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

2. Основни задаци завршног рада 2

3. Опис коришћених технологија: Python, Tkinter, Sqlite3 3

*3.1. Python* 3

*3.2. Tkinter* 3

*3.3. Sqlite3* 3

4. Имплементација Python програма за евиденцију синдикалне продаје 5

4.1. Списак команди за коришћење програма 14

5. Опис корисничког интерфејса 15

6. Инсталација и дистрибуција програма 17

*6.1.* *Pyinstaller* 17

7. ЗАКЉУЧАК 18

8. ИНДЕКС ПОЈМОВА 19

9. ЛИТЕРАТУРА 21

10. изјава о академскoј честитости 22

# 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*. Коришћењем поменутог модула добија се једна извршна датотека.

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* због једноставности у раду коју пружа као и због великог броја различитих опција које програмерима стоје на располагању.

# 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. У наставку ће на веома детаљан и свеобухватан начин бити појединачно описане све технологије коришћене у изради пројекта, мотивација за њихову употребу као и поступак имплементације свих појединачних делова програма у једну функционалну целину. Такође биће веома детаљно описани сви елементи који су део графичког корисничког интерфејса и њихове функције у раду програма. Поступак имплементације је веома детаљно објашњен кроз опис делова написаног кода као и свих функција и елемената корисничког интерфејса. Све функције и елементи су набројани и детаљно је разјашњена њихова функција у програму, такође постоји и табела која укратко описује акције које корисник може да изведе у току коришћења програма. Описани су и типови података који се очекују да их корисник унесе као и подаци које корисник може кроз употребу програма да добије.

## 3. 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* због свих предности које пружају у раду са скриптама написаним на *Python* програмском језику и у изради пројеката мањег обима. У наставку следи детаљнији опис технологија коришћених у изради пројекта.

### 3.1 *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 и структурно програмирање. Синтакса језика *Python* омогућава писање веома прегледних програма што је веома погодно за програмере почетнике и људе са мало програмерског искуства. Језик се брзо и лако учи и због своје прегледности и велике читљивости могућа је релативно лака разумљивост написаног програмског кода. 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. Неки од најпознатијих радних оквира овог програмског језика су *Flask*, *Django*, *Bottle*, *CherryPie*, *Pylons* и многи други. Филозофија овог програмског језика почива на великој читљивости програмског кода као и на стриктном поштовању увлачења редова написаног кода. Званична документација овог програмског језика је веома добро написана и корисна је како у процесу учења тако и у раду . 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 односно *Buttons*, поља за унос података односно *Entries* и лабеле за приказ података односно *Labels*. 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. *Tkinter* омогућава рад са елементима као што су Listbox односно листа елемената, Option menu листа опција, Radio button односно избор једног од више понуђених елемената, Checkbox односно избор једног или више елемената, Dropdown menu односно избор једног елемента из сажете листе, постоје и многи други елементи овог радног оквира који су веома лепо описани и документовани на страницама званичне документације. 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. Неке од наредби које се могу извршити су упис у базу података, ажурирање података из базе као и брисање података из базе. Дакле курсор служи за извршавање *SQL* наредби што га чини посредником између нас и наше базе података. Курсор може да учита само један или све редове који се слажу са *SQL* наредбом коју смо упутили бази података. Ипак са аспекта перформанси и брзине рада апликације боље је да се изведе мали број учитавања велике количине података јер често учитавање података као и учитавање велике количине података у кратком временском периоду значајно утичу на перформансе програма или апликације. Такође треба напоменути да програмски језик *Python* пружа велике могућности када је у питању рад са подацима учитаним из базе података. За *Sqlite3* није потребна инсталација јер се овај модул дистрибуира као део стандардног *Python* пакета . Аутор *Sqlite3* је Герхард Херинг.

## 4. 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

Програмски код за увоз sqlite3 модула, иницијализацију почетне базе података и метода за дохватање података из базе података.

