Notes application

Description

Notes application is a versatile tool that allows you to take notes in a simple and fast way. The particularity of this application is that it can memorize the user's past or future notes by selecting a reference date from the calendar. The data is saved in a database that can be accessed on the basis of a username and password

Technologies used

Python -Python is a dynamic, high-level, object-oriented programming language developed by Guido van Rossum in 1989

SQLite (sqlite3) - sqlite3 is an API for SQLite, it realizes a database on the physical memory medium of the computer

Google Text-to-Speech (gtts) - Google Text-to-Speech API, converts input text to audio

PyFPDF (fpdf) - PyFPDF is a library for generating PDF documents in the Python programming environment, ported from PHP

playsound (playsound3) - Audio playback module, used on multiple platforms with a single function and no dependencies

Python OS (os) - The module provides facilities for interaction between the programming environment and the operating system.

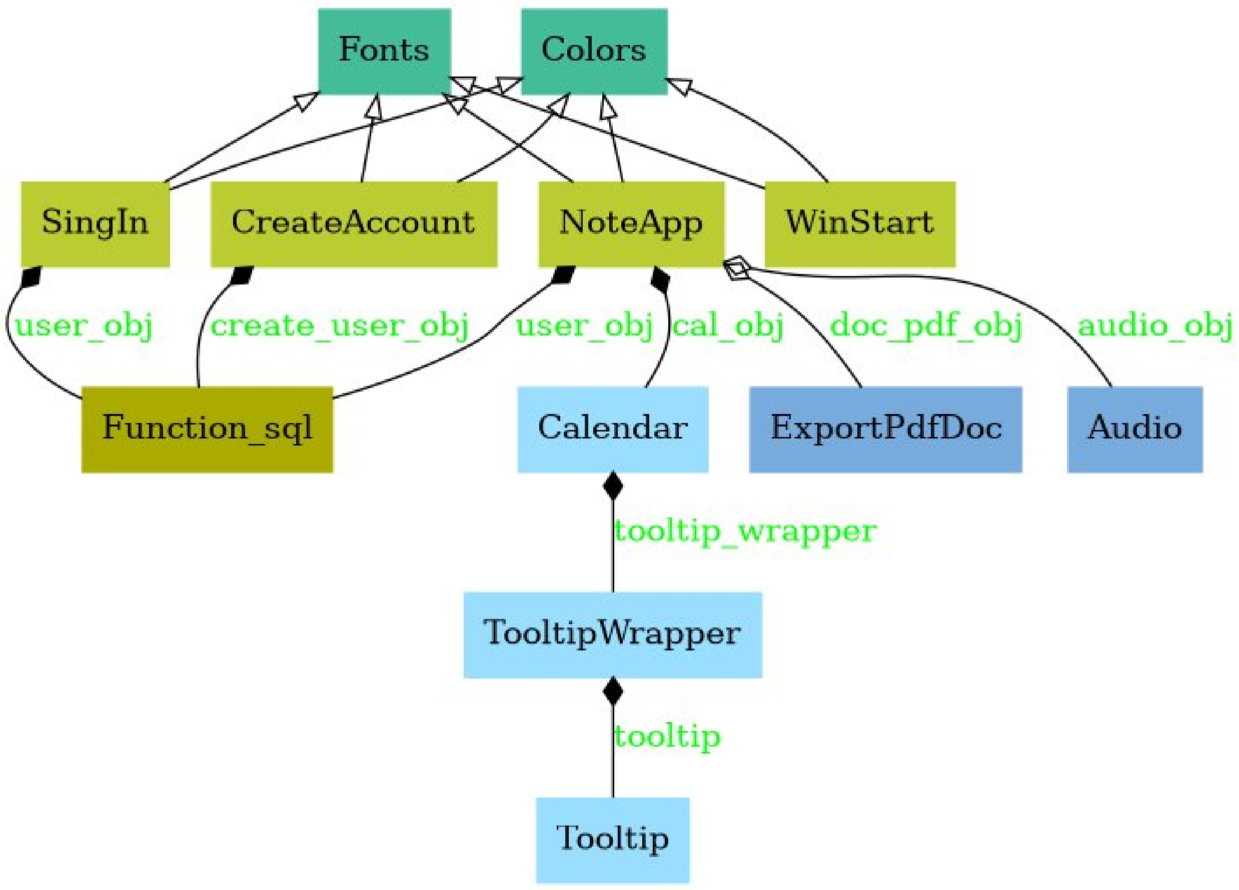
Python DateTime (datetime) - The module provides classes for manipulating date, time and time intervals, suitable for scenarios that require complex calculations and formatting

