UNIVERSITATEA "POLITEHNICA" DIN BUCUREȘTI

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT INTERFEȚE OM-MAȘINĂ

APLICAȚIE DE CALENDAR

STUDENȚI:

MIRCEA-SEBASTIAN BĂNARU

ALEXANDRU-COSMIN DAN

MIHAI-ȘTEFAN VINȚELER

GRUPA: 441A

CADRU DIDACTIC COORDONATOR:

S.L.DR. ING. ANDREEA GRIPARIS

București, 2023

Descrierea librăriilor folosite

- **tkinter –** pachet standard din python pentru a crea interfețe(GUI) și este disponibil pe majoritatea platformelor Unix, macOS, Windows.
  INSTALARE: *pip install tk*

- **Numpy –** librărie care se ocupă cu operaţii matematice aplicate pe vectori unidimensionali sau multidimensionali precum şi alte funcţii matematice
  INSTALARE: *în ultimele versiuni de python, librăria vine preinstalată sau se poate folosi pip install numpy*

- **OS –** modul ce facilitează interacţiunea dintre programul de dezvoltare şi sistemul de operare.
  INSTALARE: *modul standard preinstalat împreună cu python*

- **gTTS (Google Text-to-Speech) –** modul python care folosește API-ul de la Google Translate pentru a transforma limbajul scris în limbaj vorbit
  INSTALARE: *pip install gTTS*

- **PIL (Python Imaging Library) –** librăria de bază din python ce permite lucrul cu imagini
  INSTALARE: *pip install Pillow*

- **pygame –** modul al python conceput pentru a scrie jocuri video. Acesta include de asemenea pachete pentru lucrul cu fişiere audio
  INSTALARE: *pip install pygame*

- **io –** modul al python folosit pentru lucrul cu fişiere, de exemplu citire/scriere din fişier.
  INSTALARE: *modul standard preinstalat împreună cu python*

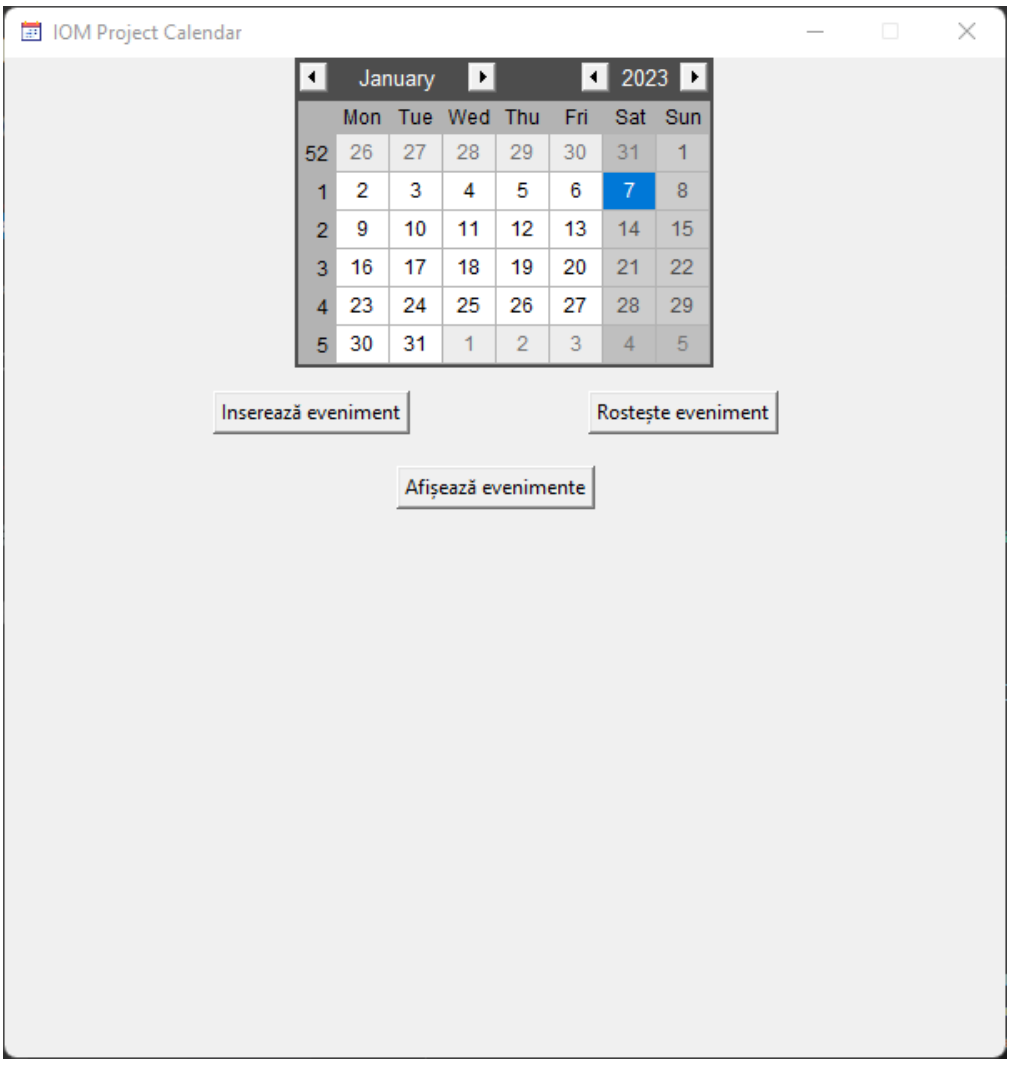
- **speech_recognition –** motor de recunoaştere a semnalelor vocale de la Google ce transcrie conţinutul unui fişier audio în text.
  INSTALARE: *pip install SpeechRecognition*

