UPRAVLJANJE SQLITE3 BAZOM PODATAKA POMOĆU PYTHON 3

Šarić, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

UPRAVLJANJE SQLITE3 BAZOM PODATAKA POMOĆU PYTHON 3

Završni rad

Marko Šarić

Osijek, 2022. godina

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 14.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Marko Šarić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	1589, 08.07.2005.
OIB Pristupnika:	73898897583
Mentor:	Izv. prof. dr. sc. Irena Galić
Sumentor:	Dr. sc. Hrvoje Leventić
Sumentor iz tvrtke:	
Naslov završnog rada:	Upravljanje sqlite3 bazom podataka pomoću Python 3
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	14.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2022.

Ime i prezime studenta:	Marko Šarić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	1589, 08.07.2005.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje sqlite3 bazom podataka pomoću Python 3**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Irena Galić

i sumentora Dr. sc. Hrvoje Leventić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Tema ovog rada je vođenje evidencije održavanja informatičke opreme kroz mnogobrojne datoteke programskog paketa Microsoft Excel automatiziranjem spremanja podataka tih evidencija u jednu strukturiranu bazu podataka korištenjem programskog jezika Python 3. Glavni problem vođenja evidencije u Microsoft Excel datotekama je ponajprije količina podataka te otežano pronalaženje različitih potrebnih podataka kada se za to ukaže potreba. Naime, da bi se došlo do nekog podatka potrebno je znati nekoliko kriterija (npr. datum posljednjeg servisa opreme, ime djelatnika koji je obavio servis) te na osnovu njih pretraživati ručno pojedinačne Microsoft Excel datoteke, što je vremenski dugotrajno i neučinkovito. Spremanjem svih podataka unutar jedne baze podataka učinkovitost pretrage višestruko se povećava, što je i razlog razvijanja ovog programskog rješenja. Kako je iz određenih razloga i dalje potrebno voditi istu evidenciju i kroz Microsoft Excel, ovaj rad će biti alat koji će olakšati pretragu, smanjiti vrijeme pretrage, dati jasniju sliku povijesti održavanja pojedinačnog uređaja te u budućnosti omogućiti razvijanje novih alata koji će višestruko povećati funkcionalnost. U radu se koristi sqlite3 baza podataka zbog lakšeg korištenja i manje zahtjevnosti, kao i nekoliko modula programskog jezika Python 3, što omogućava izvedbu cjelovitog rješenja za postizanje željenog cilja.

U radu će se najprije ukratko iznijeti najvažnije prednosti programskog jezika Python 3, a zatim će se predstaviti osnovne značajke modula Tkinter, openpyxl i sqlite3. U glavnom dijelu rada pobliže će se obrazložiti tijek izvođenja programa, od uvoza potrebnih modula za izvođenje, generiranja grafičkog sučelja do definiranja funkcija, varijabli i podataka potrebnih za automatski i ručni unos podataka u bazu.

1.1. Zadatak završnog rada

Zadatak završnog rada je izvođenje automatiziranog unosa podataka iz Microsoft Excel datoteka u sqlite3 bazu podataka korištenjem Python 3 programskog jezika, kao i omogućavanje unosa podataka u istu bazu direktnim unosom („ručno“) ali strukturirano na način jednak unosu podataka u Microsoft Excel tablicu. Zadatak je i stvaranje temelja za razvoj dodatnih funkcionalnosti u vođenju evidencije održavanja informatičke opreme u svrhu učinkovitijeg i kvalitetnijeg održavanja i pretraživanja rastuće količine podataka.

2. PREGLED LITERATURE

Postoje već gotova programska rješenja koja se bave istom tematikom kao i ovaj rad, a koja su mnogo razvijenija i sa bogatom količinom opcija dostupnih korisniku. Prema stranici Investopedia.com [1] u 2022. godini kao najbolje besplatno sveukupno rješenje u kategoriji softver za upravljanje inventarom je Zoho Inventory. Zoho Inventory svojim korisnicima nudi više-kanalnu povezanost preko Interneta te integraciju raznih drugih servisa za kupovinu i plaćanje. Nudi integriran mrežni kao i lokalizirani sustav upravljanja inventarom, kupovinama, dostavama i slično. Korisnicima su na raspolaganju i praćenje paketa pomoću GPSa (engl. *Global Positioning System*) te mnoge druge opcije [2]. Sve te opcije, iako vrlo korisne i dobro razvijene, nisu potrebne za rješenje problema kojim se bavi ovaj rad. Mrežna povezanost bi izložila podatke dodatnim opasnostima koje mrežna povezanost donosi, a nije nužna za funkcioniranje. Slična gotova rješenja vođenja inventara također nude veliki spektar opcija koje u ovom trenutku nisu potrebne za rješenje predmetnog problema.

Druga mogućnost za rješenje predmetnog problema bila bi korištenje neke druge relacijske baze podataka poput MySQL. MySQL je upravljački sustav za relacijske [3], više-nitne, baze podataka otvorenog izvornog koda, koji je razvio Michael Widenius 1995. godine. Godine 2000. MySQL je izdan pod modelom dualnih licenci koja je javno besplatno korištenje pod GNU GPL licencom (engl. *General Public License*) što je dramatično povećalo popularnost MySQL. Mnoge značajke MySQL-a doprinose njegovu statusu kao vrhunskog sustava za baze podataka. Usporedba časopisa eWEEK nekoliko baza podataka, uključujući MySQL, Oracle, MS SQL, IBM DB2 i Sybase ASE, pokazala je kako su MySQL i Oracle bili izjednačeni u najboljim performansama i najvećoj saklabilnosti. MySQL može upravljati desecima tisuća tablica i milijardama redova podataka, te uz to vrlo brzo funkcionira u upravljanju malim količinama podataka.

MySQL paket dolazi sa nekoliko programa [3]. Glavni je MySQL server, koji koristi MySQL daemon. Daemon „sluša" zahtjeve na određenom mrežnom portu preko kojeg klijent šalje naredbe. Standardni MySQL klijentski program se zove jednostavno mysql te preko njega korisnik može pristupiti svojim podacima ili upravljati istima.

Rješenje predmetnog problema imalo je određene potrebe koje su morale biti ispunjene. Jedna od potreba rješenja je bila i sigurnost, te je zbog toga izabran pristup sa SQLite bazom podataka. Predviđeno je da će se baza podatka držati na samo jednom računalu koje neće biti mrežno povezano. SQLite baza podataka je relacijska baza podatka koja sve podatke sprema u obliku tablice sa redovima i stupcima. Kompletna

baza je spremljena unutar jedne datoteke, te je time lako podatke dostaviti na drugo računalo putem prijenosnog medija, ako bi se javila potreba za time.

Sa auto-py-to-exe [4], projekt koji je razvio Brent Vollebregt, lako je napraviti izvršnu datoteku neke Python aplikacije. Iza grafičkog sučelja nalazi se PyInstaller, terminal aplikacija za stvaranje Python izvršnih datoteka za Windows, Mac i Linux operacijske sustave. Na ovaj način moguće je stvoriti jednu izvršnu datoteku koju je tada moguće pokrenuti na bilo kojem od tri glavna operacijska sustava, bez obzira na to da li je Python programski jezik instaliran na tom računalu ili ne. Cijelo riješenje se na taj način može staviti na prijenosni medij u obliku dvije datoteke, pojavi li se potreba za prijenosom podataka na druga računala, jedna datoteka kao izvršna datoteka programskog rješenja i druga datoteka kao baza podataka sa spremljenim podacima.

Rješenje problema je odabrano u obliku Python programa i SQLite baze podatka zbog navedenih razloga. Izgradnjom Python programa omogućena je potpuna prilagodba programa obliku spremljenih podataka. Podaci u Microsoft Excel datoteci su u tabličnom obliku koji koriste relacijske baze podataka, te su se Python i SQLite pokazali kao idealno i najjednostavnije rješenje problema.

3. INTEGRACIJA KORIŠTENIH TEHNOLOGIJA U IZRADI RJEŠENJA

Svrha ovog rada je unos postojećih podataka iz pojedinačnih Microsoft Excel datoteka u jedinstvenu bazu podataka. Iz tog razloga program nema posebne algoritme izvođenja, donošenje logičkih zaključaka temeljenih na unosu podataka od strane korisnika niti odstupanje od zadanih zadataka koje bi se inače moglo očekivati u programskim aplikacijama. Rad je stvoren sa specifičnom svrhom automatizacije i ubrzanja procesa unosa podataka koji bi drugim metodama bio vrlo dug i zahtijevao poveću interakciju sa korisnikom.

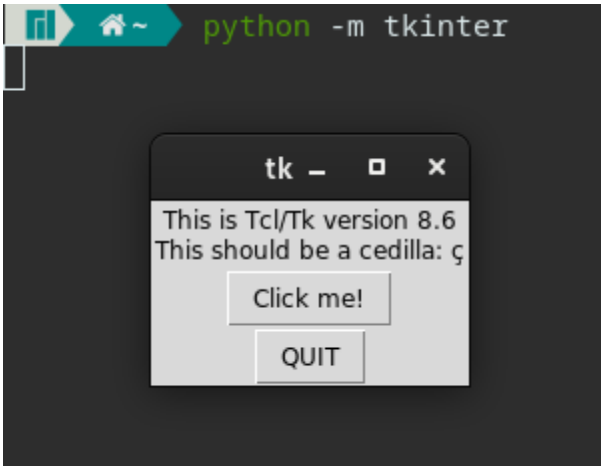
3.1. Programski jezik Python 3

Python je programski jezik otvorenog koda (engl. *open source*) koji se koristi kako za samostalne programe, tako i za pisanje skriptiranih programa u širokoj domeni primjena [5]. Besplatan je, prenosiv, snažan, te ga je relativno lako koristiti. Programeri iz svih domena industrije informacijskih tehnologija koriste Python zbog njegovog fokusa na produktivnost developera te na kvalitetu programa, što pruža stratešku prednost u velikim i malim projektima.Za mnoge programere [5] Pythonov fokus na čitljivost, koherentnost i generalna kvaliteta programa izdvaja ovaj programski jezik od ostalih skriptnih alata u industriji. Pythonov kod je dizajniran da bude čitljiv, te prema tome i ponovno iskoristiv i lako održavan. Python također ima i veliku podršku za naprednije mehanizme ponovnog korištenja programa, kao što je objektno orijentirano i funkcijsko programiranje. Većina Python programa može se izvoditi na svim većim operacijskim sustavima bez izmjena [4]. Prijenos Python programa između Microsoft Windows i Linux sustava je u glavnini slučajeva samo pitanje kopiranja izvornog koda. Python pruža više opcija stvaranja grafičkih sučelja, programa za pristup bazama podataka, web bazirane sustave, pa čak i stvaranje kompletno novih operacijskih sustava.Python dolazi sa velikom kolekcijom već izgrađenih i prenosivih funkcionalnosti koji se nazivaju standardna biblioteka (engl. *standard library*).

U nastavku će se detaljnije objasniti moduli programskog jezika Python 3 koji su korišteni u izradi programskog rješenja.

3.1.1. Modul: tkinter

Tkinter paket („Tk sučelje“) je standardno Python grafičko korisničko sučelje za Tcl/Tk set alata (eng. *Tool Command Language/Tk Graphical User Interface toolkit*) [6]. Tk kao i tkinter su dostupni na većini Unix platformi, uključujući i macOS, kao i na Windows sustavima.



