

АКАДЕМИЈА ТЕХНИЧКО-УМЕТНИЧКИХ СТРУКОВНИХ СТУДИЈА БЕОГРАД

ОДСЕК ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВА

**Рилак Никола**

***PYTHON* АПЛИКАЦИЈА ЗА ЕВИДЕНЦИЈУ ЗАПОСЛЕНИХ**

**- завршни рад –**



Београд, новембар 2021.

Кандидат: **Рилак Никола**

Број индекса: **73/18**

Студијски програм: **Нове рачунарске технологије**

Тема: **Python апликација за евиденцију запослених**

Основни задаци:

- 1. Опис технологија клијентске и серверске стране.**
- 2. Имплементација Python апликације за евиденцију запослених.**
- 3. Опис корисничког интерфејса.**

Ментор:

Београд, новембар 2021 године.

---

Др. Перица Штрбац, професор АТУСС

## **РЕЗИМЕ:**

Софтвер за евиденцију запослених је направљен са идејом и циљем да омогући једној фирми да на лакши начин испрати све информације о запосленим. На овај начин, одговорна особа има простора да руководи базом података, прегледава податке и има увид у сва дешавања у фирми. Сам софтвер омоућава надлежној особи разне могућности као и увид у неке битне информације, а неке од њих су:

- време доласка, одласка радника и укупно време проведено на радном месту;
- све информације о запосленом, као што су његови лични подаци и фотографија;
- манипулација базом података, додавање нових запослених, брисање и претрага.

Софтвер сам по себи није захтеван у смислу да не захтева додатну обуку запосленог и тиме сам себе чини лакшим за комерцијализацију и употребу у глобалу.

**Кључне речи:** Софтвер, база података, идентификација, запослени

## **ABSTRACT:**

Employee evidentication software was created with the idea and goal of enabling one company to monitor all the data and information about the employees in an easier way. In this way, the responsible person has the space to manage the database, review the data and have an insight into all events in the company. The software itself provides the competent person with various options as well as insight into some important information, and some of them are:

- time of arrival, departure of workers and total time spent at the workplace;
- all information about the employee, such as his personal data and photo;
- database manipulation, adding new employees, deleting and searching.

The software itself is not required in the sense that it does not require additional employee training and thus makes itself easier to commercialize and use globally.

**Key words:** Software, databse, identification, employee

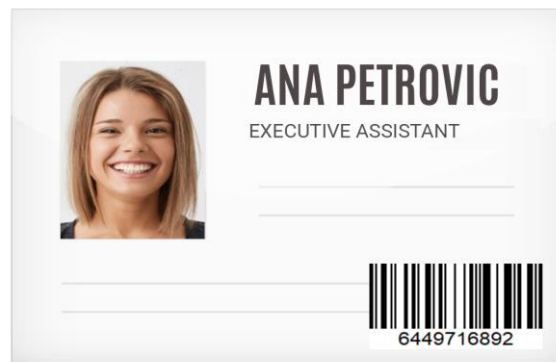
## САДРЖАЈ:

<u>1.</u>	<u>УВОД</u>	5
<u>2.</u>	<u>ОПИС СОФТВЕРА</u>	6
<u>2.1.</u>	<u>Кориснички део интерфејса</u>	6
<u>2.2.</u>	<u>Администраторски део интерфејса</u>	9
<u>2.2.1.</u>	<u>Основне опције</u>	12
<u>2.2.2.</u>	<u>Напредне опције</u>	16
<u>3.</u>	<u>ПОСТУПАК ИЗРАДЕ</u>	25
<u>3.1.</u>	<u>Идеја софтвера</u>	25
<u>3.2.</u>	<u>Реализација идеје</u>	25
<u>3.3.</u>	<u>Проблеми настали током имплементације</u>	28
<u>4.</u>	<u>ПРАКТИЧНА ПРИМЕНА СОФТВЕРА</u>	29
<u>4.1.</u>	<u>Метода употребе</u>	29
<u>5.</u>	<u>ЗАКЉУЧАК</u>	30
<u>6.</u>	<u>ИНДЕКС ПОЈМОВА</u>	31
<u>7.</u>	<u>ЛИТЕРАТУРА</u>	32
<u>8.</u>	<u>ПРИЛОЗИ</u>	33
<u>9.</u>	<u>ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ</u>	34

## 1. УВОД

Софтвер за идентификацију запослених је систем осмишљен тако да се на лак начин за брзо време изврши било која провера запосленог и његових информација као што су време доласка, радни сати или његове име и датум рођења. Идеја се остварује тако што сваки запослени добија своју личну идентификациону картицу а то можемо видети на слици 1.1. и са њом се пријављује/одјављује за посао. Пријава и одјава се врше једним скенирањем картице, која шаље податке администратору односно руководиоцу програма.

Поред своје личне фотографије, имена, презимена, датума рођења и позиције на послу, запослени такође добија свој јединствени број за идентификацију који је представљен у бар коду. Број се састоји од 10 насумично одабраних цифара и притом је сам по себи јединствен. Уз помоћ овог броја, сваки запослени је заведен у бази података и на тај начин се читају његови подаци. Пример погледати на фотографији 1.1.



Слика 1.1 Пример идентификационе картице

Главна замисао јесте да на самом улазу у фирму постоји панел са бар код читачем где се запослени пријављују и одјављују. Поред панела, физички би се сигурно додало зелено и црвено светло које означава успех и грешку. На тај начин се без проблема успоставља редовно уписивање радних сати као и све остале битне информације које касније служе и прослеђују се другим кадровима, нпр. кадру за људске ресурсе, одакле могу читати месечне радне сате на основу којих праве обрачун примања.