Descrierea aplicației

În cadrul acestui proiect am implementat o aplicație de calendar cu următoarele funcționalități:

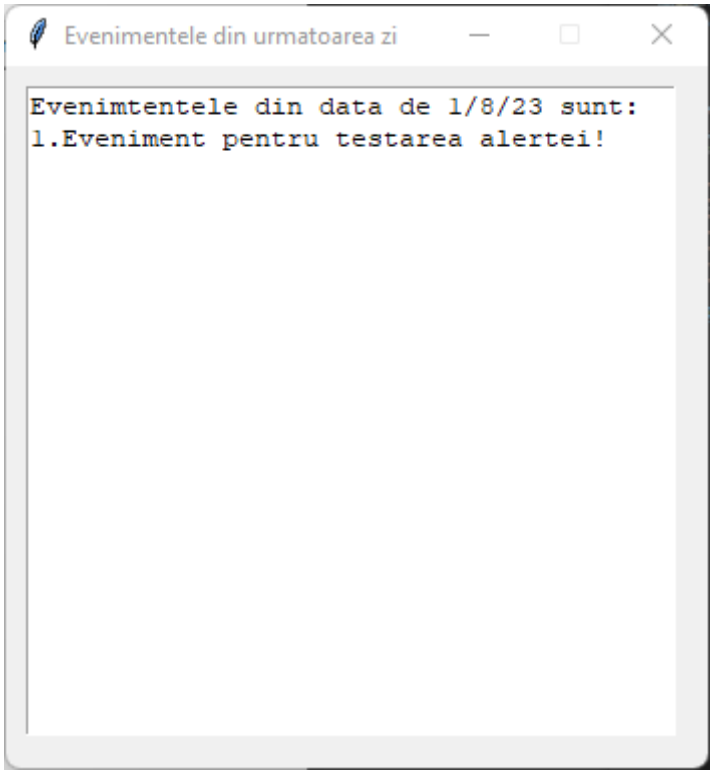
- adăugarea unui eveniment atât vocal cât și text
- alertarea utilizatorului înaintea unui eveniment
- rostirea evenimenteleor dintr-o zi

Interfața aplicației a fost construită folosind componente de bază din librăria tkinter, mai exact butoane(**Button**), calendar(**Calendar**), text(**Text**), etichete(**Label**) rezultatul fiind următorul:



În momentul deschiderii aplicației, va fi evidențiată cu albastru data curentă a sistemului iar utilizatorul va putea selecta orice dată folosind butonul Left Click al mouse-ului pentru a folosi funcționalitățile aplicației pentru data respectivă. De asemenea, tot la pornirea aplicației, utilizatorul va fi alertat într-o nouă fereastră în cazul în care există evenimente introduse în prealabil ziua următoare.