Sl. 3.1. Prikaz instalirane Tkinter verzije naredbom iz komandne linije

Naredbom **python –m tkinter** [6] iz komandne linije (Slika 3.1.) otvara se prozor koji demonstrira jednostavno Tk sučelje koje nam daje informaciju da je tkinter modul ispravno instaliran na računalu i koja verzija Tcl/Tk je instalirana. Tkinter nije samo „tanki omot“ koji omogućava njegovo korištenje unutar Python 3 okruženja, nego dodaje značajnu količinu vlastite logike kako bi korisničko iskustvo rada sa modulom bilo sličnije sintaksi Python jezika. Tcl/Tk nije jedna biblioteka nego se sastoji od nekoliko određenih modula, svaki sa svojim specifičnim funkcionalnostima i vlastitom službenom dokumentacijom. Tcl je dinamički interpretiran programski jezik poput Pythona i iako se može koristiti kao zaseban jezik, najčešće se koristi kao umetnuti jezik u C aplikacijama ili kao sučelje za Tk alat. Tk je Tcl paket implementiran u C programskom jeziku koji dodaje posebne naredbe za stvaranje i upravljanje malim gotovim programima sa grafičkim sučeljem (engl. *widget).* Ttk (engl. *Themed Tk*) je novija porodica Tk widgeta koji pružaju puno bolji izgled platformama nego što je to bilo moguće sa klasičnim Tk widgetima.

Kada Python [6] aplikacija koristi Tkinter klasu kako bi, na primjer, stvorila widget, tkinter modul prvo asemblira Tcl/Tk upravljačku naredbu, zatim je prosljeđuje internom _tkinter binarnom modulu, koji tada poziva Tcl interpretator za evaluaciju, potom Tcl interpretator poziva Tk i/ili Ttk pakete koji će slati zahtjeve u Xlib za Unix/X11 sustave, Cocoa za macOS sustave ili GDI za Windows sustave.

3.1.2. Modul: openpyxl

Openpyxl [7] je Python biblioteka za čitanje i pisanje Microsoft Excel 2010 xlsx/xlsm/xltx/xltm datoteka. Nastala je zbog nedostatka postojećih biblioteka koje mogu direktno pisati/čitati Office Open XML format

iz Pythona. Openpyxl je open source projekt kojeg održavaju volonteri u svoje slobodno vrijeme, što znači da neke poželjne opcije i funkcionalnosti ne postoje ili nisu još do kraja razvijene. Profesionalna potpora za openpyxl modul može se pronaći kod nekih privatnih tvrtki, dok se volonterska pomoć može uvijek pronaći na raznim forumima ili direktnim kontaktom jednog od developera.

Na slici 3.2. prikazan je isječak koda openpyxl modula kako je naveden u službenoj dokumentaciji.

```
1.  from openpyxl import Workbook
2.  wb = Workbook()
3.  ws = wb.active
4.  ws['A1'] = 42
5.  ws.append([1, 2, 3])
6.  import datetime
7.  ws['A3'] = datetime.datetime.now()
8.  wb.save("sample.xlsx")
```

Sl. 3.2.: Isječak koda izvornog modula openpyxl

Prva linija koda uvozi Workbook funkcije iz openpyxl modula, a druga linija definira `Workbook()` funkciju kao wb varijablu. Nakon što je Excel datoteka učitana, svi podaci će biti učitani kao Python objekti te se tada njima može upravljati kao sa ostalim Python objektima. Treća linija učitava radni list iz aktivne Microsoft Excel datoteke što omogućuje upravljanje podacima koji se nalaze u njoj, kako je i vidljivo u četvrtoj liniji koda. Peta linija dodaje redove u Microsoft Excel datoteku, u ovom slučaju dodaje listu koju će naredbom `append()` dodati u prvu praznu ćeliju podatke te ispuniti i dodatne ćelije u istom redu ovisno o količini podataka. Uvoz datetime modula vidljiv je u šestoj liniji koda, a sedma linija prikazuje automatsko konvertiranje tipa podataka u Python3 programskom jeziku. U posljednjoj, osmoj liniji koda pohranjuje se datoteka pod određenim nazivom. U trenutku pisanja rada u službenoj dokumentaciji openpyxl modula [6] nisu navedena nikakva ograničenja u vidu tipa podataka koji se mogu koristiti.

3.1.3. Modul: sqlite 3

SQLite [8] je C biblioteka koja pruža jednostavnu disk-baziranu bazu podataka koja ne zahtijeva posebni serverski proces i omogućuje pristup bazi podataka koristeći nestandardnu varijantu SQL programskog jezika. Neke aplikacije mogu koristiti SQLite za interno spremanje podataka. Moguće je i napraviti prototip aplikacije koristeći SQLite koji bi se tada mogao migrirati u veću bazu podataka koristeći PostgreSQL ili Oracle. Modul sqlite3, kojeg je napisao Gerhard Häring, pruža SQL sučelje koje odgovara

DB-API 2.0 specifikacijama opisanim u PEP 249, te je za izvođenje potrebna SQLite 3.7.15 ili novija verzija.

Za korištenje modula [8] prvo se stvara veza koja predstavlja bazu podataka. Kada se veza ostvari potrebno je stvoriti objekt koji će se pozivati sa njegovom **`execute()`** funkcijom kako bi se izvršile SQL naredbe. Spremljeni podaci ostaju u stvorenoj bazi podataka i nakon ponovnog pokretanja Python interpretatora. Za poziv podataka iz baze koristi se isti objekt sa raznim pripadajućim metodama. Kada su obavljene sve radnje sa bazom konekciju je potrebno zatvoriti kako bi se promjene spremile.

Na slici 3.3. prikazan je isječak izvornog koda predstavlja uvoz sqlite3 modula u Python 3 program, potom stvaranje baze podataka, stvaranje tablice ili strukture te novostvorene baze, definiciju vrste podataka i sami unos podataka, te na kraju spremanje promjena i prekid konekcije sa stvorenom bazom podataka. Ovo je jedan vrlo osnovan primjer koda sa sqlite3 modulom, ali vrlo jasno pokazuje koliko je učinkovit i jednostavan za korištenje kada su potrebne jednostavnije baze podataka.

```
1.  import sqlite3
2.  conn = sqlite3.connect('/home/marko/primjer.db')
3.  c = conn.cursor()
4.  c.execute("CREATE TABLE IF NOT EXISTS tablica(prvi_stupac INTEGER)")
5.  c.execute("INSERT INTO tablica(prvi_stupac) VALUES(?)", (1))
6.  conn.commit()
```

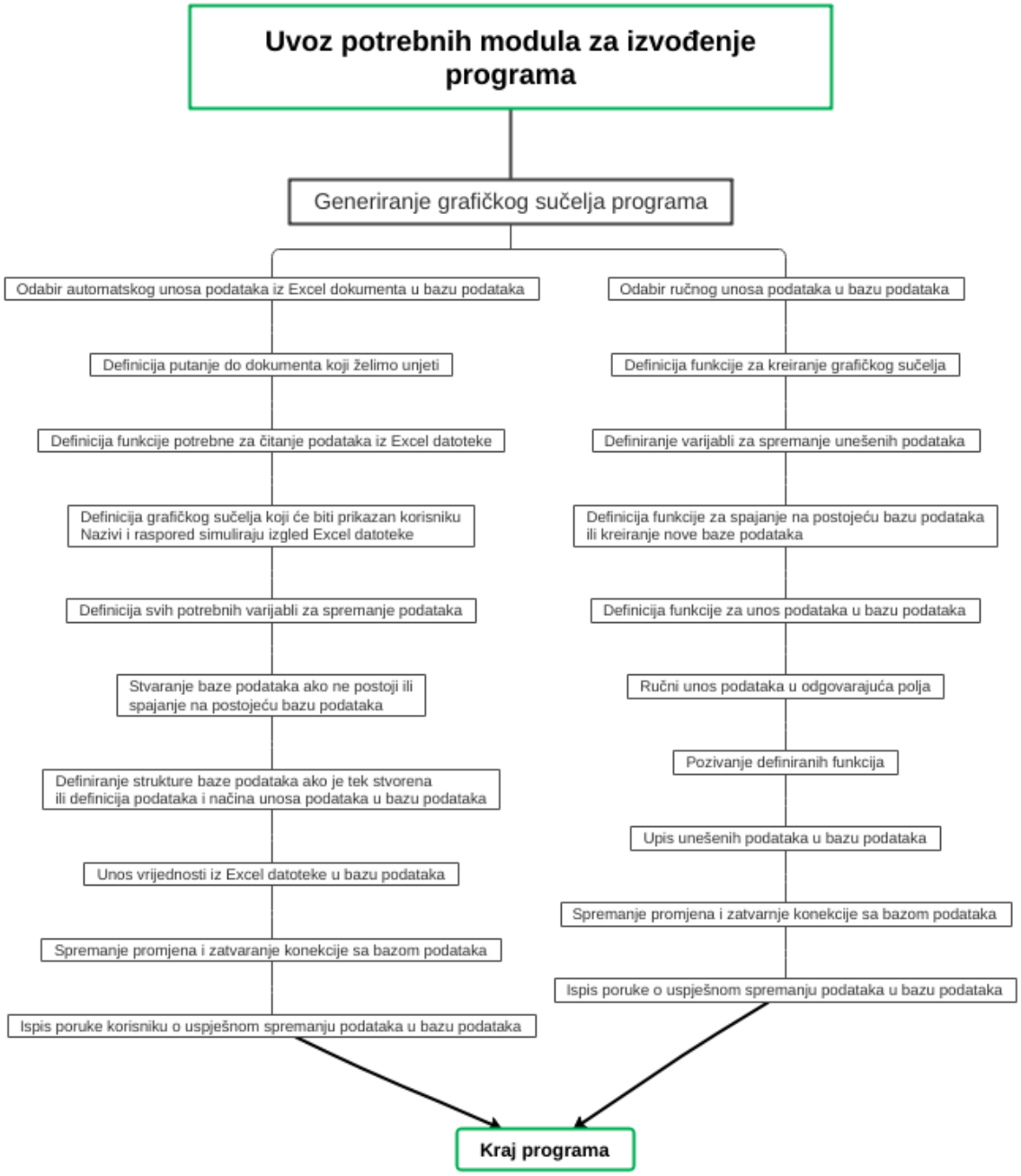
Sl. 3.3. Uvoz sqlite3 modula u Python 3

Prva linija koda uvozi potrebni sqlite3 modul, druga linija prikazuje spajanje na bazu podataka sa zadanom putanjom do iste. Treća linija koda prikazuje kursor objekt koji je potreban za pretraživanje baze podataka i druge radnje. Četvrta linija koda stvara tablicu naziva tablica, ako ona već ne postoji, ili koristi tablicu naziva tablica, ako ona postoji, također definira naziv tablice naziv stupaca u tablici te vrstu podataka u stupcima. Peta linija koda unosi podatke u tablicu a šesta linija potvrđuje promjene i zatvara vezu s bazom podataka.

Unatoč [8] jednostavnosti korištenja i brzini stvaranja, sqlite3 modul podložan je i problemima koji postoje u SQL programskom jeziku, kao što je ubacivanje zlonamjernih vrijednosti podataka (engl. *SQL injection attacks*). Zbog toga postoje preporuke o načinu strukturiranja i unosa podataka u bazu, što neće biti obrađeno u ovom radu jer baze podataka koje su rezultat ovog rada neće biti izložene mreži, niti će biti javno dostupne na bilo koji način. To podrazumijeva da ako u budućnosti dođe do promjene

funkcionalnosti ovog rada, sukladno tome će se i sigurnosni standardi morati uvesti u krajnju verziju programa.