# 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()  
# function for closing connection to database # # # # # # # # # # # # # # # # #  
def \_\_del\_\_(self):  
 self.conn.close()

Функција за унос података у базу као и функција за затварање конекције ка бази.

Скрипта увози sqlite3 модул, направљена је класа базе података и написане су функције за иницијализацију нове базе ако стара не постоји као и функције за унос, брисање измену и читање свих података из 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()))

Део кода који служи за импортовање библиотека и модула неопходних за рад програма. Такође приказан је и програмски код за иницијализацију базе података као и функција за проверу уписаних вредности у поља за унос података, која ради тако што прво покреће функцију за проверу формата унетих података па ако та провера прође онда врши рачунање и постављање вредности појединачних рата тако што од укупног износа одузме износ уплаћен при куповини и добијени број подели са бројем рата које је корисник изабрао.

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 prezime\_entry.get().\_\_len\_\_() < 3 or prezime\_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

Функција за проверу формата свих унетих података и приказ одговарајућих порука упозорења и савета на који начин да се исправно унесу подаци. Функција се покреће пре сваког покушаја уноса или измене података у бази података. Такође функција проверава да ли су унети називи за синдикат, име и презиме дужи од три и краћи од тридесет слова и у случају да нису у датом опсегу приказује се одговарајућа порука упозорења. У наставку су провере за укупан износ и износ уплаћен при куповини које осигуравају да је корисник унео искључиво позитивне бројеве, а постоје и провера унетог броја рата који мора бити у опсегу од један до дванаест као и провера унетог датума прве рате где се проверава да ли је датум унет у одговарајућем формату и да ли је датум старији од дана куповине и у случају да није приказује се одговарајућа порука.

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])))

Функција за читање свих података из базе података која затим форматира податке и попуњава *Listbox* елемент прочитаним подацима. У коментарима су приказане различите опције за форматирање и приказ података корисницима.

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 in dataset # # # # # # # # # # # # # # #

for i in range(100):

if parts\_list.get(i):

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

Функција која чита податке из базе података ред по ред и затим попуњава *Listbox* елемент прочитаним подацима уколико испуне постављене услове после чега следи филтрирање према датуму и истицање добијених података црвеном бојом да би се нагласило да су то рате за плаћање које ускоро стижу на наплату.

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]))

Функција која служи за читање и приказ података из базе података који су филтрирани на основу изабраног појма за претрагу који корисник бира преко корисничког интерфејса као што су на пример назив синдиката, име или презиме корисника.

def add\_item(): # function for adding items to Listbox and to database # #

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()

Функција која служи за додавање података у базу података тако што прво покреће функцију за проверу исправности унетих података и ако та провера прође без проблема, унети подаци се уписују у базу података али и у *Listbox* елемент на графичком корисничком интерфејсу, такође покрећу се и функције за чишћење поља за унос података као и функција за попуњавање елемента за приказ података.

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

try:

global selected\_item

global counter

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

lista = [] # list of all items

lista\_date = [] # list of items filtered by date

lista\_search = [] # list of items filtered by search word

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() # clear all entry fields

populate\_list() # populate listbox with data

Функција која служи за селекцију и приказ изабраних података тако што када корисник изабере одговарајући ред података левим кликом миша из елемента за приказ података функција тим подацима попуњава поља за унос података и на тај начин се подаци приказују кориснику и на располагању су му за читање, брисање или измену.

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()

Функција која служи за брисање изабраних података из базе података. Пре брисања података кориснику се приказује упозорење и тражи се потврда са питањем да ли заиста жели да обрише изабране податке да не би дошло до случајног брисања података.

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

sindikat\_entry.delete(0, END) # clear sindikat entry

ime\_entry.delete(0, END) # clear ime entry

prezime\_entry.delete(0, END) # clear prezime entry

ukupan\_iznos\_entry.delete(0, END) # clear ukupan iznos entry

broj\_rata\_entry.delete(0, END) # clear broj rata entry

datum\_prve\_rate\_entry.delete(0, END) # clear date entry

pojedinacna\_rata\_entry.delete(0, END) # clear single rate value entry

uplaceno\_entry.delete(0, END) # clear uplaceno entry

# Create window object # # # # # # # # # # # # # # # # # # # # # # # # #

app = Tk()

Функција за брисање свих података из поља за унос података, веома корисно при уносу велике количине података и креирање главног радног прозора.

