АКАДЕМИЈА ТЕХНИЧКО-УМЕТНИЧКИХ СТРУКОВНИХ СТУДИЈА БЕОГРАД

ОДСЕК ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВА

**Вренчев Александар**

***Python* програм за евиденцију синдикалне продаје**

**- завршни рад -**



Београд, јул 2022.

Кандидат: **Вренчев Александар**

Број индекса: **НРТ-97/19**

Студијски програм: **Нове рачунарске технологије**

Тема: ***Python* програм за евиденцију синдикалне продаје**

Основни задаци:

**1.** Опис коришћених технологија: *Python, Tkinter, Sqlite3.*

**2.** Имплементација *Python* програма за евиденцију синдикалне продаје.

**3.** Опис корисничког интерфејса.

Ментор:

Београд, јул 2022 годинe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Перица Штрбац, професор ВИШЕР

Резиме:

Завршни рад из предмета Функционално програмирање на тему *Python* програм за евиденцију синдикалне продаје. Програм има могућности за унос,измену,брисање и филтрирање података . Такође програм је мултиплатформски, модуларан и преносив. У изради завршног рада коришћени су *Python* програмски језик и радни оквири *Tkinter* и *Sqlite3* и *Pyinstaller*.

**Кључне речи:** *Python*, *Tkinter*, *Sqlite3*, *Pyinstaller*

ABSTRACT:

Рад је посвећен аутоматизацији и вођењу евиденције у дигиталној форми што би требало да убрза и олакша рад са великом количином података. Такође овај рад би могао да послужи као пример добре праксе у даљем процесу дигитализације радних задатака и услуга у областима рада где се од радника са минималном информатичком писменошћу очекује вођење евиденције и рад са подацима клијената.

**Key words**: Рад, Евиденција, Дигитализација

САДРЖАЈ:

[1. УВОД 1](#_Toc108559616)

[2. Основни задаци завршног рада 2](#_Toc108559617)

[2.1. Опис коришћених технологија: Python, Tkinter, Sqlite3 2](#_Toc108559618)

[*2.1.1.* *Python* 2](#_Toc108559619)

[*2.1.2.* *Tkinter* 2](#_Toc108559620)

[*2.1.3.* *Sqlite3* 2](#_Toc108559621)

[2.2. Имплементација Python програма за евиденцију синдикалне продаје 3](#_Toc108559622)

[2.2.1. Списак команди за коришћење програма 11](#_Toc108559623)

[2.3. Опис корисничког интерфејса 12](#_Toc108559624)

[3. Инсталација и дистрибуција програма 14](#_Toc108559625)

[*3.1.1.* *Pyinstaller* 14](#_Toc108559626)

[6. ЗАКЉУЧАК 17](#_Toc108559629)

[9. ИНДЕКС ПОЈМОВА 20](#_Toc108559632)

[12. ЛИТЕРАТУРА 23](#_Toc108559635)

[15. изјава о академскoј честитости 26](#_Toc108559638)

# УВОД

Посматрањем процеса рада у продаји риболовачке опреме приметио сам да се пуно времена губи на рад са папирологијом, на пример административне забране, синдикална продаја, чекови грађана, требовање робе, итд. Бавећи се административним пословима радници немају довољно времена за рад са муштеријама. Тако сам дошао на идеју да им помогнем својим информатичким знањем и направио сам програм за евиденцију синдикалне продаје. Будући корисници били су укључени у цео процес писања софтвера, што укључује пројектовање програмске логике, пројектовање базе податка, израду корисничког интерфејса, тестирање софтвера као и планирање даљег развоја софтвера.

Поглавље под насловом основни задаци завршног рада садржи опис коришћених технологија што укључује *Python*, *Tkinter* и *Sqlite3* као иимплементацију *Python* програма за евиденцију синдикалне продаје. Такође ово поглавље садржи и опис корисничког интерфејса у којем су на једноставан начин описане опције и могућности за употребу као и списак команди које су корисницима на располагању. Читаоцима је на располагању и табела која описује функционалност сваког дугмета у програму као и слике које приказују кориснички интерфејс у целини као и поруке о грешци.

Поглавље под насловом инсталација и дистрибуција програма укратко описује разлог употребе као и начин рада модула под називом *Pyinstaller* који је употребљен за паковање, дистрибуцију и инсталацију програма. *Pyinstaller* представља добар избор пре свега због једноставности употребе, али и због великог избора опција које пружа у раду са скриптама написаним на програмском језику *Python*.

С обзиром на то да живим у граду који има велики број синдиката како у приватним тако и у јавним предузећима, израда и коришћење једног оваквог софтвера свакако има смисла. Основну логику програма тестирао сам прво уз помоћ програма *Excel* са реалним подацима иуз помоћ функција које овај софтвер поседује, а касније сам користећи *Tkinter* и *Sqlite3* направио кориснички интерфејс и базу података који су модуларни, преносиви и веома једноставни за коришћење. За дистрибуцију програма користио сам библиотеку *Python* програмског језика под називом *Pyinstaller*.

# Основни задаци завршног рада

Основни задаци завршног рада обухватају опис коришћених технологија, имплементацију програма за евиденцију синдикалне продаје као и опис корисничког интерфејса и свих његових елемената.

## Опис коришћених технологија: Python, Tkinter, Sqlite3

За израду пројекта одлучио сам се за коришћење *Python* програмског језика и његових радних оквира *Tkinter* и *Sqlite3* као и пакета *Pyinstaller*. У наставку следи опис коришћених технологија.