Na slici 3.4. dan je okvirni prikaz tijeka izvođenja programa, razvrstan u faze po kojima je razvijeno rješenje problema.



Sl 3.4. Dijagram tijeka izvođenja programa

3.2. Uvoz potrebnih modula za izvođenje programa

Rješenje problema zahtijevalo je interakciju sa nekoliko modula dostupnih u Python 3 programskom jeziku. Ranije u ovom radu opisani su glavni moduli te neka njihova glavna svojstva. U nastavku će korištenje modula u ovom programu biti detaljnije razjašnjeno.

Prvo se uvozi tkinter modul te mu se pridaje oznaka „tk“ zbog lakšeg korištenja u daljnjem pisanju koda. Ovaj modul koristi se za stvaranje glavnog grafičkog sučelja cijelog programa. Potom se iz tkinter modula uvozi ttk modul koji u sebi sadrži *notebook widget*. *Notebook widget* se u programu koristi za generiranje kartica (engl. *tabs*) u grafičkom sučelju. Koristeći kartice korisniku se pruža opcija odabira automatskog ili ručnog unosa podataka na jasan i jednostavan način. Nadalje, uvozi se sve dostupno iz tkinter modula. Dakako, moguće je uvesti samo **`StringVar()`** metodu koja je potrebna za izvođenje programa, međutim uobičajena praksa je uvesti sve dostupno kada veličina programa u memoriji nije problematična. Na taj način se mogu izbjeći naknadni problemi ako se, na primjer, poziva neku metodu iz tkinter modula koja nije eksplicitno uvezena, što bi dovelo do prekida izvršavanja programa. Nastavkom razvoja ovog rješenja u budućnosti, te implementacijom dodatnih funkcionalnosti ili metoda pristupa, ovo bi se moglo promijeniti zbog smanjenja veličine programa i brzine izvođenja, no za sada nije potrebno.

Na kraju se uvoze openpyxl i sqlite3 moduli koji su ključni za izvođenje ovog rješenja problema. Modul openpyxl, kao što je već napomenuto, omogućuje interakciju sa Microsoft Excel dokumentima. Modul sqlite3 pruža mogućnost stvaranja baze podataka te upravljanje istom.

Naredbe za uvoz svih potrebnih modula za izvođenje ovog programa prikazane su na slici 3.5.:

```
1.  import tkinter as tk
2.  from tkinter import ttk
3.  from tkinter import *
4.  import openpyxl, sqlite3
```

Sl 3.5. Uvoz modula Tkinter, openpyxl i sqlite3

Prva linija koda uvozi tkinter modul u program pod nazivom tk. Druga linija koda iz tkinter modula uvozi funkciju ttk koja sadrži notebook widget. Treća linija koda uvozi sve funkcije iz tkinter modula – ovime se, između ostalog, uvozi **`StringVar()`** funkcija. Četvrta linija koda uvozi openpyxl modul za interakciju s Microsoft Excel tablicama, te sqlite3 modul za interakciju sa bazama podataka.

3.3.Generiranje grafičkog sučelja

Korištenjem tkinter modul započinje izrada grafičkog sučelja programa. Najprije se definira glavni ili roditeljski direktorij (engl. *Parent Directory*) za što se koristi **`Tk()`** klasa koja se instancira bez ikakvih argumenata. Slijedeći korak je definiranje veličine grafičkog sučelja pozivanjem funkcije **`geometry()`** kojoj se pridaje željena vrijednost. Za izvršenje programa određena je veličina od 800x650 piksela, tj. minimalno potrebna veličina za smještanje svega na uredan način unutar konačnog izgleda grafičkog sučelja. Moguće je napraviti program tako da reagira na veličinu ekrana na kojem je prikazan, no to nije bilo potrebno za rješenje glavnog problema, stoga je grafičko sučelje izvedeno na najmanji mogući način kako bi se lako moglo prikazati i na ekranima niske rezolucije. Zatim se definira naslov roditelj direktorija funkcijom **`title()`**. Odabran je naslov „FERIT IT oprema“ jer je za to program namijenjen.

Početak izrade grafičkog sučelja prikazan je na slici 3.6.

```
1.  root = tk.Tk()
2.  root.geometry('800x650')
3.  root.title('FERIT IT oprema')
```

Sl 3.6. Početak izrade grafičkog sučelja

Prva linija koda stvara roditelj–direktorij (engl. *parent directory*). Druga linija koda definira veličinu početnog zaslona prilikom izvođenja programa, a treća linija dodjeljuje naslov početnom zaslonu.

Nakon toga se definira kontrola kartica korištenjem **`ttk.Notebook()`** funkcije. Daje se naziv „tab_control“ varijabli koja se pretvara u roditelj (engl. *parent*) widget za kartice te se ista definira kao glavni okvir (engl. *frame*) unutar cijelog grafičkog sučelja koje je nazvano „root“. Taj okvir će sadržavati djecu (engl. *child*) okvire koji su zapravo kartice nazvani „tab1“ i „tab2“. Definiraju se tabovi kao okviri unutar glavnog okvira, opisuju se i postavlja se glavni okvir u grafičko sučelje korištenjem **`pack()`** funkcije.

Sva daljnja dodavanja widgeta moraju u svojoj definiciji imati naznačeno kojem okviru pripadaju od ovih prethodno definiranih okvira.

Izrada potrebnih okvira za grafičko sučelje prikazana je na slici 3.7.

```
1.  tab_control = ttk.Notebook(root)
2.  tab1 = ttk.Frame(tab_control)
3.  tab2 = ttk.Frame(tab_control)
4.  tab_control.add(tab1, text = 'Automatski unos')
5.  tab_control.add(tab2, text = 'Ručni unos')
6.  tab_control.pack(expand=1, fill='both')
```

Sl 3.7. Izrada okvira za grafičko sučelje

Prva linija koda stvara tab kontrolu. **Notebook()** metoda prihvaća određene definirane opcije za oblikovanje dok je „tab_control“ roditelj-widget za tabove. Druga i treća linija koda dodaju kartice. **`Frame()`** widget se ponaša kao spremnik i koristi se za grupiranje ostalih widgeta. Četvrta i peta linija koda koriste **`add()`** metodu koja se nalazi u tk.ttk Notebook klasi, a koristi se za dodavanje novih kartica i Notebook widgeta. Šesta linija koda stavlja Notebook widget na zaslon. Metoda **`pack()`** koristi se kako bi se widgeti organizirali u blokove prije nego se stave u roditelj-widget. Opcija **`expand()`** osigurava jednaku raspodjelu prostora među widgetima koji imaju vrijednost proširenja različitu od nule kada se roditelj-widget proširi. Opcija **`fill='both'`** osigurava da widget zauzima prostor koji mu je dodijeljen po X i po Y osi.

Nakon izgradnje potrebnih okvira, dodaju im se funkcionalnosti. Za „tab1“ koji je opisan kao „Automatski unos“ prvo se definira varijabla naziva „putanja_automatski_unos“ te se ista postavlja kao **`StringVar()`** oblik podatka. Ova varijabla koristi se za spremanje vrijednosti koju će korisnik upisati kao putanju do lokacije gdje se nalazi Microsoft Excel datoteka koja se treba pohraniti u bazu podataka. Zatim se stvara jedan entry widget koji se koristi kao polje za unos vrijednosti, to jest polje u koje korisnik upisuje putanju do datoteke. Na kraju se izrađuje gumb koji će korisnikov unos spremiti u varijablu te pozvati sve prethodno definirane funkcije i ažurirati izgled „tab1“ okvira. Za „tab2“ koji je opisan kao „Ručni unos“ stvara se samo gumb koji će pozvati sve prethodno definirane funkcije vezane uz ručni unos, te ažurirati izgled „tab2“ okvira.

U ovom dijelu izrade grafičkog sučelja još se izrađuju gumbi koji aktiviraju automatski unos podataka ili ručni unos podataka te se navodi **`root.mainloop()`** funkcija koja se ponaša kao beskonačna petlja i koristi se za pokretanje programa.

Završetak izrade glavnog grafičkog sučelja prikazan je na slici 3.8.

```
1.  putanja_automatski_unos = StringVar()
2.  lokacija_entry = Entry(tab1, textvariable = putanja_automatski_unos, width=50)
3.  lokacija_entry.grid(row=0, column=1)
4.  ucitajIzvjestajButton = Button(tab1, width=15, height=1, text='Učitaj izvještaj')
5.  ucitajIzvjestajButton.bind("<Button-1>", unos_podataka)
6.  ucitajIzvjestajButton.grid(row=0, column=0)
7.  rucni_unos_button = Button(tab2, width=15, height=1, text='Ručni unos')
8.  rucni_unos_button.bind("<Button-1>", rucni_unos)
9.  rucni_unos_button.grid(row=0, column=0)
10. root.mainloop()
```

Sl 3.8. Završetak izrade glavnog grafičkog sučelja

Prva linija koda definira varijablu naziva putanja_automatski_unos u koju spremamo punu putanju do Microsoft Excel datoteke u **`StringVar()`** obliku. Druga i treća linija koda stvara entry widget u koji ručno upisujemo putanju te koristimo **`grid()`** metodu kako bi roditelj-widget razdvojili na redove i stupce. U četvrtoj liniji koda stvara se gumb kojeg se pritišće nakon unosa putanje do Microsoft Excel datoteke. Peta linija koda gumb povezuje s pritiskom na lijevu tipku miša te poziva ranije definiranu funkciju unos_podataka. Šesta linija koda smješta gumb na zaslon u definirani red i stupac. Sedma, osma i deveta linija koda stvaraju gumb za ručni unos podataka, smještaju ga na tab2 te pozivaju ranije definiranu funkciju rucni_unos. Deseta linija koda prikazuje **`mainloop()`** funkciju koja se ponaša kao beskonačna petlja u koristi se za pokretanje programa.

3.4. Definiranje funkcija, varijabli i podataka potrebnih za automatski unos podataka

Definira se funkcija naziva „unos_podataka“ kojoj je kao argument dana „event“ opcija jer se na taj način funkcija poziva pritiskom na gumb koji je kreiran u grafičkom sučelju. Prvi korak potreban za izvršenje je dohvaćanje vrijednosti putanje do Microsoft Excel datoteke koju je korisnik upisao u entry widget u grafičkom sučelju. Funkcija **`get()`** dodana na varijablu koja je definirana za spremanje te vrijednosti izvršava ovaj korak. Drugi korak je definiranje varijable u koju se pomoću openpyxl modula sprema lokacija Microsoft Excel datoteke. Korištenjem istog modula definira se i aktivna stranica (engl. *sheet*) unutar Microsoft Excel datoteke s kojom se radi.

Definiranje glavne funkcije za automatski unos i upotrebu openpyxl modula prikazan je na slici 3.9.

```
1.  def unos_podataka(event):
2.      putanja_text = putanja_automatski_unos.get()
3.      workbook = openpyxl.load_workbook(putanja_text)
4.      sheet = workbook.active
```

Sl 3.9. Definiranje glavne funkcije za automatski unos i upotrebu openpyxl modula

Prva linija koda definira funkciju koja sadrži sve vezano za automatski unos podataka te je definira kao event funkciju, što znači da će ona biti pozvana prilikom nekog događaja, tj. pritiskom lijeve tipke miša na gumb. Druga linija koda uzima putanju do Microsoft Excel datoteke iz entry widgeta. Treća linija koda pristupa podacima iz Microsoft Excel tablice pomoću openpyxl modula. Četvrta linija koda definira rad sa aktivnim radnim listom, to jest glavnim radnim listom u Microsoft Excel tablici kada se ona pokrene.