# 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() # store value of name

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() # store value of last name

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() # store value of ukupan iznos

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)) # range of number to choose for rates

broj\_rata\_text = IntVar() # saving choosen number of rates

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 )

Постављање елемената за унос података као и натписа који описују тип података који треба унети, такође направљени су и елементи за чување унетих података *StringVar* и *IntVar* према типу података који ће бити унети и извршена је конфигурација поменутих елемената.

# 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)

Постављање елемента који ће да приказује листу података као и постављање елемента за вертикално листање свих података прочитаних из базе података.

# 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 ) # place add button on screen

# Button for deleting data from database # # # # # # # # # # # # # # # #

remove\_btn = Button(app, text = ' Obriši podatak ', width = 12, command=remove\_item) # place remove button on screen

remove\_btn.grid( row = 9, column = 6 ) # place remove button on screen

# 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 ) # place update button on screen

# 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) # place clear button on screen

# Option menu with StringVar and OptionMenu # # # # # # # # # # # # # # #

search\_options = [' Sindikat ', ' Ime ', ' Prezime '] # options for search

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 ) #p placing option menu

# 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 ) # place charge button on screen

# Button for showing data # # # # # # # # # # # # # # # # # # # # # # #

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

show\_btn.grid( row = 10, column = 5 ) # place show button on screen

# Button for search # # # # # # # # # # # # # # # # # # # # # # # # # # #

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

search\_btn.grid( row = 10, column = 8 ) # place search button on screen

Код за прављење и постављање дугмади која служе за покретање функција за читање, измену, унос и брисање података као и функције за чишћење поља за унос података и функције за израчунавање износа појединачних рата и приказ података према изабраном критеријуму. Постављен је и елемент за избор критеријума за претрагу *Option menu* .

# app title # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

app.title(' Python program za sindikalnu prodaju ') # setting up screen title

app.geometry( '1100x500' ) # setting up size of screen # # # # # # # #

# Populate data # # # # # # # # # # # # # # # # # # # # # # # # # # # #

populate\_list()

# Start program in windowed mode without console # # # # # # # # # # #

app.mainloop()

Постављање величине и наслова главног радног прозора. Такође покренуте су функција за попуњавање елемента за приказ података као и главна програмска функција.

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. Пожељно је да све датотеке буду у истом директоријуму.

