TALLINNA TÖÖSTUSHARIDUSKESKUS

**ANDMEBAASI HALDAMINE RAAMATUKATALOOGIS**  
Arvestus töö

Õpetaja: Marina Oleinik  
Õpilane: Jelizaveta Ostapjuk  
Rühm: TARpv23

Tallinn  
2024

# SISUKORD

[SISUKORD 2](#_Toc166242482)

[SISEJUHATUS 3](#_Toc166242483)

[TEOREETILINE OSA 4](#_Toc166242484)

[1. Vahendid 4](#_Toc166242485)

[PRAKTILINE OSA 6](#_Toc166242486)

[2. Подготовка к работе 6](#_Toc166242487)

[3. Создание базы данных 6](#_Toc166242488)

[4. Создание графического интерфейса 7](#_Toc166242489)

[KASUTUSJUHEND 10](#_Toc166242490)

[KOKKUVÕTTE 11](#_Toc166242491)

[ALLIKAD 12](#_Toc166242492)

# ****SISEJUHATUS****

Цель проекта заключается в создании интерактивного приложения для управления каталогом книг. Это приложение позволяет пользователям просматривать информацию о книгах, авторах и жанрах, а также добавлять, обновлять и удалять данные в базе данных. Этим приложением может воспользоваться огромный круг людей, начиная от работников библиотеки для введения учета книг до любителей почитать для введения личной библиотеки. Актуальность приложения управлением каталога книг заключается в её эффективности и простоте управления, они нужны различным сферам деятельности, к примеру для академических целей или же для магазинов и библиотек.

Для достижения этой цели я использовала приложение Visual Studio 2022 и писала на языке Python. Так же использовала различные вспомогательные библиотеки такие как tkinter и sqlite3. Tkinter использовался для внешней оболочки, а SQLite3 для управления базой данных.

# TEOREETILINE OSA

## Vahendid

* **Python** - это язык программирования, который широко используется в интернет-приложениях, разработке программного обеспечения, науке о данных и машинном обучении (ML). Разработчики используют Python, потому что он эффективен, прост в изучении и работает на разных платформах. Программы на языке Python можно скачать бесплатно, они совместимы со всеми типами систем и повышают скорость разработки. (Что такое Python?, 2023)
* **Tkinter** — это кроссплатформенный графический интерфейс Python, позволяющий работать с библиотекой Tk. Он содержит элементы графического интерфейса пользователя (GUI — Graphical User Interface), с помощью которых можно создавать различные приложения. Изначально Tkinter был разработан в 1988 году профессором математических наук в университете Беркли Джоном Остерхаутом для собственного языка программирования Tcl. Однако впоследствии он был адаптирован для многих других ЯП, таких как Ruby Perl и т.д. С 1994 года этот программный пакет входит в стандартный дистрибутив языка программирования Python. (Tkinter, 2023)
* **Tkinter.ttk -** В состав пакета tkinter входит модуль ttk, содержащий классы более стилизованных и современных виджет. По умолчанию их внешний вид зависит от операционной системы. В ttk используется новый движок для создания виджетов. Этот движок обладает поддержкой тем и стилей оформления. Начиная с версий python 2.7 и 3.1.2 в Tkinter включён модуль для работы с ttk. (Tkinter ttk (модуль), б.д.) (Шапошникова, 2024)
* **SQLite3 -** это встраиваемая система управления базами данных для клиент-серверных приложений. Эта СУБД отличается компактностью – она занимает один файл, благодаря чему встраивается в приложения, чтобы выполнять функции СУБД автономно, без использования сервера. Также SQLite применяется в веб-разработке небольших проектов. SQLite – одна из самых популярных СУБД с открытым исходным кодом и большим комьюнити. (SQLite, б.д.)

# PRAKTILINE OSA

## Подготовка к работе

Для создания системы управления библиотекой книг с использованием Python и Tkinter, я разработаю приложение, которое позволит пользователям добавлять книги в библиотеку, просматривать их список и удалять книги из списка. В этом процессе я буду использовать базу данных для хранения информации о книгах, включая название, дату выпуска, автора и жанр.