Будући да потребе данашњих организација расту и да рачунари и софтвери свакога дана замењују неке послове радника који су се некада радили ручно, са лакоћом се може рећи да је овакав софтвер потребан свакој фирми ради лакше организације, руковођења и увида у информације. Физичке потребе оваквог софтвера нису презахтевне, с обзиром да је потребан само један сервер и барем два уређаја за пријаву и одјаву. Употреба је поједностављена и осмишљена тако да буде једноставна, тако да је софтвер идеалан за употребу у некој мањој фирми, сходно са његовим тренутним опцијама.

У глави опис софтвера говоримо о детаљном опису свих интерфејсних и кодовских аспеката, укључујући и слике и детаљна објашњења.

Глава поступак израде се састоји од детаљног описа свих имплементација над софтвером, где се детаљно описује сваки корак израде и додатно објашњавају ствари које су важне за апликацију.

У практичној примени софтвера, говоримо о употреби софтвера у данашњици као и методе употреба, где се додатно објашњавају начини употребе.

## 2. ОПИС СОФТВЕРА

Пре свега, сама идеја је настала као резултат потребе организације и брзог приступа подацима. Иако је у већини данашњих компанија сасвим сигурно имплементиран овакав систем, сигуран сам да је у мањим локалним фирмама, где је број радника између 10 и 20, заиста неопходан овакав тип организације. За коришћење софтвера, како администратора тако и радника, није потребно никакво обучавање јер је софтвер прилагођен тако да било ко може да рукује. Важно је поменути да администратор има много више опција у самом софтверу него запослени, односно радник.

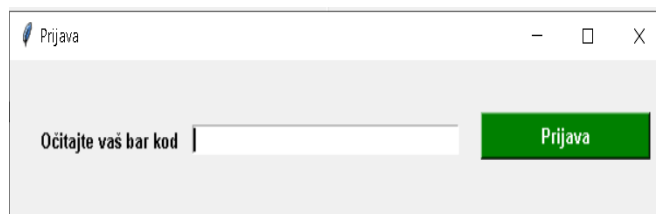
### 2.1. Кориснички део интерфејса

Кориснички део захтева претходно укључен сервер, односно администраторски део софтвера. Након укљученог сервера, кориснички део се повезује и остварује конекцију са истим, након тога софтвер је у потпуности спреман за коришћење.



Слика 2.1.1 Кориснички део интерфејса

Панел односно интерфејс за корисника сам по себи садржи само основне неопходне функције као што су пријава, одјава и увид у тренутно време (погледати слику 2.1.1.). Након одабира једне од ове опције (пријава/одјава), кориснику се нуди да прочита или упише свој јединствени број односно бар код, што можемо видети на слици 2.1.2.



Слика 2.1.2 Панел за унос бар кода

Важно је напоменути да је панел за унос бар кода опције одјаве скроз идентичан као панел пријаве. У продужетку можемо видети код за пријаву који се састоји из фазе провере бар кода, читање базе и слање информација серверу.

## prijava.py

- Повезивање на сервер

```
host = "127.0.0.1" #koristimo ip adresu lokalnog hosta
port = 12345 #uzimamo port 12345 pod pretpostavkom da je otvoren
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
def poveziSe(): #funkcija za povezivanje na server
    s.connect((host, port)) # kreiramo objekat s u koji smestamo dve glavne stavke,
                           # prva oznacava ipv4 adresu dok druga oznacava da je u
                           # pitanju tcp protokol
    print("Povezan na: ", host)
mainThread = threading.Thread(target=poveziSe, args=()).start() #smestanje funkcije u nit
```

## prijava.py

- Провера бар кода

```
flag = "PRIJAVA" #koristimo ovaj flag da bi server znao koji podatak da procita. U slucaju odjave,
flag bi bio "ODJAVA"
def proverBarkod(): #Proveravamo unos odnosno bar kod
    if len(e1.get()) == 0: #postavljanje svih uslova kako ne bi doslo do greske
        messagebox.showerror(title="Greška", message="Niste uneli barcode!")
    elif len(e1.get()) > 1:
        conn = sqlite3.connect("zaposleni.db")
        cursor = conn.cursor()
        bar_kod = e1.get()
        upit = f"""
                SELECT *
                FROM zaposleni
                WHERE id = '{bar_kod}' AND naPoslu = 'Ne'
            """
        cursor.execute(upit)
        rezultat = cursor.fetchall()
        conn.close()
        if rezultat:
            messagebox.showinfo(title="Uspešna prijava", message="Uspešno ste se prijavili za
svoje radno vreme.")
            prijava()
            sys.exit()
```

```
else:
    messagebox.showerror(title="Greška", message="Nije pronađen nijedan zaposlen ili ste već prijavljeni!")
```

## prijava.py

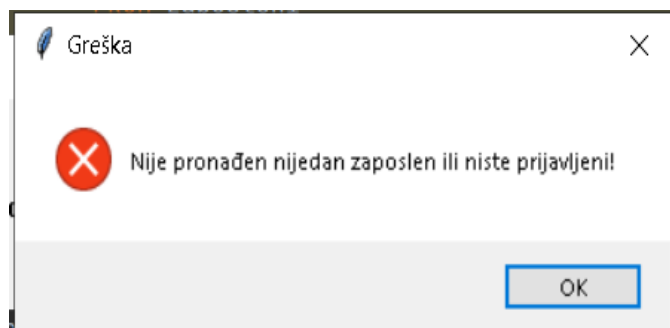
- Слање повратних информација серверу

```
def prijava(): # Saljemo podatke server sa flagom "PRIJAVA" kako bi server znao da se
               #klijent prijavljuje

    #upit
    radniSati = "START"
    zahtev = e1.get() + " " + flag + " " + radniSati
    s.sendall(zahtev.encode())

# FORMAT ---- > 11/11/2021 - 22:03:36: zaposleni 6449716892 se PRIJAVIO.
```

Међутим уколико бар код није тачан или уколико тај радник једноставно не постоји у бази података, запосленом излази порука у којој га обавештава да је настала грешка и тај бар код није валидан или да једноставно је већ пријављен/одјављен. (погледати слику 2.1.3 и слику 2.1.4.).