Zatim se očitavaju vrijednosti iz Microsoft Excel datoteke te se kreira label widgeta u kojega se očitane vrijednosti spremaju. Label widget je dio tkinter modula te se ovdje primarno koristi u svrhu oponašanja približnog izgleda Microsoft Excel datoteke a istovremeno i za spremanje i prikazivanje očitanih podataka. U automatskom unosu ovo je posljednja prilika korisniku za pregled svih informacija prije pohrane u bazu podataka. Uoči li se potreba za izmjenom nekog podatka, korisnik mora izmijeniti podatak u Microsoft Excel datoteci, te nakon izmjene ponovno učitati datoteku u program. Ovo je ujedno i razlog zašto je stvoren ručni unos u ovom programu iako nije bio potreban za rješenje glavnog problema. Prilikom izrade rada postalo je očito da bi logičan slijed rješavanja ovog problema bio potpuno izbacivanje Microsoft Excel datoteka iz procesa unosa podataka. Međutim, kako već postoji velika količina Microsoft Excel datoteka koje je potrebno unijeti u bazu podataka, opcija za automatski unos je zadržana.

Label widget se stvara spremanjem u varijablu, definiranjem okvira kojem pripada te dodavanjem opisnog teksta. Korištenjem openpyxl modula i opcije „f-strings“ u svakom label widgetu opisni tekst naveden je kao naziv polja iz Microsoft Excel datoteke. Također, korišten je openpyxl da se definira iz koje ćelije u Microsoft Excel datoteci se vrijednost očitava, a sa „f-strings“ metodom u label widgetu je ispisana očitana vrijednost.

Upotreba label widgeta u automatskom unosu prikazana je na slici 3.10.

```
1.  djelatnik = Label(tab1, text=f"Djelatnik: {sheet.cell(row=5, column=3).value}")
2.  djelatnik.grid(row=1, column=0, sticky="E")
```

Sl 3.10. Upotreba *label widgeta* u automatskom unosu

U ovim linijama koda čitaju se vrijednosti izravno iz Microsoft Excel datoteke pomoću **`sheet.cell().value`** opcije, te se one smještaju unutar label widget*a* sa lijeve strane programa pomoću **`grid()`** funkcije. Definira se label, kartica unutar kojeg se on nalazi, tekst koji će biti ispisan u label widgetu te se na kraju „sticky*“* opcijom poravnava „istočno“ tj. desno, za uredniji izgled. Isti postupak ponavlja se za sve ostale vrijednosti koje se unose.

Analogno lijevoj strani izvodi se i desnu stranu dokumenta, te se istom metodom dolazi do podataka. Preostaje dodavanje podatke o djelatniku i datumu kako bi se upotpunio izgled Microsoft Excel dokumenta. Ovime je očitavanje podataka gotovo, sukladno tome i openpyxl modul je gotov sa svojim doprinosom rješenju.

Nakon ovog koraka grafičko sučelje za automatski unos podataka u potpunosti je završeno i svi potrebni podaci su očitani iz Microsoft Excel datoteke. Ovi koraci nisu bili nužni za izvršenje rješenja problema, no praksa je pokazala da je korisno imati dodatni korak provjere ispravnosti podataka. Grafičko sučelje je baš iz tog razloga uvedeno u rješenje ovog problema, kako bi korisnik imao priliku još jednom prije unosa podataka u bazu ispraviti netočnosti u samim podacima, ako postoje. Time se točnost podataka povećava, kasniji pregled podataka postaje jasniji i potreba za objašnjavanjem među korisnicima svodi se na minimum.

Nastavlja se definiranje putanje do baze podataka koja će se koristiti te se stvara objekt pokazivač koji je potreban za kasnije dodatno upravljanje bazom podataka. Ovisno o tome na kojem sustavu se program izvodi (Unix, macOS, Windows) baza može biti stvorena na različitim lokacijama ako se ne definira točno potpuna putanja do lokacije na kojoj se želi smjestiti baza podataka. Na nekim Linux distribucijama, kao Manjaro ili Linux Mint, prilikom testiranja programa uz definiciju samo naziva baze, ne i putanje, sustav bi smjestio bazu u direktorij u kojem se nalazi i programska datoteka. Druge Linux distribucije, poput Ubuntu ili Fedora, bi smjestile bazu u /home/user/ direktorij. Kako bi se izbjegle nedoumice oko lokacije baze podataka, preporuča se točno definirati punu putanju i naziv za bazu podataka u ovom koraku. Objekt pokazivač može se spremiti u neku varijablu radi lakšeg pisanja koda.

Putanja do baze podataka i pokazivač objekt prikazani su na slici 3.11.

```
1.  conn = sqlite3.connect('/home/marko/zavrsni_rad/izvjestaji_baza.db')
2.  c = conn.cursor()
```

Sl 3.11. Putanja do baze podataka i pokazivač objekt

Prva linija koda spaja korisnika na bazu podataka sa danom putanjom do njezine lokacije ili stvara novu bazu podataka na toj lokaciji ako ona ne postoji. Druga linija koda definira pokazivač koji je potreban za pretraživanje podataka. Bez pokazivača se može samo upisivati u bazu, ali ako korisnik treba pretraživati bazu, potreban je pokazivač.

Definira se funkcija „stvori_tablicu_automatski“ te se koristi stvoreni pokazivač objekt za upis oblika tablice u bazu podataka. Definiraju se nazivi stupaca u tablici te vrsta podatka koji će biti upisani u njih. Ako na toj lokaciji već postoji baza podataka sa strukturom tablice identičnom onoj koju se definira, program koristi postojeću bazu. U protivnom, program će stvoriti novu bazu sa definiranom strukturom.

Stvaranje tablice u bazi podataka ili korištenje postojeće tablice za automatski unos podataka prikazani su na slici 3.12.

```
1.  def stvori_tablicu_automatski():
2.      c.execute("CREATE TABLE IF NOT EXISTS izvjestaj_tablica(inventurni_broj INTEGER)")
```

Sl 3.12. Stvaranje tablice u bazi podataka ili korištenje postojeće tablice

Prva linija koda stvara tablicu, ako ona već ne postoji, sa svim potrebnim unosima. U drugoj liniji koda za svako polje u koje se unosi podatak, definira se vrstu podatka. SQL naredbe pišu se velikim slovima, dok pisanje naziva varijabli, stupaca i ostalih podataka može biti proizvoljno, ali je primjer pisanja dan na slici 3.12 ustaljena praksa. U primjeru koda navedena je samo jedna varijabla i vrsta podatka, no u samom kodu programa ovdje su definirane i sve ostale potrebne varijable.

Nakon što je tablica stvorena ili definirana za korištenje, definira se nova funkcija koja će upisati podatke iz vrijednosti definiranih varijabli u bazu podataka. Funkcija je nazvana „unesi_podatke_automatski“ te ona koristi sve do sada kreirane funkcije za automatski unos. Objekt pokazivač dobiva izvršnu naredbu te u definiranu tablicu i njene stupce izvodi, pomoću sqlite3 modula SQL, naredbe koje zauzvrat koriste openpyxl modul za direktno očitavanje podataka iz Microsoft Excel datoteke i upis u bazu podataka. Dakako, cijeli proces mogao bi se ubrzati izbacivanjem nekih koraka, no kako bi se osigurala što veća

točnost podataka i omogućile ostvarene metode kontrole, program je izveden na ovaj način. Završno, spremaju se podaci i prekida se veza sa bazom podataka.

Unos i spremanje podataka u bazu i prekid veze s bazom prikazani su na slici 3.13.

```
1.  def unesi_podatke_automatski():
2.      c.execute("INSERT INTO izvjestaj_tablica(inventurni_broj) VALUES(?)",
        (sheet.cell(row=5, column=3).value))
3.      conn.commit()
```

Sl 3.13. Unos i spremanje podataka u bazu i prekid veze s bazom

Prva linija koda stvara funkciju za automatski unos podataka u stvorene tablice. Druga linija koda pokazuje kako se točno definira unos, iz kojeg retka i stupca se poziva vrijednost pomoću **`sheet.cell().value`** metode. Treća linija koda potvrđuje izmjene u bazi podataka te zatvara vezu sa bazom podataka.

Kada su svi podaci učitani i korisnik je pregledao podatke na grafičkom sučelju, preostaje samo kliknuti na gumb za pohranu podataka, te ispisati poruku o izvršenim naredbama korisniku.

Definira se jedna funkcija koja će pozivati sve do sada stvorene funkcije, te se ta funkcija veže za pritisak gumba koji se kreira nakon učitavanja podataka. Kada korisnik pritisne gumb sve naredbe će biti izvršene dok će sami gumb nestati i na njegovom mjestu bit će novi label widget koji će ispisati korisniku poruku o ispunjenom automatskom upisu u bazu podataka. Ovim korakom je automatski unos podataka gotov i glavni cilj ovog rješenja ispunjen. Postoji li potreba za daljnjim unosom novih datoteka, nije potrebno prekinuti program nego je dovoljno samo promijeniti putanju do datoteke te ponovno pritisnuti na gumb „Učitaj izvještaj“. Podaci prethodne datoteke bit će spremljeni u bazu podataka, te će podaci nove datoteke biti spremni za pregled i pohranu u bazu podataka.

Završetak automatskog upisa podataka prikazano je na slici 3.14.

```
1.   def automatski_upis(event):
2.         stvori_tablicu_automatski()
3.         unesi_podatke_automatski()
4.         pohrani_podatke_automatski.destroy()
5.   gotovo_automatski_labe=Label(tab1,text="Pohranjeno!").grid(row=33, column=0)
6.   pohrani_podatke_automatski = Button(tab1, width=15, height=1)
7.   pohrani_podatke_automatski.config(text="Pohrani u BP!", fg='red', bg='silver')
8.   pohrani_podatke_automatski.bind("<Button-1>", automatski_upis)
9.   pohrani_podatke_automatski.grid(row=33, column=0)
```

Sl 3.14. Završetak automatskog upisa podataka

Prva linija koda stvara funkciju koju se poziva pritiskom na gumb. Druga i treća linija koda pozivaju ranije stvorene funkcije za automatski unos podataka. Četvrta linija koda briše gumb sa zaslona nakon što je isti aktiviran. Peta linija koda stvara novi label, koji ispisuje poruku o uspješnom spremanju podataka u bazu, na mjestu gdje se prethodno nalazio gumb. Šesta, sedma, osma i deveta linija koda stvaraju gumb koji poziva funkciju automatski_upis, definiraju widget kao gumb, njegovu lokaciju u tab1, njegovu visinu i širinu, tekst ispisan na gumbu kao i njegovu boju. Nakon toga pritisak lijeve tipke miša se povezuje sa gumbom za pozivanje funkcije i naposljetku se definira smještaj gumba na zaslonu.

3.5.Definiranje funkcija, varijabli i podataka potrebnih za ručni unos podataka

Ovaj dio programa nastao je nakon što je postalo očito u izradi rješenja da je logičan nastavak razvoja ovog programa takav da se u potpunosti izbaci Microsoft Excel i počne se izravno podatke upisivati u bazu podataka. Shodno tome ovaj dio programa koristi manje modula za izvedbu, ali ima veću količinu koda za izvođenje. Kako ovdje nije potrebno očitati vrijednosti iz Microsoft Excel datoteke, bilo je potrebno definirati po jednu varijablu za svaki podatak koji je potrebno pohraniti u bazu podataka.