Для начала необходимо установить необходимые библиотеки:

* Python
* Tkinter (Для графического интерфейса)
* SQLite (Для работы с базой данных)

## Создание базы данных

Создала базу данных SQLite, которая будет содержать таблицы для авторов, жанров и книг. Код для создания:

def create\_valitud\_tabel():

cursor = connection.cursor()

valitud\_tabel = table\_combobox.get()

if valitud\_tabel == "Autorid":

cursor.execute("""

CREATE TABLE IF NOT EXISTS Autorid (

autor\_id INTEGER PRIMARY KEY AUTOINCREMENT,

autor\_nimi TEXT NOT NULL,

sünnikuupäev DATE NOT NULL

)

""")

elif valitud\_tabel == "Žanrid":

cursor.execute("""

CREATE TABLE IF NOT EXISTS Žanrid (

žanr\_id INTEGER PRIMARY KEY AUTOINCREMENT,

žanri\_nimi TEXT NOT NULL

)

""")

elif valitud\_tabel == "Raamatud":

cursor.execute("""

CREATE TABLE IF NOT EXISTS Raamatud (

raamat\_id INTEGER PRIMARY KEY AUTOINCREMENT,

pealkiri TEXT NOT NULL,

väljaandmise\_kuupäev DATE NOT NULL,

autor\_id INTEGER,

žanr\_id INTEGER,

FOREIGN KEY(autor\_id) REFERENCES Autorid(autor\_id),

FOREIGN KEY(žanr\_id) REFERENCES Žanrid(žanr\_id)

)

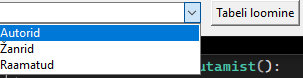
""")

else:

print("Vigane valik.")

connection.commit()

messagebox.showinfo("Edu", f"Tabel '{valitud\_tabel}' on edukalt loodud")



## Создание графического интерфейса

Используя Tkinter для создания графического интерфейса , я смогла добавить кнопки и функционировать их  
Примеры кодов и что они делают:

update\_frame и näidata\_andme\_frame - это фреймы, которые используются для группировки связанных элементов интерфейса.

autor\_nimi\_label и valitud\_autor\_nimi - создают выпадающий список для выбора имени автора.

uus\_nimi\_label и uus\_nimi\_entry - предоставляют текстовое поле для ввода нового имени автора.

update\_autor\_button - кнопка, которая вызывает функцию update\_autor\_nimi2 при нажатии.

Элементы интерфейса размещаются на экране с помощью метода pack, который автоматически выравнивает элементы по вертикали и горизонтали

def update\_autor\_nimi2():

autor\_nimi = valitud\_autor\_nimi.get()

uus\_nimi = uus\_nimi\_entry.get()

update\_autor\_nimi(connection, (uus\_nimi, autor\_nimi))

uus\_nimi\_entry.delete(0, END)

update\_frame = tk.Frame(root)

update\_frame.pack(pady=10)

autor\_nimi = [row[0] for row in execute\_read\_query(connection, "SELECT autor\_nimi FROM Autorid")]

näidata\_andme\_frame = tk.Frame(root)

näidata\_andme\_frame.pack(pady=10)

autor\_nimi\_label = tk.Label(näidata\_andme\_frame, text="Valige Autori nimi:")

autor\_nimi\_label.pack(side=tk.LEFT, padx=5)

valitud\_autor\_nimi = tk.StringVar()

autor\_nimi\_combobox = ttk.Combobox(näidata\_andme\_frame, textvariable=valitud\_autor\_nimi, values=autor\_nimi)

autor\_nimi\_combobox.pack(side=tk.LEFT, padx=5)

uus\_nimi\_label = tk.Label(näidata\_andme\_frame, text="Uus nimi:")

uus\_nimi\_label.pack(side=tk.LEFT, padx=5)

uus\_nimi\_entry = tk.Entry(näidata\_andme\_frame)

uus\_nimi\_entry.pack(side=tk.LEFT, padx=5)

update\_autor\_button = tk.Button(näidata\_andme\_frame, text="Uuenda autorinime", command=update\_autor\_nimi2)

update\_autor\_button.pack(side=tk.LEFT, padx=5)