### *Python*

*Python* је програмски језик високог нивоа опште намене и представља један од најпопуларнијих програмских језика у областима као што су машинско учење, развој вештачке интелигенције, анализа и визуелизација велике количине података, аутоматизација задатака, развој интернет апликација као и развој апликација са графичким корисничким интерфејсом. Подржава пре свега објектно оријентисан и функционални стил програмирања, али структурно програмирање. Синтакса језика *Python* омогућава писање веома прегледних програма. Језик се брзо и лако учи. За програме писане на *Python* програмском језику потребан је интерпретер који служи за превођење наших скрипти на језик разумљив компјутерском хардверу. Превођење се дешава сваки пут када покренемо скрипту. Уз интерпретер *Python* има и веома развијену бибилиотеку стандардних модула. Филозофија овог програмског језика почива на великој читљивости програмског кода као и на стриктном поштовању увлачења редова написаног кода. Аутор овог програмског језика је Гвидо ван Росум са универзитета Стичинг у Холандији.

### *Tkinter*

*Tkinter* је радни оквир *Python* програмског језика који служи за израду графичког корисничког интерфејса за десктоп апликације уз помоћ елемената као што су дугмад, поља за унос података и лабеле за приказ података. Назив *Tkinter* је скраћеница за Тк интерфејс. *Tkinter* је мултиплатформски радни оквир што значи да подједнако добро ради на свим оперативним системима. За *Tkinter* није потребна инсталација јер се овај радни оквир дистрибуира као део стандардног *Python* пакета. Аутори овог бесплатног софтвера су Гвидо ван Росум и Стин Лумхолт.

### *Sqlite3*

*SQLite* је библиотека програмског језика С која омогућава рад са базом података без употребе посебних сервера а *Sqlite3* је модул који функционише као интерфејс између *Python* програмског језика и базе података. За употребу *Sqlite3* модула потребно је направити објекат конекције који представља базу података и курсор објекат који омогућава извршавање *SQL* наредби. За *Sqlite3* није потребна инсталација јер се овај модул дистрибуира као део стандардног *Python* пакета . Аутор *Sqlite3* је Герхард Херинг.

## Имплементација Python програма за евиденцију синдикалне продаје

Основу програма чине две скрипте од којих једна служи за почетну иницијализацију односно прављење базе података и друга која садржи програмску логику. Употреба скрипте за иницијализацију базе података није неопходна у случају када се извршна датотека дистрибуира заједно са већ постојећом базом података. Изменом друге скрипте мења се функционалност програма.

import sqlite3 # imports  
  
class Database: # making database class and setting up cursor  
 def \_\_init\_\_(self, db\_sindikalna):  
 self.conn = sqlite3.connect(db\_sindikalna)  
 self.cur = self.conn.cursor()  
 self.cur.execute(  
 "CREATE TABLE IF NOT EXISTS sindikalci (id INTEGER PRIMARY KEY, sindikat text, ime text, prezime text, ukupan\_iznos int, uplaceno int, broj\_rata int,pojedinacna\_rata int, datum\_prve\_rate text)")  
 self.conn.commit()  
  
 def fetch(self): # selecting all rows and columns from database  
 self.cur.execute("SELECT \* FROM sindikalci")  
 rows = self.cur.fetchall()  
 return rows

# function for inserting new values in database

def insert(self, sindikat, ime, prezime, ukupan\_iznos, uplaceno, broj\_rata,pojedinacna\_rata, datum\_prve\_rate):  
 self.cur.execute("INSERT INTO sindikalci VALUES (NULL, ?, ?, ?, ?, ?, ?, ?, ?)",  
 (sindikat, ime, prezime, ukupan\_iznos, uplaceno, broj\_rata,pojedinacna\_rata, datum\_prve\_rate))  
 self.conn.commit()  
#function for removing values from database  
def remove(self, id):  
 self.cur.execute("DELETE FROM sindikalci WHERE id=?", (id,))  
 self.conn.commit()  
#function for updating values in database  
def update(self, id, sindikat, ime, prezime, ukupan\_iznos, uplaceno, broj\_rata,pojedinacna\_rata, datum\_prve\_rate):  
 self.cur.execute("UPDATE sindikalci SET sindikat = ?, ime = ?, prezime = ?, ukupan\_iznos = ?, uplaceno = ?, broj\_rata = ?, pojedinacna\_rata = ?, datum\_prve\_rate = ? WHERE id = ?",  
 (sindikat, ime, prezime, ukupan\_iznos, uplaceno, broj\_rata,pojedinacna\_rata, datum\_prve\_rate, id))  
 self.conn.commit()  
  
def \_\_del\_\_(self):  
 self.conn.close()

Скрипта за иницилализацију базе података.