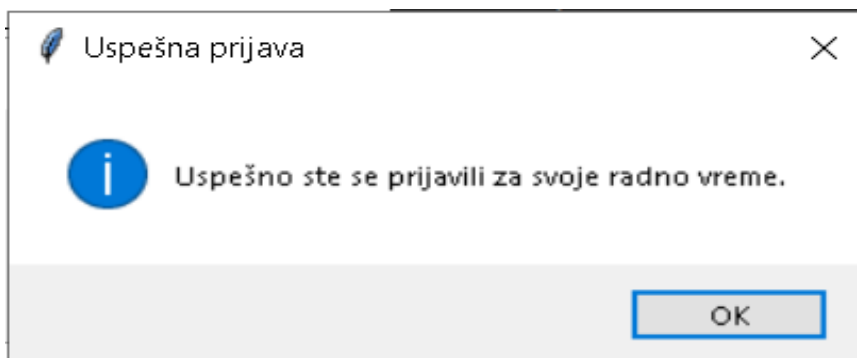


Слика 2.1.3 Грешка приликом одјаве



Слика 2.1.4 Грешка приликом пријаве

Уколико је ипак бар код валидан и уколико се запослени пријавио успешно, излази му порука са текстом да је успешно пријављен и од тог тренутка тече његово радно време и то можемо видети на слици 2.1.5.

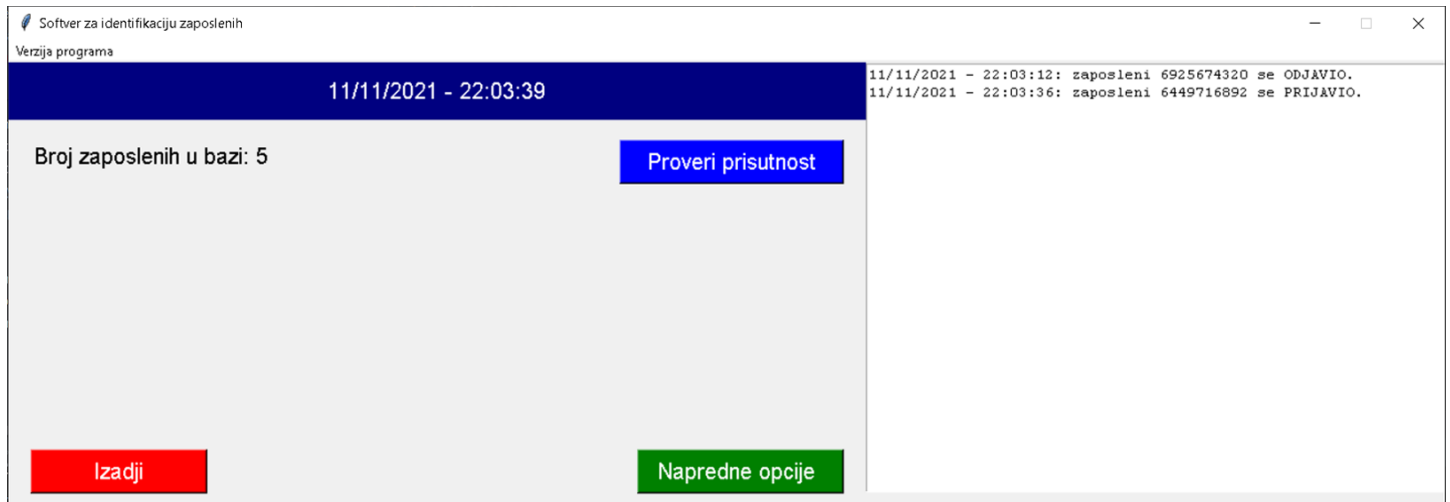


Слика 2.1.5 Успешна пријава



## 2.2. Администраторски део интерфејса

За разлику од корисничког интерфејса, админ панел односно прозор администраторски део интерфејса је доста напреднији и сложенији. Поседује разне опције као што су манипулација базом података и испитивање присутности запослених што можемо и видети на слици 2.2.1. Сам администраторски део је заправо и серверски део софтвера, преко којег се „качи“ интерфејс који смо раније поменули. Дакле, могло би се рећи да је ово главни део софтвера из којег произилазе остали сегменти. Интерфејс је лак за употребу и није потребна било каква обука, бар не у овој тренутној верзији.