Najprije se definira funkcija „rucni_unos“ koja za argument ima „event“, a ista se izvršava pritiskom na gumb. Unutar te funkcije prvo se definiraju sve potrebne varijable kao **`StringVar()`** tip podatka kako bi kasnije uneseni podaci u polja za ručni unos mogli biti spremljeni u te varijable.

Definiranje dvije varijable za ručni unos podataka prikazano je na slici 3.15.

```
1.  def rucni_unos(event):
2.          djelatnik_rucno = StringVar()
3.          naziv_racunala_rucno=StringVar()
```

Sl 3.15. Definiranje varijabli za ručni unos podataka

Prva linija koda stvara novu funkciju naziva rucni_unos, a druga i treća linija koda definiraju varijable kao **`StringVar()`** vrstu podataka.

Definiraju se preostale potrebne varijable te se dodaje naredba za uklanjanje gumba, definiranog u glavnom dijelu programa. Taj gumb aktivacijom generira grafičko sučelje za ručni unos koristeći sve prethodno definirane funkcije potrebne za izvršenje te zadaće.

Kako u ručnom unosu sve podatke koje je potrebno pohraniti u bazu podataka unosi korisnik, grafičko sučelje stvoreno je tako da približno oponaša izgled Microsoft Excel datoteke a sastoji se od widgeta label i entry. Widget label korisniku daje do znanja koja vrsta podatka se očekuje, a entry widget pruža prostor za upis podataka. Entry widget se povezuje sa definiranim varijablama preko opcije „textvariable“ a na grafičko sučelje se smještaju sa **`grid()`** funkcijom uz dodatne postavke.

Label i entry widget za ručni unos podataka prikazani su na slici 3.16.

```
1.  label_djelatnik = Label(tab2, text="Djelatnik:").grid(row=1, column=0, sticky="e")
2.  entry_djelatnik = Entry(tab2, textvariable=djelatnik_rucno, width=25).grid(row=1,
    column=1)
```

Sl 3.16. Label i entry widget za ručni unos podataka

Prva linija koda dodaje label widget, a druga linija koda dodaje entry widget u tab2. Opcija textvariable uzima vrijednost ranije definiranih varijabli kao **`StringVar()`** kako bi se učitala vrijednost koja je ručno unesena u widget entry.

Postupak je isti i za preostale grupe podataka. Kada su svi potrebni widgeti definirani i povezani sa pripadajućim varijablama te smješteni na grafičko sučelje, potrebno je ostvariti interakciju sa bazom podataka. Kao i kod automatskog unosa prvo se definira putanju do baze podataka na koju se program spaja ili ako ne postoji, stvara se nova na zadanoj lokaciji. Potom se objekt pokazivač sprema u varijablu kratkog imena radi lakše pisanja.

Spajanje na postojeću bazu podataka ili stvaranje nove baze podataka za ručni unos prikazano je na slici 3.17.

```
1.  conn = sqlite3.connect('/home(marko/zavrsni_rad/izvjestaji_baza.db')
2.  c = conn.cursor()
```

Sl 3.17. Spajanje na postojeću bazu podataka ili stvaranje nove za ručni unos

Ovaj dio koda uspostavlja vezu sa bazom podataka na danoj lokaciji te definira pokazivač (engl. *cursor*) potreban za pregled podataka u bazi.

Definira se funkcija za stvaranje tablice unutar baze podataka ili korištenje postojeće za ručni unos pomoću pokazivač objekta. Bitno je paziti da tablica bude definirana sa jednakim nazivima stupaca i vrstom podatka koji se upisuje u stupce kao i za automatski unos podataka kako ne bi došlo do neslaganja u bazi podataka. Funkcija za stvaranje tablice u bazi podataka za ručni unos ili za korištenje postojeće tablice prikazana je na slici 3.18.

```
1.  def stvori_tablicu_rucno():
2.        c.execute("CREATE TABLE IF NOT EXISTS
3.        izvjestaj_tablica(inventurni_broj INTEGER,
4.        djelatnik TEXT, naziv_racunala TEXT)")
```

Sl 3.18. Funkcija za stvaranje tablice u bazi podataka za ručni unos ili korištenje postojeće

U prvoj liniji koda definira se nova funkcija za ručni unos podataka. U ostalim linijama koda stvaraju se tablice unutar baze podataka ako one ne postoje ili koristi postojeće identičnog naziva i vrste podataka.

Slijedeći korak je definiranje funkcije za unos podataka u bazu. Funkciji se dodjeljuje naziv „unesi_podatke_rucno“, definiraju se stupci tablice u koje se unose podaci, za vrijednosti se prvo upisuju znakovi upitnika a potom se definira sama vrijednost prema pravilima sqlite3 modula.

U ovom koraku vrijednosti se dobivaju **`get()`** funkcijom koja očitava vrijednosti iz varijabli koje je korisnik upisao u entry widgete a funkcija „rucni_unos“ je vrijednosti prije toga spremila u varijable. Zato za ručni unos nije bio potreban openpyxl modul ni u kojem trenu, kao ni Microsoft Excel program.

Nakon unosa podataka, spremaju se promjene u bazi i prekida se veza. Upis ručno unesenih vrijednosti u bazu podataka, spremanje promjena i prekid veze prikazani su na slici 3.19.

```
1.  def unesi_podatke_rucno():
2.          c.execute(''INSERT INTO izvjestaj_tablica(inventurni_broj)
                    VALUES(?)'', (inventurni_broj_rucno.get()))
3.          conn.commit()
```

Sl 3.19. Upis ručno unesenih vrijednosti u bazu podataka, spremanje promjena i prekid veze

U prvoj i drugoj liniji koda definira se funkcija koja će učitavati ručno unesene podatke. Ovdje se koristi funkciju **`get()`** za povlačenje podataka, za razliku od automatskog unosa kod kojeg su korištene funkcionalnosti openpyxl modula. Treća linija koda pohranjuje podatke i prekida vezu s bazom podataka.

Nakon definicije potrebnih funkcija, stvara se jednu funkciju koja će pritiskom na gumb pozvati sve ostale, daje joj se naziv „rucni_upis“ te joj se dodaje naredbu da ukloni gumb koji je stvorio grafičko sučelje za ručni unos. Dodaje se gumb za pohranu u bazu podataka koji poziva funkciju „rucni_unos“ i Label widget koji će ispisati poruku o izvršavanju svega zadanog korisniku.

Pozivanje svih funkcija za ručni unos podataka prikazano je na slici 3.20.

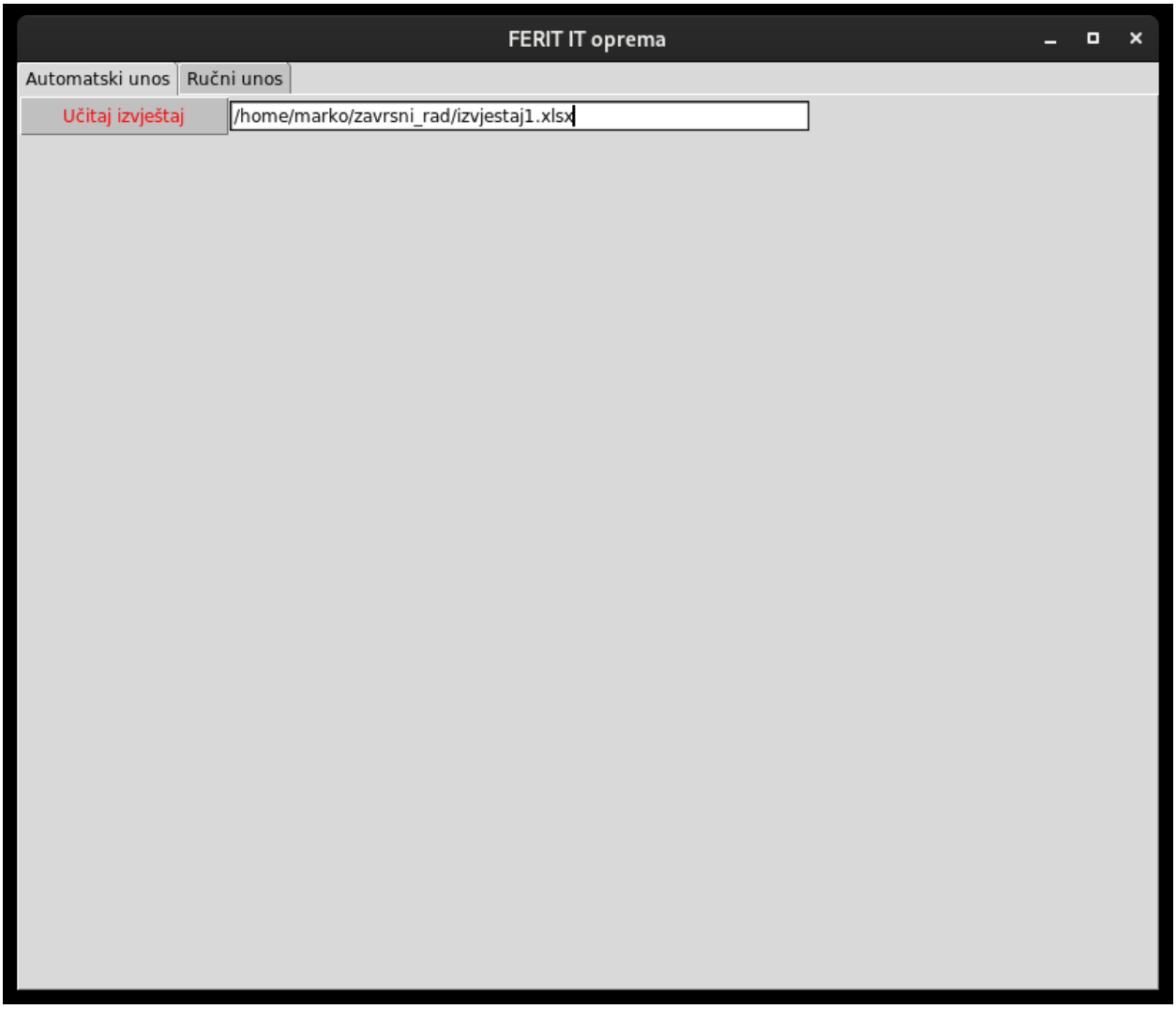
```
1.  def rucni_upis(event):
2.        stvori_tablicu_rucno()
3.        unesi_podatke_rucno()
4.        pohrani_podatke_rucno()
5.        gotovo_rucno_label = Label(tab2, text = „Pohranjeno!").grid(row=29, column=1)
6.  pohrani_podatke_rucno = Button(tab2, width=15, height=1)
7.  pohrani_podatke_rucno.config(text="Pohrani u BP!", fg="red", bg="silver")
8.  pohrani_podatke_rucno.bind("<Button-1>", rucni_upis)
9.  pohrani_podatke_rucno.grid(row=28, column=1, sticky="e")
```

Sl 3.20. Pozivanje svih funkcija za ručni unos podataka

Prva linija koda stvara novu funkciju tipa event koja će biti pozvana klikom na gumb kojeg kreiramo u dijelu koda od šeste do devete linije. Ova nova funkcija će pozivati na izvođenje sve ranije stvorene funkcije vezane za ručni unos podataka, te će nakon aktivacije gumba uništiti taj isti gumb i na njegovom mjestu stvoriti poruku o uspješnoj pohrani podataka u bazu.