# imports #

from tkinter import \*

from tkinter import messagebox

import re

from db\_sindikalna import Database

from datetime import \*

from dateutil.relativedelta import relativedelta

import tkinter.font as tkFont

# initialize database #

db\_sindikalna = Database('sindikalna.db')

# functions #

def set\_number\_of\_rates():

regex\_check()

pojedinacna\_rata\_text.set(int((ukupan\_iznos\_text.get() - uplaceno\_text.get()) / broj\_rata\_text.get()))

def regex\_check(): # function for checking data validity #

re\_sindikat\_entry = re.match("^[A-Z][-a-zA-Z]+$", sindikat\_entry.get())

# if not re\_sindikat\_entry:

if sindikat\_entry.get().\_\_len\_\_() < 3 or sindikat\_entry.get().\_\_len\_\_() > 30:

messagebox.showinfo(message="Polje 'Sindikat' je prazno ili nije odgovarajuće dužine.")

return False

re\_ime\_entry = re.match("^[A-Z][-a-zA-Z]+$", ime\_entry.get())

# if not re\_ime\_entry:

if ime\_entry.get().\_\_len\_\_() < 3 or ime\_entry.get().\_\_len\_\_() > 30:

messagebox.showinfo(message="Polje 'Ime' je prazno,počinje brojem ili nije odgovarajuće dužine.")

return False

re\_prezime\_entry = re.match("^[A-Z][-a-zA-Z]+$", prezime\_entry.get())

# if not re\_prezime\_entry:

if ime\_entry.get().\_\_len\_\_() < 3 or ime\_entry.get().\_\_len\_\_() > 30:

messagebox.showinfo(message="Polje 'Prezime' je prazno,počinje brojem ili nije odgovarajuće dužine.")

return False

re\_ukupan\_iznos\_entry = re.match("^[1-9]+[0-9]\*$", ukupan\_iznos\_entry.get())

if not re\_ukupan\_iznos\_entry:

messagebox.showinfo(message="Polje 'Ukupan iznos' nije popunjeno ili ne sadrži isključivo pozitivne brojeve.")

return False

re\_uplaceno\_entry = re.match(r"[0-9]{1,7}", uplaceno\_entry.get())

if not re\_uplaceno\_entry:

messagebox.showinfo(message="Polje 'Uplaćeno pri kupovini' nije popunjeno ili ne sadrži isključivo pozitivne brojeve.")

return False

re\_datum\_prve\_rate\_entry = re.match(r"^20[0-2][0-9]-((0[1-9])|(1[0-2]))-(0[1-9]|[1-2][0-9]|3[0-1])$", datum\_prve\_rate\_entry.get())

if not re\_datum\_prve\_rate\_entry:

messagebox.showinfo(message="Polje 'Datum prve rate' nije popunjeno ili nije upisano u ispravnom formatu.Primer 2022-06-24.")

return False

re\_broj\_rata\_entry = re.match("^(1[0-2]|[1-9])$", broj\_rata\_entry.get())

if not re\_broj\_rata\_entry:

messagebox.showinfo(message="Polje 'Broj rata' nije popunjeno ili nije u opsegu 1-12.")

return False

# if int(broj\_rata\_entry.get()) \* int(float(pojedinacna\_rata\_entry.get())) != int(ukupan\_iznos\_entry.get()) - int(uplaceno\_entry.get()):

# messagebox.showinfo(message="Broj rata pomnozen sa iznosom pojedinacnih rata nije jednak ukupnom iznosu za placanje umanjenom za iznos uplacen pri kupovini")

# return False

if int(ukupan\_iznos\_entry.get()) < int(uplaceno\_entry.get()):

messagebox.showinfo(message="Iznos uplacen pri kupovini ne moze biti veci od ukupnog iznosa.")

return False

return True

def populate\_list(): # function for populating list with items #

global counter

counter = 0

parts\_list.delete(0, END)

for row in db\_sindikalna.fetch():

# parts\_list.insert(END, "ID:"+str(row[0])+" Sindikat:"+row[1]+" Ime:"+row[2]+" Prezime:"+ str(row[3])+ " Ukupno:"+str(row[4])+" Uplaceno pri kupovini"+str(row[5])+" Br.rata:"+str(row[6])+" Iznos svake rate:"+str(row[7]))

parts\_list.insert(END, "ID: "+str(row[0])+" "+row[2]+" "+ str(row[3]))

# parts\_list.insert(END, "{:>3}{:<5}".format("ID:",str(row[0]))+"{:>12}{:<30}".format("Sindikat: ",row[1])+"{:>5}{:<30}".format("ime",row[2])+ "{:>10}{:<30}".format("Prezime:",str(row[3])))

def populate\_list\_date():# function for populating list based on date #

global counter

counter = 1

parts\_list.delete(0, END)

for row in db\_sindikalna.fetch():

first\_rate\_date = datetime.strptime(row[8], "%Y-%m-%d").date()

for i in range(row[6]):

if first\_rate\_date + relativedelta(months=i) >= datetime.now().date():

# naplata = str(row[1])+" "+row[2]+" "+row[3]+" Ukupno: "+str(row[4])+" Uplaceno: "+str(row[5])+" Datum prve rate: "+str(row[8])+" Br.rata: "+str(row[6])+" iznos svake rate: "+str(row[7])+" redni broj rate: "+str(i+1)+" sledeca rata: "+str(first\_rate\_date + relativedelta(months=i))

parts\_list.insert(END,"ID: " + str(row[0]) +' Ime: '+row[2]+ ' Prezime: '+row[3] + " Rata broj: "+ str(i+1) +" stize na naplatu: "+str(first\_rate\_date + relativedelta(months=i)))

break

# change color for every row #

for i in range(100):

if parts\_list.get(i):

parts\_list.itemconfig(i,{'fg': 'red'})

def populate\_list\_search(): # function that populate list based on search #

global counter

counter = 2

parts\_list.delete(0, END)

if sindikat\_entry.get() == '' and ime\_entry.get() =='' and prezime\_entry.get() =='':

messagebox.showinfo(message='Polja za pretragu su prazna!')