Слика 2.2.1 Администраторски део интерфејса

### main.py

- Структура главног менија користећи ткинтер (енг. Tkinter) библиотеку

```
root = tk.Tk()# root objekat u koji smestamo glavni prozor tkinter biblioteke

root.geometry("1300x400") # geometrija

root.title("Softver za identifikaciju zaposlenih")
root.resizable(False, False) # postavljamo mogucnost sirenja prozora na false

def verzijaPrograma(): # mala funkcija za prikaz verzije programa ugradjena u glavni
                        # meni prozora
    messagebox.showinfo(title="Verzija", message="v0.1")
```

```

main_menu = tk.Menu(root)
root.config(menu=main_menu)
main_menu.add_command(label='Verzija programa', command=verzijaPrograma)

root.var = tk.StringVar() # promenljiva u kojoj se smesta vrednost labele za prikaz
                           broja zaposlenih

# kontrole
label1 = tk.Label(root, text="Vreme", fg="white", bg="navy", width=70, font="15", height=2)
label1.grid(row=0, column=0, sticky=tkinter.N)

label2 = tk.Label(root, textvariable=root.var, font="10")
label2.grid(row=0, column=0, sticky=tkinter.NW, pady=70, padx=(20, 0))

koJePrisutan = tk.Button(root, padx=15, width="15", text="Proveri prisutnost", bg="blue",
fg="white", command=koJeTu, font="2")
koJePrisutan.grid(row=0, column=0, sticky=tkinter.NE, pady=70, padx=(0, 20))

nazad = tk.Button(root, padx=50, text="Izadji", bg="red", fg="white", command=ugasiProgram,
font="10")
nazad.grid(row=0, column=0, sticky=tk.NW, pady=(350, 0), padx=(20, 0))

napredneOpcijeDugme = tk.Button(root, padx=15, text="Napredne opcije", bg="green", fg="white",
command=napredneOpcije, justify=tk.LEFT, anchor="w", font="10")
napredneOpcijeDugme.grid(row=0, column=0, sticky=tkinter.NE, pady=(350, 0), padx=(0, 20))

text1 = tkinter.Text(root, width="70")
text1.grid(row=0, column=1)

```

## main.py

- Функција за слушање конекција и порука клијента

```

local_pc = "" # ip adresu odnosno host postavljamo na prazno zato sto server sam
               uzima svoju ip adresu

port = 12345 # uzimamo port 12345 pod pretpostavkom da je slobodan

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # kreiramo objekat s u koji smestamo
               dve glavne stavke, prva oznacava ipv4 adresu dok druga oznacava da je u pitanju tcp protokol

```

```

s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1) # postavljamo mogucnost
                    ponovnog koriscenja iste adrese

def slusajKonekcije(): # funkcija za slusanje svih konekcija, koja se kasnije smesta u nit
    s.bind((local_pc, port)) # postavljanje prethodno definisane ip adrese I porta u nas objekat
    odnosno socket

    s.listen() # postavljamo objekat u stanje slusanja novih konekcija

    connection = sqlite3.connect("zaposleni.db") # povezivanje sa bazom podataka

    cursor = connection.cursor() # pravljenje objekta uz pomoc kojeg rukovodimo bazom

    print("Slusam...")

    global start_time

    while True: # petlja koja regulise uspesno povezivanje klijenta

        conn, addr = s.accept() # dva najvaznija objekta uz pomocu kojih mozemo da vidimo sadrzaj
        poruke I adresu sa koje je poslata poruka

        print("Nadolazeca konekcija: ", addr)

        try:

            while True:

                data = conn.recv(1024) # objekat koji sadrzi primljenu poruku sa baferom velicine
                1024 bajta

                zahtev = data.decode()# desifrujemo poruku buduci da se sav sadrzaj preko mreze
                salje u enkodovanom obliku

                if("PRIJAVA" in zahtev): # pravimo upite da bi razgranicili sta je prijava a sta
                odjava u poruci

                    trenutnoVreme = datetime.datetime.now().strftime("%d" + "/" + "%m" + "/" + "%Y
                    - %H:%M:%S")

                    name = zahtev.split(" ")[0]

                    text1.insert(END, trenutnoVreme + ": zaposleni " + name + " se PRIJAVIO. \n")

                    f = open("log.txt", "a")

                    f.write(trenutnoVreme + ": zaposleni " + name + " se PRIJAVIO. \n")

                    f.close()

                    start_time = time.time()

                    start_time_baza = (int(start_time))

                    cursor.execute("UPDATE zaposleni SET naPoslu=?, start_time=? WHERE id=? ",
                                ('Da', start_time_baza, name))

```

```

        connection.commit()

    if("ODJAVA" in zahtev):
        trenutnoVreme = datetime.datetime.now().strftime("%d" + "/" + "%m" + "/" + "%Y
- %H:%M:%S")

        name = zahtev.split(" ")[0]
        text1.insert(END, trenutnoVreme + ": zaposleni " + name + " se ODJAVIO. \n")

        upit1 = f"""
            SELECT sekunde
            FROM zaposleni
            WHERE id={name}

        """

        cursor.execute(upit1)
        sekunde = cursor.fetchall()[0]
        rowsCut1 = str(sekunde).split('(', 1) # kratimo string pre broja
        preString1 = rowsCut1[1]
        posleString1 = preString1.split(",", 1) # kratimo string posle broja
        brojSekundi = posleString1[0] # rezultat kao ceo broj
        print(brojSekundi)

        upit2 = f"""
            SELECT start_time
            FROM zaposleni
            WHERE id={name}

        """

        cursor.execute(upit2)
        start = cursor.fetchall()[0]
        row = str(start).split('(', 1) # kratimo string pre broja

```

```

        pre = row[1]
        posle = pre.split(",", 1) # kratimo string posle broja
        broj_start = posle[0] # rezultat kao ceo broj

        f = open("log.txt", "a")
        f.write(trenutnoVreme + ": zaposleni " + name + " se ODJAVIO. \n")
        f.close()

        e = int(time.time() - float(broj_start))
        print(e)
        e2 = int(brojSekundi)
        e3 = e + e2
        print(e3)

        hour = ('{:02d}:{:02d}'.format(e3 // 3600, (e3 % 3600 // 60)))
        print(hour)

        cursor.execute("UPDATE zaposleni SET naPoslu=?, radniSati=?, sekunde=? WHERE
id=? ", ('Ne', hour, e3, name))

        connection.commit()

    break

except:
    print("---")

tk1 = threading.Thread(target=slusajKonekcije, args=()).start() # Stavljanje funkcije pod nit da
bi mogla uvek da prima konekcije, bez obzira da li se dešavaju neke radnje ili ne