Python Time (time) - module that provides various methods to work with time-related operations in particular for measuring execution time, pausing execution and retrieving the current time.

Tk (tkinter) - is a standard Python GUI (Graphical User Interface) library that provides a set of tools and widgets to create desktop applications with graphical user interfaces.

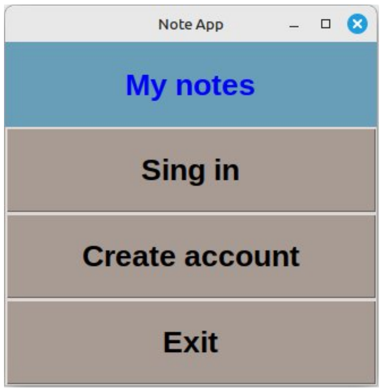
Application structure

The adjacent diagram illustrates the hierarchical architecture of the classes and their functional association

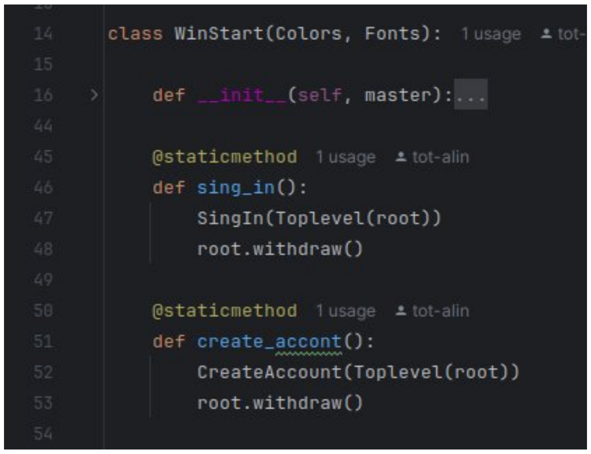


Application features

Initial window

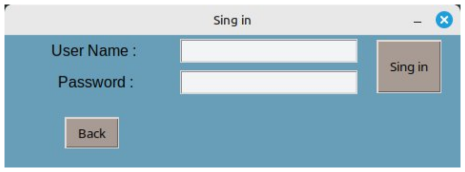


WinStart class methods

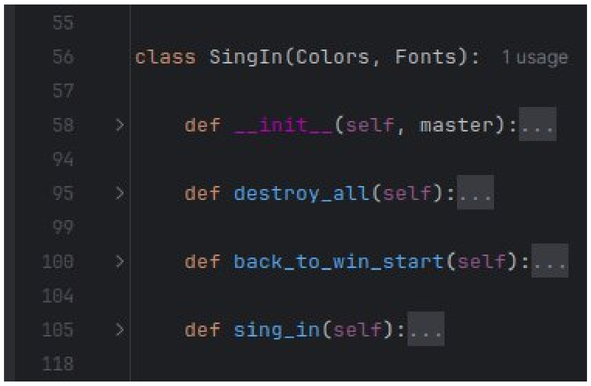


```
class WinStart(Colors, Fonts):  1 usage  tot-

    def __init__(self, master):...

    @staticmethod  1 usage  tot-alin
    def sing_in():
        SingIn(Toplevel(root))
        root.withdraw()

    @staticmethod  1 usage  tot-alin
    def create_accont():
        CreateAccount(Toplevel(root))
        root.withdraw()
```