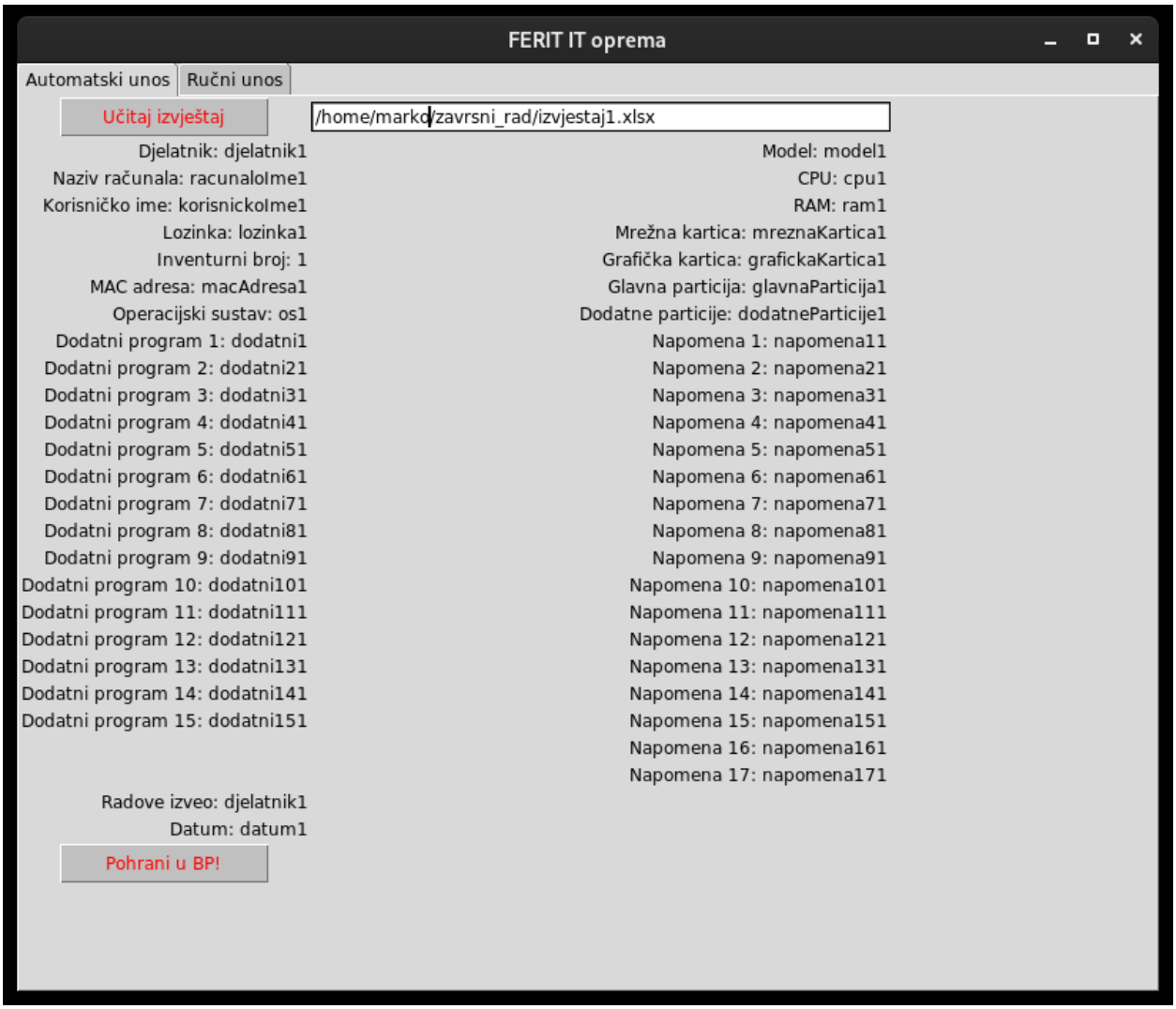
4. GRAFIČKO SUČELJE PROGRAMA

Konačni rezultat cijelog programa je grafičko sučelje u kojem se mogu pregledati podaci koji će se pohraniti u bazu podataka prilikom automatskog unosa iz Microsoft Excel datoteke ili direktnim unosom u Python 3 program pa u bazu podataka.



Sl 4.1 Grafičko sučelje za automatski unos, primjer putanje

Na slici 4.1 prikazan je izgled grafičkog sučelja sa gumbom za unos putanje do Microsoft Excel datoteke. Unesena je potpuna putanja do primjera Microsoft Excel datoteke kakva se koristi.

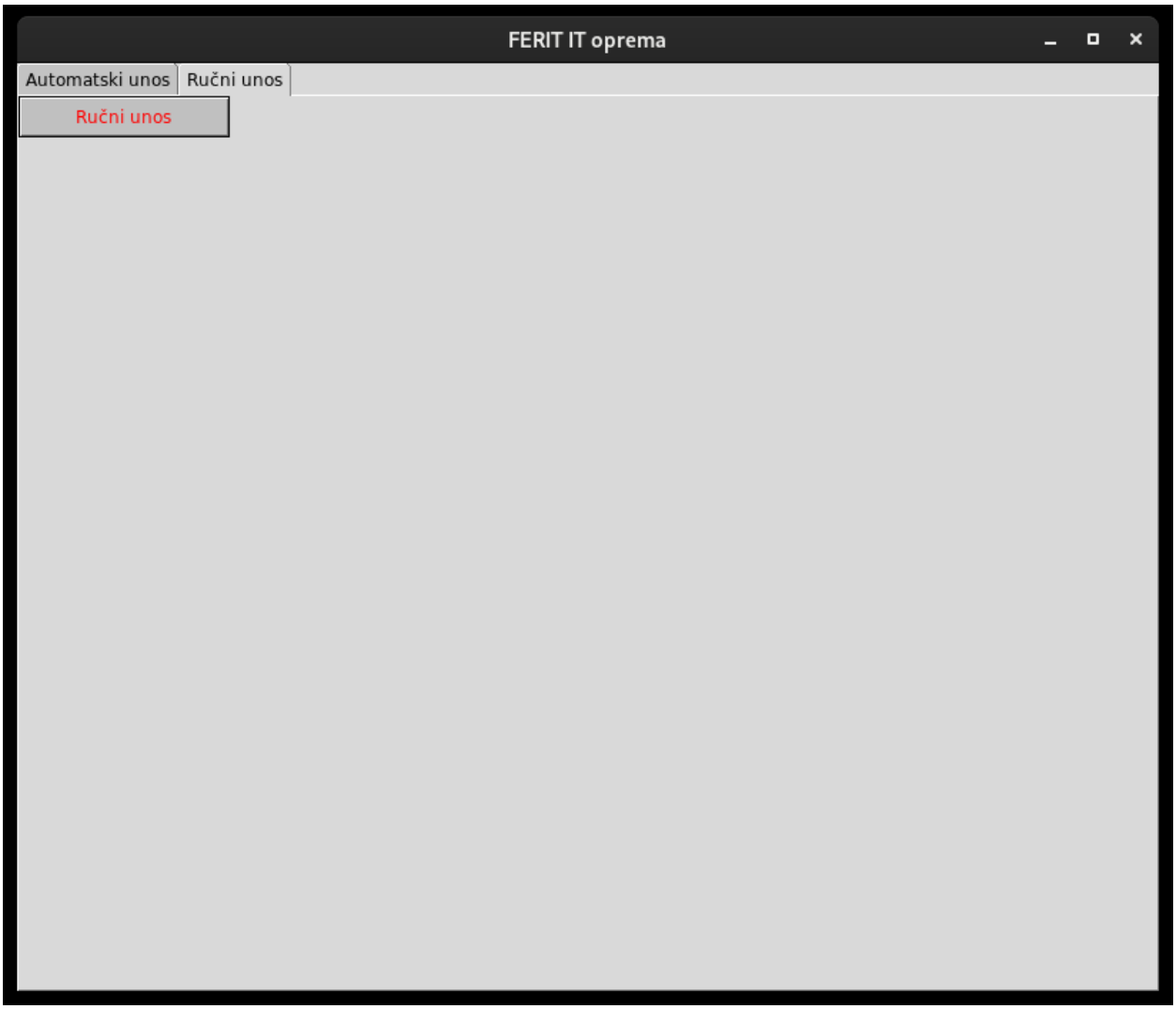


Sl 4.2 Grafičko sučelje za automatski unos nakon učitane Microsoft Excel datoteke

Sl 4.3 Grafičko sučelje za automatski unos nakon pohrane podataka u bazu podataka, ispis poruke o ispunjenom zadatku

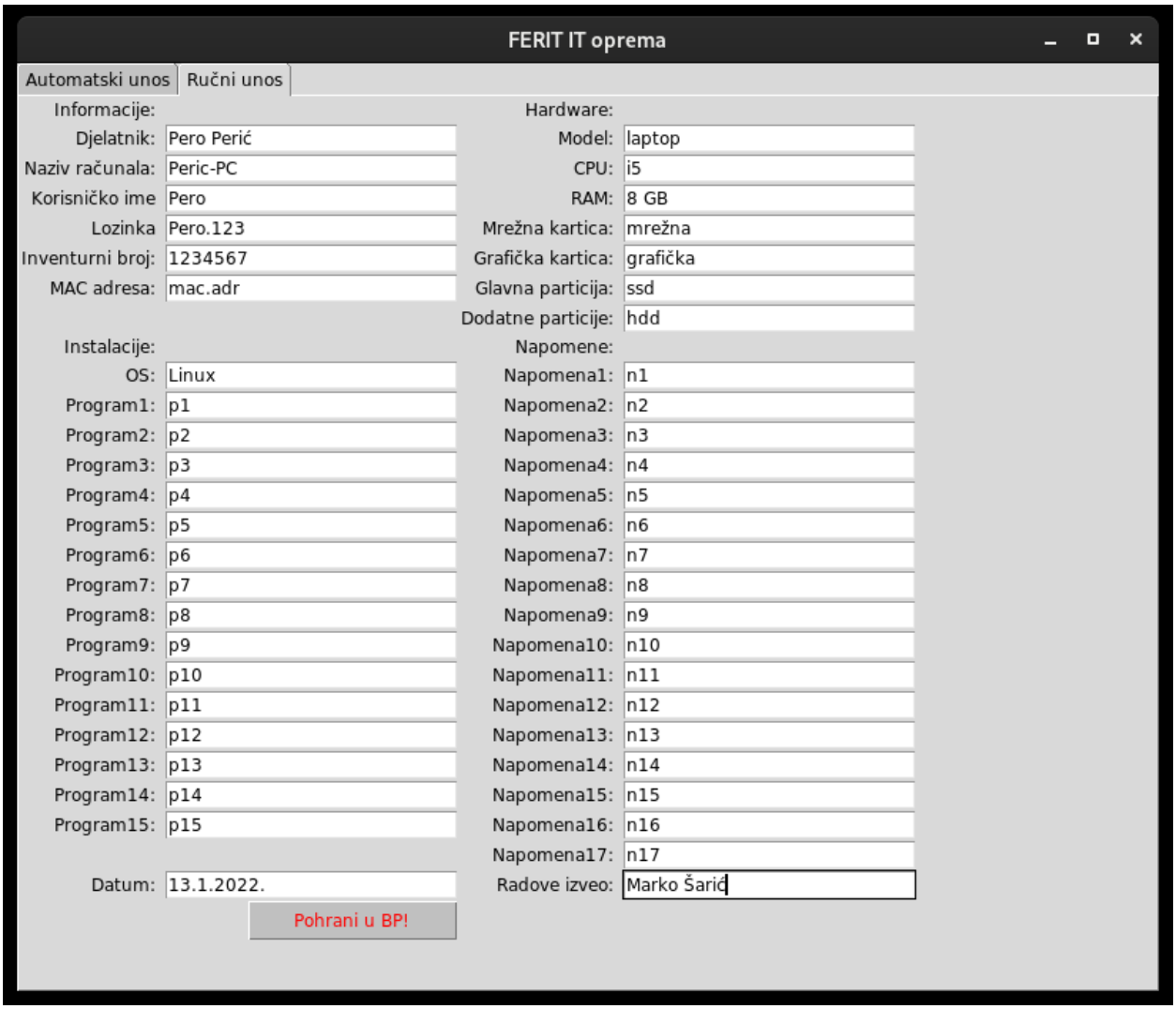
Na slikama 4.2 i 4.3 prikazan je izgled grafičkog sučelja za automatski unos kada se učitaju podaci iz Microsoft Excel datoteke, te nakon što je korisnik pritisnuo gumb „Pohrani u BP!“. Poruka o izvršenoj naredbi se pojavljuje umjesto gumba za pohranu u obliku label widgeta sa tekstom „Pohranjeno!“.