```

### 2.2.1 Основне опције

Ако погледамо претходну фотографију 2.2.1, можемо видети неке од опција које има администратор. Једна од најважнијих опција јесте могућност да администратор провери присутност и тиме се врло лако добије информација ко је тренутно на послу а ко не. У продужетку следи код функције за динамички приказ запослених, где важи правило:

- Радници који **нису** присутни имају сиве фотографије
- Радници који **јесу** присутни имају фотографије у боји

- Функција за креирање интерфејса присутних радника

```

class DinamicniGrid(tk.Frame): # U ovom slucaju smo koristili klasu zbog lakse dinamike
                                prikazivanja podataka
    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, *args, **kwargs)
        frame = tk.Frame(self, width=300, height=300) # objekat frame koji vec koristi
                                                         kao roditelja tkinter root prozor

        frame.pack(expand=True, fill=tk.BOTH)
        canvas = tk.Canvas(frame, bg='FFFFFF', width=300, height=300, scrollregion=(0, 0, 500,
500))

        hbar = tk.Scrollbar(frame, orient=tk.HORIZONTAL)
        hbar.pack(side=tk.BOTTOM, fill=tk.X) # dodajemo scrollbar
        hbar.config(command=canvas.xview)
        # vbar = tk.Scrollbar(frame, orient=tk.VERTICAL)
        # vbar.pack(side=tk.RIGHT, fill=tk.Y)
        # vbar.config(command=canvas.yview)
        canvas.config(width=300, height=300)
        canvas.config(xscrollcommand=hbar.set)
        canvas.pack(side=tk.LEFT, expand=True, fill=tk.BOTH)

        self.text = tk.Text(canvas, wrap="char", borderwidth=0, highlightthickness=0,
                             state="disabled", cursor="arrow")

        self.text.pack(fill="both", expand=True)
        self.bboxes = []

    def add_box(self, color=None): # funkcija pravljenje frejma odnosno "kutije"

        conn = sqlite3.connect("zaposleni.db")
        cursor = conn.cursor()

        upit = f"""
                SELECT *
                FROM zaposleni
                """

        cursor.execute(upit)
        rezultat = cursor.fetchall()

```

```

conn.close()

for i in rezultat:
    box = tk.Frame(self.text, bd=1, relief="sunken",
                    width=25, height=25)
    bar_kod = f"{i[0]}"
    naPoslu = f"{i[7]}"
    slika = f"{i[9]}"

    img = (Image.open("" + slika))
    if("Ne" in naPoslu):
        resized_image = img.resize((150, 150), Image.ANTIALIAS).convert('L')
    else:
        resized_image = img.resize((150, 150), Image.ANTIALIAS)
    new_image = ImageTk.PhotoImage(resized_image)
    label2 = tk.Label(box, image=new_image)
    label2.image = new_image
    label2.grid()

    btn = tk.Button(box, text=bar_kod, cursor="hand2")
    btn.grid()
    self.bboxes.append(box)
    self.text.configure(state="normal")
    self.text.window_create("end", window=box)
    self.text.configure(state="disabled")

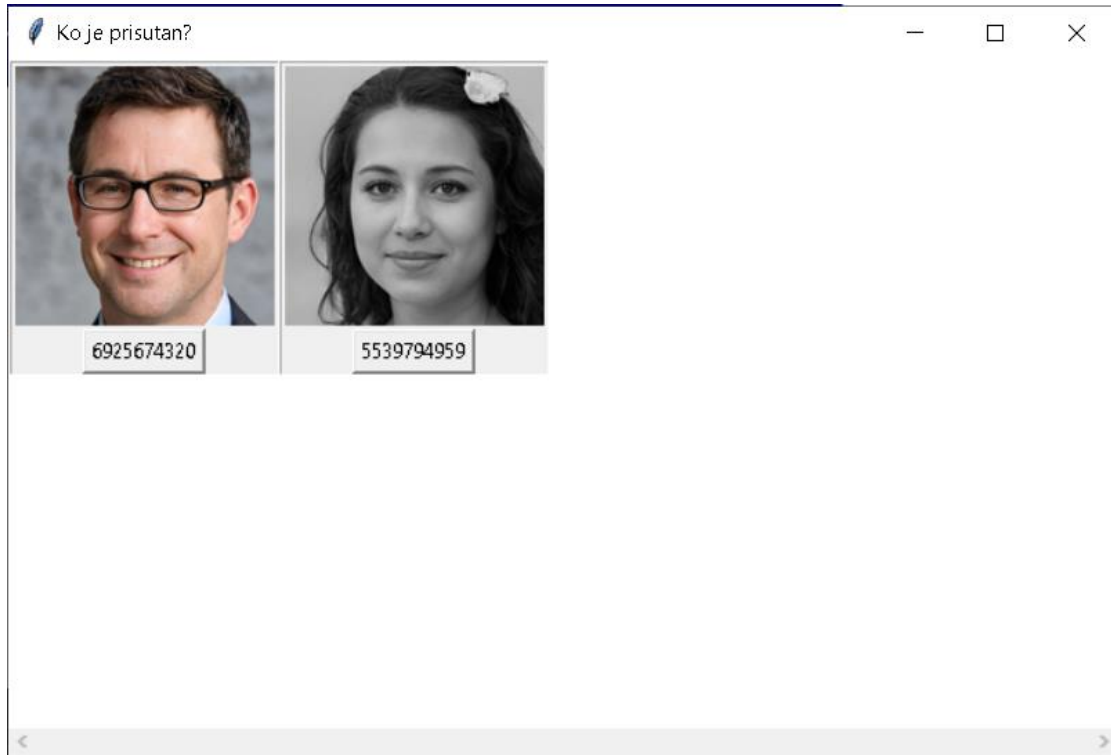
class prikaz(object):
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Ko je prisutan?")
        self.dg = DynamicGrid(self.root, width=500, height=200)
        self.dg.pack(side="top", fill="both", expand=True)

    def start(self):
        self.dg.add_box()
        self.root.mainloop()

prikaz().start()