parts\_list.insert(END, 'Upišite pojam za pretragu u polje za naziv sindikata ili u polje za ime ili prezime!')

for row in db\_sindikalna.fetch():

if row[1] == sindikat\_entry.get() and options\_variable.get() == 'Sindikat' or row[2] == ime\_entry.get() and options\_variable.get() == 'Ime' or row[3] == prezime\_entry.get() and options\_variable.get() == 'Prezime':

parts\_list.insert(END, "ID: "+str(row[0])+" "+row[2]+" "+ str(row[3]))

def add\_item():

if regex\_check():

db\_sindikalna.insert(sindikat\_text.get(), ime\_text.get(),

prezime\_text.get(), ukupan\_iznos\_text.get(),

uplaceno\_text.get(),broj\_rata\_text.get(),

pojedinacna\_rata\_text.get(),datetime.strptime(datum\_prve\_rate\_text.get(), '%Y-%m-%d').date())

parts\_list.delete(0, END)

parts\_list.insert(END, (sindikat\_text.get(), ime\_text.get(),prezime\_text.get(), ukupan\_iznos\_text.get(), uplaceno\_text.get(), broj\_rata\_text.get(), pojedinacna\_rata\_text.get(), datum\_prve\_rate\_text.get()))

clear\_text()

populate\_list()

def select\_item(event):# function with event for selecting item from list #

try:

global selected\_item

global counter

index = parts\_list.curselection()[0]

lista = []

lista\_date = []

lista\_search = []

if counter == 0:

for row in db\_sindikalna.fetch():

lista.append(row)

# selected\_item = lista[index+counter] #

selected\_item = lista[index]

sindikat\_entry.delete(0, END)

sindikat\_entry.insert(END, selected\_item[1])

ime\_entry.delete(0, END)

ime\_entry.insert(END, selected\_item[2])

prezime\_entry.delete(0, END)

prezime\_entry.insert(END, selected\_item[3])

ukupan\_iznos\_entry.delete(0, END)

ukupan\_iznos\_entry.insert(END, selected\_item[4])

uplaceno\_entry.delete(0, END)

uplaceno\_entry.insert(END, selected\_item[5])

broj\_rata\_entry.delete(0, END)

broj\_rata\_entry.insert(END, selected\_item[6])

pojedinacna\_rata\_entry.delete(0, END)

pojedinacna\_rata\_entry.insert(END, selected\_item[7])

datum\_prve\_rate\_entry.delete(0, END)

datum\_prve\_rate\_entry.insert(END, selected\_item[8])

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

if counter == 1:

for row in db\_sindikalna.fetch():

first\_rate\_date = datetime.strptime(row[8], "%Y-%m-%d").date()

for i in range(row[6]):

if first\_rate\_date + relativedelta(months=i) >= datetime.now().date():

lista\_date.append(row)

break

# selected\_item = lista[index+counter] #

selected\_item = lista\_date[index]

sindikat\_entry.delete(0, END)

sindikat\_entry.insert(END, selected\_item[1])

ime\_entry.delete(0, END)

ime\_entry.insert(END, selected\_item[2])

prezime\_entry.delete(0, END)

prezime\_entry.insert(END, selected\_item[3])

ukupan\_iznos\_entry.delete(0, END)

ukupan\_iznos\_entry.insert(END, selected\_item[4])

uplaceno\_entry.delete(0, END)

uplaceno\_entry.insert(END, selected\_item[5])

broj\_rata\_entry.delete(0, END)

broj\_rata\_entry.insert(END, selected\_item[6])

pojedinacna\_rata\_entry.delete(0, END)

pojedinacna\_rata\_entry.insert(END, selected\_item[7])

datum\_prve\_rate\_entry.delete(0, END)

datum\_prve\_rate\_entry.insert(END, selected\_item[8])

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

if counter == 2:

for row in db\_sindikalna.fetch():

if row[1] == sindikat\_entry.get() and options\_variable.get() == 'Sindikat' or row[2] == ime\_entry.get() and options\_variable.get() == 'Ime' or row[3] == prezime\_entry.get() and options\_variable.get() == 'Prezime':

lista\_search.append(row)

selected\_item = lista\_search[index]

sindikat\_entry.delete(0, END)

sindikat\_entry.insert(END, selected\_item[1])

ime\_entry.delete(0, END)

ime\_entry.insert(END, selected\_item[2])

prezime\_entry.delete(0, END)

prezime\_entry.insert(END, selected\_item[3])

ukupan\_iznos\_entry.delete(0, END)

ukupan\_iznos\_entry.insert(END, selected\_item[4])

uplaceno\_entry.delete(0, END)

uplaceno\_entry.insert(END, selected\_item[5])

broj\_rata\_entry.delete(0, END)

broj\_rata\_entry.insert(END, selected\_item[6])

pojedinacna\_rata\_entry.delete(0, END)

pojedinacna\_rata\_entry.insert(END, selected\_item[7])

datum\_prve\_rate\_entry.delete(0, END)

datum\_prve\_rate\_entry.insert(END, selected\_item[8])

except IndexError:

pass

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

def remove\_item(): # function for removing data from database #

remove\_boolean = messagebox.askyesno(message="Da li zaista želite da obrišete izabrani podatak?")