Остальные обновления данных сделаны похожим образом

Добавление данных в базу данных

def add\_autor():

autor\_andmed\_entry = tk.Entry(root)

autor\_andmed\_entry.focus\_set()

autor\_andmed\_frame = tk.Toplevel(root)

autor\_andmed\_frame.title("Lisa autor")

tk.Label(autor\_andmed\_frame, text="Autori nimi:").grid(row=1, column=0)

autor\_nimi\_entry = tk.Entry(autor\_andmed\_frame)

autor\_nimi\_entry.grid(row=1, column=1)

tk.Label(autor\_andmed\_frame, text="Sünnipäev:").grid(row=2, column=0)

autor\_sünnipäev\_entry = tk.Entry(autor\_andmed\_frame)

autor\_sünnipäev\_entry.grid(row=2, column=1)

def add\_autor\_andmebaasi():

autor\_andmed = ( autor\_nimi\_entry.get(), autor\_sünnipäev\_entry.get())

add\_autor\_query(connection, autor\_andmed)

autor\_andmed\_frame.destroy()

add\_btn = tk.Button(autor\_andmed\_frame, text="Lisa autor", command=add\_autor\_andmebaasi)

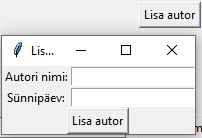
add\_btn.grid(row=3, column=0, columnspan=2)

add\_frame = tk.Frame(root)

add\_frame.pack(pady=10)

add\_autor\_btn = tk.Button(add\_frame, text="Lisa autor", command=add\_autor)

add\_autor\_btn.pack(side=tk.LEFT, padx=5)



Добавление жанров и книг сделаны похожим способом

Удаление данных из базы данных

def delete\_raamat\_pealkirja\_järgi():

uus\_window = tk.Toplevel(root)

uus\_window.title("Kustuta Raamat pealkirja järgi")

juhendamise = tk.Label(uus\_window, text="Sisestage kustutamiseks raamatu pealkiri:")

juhendamise.pack(pady=10)

global pealkiri\_entry

pealkiri\_entry = tk.Entry(uus\_window)

pealkiri\_entry.pack(pady=10)

def kinnitada\_kustutamist():

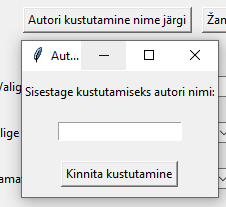
pealkiri = pealkiri\_entry.get()

delete\_raamat\_pealkiri(connection, pealkiri)

uus\_window.destroy()

kinnitusnupp = tk.Button(uus\_window, text="Kinnita kustutamine", command=kinnitada\_kustutamist)

kinnitusnupp.pack(pady=10)



Удаление жанров и книг сделано похожим образом

Просмотр таблиц

def table\_autorid(conn):

window\_autorid = tk.Toplevel()

window\_autorid.title("Autorite tabel")

tree = ttk.Treeview(window\_autorid, column=("autor\_id", "autor\_nimi", "sünnikuupäev"), show="headings")

tree.column("autor\_id", anchor=CENTER)

tree.heading("autor\_id", text="autor\_id")

tree.column("autor\_nimi", anchor=CENTER)

tree.heading("autor\_nimi", text="autor\_nimi")

tree.column("sünnikuupäev", anchor=CENTER)

tree.heading("sünnikuupäev", text="sünnikuupäev")

try:

read = execute\_read\_query(conn, "SELECT \* FROM Autorid")

for row in read:

tree.insert("", END, values=row)

except Exception as e:

print(f"Viga tabelis autorid: {e}")

tree.pack()