```

Анализом претходног кода, можемо видети начин структуре и уписивање одговарајућих података у сам панел са детаљима. Користи се упит из базе података који враћа све запослене па се на основу тога касније у продужетку исте те функције креирају слике запослених. Саме фотографије се боје у сиву ако је запослени одсутан а у боји остају ако је присутан. Ово се врши простим упитом чији је резултат приказан на слици 2.2.1.2.



**Слика 2.2.1.2 Интерфејс „Ко је присутан“**

Сама идеја је као што смо већ и рекли, да слике запослених који су присутни су у боји док са друге стране, запослени који нису на послу имају сиву слику. Са ове тачке гледишта је сасвим довољно информација (ИД број), међутим у будућности би сигурно постајала опција за приказ свих информација тог запосленог као и рецимо зелено или црвено светло поред слика, због лакшег препознавања. Свакако треба узети обзир то да како потребе фирме расту, тако би се додавало више опција и разне могућности.

У продужетку описивања кода, са десне стране интерфејса можемо видети постојање текст бокса односно текстуалног поља у којем се исписују информације о томе када се радник пријавио или одјавио. Такође се поред тога налази датум и време пријаве/одјаве и јединствени број радника (бар код). Ове се информације такође уписују у текстуелну датотеку „лог“ (у наставку је пример). Изузетна важност ових података јесте та што администратору даје комплетан увид у време пријаве односно одјаве као и радника који се пријавио или одјавио. Без ове опције, било би тешко испратити ко је када дошао на посао или ко је када отишао са истог.



06/11/2021 - 20:30:47: zaposleni 6449716892 se ODJAVIO.  
 06/11/2021 - 20:30:59: zaposleni 5539794959 se ODJAVIO.  
 06/11/2021 - 20:31:10: zaposleni 6925674320 se ODJAVIO.  
 06/11/2021 - 20:31:28: zaposleni 5539794959 se PRIJAVIO.  
 06/11/2021 - 20:38:05: zaposleni 6925674320 se PRIJAVIO.  
 06/11/2021 - 20:44:46: zaposleni 6925674320 se ODJAVIO.  
 06/11/2021 - 20:44:49: zaposleni 5539794959 se ODJAVIO.  
 06/11/2021 - 20:46:40: zaposleni 5539794959 se PRIJAVIO.  
 06/11/2021 - 20:49:20: zaposleni 6449716892 se PRIJAVIO.  
 06/11/2021 - 20:51:52: zaposleni 6449716892 se ODJAVIO.  
 06/11/2021 - 20:51:57: zaposleni 5539794959 se ODJAVIO.  
 06/11/2021 - 20:55:55: zaposleni 5539794959 se PRIJAVIO.  
 06/11/2021 - 20:57:30: zaposleni 5539794959 se ODJAVIO.  
 06/11/2021 - 21:28:49: zaposleni 6925674320 se PRIJAVIO.  
 06/11/2021 - 21:29:05: zaposleni 6449716892 se PRIJAVIO.  
 07/11/2021 - 14:08:11: zaposleni 6925674320 se ODJAVIO.  
 07/11/2021 - 14:08:15: zaposleni 6449716892 se ODJAVIO.  
 08/11/2021 - 12:06:15: zaposleni 5539794959 se PRIJAVIO.  
 08/11/2021 - 12:07:04: zaposleni 6925674320 se PRIJAVIO.  
 08/11/2021 - 12:07:16: zaposleni 6449716892 se PRIJAVIO.  
 08/11/2021 - 12:07:31: zaposleni 5539794959 se ODJAVIO.  
 08/11/2021 - 12:16:08: zaposleni 6449716892 se ODJAVIO.  
 08/11/2021 - 12:18:40: zaposleni 6449716892 se PRIJAVIO.

Део документа који  
 бележи све пријаве и одјаве.

Постоји идеја као и могућност да постоји посебна база са свим овим информацијама, па би на основу тога могле да се додају све опције за рад са овим подацима. Нпр. претрага када се запослени под редним бројем 23241414 пријавио или одјавио или да ли се исти тај радник тог дана одјавио у толико сати или раније.