### 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 у току рада.

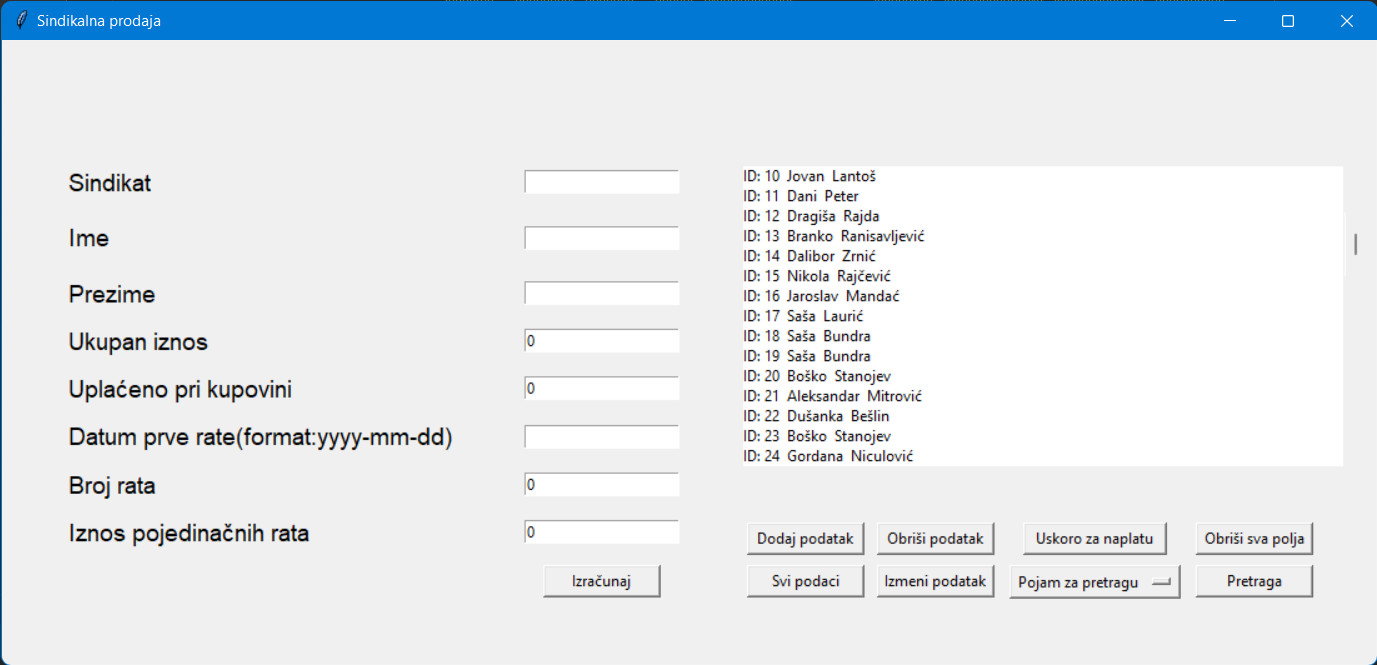
Табела 1.1 – Списак команди.

