

Wojciech Matraś WCY21KA1S1 80742

Sprawozdanie PJF nr 2.

Aplikacja do notatek dla osób z ograniczonym widzeniem

Wymagania:

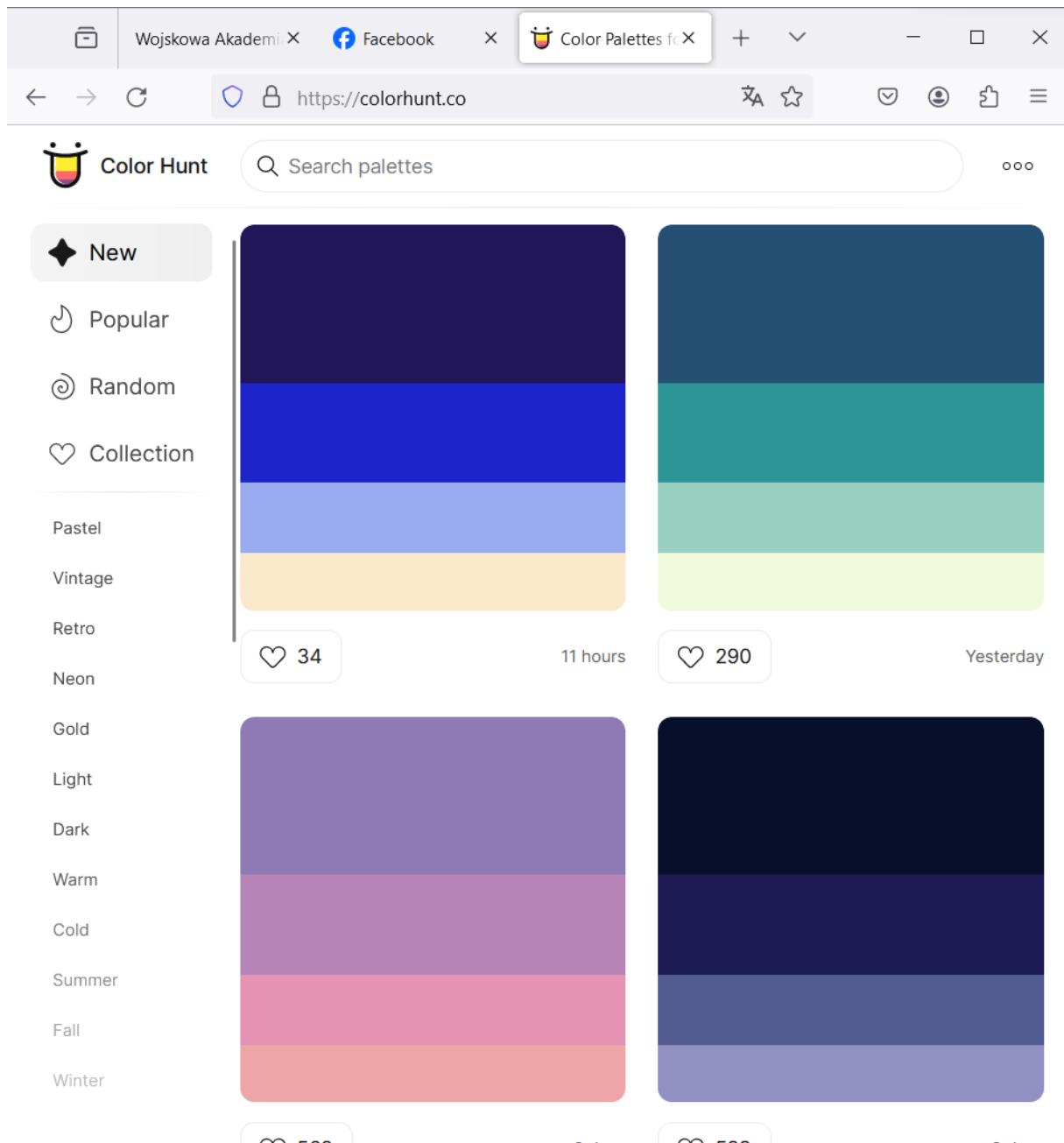
Aplikacja powinna:

- umożliwiać budowę notatki
- segregowanie notatek
- usuwanie notatek

Stworzona aplikacja:

Założenia:

Program będzie miał dużą czcionkę, by osobom niedowidzącym łatwiej było czytać. Interface musi mieć duże kontrasty. By ułatwić używanie programów stworzyłem 4 różne interface'y, wykorzystałem palety barw ze strony: <https://colorhunt.co/>



Ekran startowy:

Notatnik

Domyślny

Tytuł notatki:

Treść notatki:

Dodaj notatkę

Usuń notatkę

Sortuj według:

Tytułu

3 inne kolorystyki:

Notatnik

Ciemny

Tytuł notatki:

Treść notatki:

Dodaj notatkę

Usuń notatkę

Sortuj według:

Tytułu

Notatnik

Biało-czarny

Tytuł notatki:

Treść notatki:

Dodaj notatkę

Usuń notatkę

Sortuj według:

Tytułu

Notatnik

Dla daltonistów

Tytuł notatki:

Treść notatki:

Dodaj notatkę

Usuń notatkę

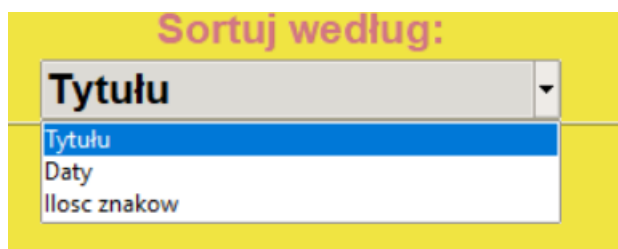
Sortuj według:

Tytułu

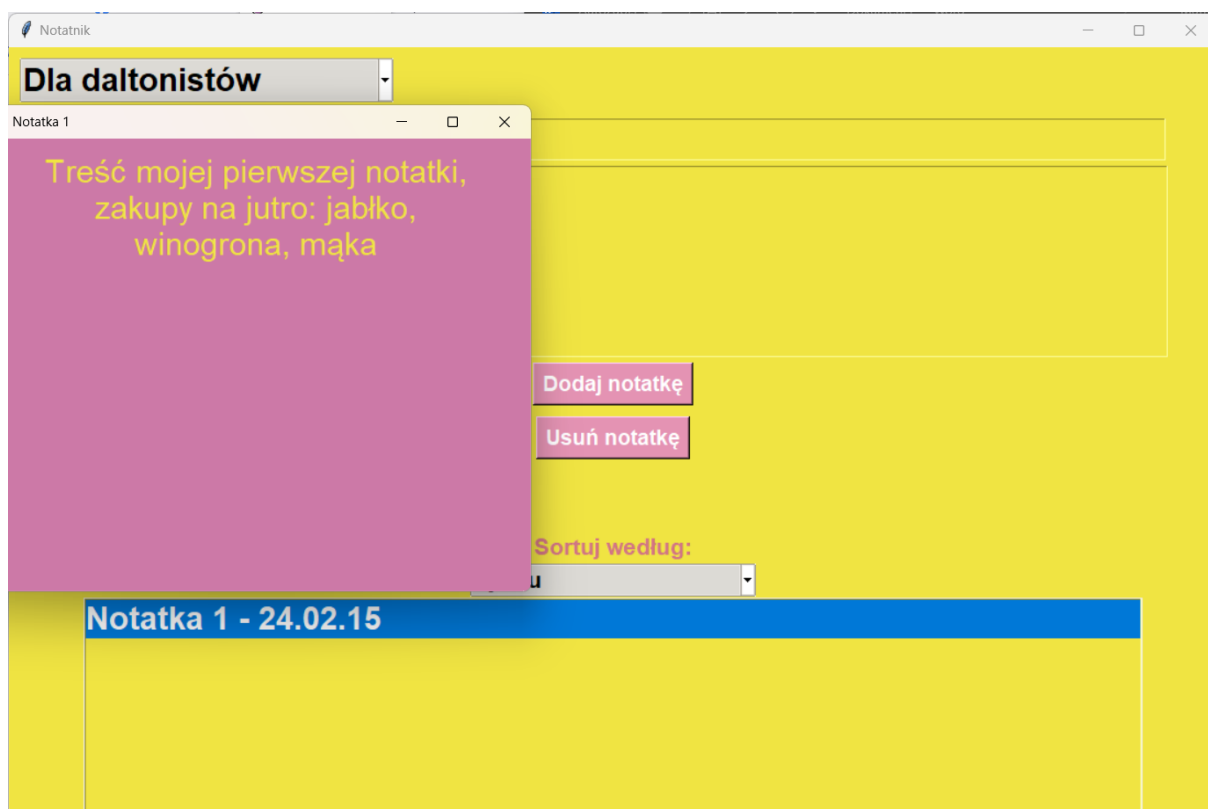
Opcje notatnika:

Notatnik jest bardzo prosty, ponieważ target jakim są osoby niedowidzące w większości są w wieku starszym dlatego obsługa skomplikowanych aplikacji może sprawiać im problemy.