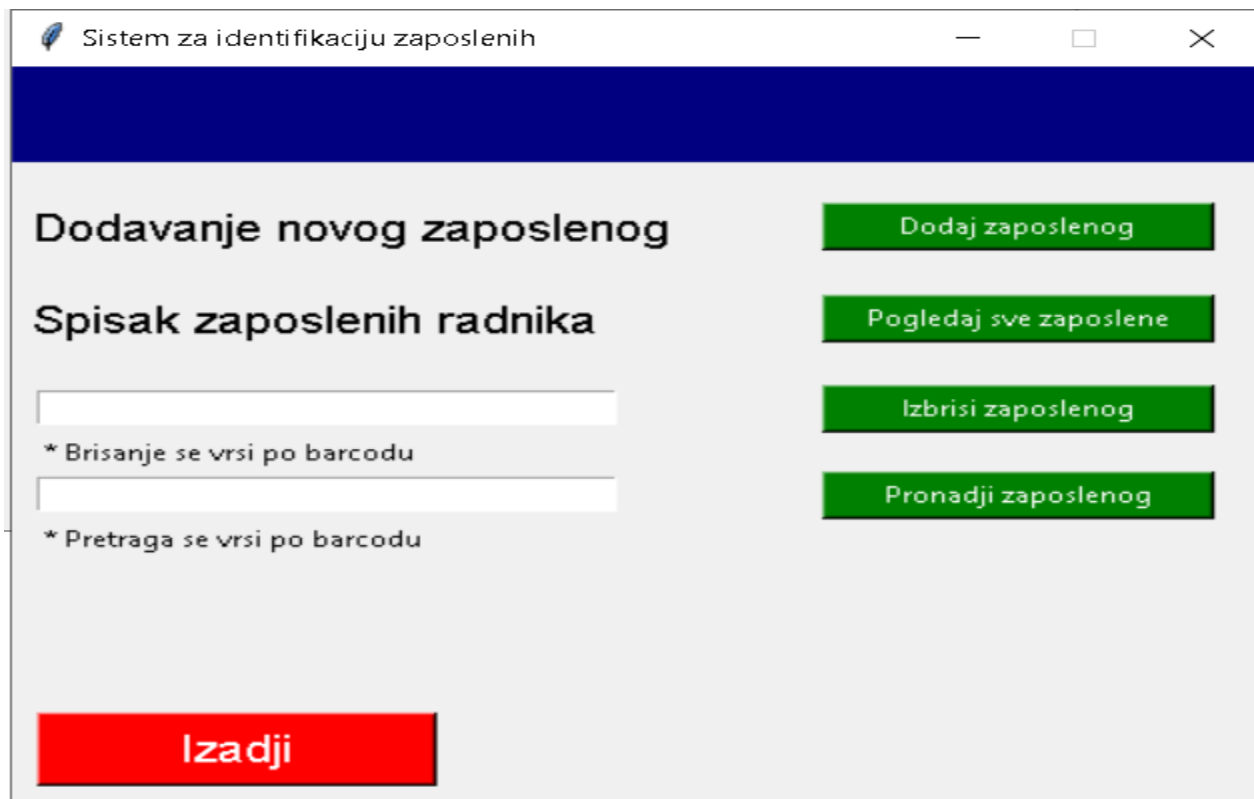
- Дан/Датум/Година - Сат:Минут:Секунда: запослени **ИД\_БРОЈ** се пријавио/одјавио.

Конкретан пример следи:

- 14/08/2021 – 14:39:01: запослени **6548954123** се пријавио.
- 14/08/2021 – 17:10:48: запослени **1264333994** се одјавио.

## 2.2.2 Напредне опције

Поред основних ту су и напредне опције којима администратор може да приступи па тако и на слици испод можемо запазити да постоји разни начини да се руководи базом података. Прва и најважнија опција јесте додавање новог радника коју ћемо касније описати детаљно. Поред ове главне функције, постоје функције за списак, брисање и претрагу, као што можемо видети на слици 2.2.2.1. Свака функција се врши на основу јединственог броја запосленог.



Слика 2.2.2.1 Напредне опције администратора

Напредне опције обухватају:

- додавање новог запосленог;
- списак свих запослених;
- брисање запосленог;
- претрагу запосленог.

Важно је напоменути да се како претрага тако и брисање врше преко бар кода. Разлог зашто се користи бар код јесте тај што је он јединствен за сваког радника и третира се као идентификациони број (енг. ID). Да би брисање и претрага били прецизни, потребно је доставити и прецизне податке за валидацију, што нпр. име или презиме нису случај. Водећи се логиком да рецимо користимо име за претрагу, приликом претраге постоји десет запослених са именом Милош и то је у најбољем случају, увек треба узети у обзир да може постојати и 100 запослених по имену Милош. Па на основу тога може доћи до загушења базе и успорити саобраћај. Свакако је боља опција вршити претрагу по бар коду али исто тако сматрам да би у будућности сигурно додао претрагу по другим параметрима, како би употпунио ове две функције.

У продужетку можемо видети кодове за претрагу и брисање као и начин на који сам ја решио услове и поставио границе функција и такође решио све могуће грешке. Обе функције се врше на исти начин, поштујући исти редослед алгоритма.

## napredne\_opcije.py

- Функција за претрагу запосленог по бар коду

```
def pronadjiZaposlenog():
    if len(e2.get()) == 0: # Mala provera entry polja da ne bi doslo do provere
                           praznog polja.
        messagebox.showerror(title="Greska", message="Niste uneli barcode!")
    elif len(e2.get()) > 1: # Ponovna provera da li je polje prazno.
                           Ocekivano je da nije, iz tog razloga funkcija nastavlja

        conn = sqlite3.connect("zaposleni.db") # Otvaranje konekcije sa bazom
        cursor = conn.cursor()
        bar_kod = e2.get()
        upit = f"""
                SELECT *
                FROM zaposleni
                WHERE id = '{bar_kod}'
            """ # Pravljenje upita
        cursor.execute(upit)
        global rezultat
        rezultat = cursor.fetchall()
        if rezultat:
            rezultatWindow()
        else:
            messagebox.showerror(title="Greška", message="Nije pronadjen nijedan zaposlen pod tim
ID-om!")
```