if remove\_boolean:

db\_sindikalna.remove(selected\_item[0])

clear\_text()

populate\_list()

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

def update\_item(): # function for updating data in database #

if regex\_check():

db\_sindikalna.update(selected\_item[0], sindikat\_text.get(), ime\_text.get(),

prezime\_text.get(), ukupan\_iznos\_text.get(),uplaceno\_entry.get(),

broj\_rata\_text.get(), pojedinacna\_rata\_text.get(), datum\_prve\_rate\_text.get())

populate\_list()

def clear\_text(): # function for clearing entry fields #

sindikat\_entry.delete(0, END)

ime\_entry.delete(0, END)

prezime\_entry.delete(0, END)

ukupan\_iznos\_entry.delete(0, END)

broj\_rata\_entry.delete(0, END)

datum\_prve\_rate\_entry.delete(0, END)

pojedinacna\_rata\_entry.delete(0, END)

uplaceno\_entry.delete(0, END)

# Create window object #

app = Tk()

# Labels and Entry fields #

# Union StringVar, Label and Entry #

sindikat\_text = StringVar()

sindikat\_label = Label(app, text='Sindikat', font=('bold', 14))

sindikat\_label.grid(row=2, column=0, sticky=W, pady=(100,10), padx=(50,5))

sindikat\_entry = Entry(app, textvariable=sindikat\_text, width=20)

sindikat\_entry.grid(row=2, column=1, pady=(100,10), padx=30)

# First name StringVar, Label and Entry #

ime\_text = StringVar()

ime\_label = Label(app, text='Ime', font=('bold', 14))

ime\_label.grid(row=3, column=0, sticky=W, pady=(0,10), padx=(50,5))

ime\_entry = Entry(app, textvariable=ime\_text)

ime\_entry.grid(row=3, column=1, pady=(0,10), padx=30)

# Last name StringVar, Label and Entry #

prezime\_text = StringVar()

prezime\_label = Label(app, text='Prezime', font=('bold', 14))

prezime\_label.grid(row=4, column=0, sticky=W, pady=(0,10), padx=(50,5))

prezime\_entry = Entry(app, textvariable=prezime\_text)

prezime\_entry.grid(row=4, column=1, pady=(0,10), padx=30)

# Total amount IntVar and Label #

ukupan\_iznos\_text = IntVar()

ukupan\_iznos\_label = Label(app, text='Ukupan iznos', font=('bold', 14))

ukupan\_iznos\_label.grid(row=5, column=0, sticky=W, pady=(0,10), padx=(50,5))

ukupan\_iznos\_entry = Entry(app, textvariable=ukupan\_iznos\_text)

ukupan\_iznos\_entry.grid(row=5, column=1, pady=(0,10), padx=30)

# Label and Entry for the amount paid at the time of purchase

uplaceno\_label = Label(app, text='Uplaćeno pri kupovini', font=('bold', 14))

uplaceno\_label.grid(row=6, column=0, sticky=W, pady=(0,10), padx=(50,5))

uplaceno\_text = IntVar()

uplaceno\_entry = Entry(app, textvariable=uplaceno\_text)

uplaceno\_entry.grid(row=6, column=1, pady=(0,10), padx=30)

# Date for first rate in format yyyy – mm - dd

datum\_prve\_rate\_text = StringVar()

datum\_prve\_rate\_label = Label(app, text='Datum prve rate(format:yyyy-mm-dd)', font=('bold', 14))

datum\_prve\_rate\_label.grid(row=7, column=0, sticky=W, pady=(0,10), padx=(50,5))

datum\_prve\_rate\_entry = Entry(app, textvariable=datum\_prve\_rate\_text)

datum\_prve\_rate\_entry.grid(row=7, column=1, pady=(0,10), padx=50)

# Number of rates

BROJEVI = list(range(1,13))

broj\_rata\_text = IntVar()

broj\_rata\_label = Label(app, text='Broj rata', font=('bold', 14))

broj\_rata\_label.grid(row=8, column=0, sticky=W, pady=(0,10), padx=(50,5))

broj\_rata\_entry = Entry(app, textvariable=broj\_rata\_text)

broj\_rata\_entry.grid(row=8, column=1, sticky=W, pady=(0,10), padx=(50,5))

# Label and Entry for value of single rate

pojedinacna\_rata\_label = Label(app, text='Iznos pojedinačnih rata', font=('bold', 14))

pojedinacna\_rata\_label.grid(row=9, column=0, sticky=W, pady=(0,10), padx=(50,5))

pojedinacna\_rata\_text = IntVar()

pojedinacna\_rata\_entry = Entry(app, textvariable=pojedinacna\_rata\_text)

pojedinacna\_rata\_entry.grid(row=9, column=1, pady=(0,10), padx=30)

izracunaj\_rate\_btn = Button(app, text='Izračunaj', width=12, command=set\_number\_of\_rates)

izracunaj\_rate\_btn.grid(row=10, column=1)

# Parts List (Listbox)

# small\_font = tkFont.Font(size=10)

parts\_list = Listbox(app, height=15, width=80, border=0)

parts\_list.grid(row=2, column=5, columnspan=6, rowspan=6, pady=(100,0), padx=(0,0))

# Create scrollbar #

scrollbar = Scrollbar(app, orient='vertical')

scrollbar.grid(row=3, column=11, sticky=NS)

# Set scroll to listbox #

parts\_list.configure(yscrollcommand=scrollbar.set)

scrollbar.configure(command=parts\_list.yview)