Alerta arată astfel:



Codul care gestionează alerta este următorul:

```
def CheckNextDayEvents():
    Events = GetAllScheduledEvents()
    tommorow = (date + timedelta(1)).strftime('%#m/%#d/%#y')
    current_events = ""
    index = 0
    for event_date, text in Events:
        if event_date == tommorow:
            index += 1
            current_events = current_events + str(index) + '.' + text +
"\n"
    if current_events != "":
        NewWindow(tommorow, current_events)

def NewWindow(date, text):
    UpcomingEventsWindow = Toplevel()
    UpcomingEventsWindow.grab_set()
    UpcomingEventsWindow.title(NEXT_DAY_TITLE)
    UpcomingEventsWindow.resizable(False, False)
    UpcomingEventsWindow.geometry("350x350")
    EventsText = Text(UpcomingEventsWindow, width=40, height=20)
    EventsText.insert('end', "Evenimtentele din data de " + date + "
sunt:\n")
    EventsText.insert('end', text)
    EventsText.configure(state='disabled')
    EventsText.place(x=10, y=10)
```

Funcţia **CheckNextDayEvents** extrage evenimentele din ziua următoare a sistemului folosind funcţia **GetAllScheduledEvents**, iar în cazul în care există

evenimente introduse în acea zi se va crea o nouă fereastră folosind funcţia **NewWindow**, în caz contrar nu se întâmplă nimic.

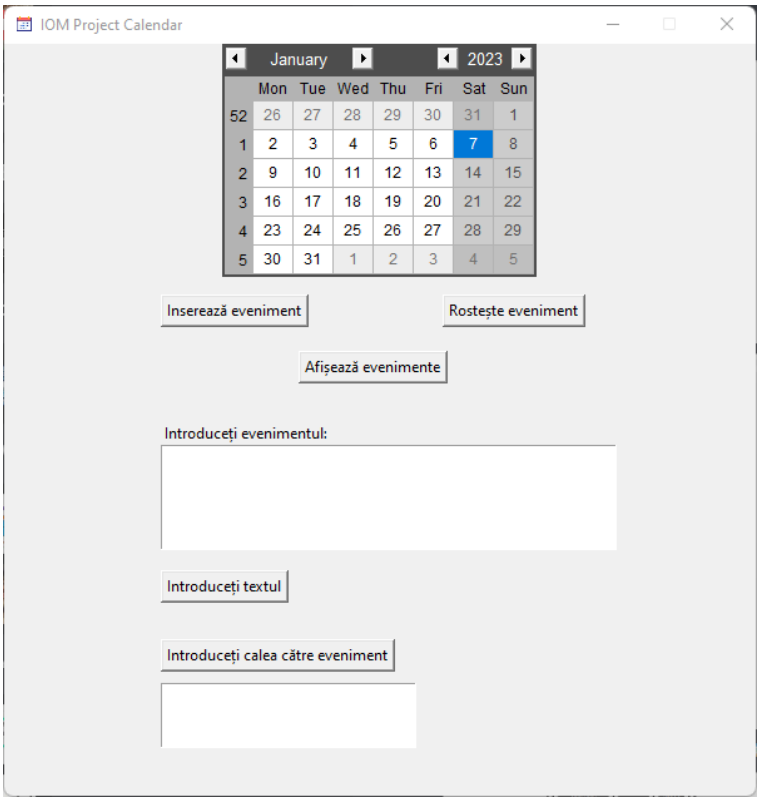
```
def GetAllScheduledEvents():
    schedule = []
    global EventsText
    try:
        with open('EventData.csv', 'r') as file:
            scheduled_events = csv.reader(file)
            for row in scheduled_events:
                schedule.append((row[0], row[1]))
    except:
        print("No events yet!")
    return schedule
```

Această funcţie extrage toate evenimentele ce se vor petrece în ziua următoare celei curente din fişierul **EventData.csv,** ce stochează toate evenimentele introduse în cadrul aplicaţiei.

Butonul **Inserează eveniment** :

La apăsarea butonului, în interfaţă apar două componente de tip TextBox şi două butoane ce facilitează inserarea unui eveniment atât scris, cât şi vocal.

```
InsertEventButton = Button(root, text=INSERT_BUTTON_TEXT, command= lambda:
[ShowInsertEventTextBox(), ShowInsertVocalText()]).place(x=125, y=200)
```



De asemenea, la apăsarea butonului se vor apela următoarele două funcţii: **ShowInsertEventTextBox** şi **ShowInsertVocalText.**

```
def ShowInsertEventTextBox():
    global InsertIntoCSVButton, InsertEventLabel, data, InsertEventText
    remove_parts()
    InsertEventLabel = Label(root, text='Introduceţi evenimentul:')
    InsertEventLabel.place(x=125, y=300)
    InsertEventText = Text(root, height=5, width=45)
    InsertEventText.place(x=125, y=320)
    InsertIntoCSVButton = Button(root, text='Introduceti textul',
command=InsertEvent)
    InsertIntoCSVButton.place(x=125, y=420)
```