Grafičko sučelje za ručni unos podataka sličnog je izgleda. Obje metode kopiraju izgled Microsoft Excel tablice radi lakšeg snalaženja korisnika.

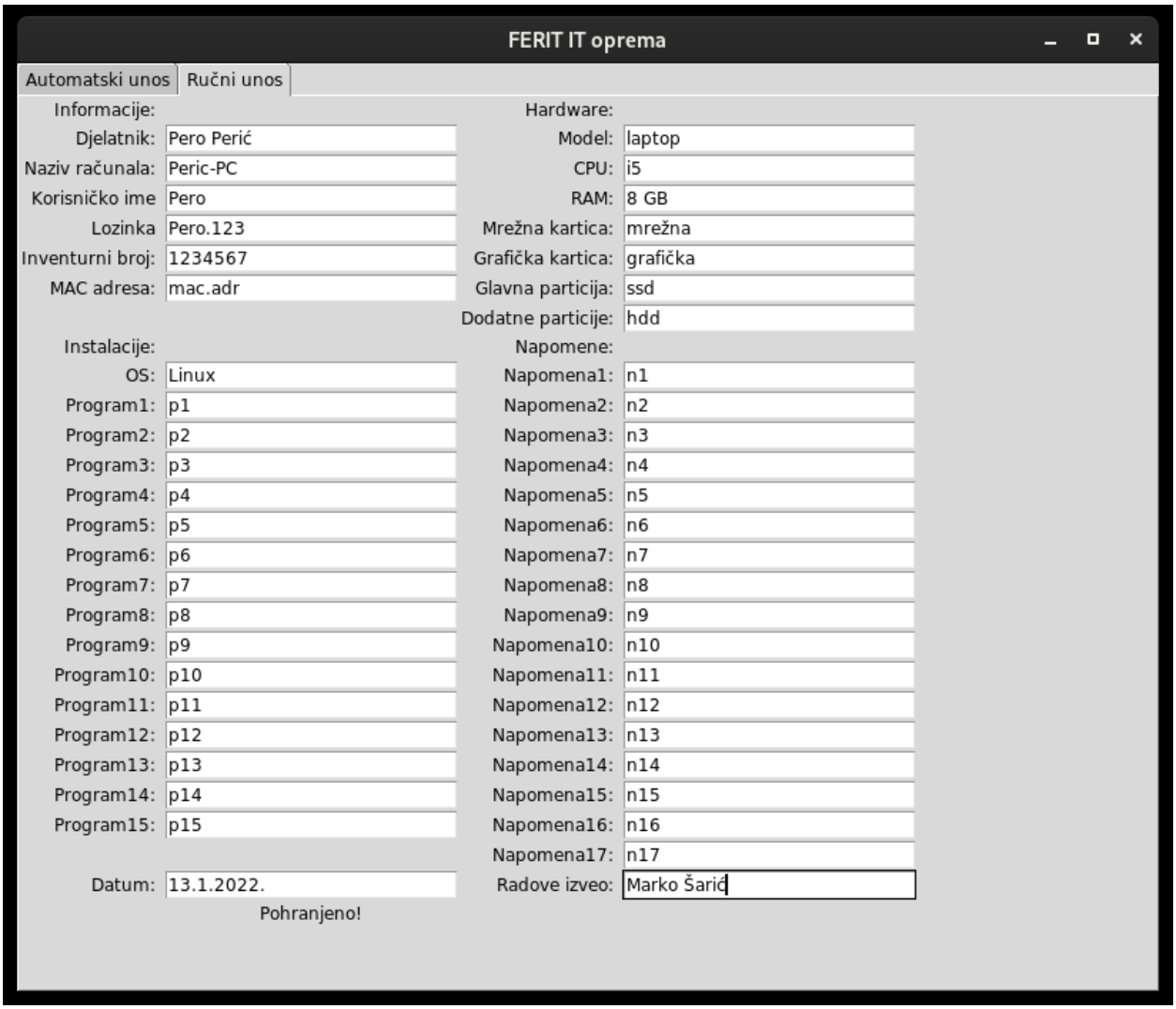


Sl 4.4 Grafičko sučelje za ručni unos podataka, gumb za aktivaciju sučelja

Sučelje za ručni unos nije automatski stvoreno aktivacijom drugog taba jer na ovaj način, u slučaju da korisnik nije planirao ručni upis već je nehotice aktivirao krivu karticu, program brže funkcionira bez generiranog grafičkog sučelja pa je ovime izvođenje programa ujedno i neznatno ubrzano.



Sl 4.5 Grafičko sučelje za ručni unos, prije pohrane u bazu



Sl 4.6 Grafičko sučelje za ručni unos, nakon pohrane u bazu

Na slikama 4.5 i 4.6 prikazan je izgled grafičkog sučelja za ručni unos podataka prilikom ručnog unosa te nakon pohrane podataka.

Baza podataka može se pregledavati raznim programskim paketima od kojih je jedan od korisnijih „DB Browser for SQLite“ open source program. Svi unosi u bazu podataka nakon spremanja ostaju u bazi podataka, bez obzira nastavlja li program sa radom ili ne. Unosom podataka o istoj opremi nekoliko puta u bazu podataka, to jest svaki puta kada su na opremi obavljeni neki radovi, stvaraju se vrlo korisne informacije o opremi kao i povijest radova na njoj. Takva baza podataka lako se izvozi na druge uređaje, moguće ju je kvalitetno zaštititi jer joj za rad nije neophodan internetski pristup, te je izvediva i na vrlo slabim ili starim uređajima koje je teže upotrijebiti u neke druge svrhe.

5. REZULTATI PROGRAMA

Za pregled baze podataka korišten je DB Browser for SQLite [9], alat otvorenog koda, visoke kvalitete sa grafičkim sučeljem za stvaranje, dizajn i izmjenjivanje baza podataka kompatibilnih sa SQLite. Koriste ga korisnici i developeri koji žele stvarati, pretraživati te izmjenjivati svoje baze podataka. Grafičko sučelje koristi poznati izgled poput Microsoft Excel datoteke te omogućuje upravljanje bazama podataka bez poznavanja SQL naredbi.

Program sadrži mnoge opcije kao što su: stvaranje i sažimanje baze podataka, stvaranje, definiranje, uređivanje i brisanje tablica, stvaranje, definiranje i brisanje indeksa, pretraživanje zapisa, uvoz i izvoz zapisa kao tekst, uvoz i izvoz zapisa u CSV datoteku. U ovom radu DB Browser for SQLite korišten je isključivo za pretraživanje zapisa.

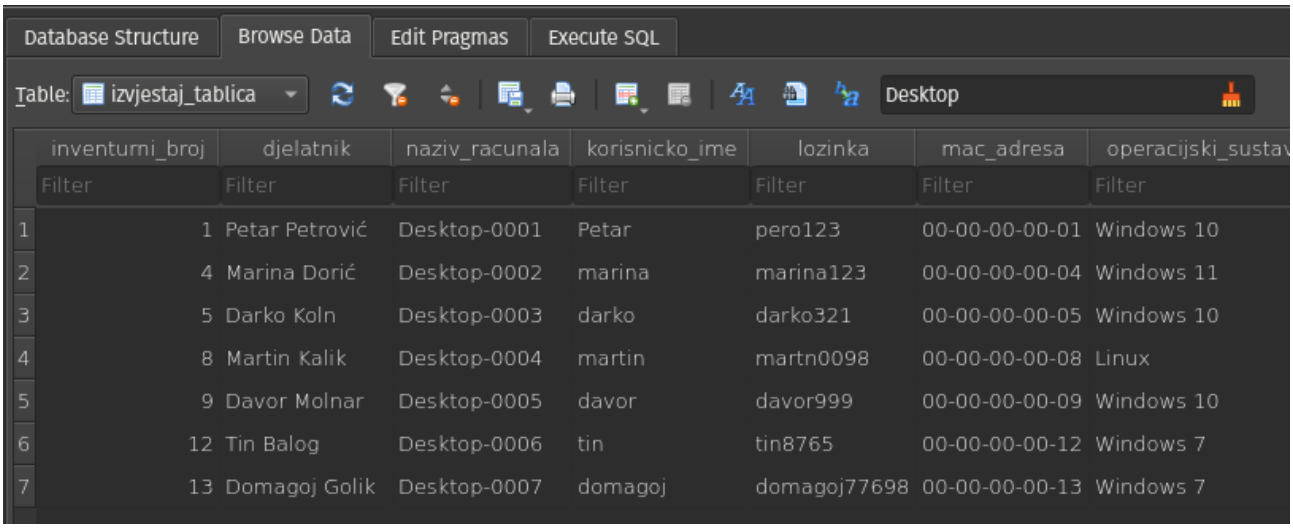
Nakon što su razvijenim programskim rješenjem podaci uneseni u bazu podataka, automatski ili ručnom metodom, rezultati mogu biti pregledani korištenjem DB Browser for SQLite programa. U slijedećim primjerima prikazani su rezultati rada programskog rješenja. Stvorena je baza podataka sa unosima iz 15 Microsoft Excel datoteka kako bi se prikazale mogućnosti razvrstavanja podataka primjenom određenog filtera. Zbog zaštite podataka, podaci koji će biti prikazani kao rezultat ovog predmetnog rješenja su u potpunosti izmišljeni i ne odnose se na stvarne osobe i informatičku opremu.

Na slici 5.1 prikazan je primjer baze podataka sa 15 izmišljenih unosa Microsoft Excel datoteka.

	inventurni_broj	djelatnik	naziv_racunala	korisnicko_ime	lozinka	mac_adresa	operacijski_sustav
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Petar Petrović	Desktop-0001	Petar	pero123	00-00-00-00-01	Windows 10
2	2	Marko Marković	Markovic-NB	marko	marko123	00-00-00-00-02	Windows 10
3	3	Ivona Morak	Morak-NB	ivona	ivona123	00-00-00-00-03	Windows 11
4	4	Marina Dorić	Desktop-0002	marina	marina123	00-00-00-00-04	Windows 11
5	5	Darko Koln	Desktop-0003	darko	darko321	00-00-00-00-05	Windows 10
6	6	Marin Levak	Levak-NB	marin	marin456	00-00-00-00-06	Windows 11
7	7	Denis Krolo	Krolo-NB	denis	denis222	00-00-00-00-07	Linux
8	8	Martin Kalik	Desktop-0004	martin	martn0098	00-00-00-00-08	Linux
9	9	Davor Molnar	Desktop-0005	davor	davor999	00-00-00-00-09	Windows 10
10	10	Igor Novak	Novak-NB	igor	igor123	00-00-00-00-10	Windows 8.1
11	11	Marija Livaja	Livaja-NB	marija	marija444	00-00-00-00-11	Windows 8.1
12	12	Tin Balog	Desktop-0006	tin	tin8765	00-00-00-00-12	Windows 7
13	13	Domagoj Golik	Desktop-0007	domagoj	domagoj77698	00-00-00-00-13	Windows 7
14	14	Vedran Bajer	Bajer-NB	vedran	vedran123	00-00-00-00-14	Windows 7
15	15	Danijel Brezak	Brezak-NB	danijel	danijel897	00-00-00-00-15	Windows 11

Slika 5.1 Primjer baze podataka sa unosima Microsoft Excel datoteka

Ovakvo rješenje pruža korisniku brzi pregled svih potrebnih podataka sa opcijama filtriranja. Pojavi li se potreba za informacijom o ukupnoj količini dostupnih računala, jednostavnom primjenom filtera za pretragu korisnik može doći do te informacije, bez potrebe za pregledom velike količine pojedinačnih datoteka. Na slici 5.2 prikazan je prikaz podataka sa primijenjenim filterom “Desktop” kako bi se dobio ukupan broj desktop računala.