# Bind select #

parts\_list.bind('<<ListboxSelect>>', select\_item)

# Buttons #

# # Button for adding data to database #

add\_btn = Button(app, text='Dodaj podatak', width=12, command=add\_item)

add\_btn.grid(row=9, column=5)

# Button for deleting data from database #

remove\_btn = Button(app, text='Obriši podatak', width=12, command=remove\_item)

remove\_btn.grid(row=9, column=6)

# Button for updating data in database #

update\_btn = Button(app, text='Izmeni podatak', width=12, command=update\_item)

update\_btn.grid(row=10, column=6)

# Button for clearing entry fields #

clear\_btn = Button(app, text='Obriši sva polja', width=12, command=clear\_text)

clear\_btn.grid(row=9, column=8)

# Option menu with StringVar and OptionMenu #

search\_options = ['Sindikat', 'Ime', 'Prezime']

options\_variable = StringVar()

options\_variable.set('Pojam za pretragu')

search\_option\_menu = OptionMenu(app, options\_variable, \*search\_options)

search\_option\_menu.grid(row=10, column=7)

# Button for changing data #

charge\_btn = Button(app, text='Uskoro za naplatu', width=15, command=populate\_list\_date)

charge\_btn.grid(row=9, column=7)

# Button for showing data #

show\_btn = Button(app, text='Svi podaci', width=12, command=populate\_list)

show\_btn.grid(row=10, column=5)

# Button for search #

search\_btn = Button(app, text='Pretraga', width=12, command=populate\_list\_search)

search\_btn.grid(row=10, column=8)

# app title #

app.title('Sindikalna prodaja')

app.geometry('1100x500')

# Populate data #

populate\_list()

# Start program #

app.mainloop()

Скрипта која садржи програмску логику.

Скрипта за израду базе података користи се само једанпут и то пре прве употребе програма уколико са програмом није дистрибуирана и база података, друга скрипта служи за измену програма а извршна датотека се користи за употребу програма.

### Списак команди за коришћење програма

У наставку следи табела 1.1 са кратким описом команди које корисницима стоје на располагању и које програм може да изврши.

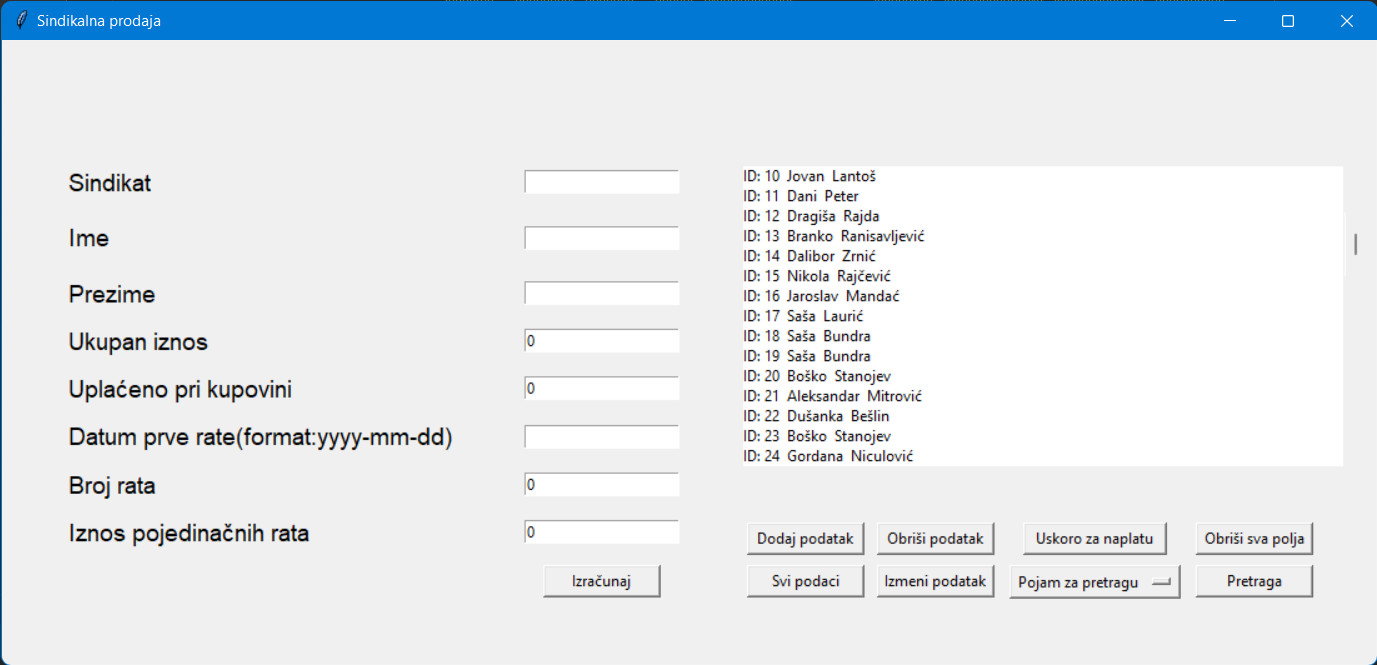
|  |  |
| --- | --- |
| Команда | Акција |
| Сви подаци | Приказ свих података |
| Додај податак | Додавање новог податка у базу |
| Обриши податак | Брисање податка из базе |
| Измени податак | Измена податка из базе |
| Израчунај | Рачунање износа појединачних рата |
| Ускоро за наплату | Филтрирање података на основу приспећа рате |
| Обриши сва поља | Брисање свих поља за унос података |
| Претрага | Филтрирање на основу изабраног критеријума |
| Појам за претрагу | Избор критеријума за претрагу |

Табела 1.1 – Списак команди.