Sing in window

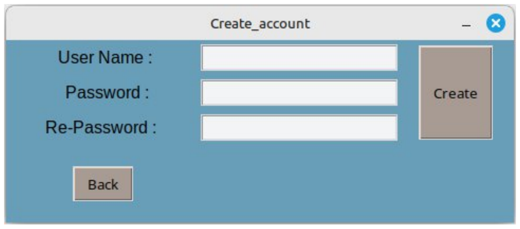


Sing In class methods

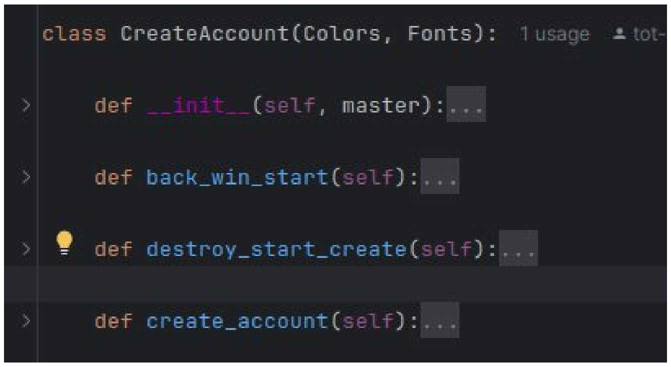


```
class SingIn(Colors, Fonts):  1 usage

    def __init__(self, master):...

    def destroy_all(self):...

    def back_to_win_start(self):...

    def sing_in(self):...
```

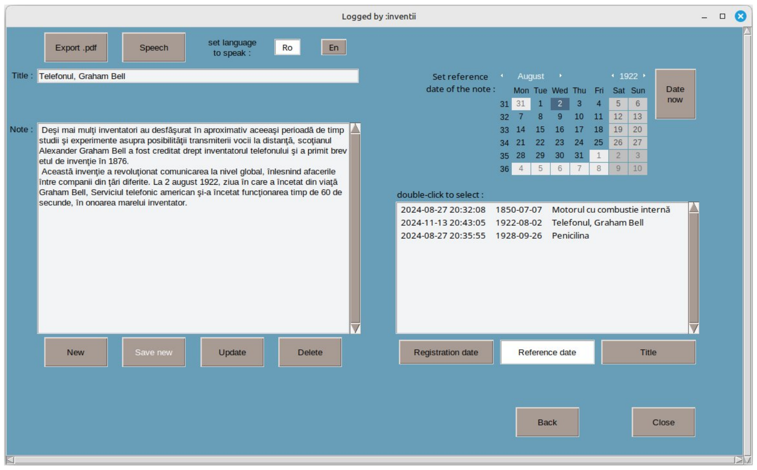
Window create account



Methods of class create account



```
class CreateAccount(Colors, Fonts):  1 usage  tot-

    def __init__(self, master):...

    def back_win_start(self):...

    def destroy_start_create(self):...

    def create_account(self):...
```