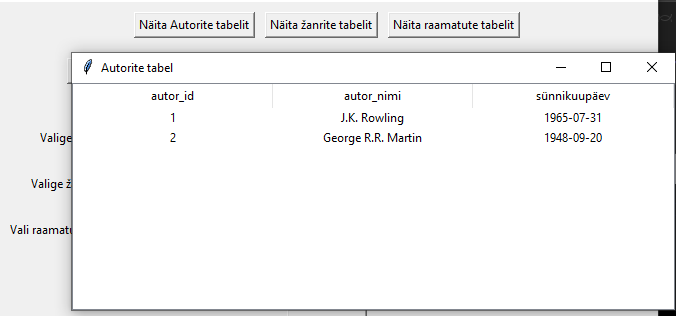
window\_autorid.mainloop()

näidata\_andme\_frame = tk.Frame(root)

näidata\_andme\_frame.pack(pady=10)

näidata\_autori\_btn = tk.Button(näidata\_andme\_frame, text="Näita Autorite tabelit", command=näita\_autorid\_tabelit)

näidata\_autori\_btn.pack(side=tk.LEFT, padx=5)



Просмотр таблиц с жанрами и книгами сделан похожим образом

Добавление дефолтных записей в таблицу

def insert\_initial\_data():

valitud\_tabel = algsed\_andmed\_combobox.get()

alg\_andmed\_päringud = {

"Autorid": [

"INSERT INTO Autorid(autor\_nimi, sünnikuupäev) VALUES ('J.K. Rowling', '1965-07-31')",

"INSERT INTO Autorid VALUES ('George R.R. Martin', '1948-09-20')"

],

"Žanrid": [

"INSERT INTO Žanrid VALUES ('Fantaasia')",

"INSERT INTO Žanrid VALUES ('Sci-Fi')"

],

"Raamatud": [

"INSERT INTO Raamatud VALUES ('Harry Potter ja filosoofi kivi', '1997-06-26', 1, 1)",

"INSERT INTO Raamatud VALUES ('Troonide mäng', '1996-08-01', 2, 2)"

]

}

if valitud\_tabel in alg\_andmed\_päringud:

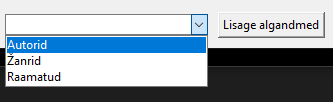
for query in alg\_andmed\_päringud[valitud\_tabel]:

execute\_query(connection, query)

messagebox.showinfo("Edu", f"Esialgsed andmed sisestati {valitud\_tabel} tabelisse edukalt")

else:

messagebox.showerror("Viga", "Kehtetu tabeli nimi")



Удаление таблицы по названию

def delete\_table\_clicked():

tabel\_nimi = tabel\_nimi\_entry.get()

query = f"DROP TABLE IF EXISTS {tabel\_nimi}"

delete\_tabel(connection, query)

window.destroy()

def kustuta\_tabeli\_liides():

global window

window = tk.Toplevel(root)

window.title("Kustuta Tabel")

tabel\_nimi\_label = tk.Label(window, text="Tabeli nimi:")

tabel\_nimi\_label.pack(pady=10)

global tabel\_nimi\_entry

tabel\_nimi\_entry = tk.Entry(window)

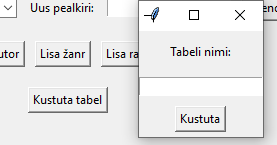
tabel\_nimi\_entry.pack(pady=5)

delete\_button = tk.Button(window, text="Kustuta", command=delete\_table\_clicked)

delete\_button.pack(pady=5)

delete\_table\_btn = tk.Button(root, text="Kustuta tabel", command=kustuta\_tabeli\_liides)

delete\_table\_btn.pack(pady=10)



# KASUTUSJUHEND

# KOKKUVÕTTE

# ALLIKAD

*SQLite*. (б.д.). Получено из workspace: https://workspace.ru/tools/database/sqllite/

*Tkinter*. (04 09 2023 г.). Получено из skillfactory: https://blog.skillfactory.ru/glossary/tkinter/

*Tkinter ttk (модуль)*. (б.д.). Получено из ts-python.fandom: https://ts-python.fandom.com/ru/wiki/Tkinter\_ttk\_(%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C)

*Что такое Python?* (2023). Получено из Amazon: https://aws.amazon.com/ru/what-is/python/

Шапошникова, С. (2024). *Модуль tkinter.ttk*. Получено из younglinux: https://younglinux.info/tkinter/ttk