Każda notatka dodana, ma tytuł datę dodania, oraz oczywiście treść.
Jest możliwość sortowania notatek na 3 sposoby:



Ostatecznie odczytywanie notatek wygląda w ten sposób:



Program w pythonie:

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from datetime import datetime

class NoteApp:
    def __init__(self, gui):
        self.gui = gui
        self.gui.title("Notatnik")
        self.gui.geometry("1100x700")
        self.gui.configure(bg="#37306B")

        self.colorSet = {
            "Domyślny": {"bg": "#37306B", "fg": "#D27685"},
            "Ciemny": {"bg": "#070F2B", "fg": "#9290C3"},
        }
```

```

        "Biało-czarny": {"bg": "white", "fg": "black"},
        "Dla daltonistów": {"bg": "#f0e442", "fg": "#cc79a7"}
    }

    self.colorWart = tk.StringVar()
    self.colorWart.set("Domyślny")
    colorZmienna = list(self.colorSet.keys())
    self.colorChanger = ttk.Combobox(self.gui,
textvariable=self.colorWart, values=colorZmienna, state="readonly",
font=('Helvetica', 22, 'bold'))
    self.colorChanger.place(x=10, y=10)
    self.colorChanger.bind("<<ComboboxSelected>>", self.colorChange)
    style = ttk.Style()
    style.theme_use('classic')

    self.Frame = tk.Frame(self.gui, bg=self.colorSet["Domyślny"]["bg"])
    self.Frame.pack(pady=60, padx=10)

    self.tytul = tk.Label(self.Frame, text="Tytuł notatki:",
font=('Helvetica', 22, 'bold'), bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"])
    self.tytul.grid(row=0, column=0, padx=0, pady=0, sticky="e")
    self.SortujWedlug = tk.Label(self.gui, text="Sortuj według:",
font=('Helvetica', 16, 'bold'),
                                bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"])
    self.SortujWedlug.pack(side=tk.TOP, padx=0, pady=0)
    self.SortujZmienna = tk.StringVar()
    self.SortujZmienna.set("Tytułu")
    sort_by_options = ["Tytułu", "Daty", "Ilość znaków"]

    style = ttk.Style()
    style.theme_use('clam')
    #style.configure("TCombobox", fieldbackground="orange",
background="white")

    self.SortujWybrana = ttk.Combobox(self.gui,
textvariable=self.SortujZmienna, values=sort_by_options, state="readonly",
font=('Helvetica', 16, 'bold'))
    self.SortujWybrana.pack(side=tk.TOP, padx=0, pady=0)
    self.SortujWybrana.bind("<<ComboboxSelected>>", self.sort)

    self.TytulBox = tk.Entry(self.Frame, width=50, font=('Helvetica',
22, 'bold'), bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"])
    self.TytulBox.grid(row=0, column=1, padx=(0, 5), pady=5)

    self.Label1 = tk.Label(self.Frame, text="Treść notatki:",
font=('Helvetica', 22, 'bold'), bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"])
    self.Label1.grid(row=1, column=0, padx=5, pady=5, sticky="e")

    self.BoxNotatki = tk.Text(self.Frame, width=50, height=5,
font=('Helvetica', 22, 'bold'), bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"], wrap=tk.WORD)
    self.BoxNotatki.grid(row=1, column=1, padx=0, pady=0)
    self.BoxNotatki.bind("<Return>", lambda event: "break")

    self.AddButton1 = tk.Button(self.Frame, text="Dodaj notatkę",
command=self.dodajNotatke, bg="#E493B3", fg="white", font=('Helvetica', 14,
'bold'))

```

```

        self.AddButton1.grid(row=2, columnspan=2, padx=5, pady=5)

        self.Button2 = tk.Button(self.Frame, text="Usuń notatkę",
command=self.delete, bg="#E493B3", fg="white", font=('Helvetica', 14,
'bold'))
        self.Button2.grid(row=3, columnspan=2, padx=5, pady=5)
        print("TWOJ STARTY!!!!!!!!!!!!!!!!!!!!!!!!!!!!")