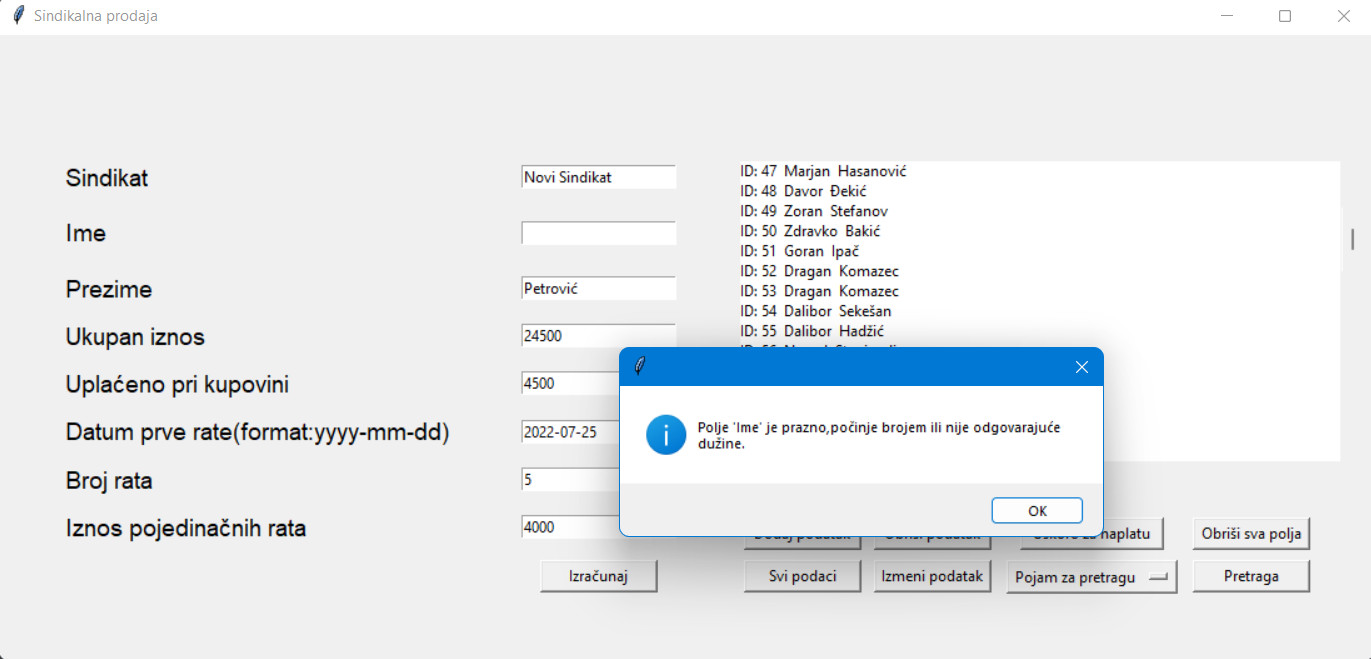


## Опис корисничког интерфејса

Кориснички интерфејс се састоји од поља за унос података, лабела које описују тип података за унос, дугмади за рачунање износа појединачних рата, унос, измену, брисање и филтрирање података као и поља за селекцију и приказ изабраних података. У случају неуношења тражених података или у случају уноса података у погрешном формату кориснички интерфејс приказује упозорење и поруку о грешци. Провера се врши за свако поље за унос података посебно тако што се проверава да ли су поља празна, да ли су унети подаци у одговарајућем опсегу као и да ли су у исправном формату. Слике 1.1 и 1.2 приказују изглед корисничког интерфејса и проверу уноса података са упозорењем и приказом одговарајуће поруке. Програм пружа могућност за унос, измену, брисање и филтрирање података. Филтрирање се врши по једном од предложених критеријума, на пример филтрирање података по презимену или по имену корисника или филтрирање по називу синдиката, а постоји и посебно дугме које филтрира и приказује податке на основу датума исплате следеће рате за сваког корисника. Филтрирање корисника на основу датума следеће рате је веома корисно јер пружа увид у стање платног биланса који се очекује у наредном периоду. Постоји и дугме које служи за брисање свих података унетих у поља за унос, што је веома корисно при уносу великог броја клијената у кратком временском периоду али и при филтрирању података. Дугме за рачунање израчунава износ појединачних рата тако што од укупног износа одузима износ уплаћен при куповини и остатак дели са уписаним бројем рата на једнаке делове, такође функционалност која стоји иза овог дугмета проверава да ли се унети подаци слажу са формулом за израчунавање у случају када корисник накнадно промени број рата или неку другу бројну вредност која се користи у рачунању појединачних износа за сваку рату.



Слика 1.1 Изглед корисничког интерфејса.



Слика 1.2 Провера унетих података и порука о грешци.

# Инсталација и дистрибуција програма

За инсталацију и дистрибуцију програма користио сам *Pyinstaller* пакет програмског језика *Python* који омогућава веома лако и веома брзо паковање, дистрибуцију и инсталацију програма израдом извршне датотеке на свим оперативним системима који имају инсталиран *Python*, што значи да је *Python* програм за евиденцију синдикалне продаје модуларан, преносив и мултиплатформски програм.