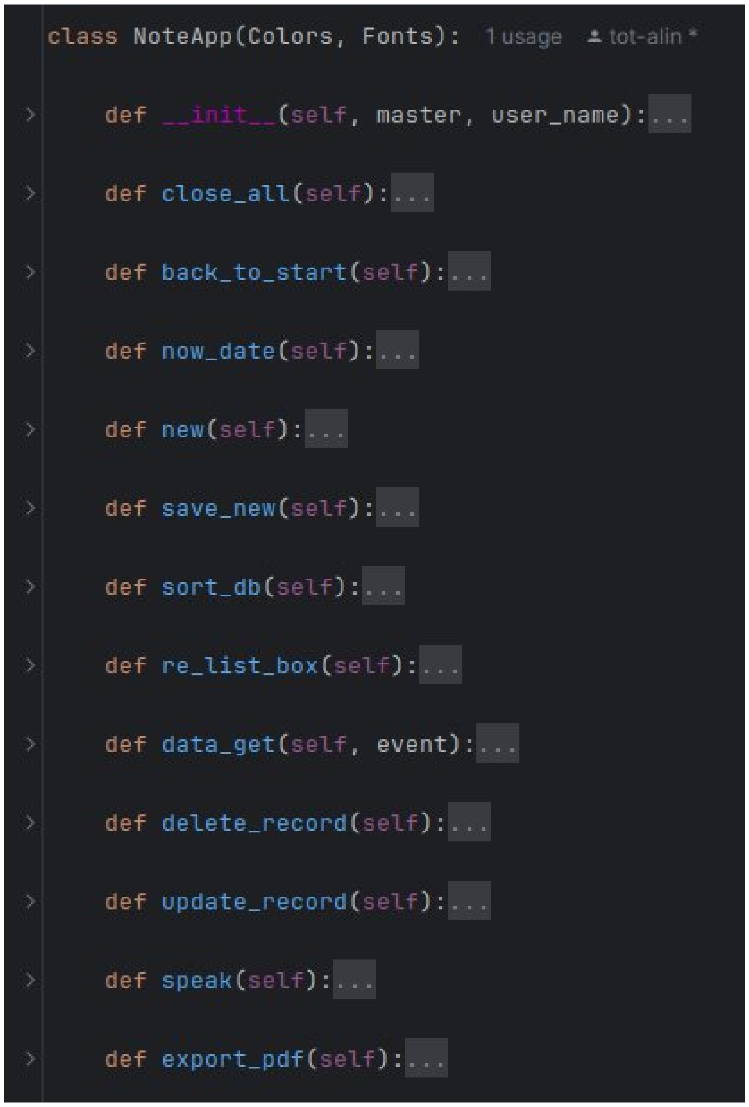
Title : Telefonul, Graham Bell

Note : Deşi mai mulţi inventatori au desfăşurat în aproximativ aceeaşi perioadă de timp studii şi experimente asupra posibilităţii transmiterii vocii la distanţă, scoţianul Alexander Graham Bell a fost creditat drept inventatorul telefonului şi a primit brevetul de invenţie în 1876.
Această invenţie a revoluţionat comunicarea la nivel global, înlesnind afacerile între companii din ţări diferite. La 2 august 1922, ziua în care a încetat din viaţă Graham Bell, Serviciul telefonic american şi-a încetat funcţionarea timp de 60 de secunde, în onoarea marelui inventator.

double-click to select :

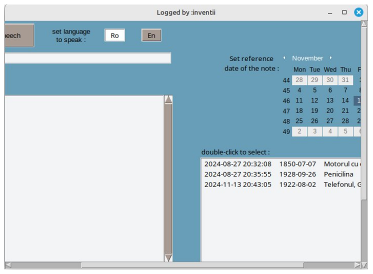
```
2024-08-27 20:32:08   1850-07-07   Motorul cu combustie internă
2024-11-13 20:43:05   1922-08-02   Telefonul, Graham Bell
2024-08-27 20:35:55   1928-09-26   Penicilina
```

Notes App window



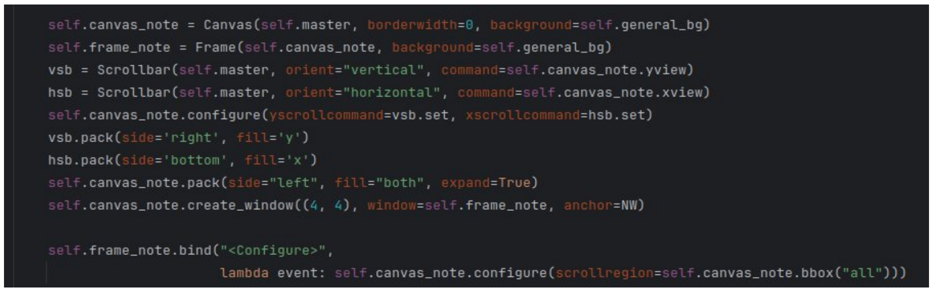
```
class NoteApp(Colors, Fonts):  1 usage  tot-alin *

    def __init__(self, master, user_name):...

    def close_all(self):...

    def back_to_start(self):...

    def now_date(self):...

    def new(self):...

    def save_new(self):...

    def sort_db(self):...

    def re_list_box(self):...

    def data_get(self, event):...

    def delete_record(self):...

    def update_record(self):...

    def speak(self):...

    def export_pdf(self):...
```

NoteApp class methods



```
2024-08-27 20:32:08    1850-07-07    Motorul cu
2024-08-27 20:35:55    1928-09-26    Penicilina
2024-11-13 20:43:05    1922-08-02    Telefonul, G
```