        self.NotatkiWszystkie = tk.Listbox(self.gui, width=60, height=15,
font=('Helvetica', 22, 'bold'), bg=self.colorSet["Domyślny"]["bg"],
fg=self.colorSet["Domyślny"]["fg"])
        self.NotatkiWszystkie.pack(padx=1, pady=1)

        self.NotatkiWszystkie.bind("<ButtonRelease-1>", self.display_note)
        self.TytulBox.bind("<Return>", lambda event: self.dodajNotatke())

def colorChange(self, event=None):
    colorSet = self.colorWart.get()
    bgColor = self.colorSet[colorSet]["bg"]
    fgColor = self.colorSet[colorSet]["fg"]
    self.gui.configure(bg=bgColor)
    self.SortujWedlug.configure(bg=bgColor)
    self.Frame.configure(bg=bgColor)
    self.tytul.configure(bg=bgColor, fg=fgColor)
    self.TytulBox.configure(bg=bgColor, fg=fgColor)
    self.Label1.configure(bg=bgColor, fg=fgColor)
    self.BoxNotatki.configure(bg=bgColor, fg=fgColor)
    self.AddButton1.configure(bg="#E493B3", fg="white")
    self.NotatkiWszystkie.configure(bg=bgColor, fg=fgColor)
    self.Button2.configure(bg="#E493B3", fg="white")

def dodajNotatke(self, event=None):
    title = self.TytulBox.get().strip()
    BoxNotatki = self.BoxNotatki.get("1.0", "end-1c").strip()
    if title:
        note_title = title
    elif len(BoxNotatki) > 20:
        note_title = BoxNotatki[:20]
    else:
        note_title = BoxNotatki

    date_str = datetime.now().strftime('%y.%m.%d Godzina: %H:%M')
    self.NotatkiWszystkie.insert(tk.END, f"{note_title} - {date_str}")
    with open(f"{note_title}.txt", "w") as f:
        f.write(BoxNotatki)
    self.TytulBox.delete(0, tk.END)
    self.BoxNotatki.delete("1.0", tk.END)

def delete(self):
    zaznaczony = self.NotatkiWszystkie.curselection()
    if zaznaczony:
        note_title = self.NotatkiWszystkie.get(zaznaczony).split(" -
") [0]

        self.NotatkiWszystkie.delete(zaznaczony)

        with open(f"{note_title}.txt", "r") as f:
            BoxNotatki = f.read()

```



```

        messagebox.showinfo(note_title, BoxNotatki)

    else:
        messagebox.showwarning("Uwaga", "Wybierz notatkę do
usunięcia.")

    def display_note(self, event=None):
        zaznaczony = self.NotatkiWszystkie.curselection()
        if zaznaczony:
            note_title = self.NotatkiWszystkie.get(zaznaczony).split(" -
") [0]

            with open(f"{note_title}.txt", "r") as f:
                BoxNotatki = f.read()
            note_window = tk.Toplevel(self.gui)
            note_window.title(note_title)

note_window.configure(bg=self.colorSet[self.colorWart.get()]["fg"])
            note_window.geometry("500x500")
            Label1 = tk.Label(note_window, text=BoxNotatki,
font=("Helvetica", 22),

bg=self.colorSet[self.colorWart.get()]["fg"],

fg=self.colorSet[self.colorWart.get()]["bg"], wraplength=400)
            Label1.pack(padx=10, pady=10)

    def sort(self, event=None):
        sort_by = self.SortujZmienna.get()
        notes = list(self.NotatkiWszystkie.get(0, tk.END)) # Pobranie
notatek z listy
        if sort_by == "Tytułu":
            self.NotatkiWszystkie.delete(0, tk.END)
            sorted_notes = sorted(notes)
            for note in sorted_notes:
                self.NotatkiWszystkie.insert(tk.END, note)
        elif sort_by == "Daty":
            self.NotatkiWszystkie.delete(0, tk.END)
            notatkiDaty = [(note.split(" - ")[1], note) for note in notes]
            sorted_notes = sorted(notatkiDaty)
            for date, note in sorted_notes:
                self.NotatkiWszystkie.insert(tk.END, note)

def main():
    root = tk.Tk()
    root.title("Notatnik")
    app = NoteApp(root)
    root.bind("<Return>", app.dodajNotatke)
    root.mainloop()

if __name__ == "__main__":
    main()

```