Funcţia creează, aşează şi configurează componentele din tkinter necesare introducerii evenimentului sub formă de text. Pentru introducerea propriu-zisă a evenimentului se foloseşte un alt buton, **Introduceţi textul** ce apelează funcţia **InsertEvent.**

```
def InsertEvent():
    global insert_label
    data = cal.get_date()
    event = InsertEventText.get(1.0, END)
    formatted_event = event.rstrip(event[-1])
    InsertedData = [data, formatted_event]
    print(InsertedData)
    with open('EventData.csv', 'a') as file:
        writer = csv.writer(file, lineterminator='\n')
        writer.writerow([data, formatted_event])
    insert_label = Label(root, text='Eveniment introdus')
    insert_label.place(x=235, y=280)
```

Această funcţie introduce în fişierul **EventData.csv** evenimentul scris în TextBox şi data selectată din calendar. În urma introducerii evenimentului se va afişa o etichetă care va semnala succesul acestei acţiuni.

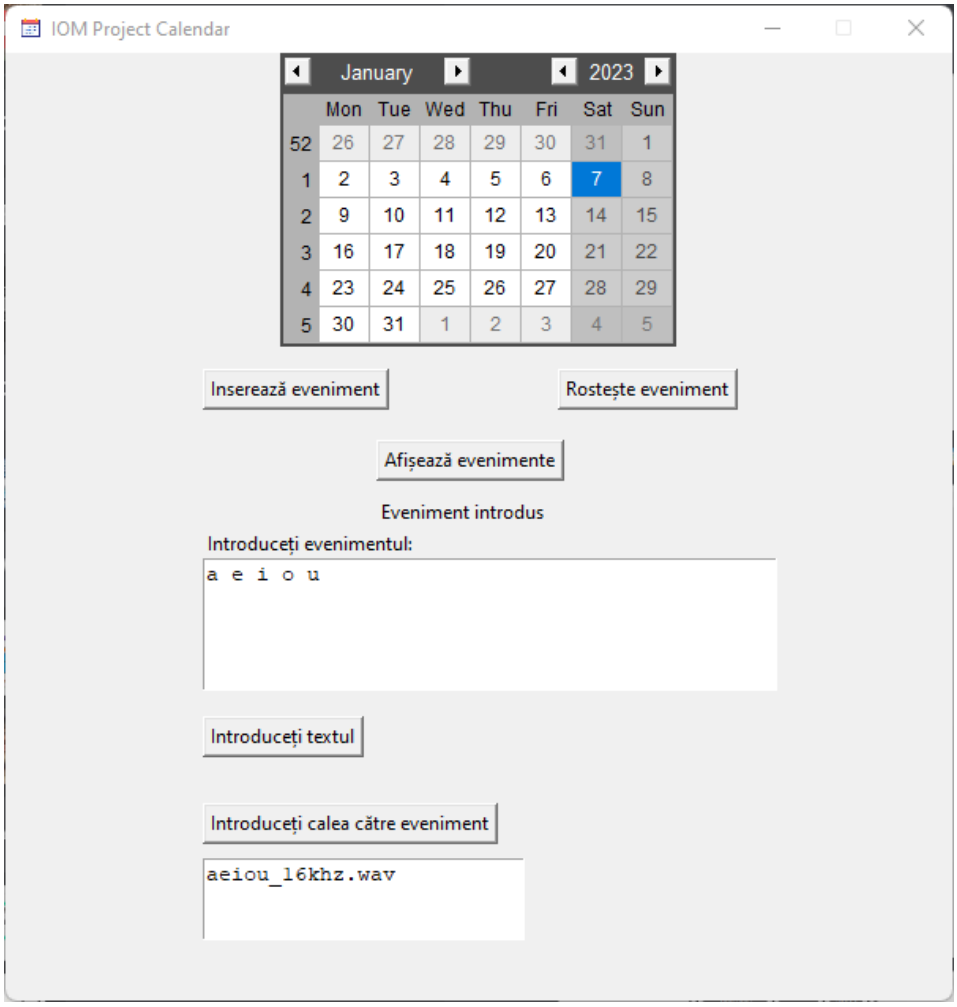
```
def ShowInsertVocalText():
    global InsertPathButton, InsertEventPath
    InsertEventPath = Text(root, height=3, width=25)
    InsertEventPath.place(x=125, y=510)
    InsertPathButton = Button(root, text='Introduceti calea către
eveniment', command=open_file)
    InsertPathButton.place(x=125, y=475)
```