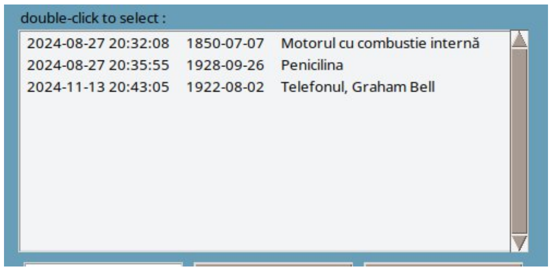
Illustration of scroll bars



```python
self.canvas_note = Canvas(self.master, borderwidth=0, background=self.general_bg)
self.frame_note = Frame(self.canvas_note, background=self.general_bg)
vsb = Scrollbar(self.master, orient="vertical", command=self.canvas_note.yview)
hsb = Scrollbar(self.master, orient="horizontal", command=self.canvas_note.xview)
self.canvas_note.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
vsb.pack(side='right', fill='y')
hsb.pack(side='bottom', fill='x')
self.canvas_note.pack(side="left", fill="both", expand=True)
self.canvas_note.create_window((4, 4), window=self.frame_note, anchor=NW)

self.frame_note.bind("<Configure>",
                     lambda event: self.canvas_note.configure(scrollregion=self.canvas_note.bbox("all")))
```

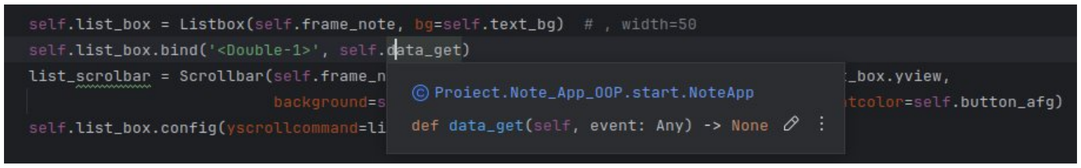
Code that scrolls the Note App window

Selection window



```
2024-08-27 20:32:08    1850-07-07    Motorul cu combustie internă
2024-08-27 20:35:55    1928-09-26    Penicilina
2024-11-13 20:43:05    1922-08-02    Telefonul, Graham Bell
```

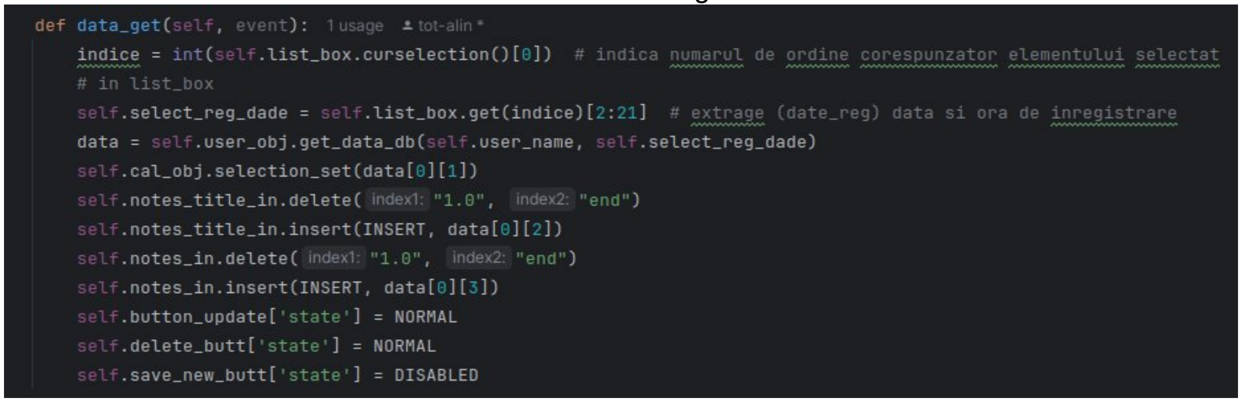
The "bind" method for double-click selection



```python
self.list_box = Listbox(self.frame_note, bg=self.text_bg)  # , width=50
self.list_box.bind('<Double-1>', self.data_get)
list_scrolbar = Scrollbar(self.frame_n                        _box.yview,
                          background=s                        tcolor=self.button_afg)
self.list_box.config(yscrollcommand=li
```

