com', 587)  # use gmail with port
session.starttls()  # enable security
session.login(email_sender, email_password)  # login with mail_id and password
content = em.as_string()
session.sendmail(email_sender, email_reciver, content)
session.quit()
```

Moim zadaniem było także dodanie dodatkowej funkcji, czyli dodawania załącznika do emaila. Poniżej można zobaczyć screen z częścią programu, który wykonuję tą funkcję.

```python
def AddAtach(self):
    # get content from attachment entry
    attachment_path = self.entry_attachment.get()
    # check if path exists
    if not os.path.exists(attachment_path):
        tkinter.messagebox.showerror(title="Error", message="Wrong path name")
        return
    # append list of attachments
    attachments.append(attachment_path)
    self.entry_attachment.delete('0', 'end')
    # update dropdown menu
    self.check_attach.configure(values=attachments)

siata103
def CheckAttach(self, choice):
    # insert attachment path into entry
    self.entry_attachment.delete('0', 'end')
    self.entry_attachment.insert('0', choice)
    self.check_attach.set("Attachments")
```

W tej części programu dodajemy załącznik i sprawdzamy.

Podsumowanie:

Celem projektu było zaprojektowanie i zaimplementowanie aplikacji, która umożliwia użytkownikom prowadzenie notatek i wysyłanie wiadomości e-mail bezpośrednio z programu. W trakcie pracy nad projektem, zespół zastosował różne biblioteki i narzędzia w języku Python, takie jak Tkinter do budowy interfejsu graficznego użytkownika, oraz moduł smtplib do wysyłania wiadomości e-mail. Dzięki temu projektowi, członkowie grupy mieli możliwość nauczenia się różnych umiejętności, takich jak programowanie w języku Python, współpraca w grupie, używanie GitHub'a i Git'a, organizacja pracy oraz rozwiązywanie problemów technicznych. Ostatecznie, projekt zakończył się powodzeniem, a stworzona aplikacja jest w pełni funkcjonalna.