Pentru introducerea unui eveniment vocal, această funcţie va crea un buton folosit pentru alegerea fişierului vocal ce va fi transformat în text şi apoi introdus în fişierul .csv. De asemenea, această metodă creează un Label pentru afişarea numelui fişierului audio selectat. În momentul în care se alege fişierul audio, se apelează funcţia **open_file.**

```
def open_file():
    global IntroducedEvent
    filepath = filedialog.askopenfilename(title = 'pen')
    file_name = filepath.split('/')[-1]
    InsertEventPath.insert (END, file_name)
    SpeechRecognizer = sr.Recognizer()
```

```
    myaudiofile = sr.AudioFile(filepath)
    with myaudiofile as source:
        myaudio = SpeechRecognizer.record(myaudiofile)
        IntroducedEvent = SpeechRecognizer.recognize_google (myaudio,
language = "ro-RO", show_all = False)
        print('Evenimentul introdus vocal este:', IntroducedEvent)
    InsertEventText.insert(END, IntroducedEvent)
```

În cadrul acestei funcţii se realizează transformarea din semnal vocal în text(în limba română) ca apoi textul sintetizat să fie introdus în TextBox-ul de la introducerea text pentru a putea fi verificată corectitudinea evenimentului transmis. Finalizarea acestui tip de introducere se realizează apăsând butonul **Introduceţi textul**.



Butonul **Rostește eveniment** :

```
        SpeechEventButton = Button(root, text=SPEECH_BUTTON_TEXT,
command=speech_event).place(x=350, y=200)
```

La apăsarea butonului, se va apela funcţia **speech_event.**

```
def speech_event():
    global EventData
    remove_parts()
    EventData = GetAllScheduledEvents()
    current_events = EMPTY_STRING
    for event_date, text in EventData:
        if event_date == cal.get_date():
            current_events = current_events + text + ", "
    ReadEventText(current_events)
def ReadEventText(text):
    global date_label
    remove_parts()
    date_label = Label(root, text='')
    date_label.place(x=220, y=280)
    mp3 = BytesIO()
    mixer.init()
    try:
        eventRead = gTTS(text, lang="ro", slow=False)
        eventRead.write_to_fp(mp3)
        mixer.music.load(mp3, "mp3")
        mixer.music.play()
    except:
        date_label.config(text="Nu există evenimente pentru data
selectată!")
```

În cadrul funcţiei **speech_event** se extrag cu ajutorul funcţiei **GetAllScheduledEvents** toate evenimentele din data curentă selectată în interfaţă, iar apoi se apelează funcţia **ReadEventText**. Aceasta cu ajutorul librăriei **gTTS** se vor transforma evenimentele text extrase în semnal vocal. Apoi în mod automat acest semnal va fi redat în sistemul audio al dispozitivului pe care este rulată aplicaţia. Dacă nu există niciun eveniment se va afişa un Label ce va semnala acest lucru.

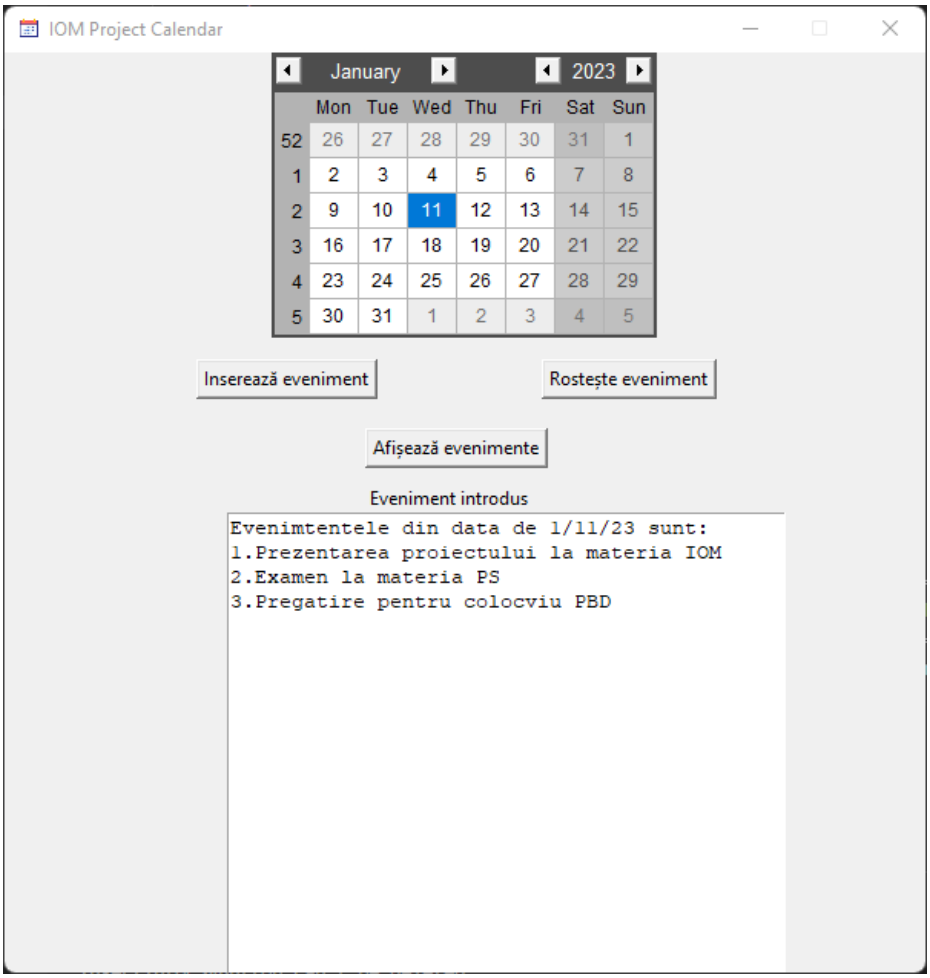
Butonul **Afişează evenimente** :

```
ShowEventsButton = Button(root, text=SHOW_EVENTS_BUTTON_TEXT,
command=show_events).place(x=235, y=245)
```