|  |  |
| --- | --- |
| 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 |

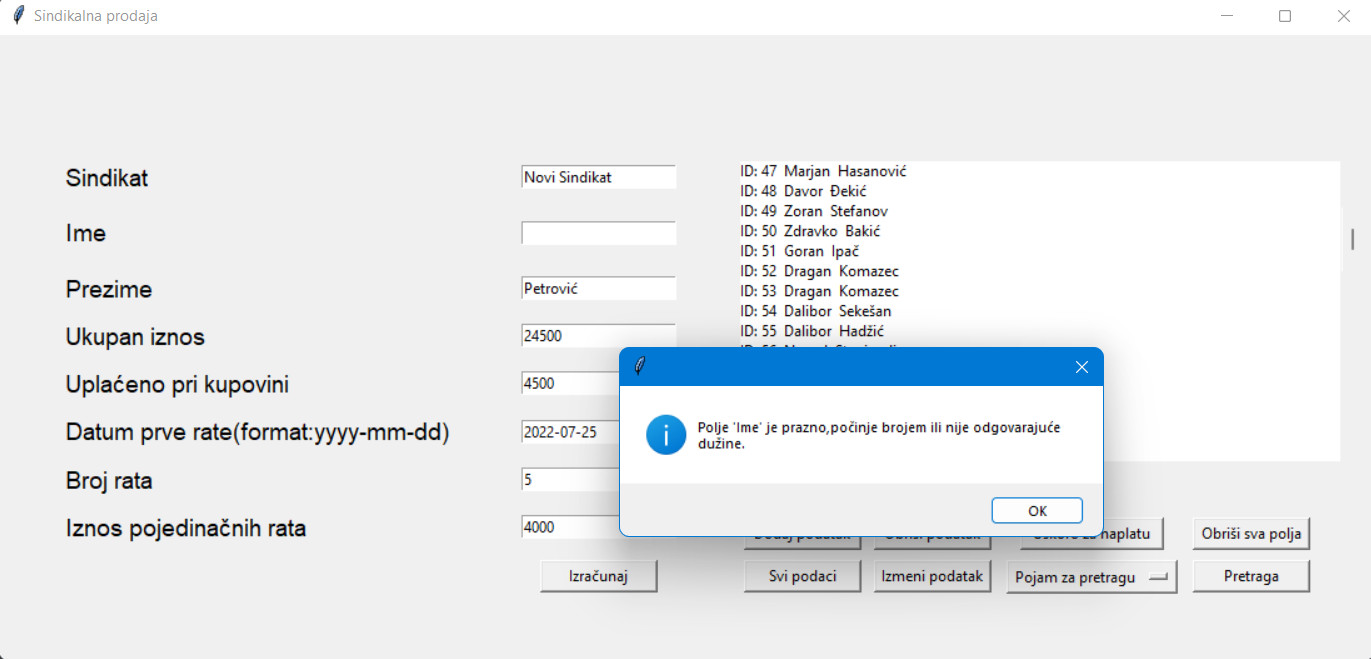


## Опис корисничког интерфејса

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. Дакле у поље за унос података о називу синдиката моду се унети слова, бројеви и специјални знакови јер неки синдикати у свом називу садрже те типове података, у поља за име и презиме корисника могу се унети само слова и прво слово мора бити велико, такође за наведена поља постоји ограничење у виду дужине уписаних података у опсегу од три до тридесет карактера. У пољима за укупан износ, износ уплаћен при куповини и износ појединачних рата могуће је ушисивање искључиво бројева, избор броја рата такође мора бити нумеричка вредност и ограничена је на опсег од један до дванаест. Датум прве рате мора бити уписан у одговарајућем формату што је назначено у лабели поред поља за унос податка и у поруци о грешци која се генерише у случају да корисник унесе датум у погрешном формату. Износ преосталих рата је могуће променити, али само ако се укупан износ умањен за износ уплаћен при куповини и умањен за износ уплаћен кроз претходно уплаћене рате подели на једнаке износе преосталих рата. Подаци који се могу добити из базе података су сви подаци о појединачним клијентима или подаци филтрирани према изабраном критеријуму што укључује филтрирање према називу синдиката, према имену корисника и према презимену корисника. Такође могу се добити и подаци филтрирани према датуму плаћања следеће рате и могућ је преглед свих клијената сортиран према редоследу уношења података у базу података.



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.

### 6.1 *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, --workpath за дефинисање локације привремених датотека, --clean за чишћење кеш меморије и брисање привремених датотека пре израде извршне датотеке, --onedir за паковање свих датотека и директоријума у једну датотеку, ова опција важи и без екслпицитне активације, --name опција за давање назива апликације, --console опција која омогућава приказ конзолног прозора у току рада апликације, ова опција се искључује уколико је екстензија датотеке *.pyw* на *windows* оперативном систему.

# 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, филтрирање према различитим критеријумима на пример према томе да ли је корисник исплатио све рате или филтрирање према изабраном датуму, сортирање података по азбуци или абецеди, сортирање према цени растући или опадајући или сортирање по преосталом броју рата за плаћање. 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, шпанском, турском, руском, француском 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, писање функција, рад са различитим типовима података, рад са реалним подацима, тестирање различитих елемената програмског језика и његових библиотека и радних оквира у сврху проналажења најбољег избора за решавање програмерских задатака 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 у складу са прописаним смерницама, што ће ми свакако бити од користи у будућности. Радећи на пројекту *Python* програм за евиденцију синдикалне продаје стекао сам знање и искуство у изради различитих модуларних целина које повезане на прави начин чине једну функционалну целину односно апликацију.

# ИНДЕКС ПОЈМОВА

A

архивирање 14

Б

база података 10

Д

дистрибуција 1,13

дугмад 1,2,11

Е

Excel 1,14

И

инсталација 1,13

извршна датотека 10

К

конверзија 14

кориснички интерфејс 1,11

Л

лабела 11

М

модуларан 13

мултиплатформски 13

П

програмски језик 2

преносив 1,13

пакет 13

С

слика 11,12

скрипта 3,10

сортирање 19

Т

табела 11

Ф

форматирање 14

У

унапређење 14

# 9 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

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