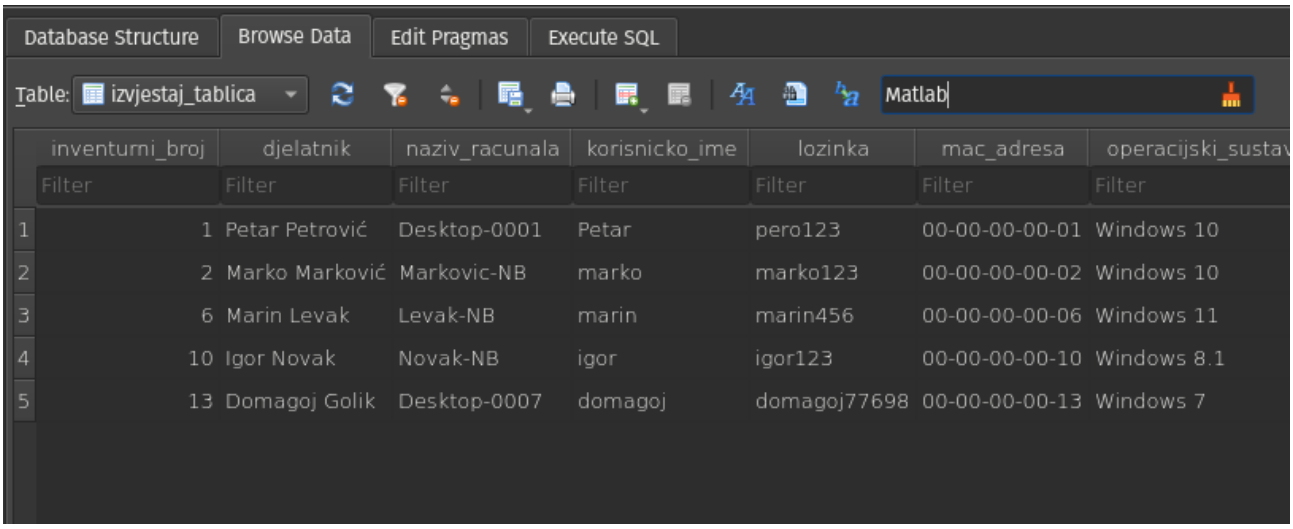


	inventurni_broj	djelatnik	naziv_racunala	korisnicko_ime	lozinka	mac_adresa	operacijski_sustav
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Petar Petrović	Desktop-0001	Petar	pero123	00-00-00-00-01	Windows 10
2	4	Marina Dorić	Desktop-0002	marina	marina123	00-00-00-00-04	Windows 11
3	5	Darko Koln	Desktop-0003	darko	darko321	00-00-00-00-05	Windows 10
4	8	Martin Kalik	Desktop-0004	martin	martn0098	00-00-00-00-08	Linux
5	9	Davor Molnar	Desktop-0005	davor	davor999	00-00-00-00-09	Windows 10
6	12	Tin Balog	Desktop-0006	tin	tin8765	00-00-00-00-12	Windows 7
7	13	Domagoj Golik	Desktop-0007	domagoj	domagoj77698	00-00-00-00-13	Windows 7

Slika 5.2 Pretraga podataka sa filterom “Desktop”

Prilikom rada može se pojaviti potreba za ažuriranjem određenog programa na računalima. Pojavi li se uz to i potreba za kupovinom licence za ažuriranje programa, korisnik ovog rješenja može primijeniti filter sa imenom potrebnog programa te u par sekundi dobiti informaciju o količini potrebnih licenci za ažuriranje. Alternativa ovome je otvaranje velike količine datoteka te ručni pregled svake datoteke ili fizički odlazak do svakog računala i provjera.

Na slici 5.3 prikazan je primjer pretrage podataka sa filterom “Matlab”.



	inventurni_broj	djelatnik	naziv_racunala	korisnicko_ime	lozinka	mac_adresa	operacijski_sustav
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Petar Petrović	Desktop-0001	Petar	pero123	00-00-00-00-01	Windows 10
2	2	Marko Marković	Markovic-NB	marko	marko123	00-00-00-00-02	Windows 10
3	6	Marin Levak	Levak-NB	marin	marin456	00-00-00-00-06	Windows 11
4	10	Igor Novak	Novak-NB	igor	igor123	00-00-00-00-10	Windows 8.1
5	13	Domagoj Golik	Desktop-0007	domagoj	domagoj77698	00-00-00-00-13	Windows 7

Slika 5.3 Pretraga podataka sa filterom „Matlab“

Dostupnost podataka je ovim programskim rješenjem mnogostruko poboljšana. Mogućnosti stvaranja novih korisnih podataka za korisnika su znatno veće, a brzina i učinkovitost upravljanja informacijama su višestruko poboljšane.

6. ZAKLJUČAK

Cilj ovog završnog rada je automatizirati proces spremanja podataka iz mnogobrojnih Microsoft Excel datoteka u jednu jednostavnu bazu podataka zbog lakše pretrage. Prilikom izrade samog rada uočeno je da bi od velike koristi bila funkcija ručnog unosa podataka u istu bazu te je ta funkcija i dodana. Time je ostvaren značajan napredak u pregledu i pretraživanju potrebnih informacija, sigurnosti informacija te prijenosu velike količine podataka u drugi medij.

Pogodnosti ovog programskog rješenja su mnogobrojne od kojih su korisniku najznačajnije brzina i učinkovitost pretrage, kao i opcija razvijanja funkcionalnosti dodavanjem novih alata i mogućnosti u budućnosti. Ovim radom stvoren je temelj na kojem se može razviti naprednija aplikacija sa većim brojem primjena. Koristeći module koji su već prisutni moguće je napraviti sustav generiranja izvještaja u Microsoft Excel datoteku korištenjem podataka spremljenih u bazi podataka, što se nameće kao prvo najkorisnije proširenje funkcionalnosti.

Iako je nekoliko puta u radu rečeno kako ovaj program može funkcionirati bez spajanja na internet pa je time relativno siguran, naravno da bi se korisnost programa povećala uvođenjem modula koji bi program pretvorili u web aplikaciju, poput Django i Flask modula. Time bi se omogućilo kreiranje i baze ovlaštenih korisnika koji imaju pristup bazi podataka, kao i korištenje neovisno o lokaciji. Kao što vidimo, ovaj rad je ispunio svoj zadani cilj te je u isto vrijeme stvorio solidan temelj za mnogobrojna proširenja mogućnosti.

7. LITERATURA

[1] Službena stranica Investopedy [Mrežno]. Dostupno: https://www.investopedia.com/best-inventory-management-software-5088863 (Pokušaj pristupa 6.7.2022.)

[2] Službena stranica Zoho Inventory [Mrežno]. Dostupno: https://www.zoho.com/inventory/ (Pokušaj pristupa 6.7.2022.)

[3 Russell J.T. Dyer, MySQL in a Nutshell, Second Edition, O'Reilly Media, Travanj 2008.

[4] Web stranica tom'sHARDWARE [Mrežno]. Dostupno: https://www.tomshardware.com/how-to/create-python-executable-applications (Pokušaj pristupa 25.8.2022.)[5] M. Lutz, Learning Python, O'Reilly Media, 6. srpanj 2013.

[6] Službena stranica tkinter modula [Mrežno]. Dostupno: https://docs.python.org/3/library/tkinter.html (Pokušaj pristupa 15.2.2022.)

[7] Službena stranica openpyxl modula [Mrežno]. Dostupno: https://openpyxl.readthedocs.io/en/stable/ (Pokušaj pristupa 15.2.2022.)

[8] Službena stranica sqlite3 modula [Mrežno].Dostupno: https://www.sqlite.org/index.html (Pokušaj pristupa 15.2.2022.)

[9] Službena stranica DB Browser for SQLite [Mrežno]. Dostupno: https://sqlitebrowser.org/ (Pokušaj pristupa 1.9.2022.)

8. SAŽETAK

Naslov: Upravljanje sqlite3 bazom podataka pomoću Python3

Tema ovog rada je problem pohranjivanja veće količine podataka o održavanju računalne opreme koji se vode u pojedinačnim Microsoft Excel datotekama, s obzirom da takav način pohrane podataka nije prikladan za učinkovitu pretragu, kao ni za točno i jasno vođenje evidencije održavanja opreme. Pohranjivanjem svih podataka iz pojedinačnih Microsoft Excel datoteka u jednu bazu podataka riješio bi se postojeći problem otežane pretrage i korištenja podataka.

Prvenstveno je bilo potrebno omogućiti unos već postojećih podataka iz Microsoft Excel datoteka, ali i omogućiti izravan unos novih podataka u bazu. Rješenje je ostvareno upotrebom Python3 programskog jezika te modula tkinter, openpyxl i sqlite3. Python3 automatizira proces, tkinter modul pruža grafičko sučelje, openpyxl modul omogućuje interakciju sa Microsoft Excel datotekama, a sqlite3 modul omogućava sve radnje potrebne za bazu podataka.

Programsko rješenje ispunilo je cilj automatizacije pohrane već postojećih podataka te je ujedno pružilo i dodatnu funkcionalnost za unos novih podataka. Uz navedeno, ostvaren je i dobar temelj za razvoj daljnjih funkcionalnosti koje će poboljšati kvalitetu ovog programskog rješenja i pružiti više mogućnosti korisnicima.

Ključne riječi: Python 3, openpyxl, sqlite3, tkinter, Excel, baza podataka, automatizacija

9. ABSTRACT

Title: Sqlite3 database management using Python3

This paper focused on the problem of storing large amounts of computer maintenance data from individual Microsoft Excel files. Storing data in individual Microsoft Excel files facilitates neither an efficient data search, nor keeping an accurate and clear account of equipment maintenance. Saving all data from individual Microsoft Excel files into a single database would solve the existing problem of inefficient data search and data usage.

The primary requirement was to enable import of existing data from Microsoft Excel files, but also to enable direct entry of new data into the database. This was achieved by using Python3 programming language and Python3 modules tkinter, openpyxl and sqlite3. Python3 handles the automation process, the tkinter module provides the graphical interface, the openpyxl module enables interaction with Microsoft Excel, and the sqlite3 module allows for all necessary database management.

This program achieved its primary goal of automating existing data storage, it also added the functionality of inputting new data. In addition, it established the basis for development of further functionalities that will increase the program's overall quality and provide more options for users.

Keywords: Python3, openpyxl, sqlite3, tkinter, Excel, database, automation