La apăsarea butonului, se va apela funcţia **show_events.**

```
def show_events():
    global EventsText
    remove_parts()
    Events = GetAllScheduledEvents()
    EventsText = Text(root, width=45, height=50)
    EventsText.insert('end', "Evenimtentele din data de " + cal.get_date()
+ " sunt:\n")
    index = 0
    for event_date, text in Events:
        if event_date == cal.get_date():
            index += 1
            numberOfEvent = str(index) + '.'
            EventsText.insert('end', numberOfEvent + text + '\n')
    EventsText.configure(state='disabled')
    EventsText.place(x=145, y=300)
```

În cadrul acestei funcţii se extrag cu ajutorul funcţiei **GetAllScheduledEvents** toate evenimentele din data curentă selectată în interfaţă şi sunt introduse într-un TextBox ce nu poate fi modificat. În urma apăsării acestui buton, modificările din interfaţă sunt următoarele:



Pentru a ascunde elementele meniului curent la trecerea într-un alt meniu, am folosit funcţia **remove_parts.**

```
def remove_parts():
    try:
        EventsText.place_forget()
    except:
        print("EventsText can't be deleted")
    try:
        date_label.place_forget()
    except:
        print("date_label can't be deleted")
    try:
        InsertIntoCSVButton.place_forget()
    except:
        print("InsertIntoCSVButton can't be deleted")
    try:
        InsertEventLabel.place_forget()
    except:
        print("InsertEventLabel can't be deleted")
```

```
    try:
        InsertEventText.place_forget()
    except:
        print("InsertEventText can't be deleted")
    try:
        insert_label.place_forget()
    except:
        print("insert_label can't be deleted")
    try:
        InsertEventPath.place_forget()
    except:
        print("InsertEventPath can't be deleted")
    try:
        InsertPathButton.place_forget()
    except:
        print('InsertPathButton can\'t be deleted')
```

Pentru rularea aplicației, codul este disponibil pe pagina de GitHub a proiectului nostru atașată la primul punct al bibliografiei. După descărcarea / clonarea proiectului de pe Git, se instalează librăriile conform indicațiilor despre instalare din cadrul **Descrierii librăriilor folosite** iar apoi se rulează scriptul **main.py** pentru explorarea tuturor funcționalităților descrise în cadrul acestei documentații.

Bibliografie

- pagina de git a proiectului: *https://github.com/banarutz/Proiect-IOM-Aplicatie-de-calendar*
- *https://docs.python.org/3/library/tkinter.html*
- *https://en.wikipedia.org/wiki/NumPy*
- *https://docs.python.org/3/library/os.html*
- *https://gtts.readthedocs.io/en/latest/*
- *https://pillow.readthedocs.io/en/stable/*
- *https://en.wikipedia.org/wiki/Pygame*
- *https://docs.python.org/3/library/io.html*
- *https://pypi.org/project/SpeechRecognition/*
- *laboratoare IOM*