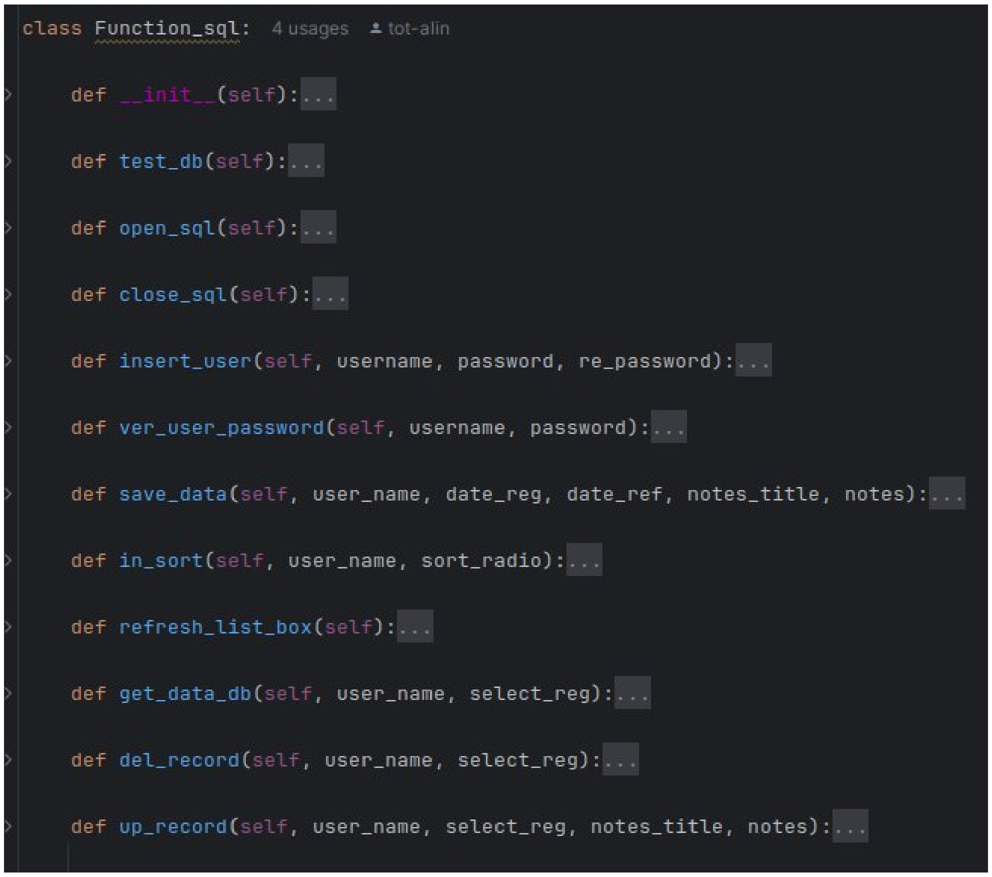
```
Proiect.Note_App_OOP.start.NoteApp
def data_get(self, event: Any) -> None
```

The window reordering method



```python
def data_get(self, event):  1 usage  tot-alin *
    indice = int(self.list_box.curselection()[0])  # indica numarul de ordine corespunzator elementului selectat
    # in list_box
    self.select_reg_dade = self.list_box.get(indice)[2:21]  # extrage (date_reg) data si ora de inregistrare
    data = self.user_obj.get_data_db(self.user_name, self.select_reg_dade)
    self.cal_obj.selection_set(data[0][1])
    self.notes_title_in.delete( index1: "1.0",  index2: "end")
    self.notes_title_in.insert(INSERT, data[0][2])
    self.notes_in.delete( index1: "1.0",  index2: "end")
    self.notes_in.insert(INSERT, data[0][3])
    self.button_update['state'] = NORMAL
    self.delete_butt['state'] = NORMAL
    self.save_new_butt['state'] = DISABLED
```