### *Pyinstaller*

*Pyinstaller* чита *Python* скрипту коју сте написали, затим је анализира и тражи модуле и билиотеке које сте употребили, а затим копира све неопходне датотеке укључујући и *Python* интерпретер и све библиотеке, модуле и остале датотеке неопходне за рад вашег програма у један директоријум заједно са извршном датотеком вашег програма. Покретањем *pyinstaller* команде настају два директоријума под називом build и dist. Први од наведених директоријума чува датотеке које су неопходне за рад програма, а директоријум под називом dist садржи извршну датотеку вашег програма, дакле програм се сада покреће преко извршне датотеке или преко пречице до извршне датотеке као што је то углавном случај после инсталације десктоп апликације. Поред многих опција које *Pyinstaller* нуди издвојио бих две које сам употребио у изради програма, а то су опција –onefile за паковање програма у једну извршну датотеку и опција –noconsole за искључивање приказа конзолног прозора у режиму рада програма са графичким корисничким интерфејсом. Од осталих доступних опција као веома корисне издвојио бих --help за приказ поруке за помоћ, --version за приказ верзије програма и –distpath DIR за избор дестинације за dist директоријум.



# ЗАКЉУЧАК

*Python* програм за евиденцију синдикалне продаје је због једноставности употребе и свих предности које пружа у раду са великом количином података већ ушао у употребу и активно се користи. Када је у питању унапређење и даљи развој овог софтвера, постоје различите опције као на пример извоз извештаја у *Excel* формат, увоз података из *Excel* формата, избор различитих износа за сваку рату појединачно, слање података електронском поштом, архивирање података као и тестирање због побољшања перформанси и безбедности података. Што се тиче тестирања пре свега би требало тестирати унос велике количине података и унос различитих типова података, такође тестирање једноставности употребе корисничког интерфејса би свакако имало смисла. Једна од могућности за даљи развој и унапређење програма за евиденцију синдикалне продаје свакако би било и увођење могућности избора између латиничног и ћириличног писма као и могућност избора језика свих текстуалних елемената, на пример на енглеском, немачком, италијанском или неком другом језику. Радећи на овом пројекту имао сам прилику да проширим своје знање из многих области програмирања као што су планирање и пројектовање, форматирање и конверзија података, као и тестирање софтвера уз помоћ реалних података и у сарадњи са будућим корисницима. Такође имао сам прилику да научим како се пише документација једног академског рада, што ће ми свакако бити од користи у будућности.



# ИНДЕКС ПОЈМОВА

1.Python 1,2,3,13,14

2.*Tkinter* 1,2,4

3.*Sqlite3* 1,2,3

4. кориснички интерфејс 1,11

5. база података 10

6. *Pyinstaller* 1,13

7. програмски језик 2

8. дистрибуција 1,13

9. табела 11

10. слика 11,12

11. Excel 1,14

12. модуларан 13

13. преносив 1,13

14. мултиплатформски 13

15. скрипта 3,10

16. инсталација 1,13

17. извршна датотека 10

18. лабела 11

19. дугмад 1,2,11

20. пакет 13

21. форматирање 14

22. конверзија 14

23. архивирање 14

24. унапређење 14



# ЛИТЕРАТУРА

[1] Steven Lott: "Functional Python Programming", Packt Publishing, 2015.

[2]Званична документација Python програмског језика <https://docs.python.org/3/> Јул 2022.

[3]Traversy Media Desktop GUI App With Python and Tkinter shorturl.at/hCKTY Jul 2022.

[4]w3schools python tutorial <https://www.w3schools.com/python/> Jul 2022.

[5]Python GUI programming with tkinter <https://realpython.com/python-gui-tkinter/> Jul 2022.

[6]Pyinstaller Manual <https://pyinstaller.org/en/stable/> Jul 2022.

[7]Tutorialspoint SQLite – Python [https://www.tutorialspoint.com/sqlite/sqlite\_python.htm Jul 2022](https://www.tutorialspoint.com/sqlite/sqlite_python.htm%20Jul%202022).

[8]Python SQLite tutorial using sqlite3 <https://pynative.com/python-sqlite/> Jul 2022.

[9]Python – GUI Programming (Tkinter) <https://www.tutorialspoint.com/python/python_gui_programming.htm> Jul 2022.

# изјава о академскoј честитости

**ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

|  |  |
| --- | --- |
| **Студент (име, име једног родитеља и презиме):** | Александар,Никола,Вренчев |
| **Број индекса:** | НРТ-97/19 |

Под пуном моралном, материјалном, дисциплинском и кривичном одговорношћу изјављујем да је завршни рад, под насловом:

Python програм за евиденцију синдикалне продаје

1. резултат сопственог истраживачког рада;
2. да овaj рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
3. да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
4. да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
5. да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
6. да сам свестан/свесна да је плагијат коришћење туђих радова у било ком облику (као цитата, прафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, вебсајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима), као и других закона и одговарајућих аката Високе школе електротехнике и рачунарства струковних студија у Београду;
7. да је електронска верзија овог рада идентична штампаном примерку овог рада и да пристајем на његово објављивање под условима прописаним актима Високе школе електротехнике и рачунарства струковних студија у Београду;
8. да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, \_\_.\_\_.202\_. године Својеручни потпис студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_