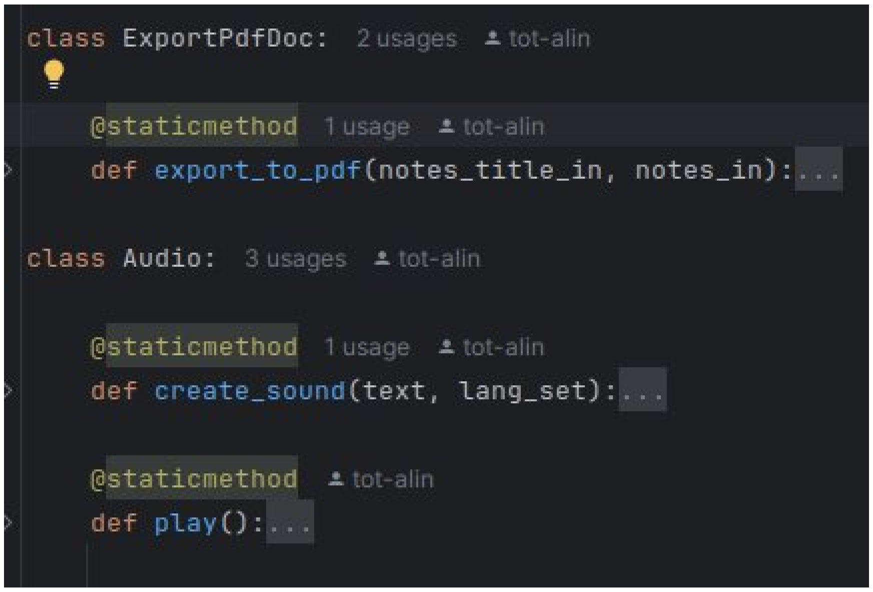
Methods of class Function_sql



```
class Function_sql:  4 usages  tot-alin

    def __init__(self):...

    def test_db(self):...

    def open_sql(self):...

    def close_sql(self):...

    def insert_user(self, username, password, re_password):...

    def ver_user_password(self, username, password):...

    def save_data(self, user_name, date_reg, date_ref, notes_title, notes):...

    def in_sort(self, user_name, sort_radio):...

    def refresh_list_box(self):...

    def get_data_db(self, user_name, select_reg):...

    def del_record(self, user_name, select_reg):...

    def up_record(self, user_name, select_reg, notes_title, notes):...
```

test_db - test if the database exists
open_sql - open and connect to the database
close_sql - close the cursor and the connection to the database
insert_user - write user password, and create new table
ver_user_password - check user and password in the intended table (users)
save_data - save notes to table named after user name
in_sort - pre-sort table information to refresh_list_box sorted in the form requested by list_box
refresh_list_box - send list_box the requested data
get_data_db - search database for user information and record parameters
del_record - delete a record
up_record - modify a record

ExportPdfDoc and Audio class methods



```
class ExportPdfDoc:  2 usages  tot-alin

    @staticmethod  1 usage  tot-alin
    def export_to_pdf(notes_title_in, notes_in):...

class Audio:  3 usages  tot-alin

    @staticmethod  1 usage  tot-alin
    def create_sound(text, lang_set):...

    @staticmethod  tot-alin
    def play():...
```

class ExportPdfDoc
- The export_to_pdf method is called from the NoteApp class with the parameters title and content text. This method generates a PDF document according to a given template.

class Audio
- The create_sound method prepends the text and language setting using the gTTs API and passes it to a server which converts it to an audio file
- The play method prepends the file created by the create_sound method and plays it, after which it will delete the audio file

Bibliography:

https://ro.wikipedia.org/wiki/Python

https://pyfpdf.readthedocs.io/en/latest/

https://github.com/pndurette/gTTS

https://pypi.org/project/playsound3/

https://docs.python.org/3/library/sqlite3.html

https://docs.python.org/3/library/datetime.html

https://levelup.gitconnected.com/time-module-vs-datetime-module-in-python-f4a5e818350a

https://www.geeksforgeeks.org/introduction-to-tkinter/