## napredne\_opcije.py

- Функција “rezultatWindow()” која враћа интерфејс са свим подацима претраге.
- Без ње би било безвредно вршити претрагу, с обзиром да свака функција тежи ка резултату.

```
def rezultatWindow():  
    global rezultat  
  
    id = rezultat[0][0]  
    ime = rezultat[0][1]  
    prezime = rezultat[0][2]  
    datumRodjenja = rezultat[0][3]  
    pozicija = rezultat[0][4]  
    radniSati = rezultat[0][5]  
    naPoslu = rezultat[0][7]  
    slika = rezultat[0][9]  
  
    master = tk.Toplevel()  
    master.title("Rezultat")  
    master.geometry("360x450")  
    master.resizable(False, False)  
  
    idLabel = tk.Label(master, text="ID zaposlenog", font=8, justify=tk.LEFT)  
    idLabel.grid(row=0, column=1, sticky=tk.W)  
  
    #  
    e1 = tk.Entry(master, width=25)  
    e1.grid(row=0, column=2, padx=(13, 0))  
    e1.insert(0, id)  
    e1.configure(state=tk.DISABLED)  
  
    #  
    imeLabel = tk.Label(master, text="Ime zaposlenog", font=8, justify=tk.LEFT)  
    imeLabel.grid(row=1, column=1, sticky=tk.W)
```

```

#
e2 = tk.Entry(master, width=25)
e2.grid(row=1, column=2, padx=(13, 0))
e2.insert(0, ime)
#
prezimeLabel = tk.Label(master, text="Prezime zaposlenog", font=8, justify=tk.LEFT)
prezimeLabel.grid(row=2, column=1, sticky=tk.W)
#
e3 = tk.Entry(master, width=25)
e3.grid(row=2, column=2, padx=(13, 0))
e3.insert(0, prezime)
#
godinaLabel = tk.Label(master, text="Datum rodjenja", font=8, justify=tk.LEFT)
godinaLabel.grid(row=3, column=1, sticky=tk.W)
#
e4 = tk.Entry(master, width=25)
e4.grid(row=3, column=2, padx=(13, 0))
e4.insert(0, datumRodjenja)
#
pozicijaLabel = tk.Label(master, text="Pozicija zaposlenog", font=8, justify=tk.LEFT)
pozicijaLabel.grid(row=4, column=1, sticky=tk.W)
#
e5 = tk.Entry(master, width=25)
e5.grid(row=4, column=2, padx=(13, 0))
e5.insert(0, pozicija)

radniSatiLabel = tk.Label(master, text="Radni sati", font=8, justify=tk.LEFT)
radniSatiLabel.grid(row=5, column=1, sticky=tk.W)
#
e6 = tk.Entry(master, width=25)
e6.grid(row=5, column=2, padx=(13, 0))
e6.insert(0, radniSati)

```

```

naPosluLabel = tk.Label(master, text="Na poslu:", font=8, justify=tk.LEFT)

naPosluLabel.grid(row=6, column=1, sticky=tk.W)

#
e7 = tk.Entry(master, width=25)
e7.grid(row=6, column=2, padx=(13, 0))
e7.insert(0, naPoslu)

#
# labelPrazna = tk.Label(master, height=10)
# labelPrazna.grid(row = 5, column = 2)

#
label = tk.Label(master, text="")
label.grid(row=7, column=2)


img = (Image.open("" + slika))
smanjena_slika = img.resize((180, 180), Image.ANTIALIAS)
nova_slika = ImageTk.PhotoImage(smanjena_slika)

label2 = tk.Label(master, image=nova_slika, width=150, height=245)
label2.image = nova_slika
label2.grid(row=7, column=1

```

### napredne\_opcije.py

- **Функција за брисање запосленог из базе**
- Једина функција која захтева потврду администратора како не би дошло до нежељене грешке

```

def validacijaZaBrisanje():
    if len(e3.get()) == 0:
        messagebox.showerror(title="Greška", message="Niste uneli barcode!")
    elif len(e3.get()) > 1:
        conn = sqlite3.connect("zaposleni.db")
        cursor = conn.cursor()
        bar_kod = e3.get()

```

```

upit = f"""

        SELECT *

        FROM zaposleni

        WHERE id = '{bar_kod}'

        """

cursor.execute(upit)
global rezultat
rezultat = cursor.fetchall()
if rezultat:
    id = rezultat[0][0]
    ime = rezultat[0][1]
    prezime = rezultat[0][2]
    datumRodjenja = rezultat[0][3]
    pozicija = rezultat[0][4]

    msg = messagebox.askquestion(title="Potvrda", message="Sledeći zaposleni ce biti
    izbrisan iz baze: \t\n\n"

                                "ID: " + id + "\n" +
                                "Ime: " + ime + "\n" +
                                "Prezime: " + prezime + "\n" +
                                "Datum rodjenja " + datumRodjenja + "\n" +
                                "Pozicija: " + pozicija + "\n")

    if msg == 'yes':
        print("Uspešno")
        upit = f"""

                DELETE FROM zaposleni

                WHERE id = {id}

                """

        cursor.execute(upit)
        conn.commit()

        messagebox.showinfo("Uspešno brisanje!", message=id + " je izbrisan!")
    else:
        messagebox.showerror(title="Greška", message="Nije pronadjen nijedan zaposlen pod tim
        ID-om!")

```

Као што смо раније поменули, главна функција јесте опција за додавање новог запосленог којом администратор може, ручно путем интерфејса, допунити базу са новим радником укључујући све његове произвољне атрибуте па и јединствени број односно бар код. Ово је од посебне важности јер се администратору оставља опција да промени унапред генерисани број од 10 цифара у неки други број. Наравно, треба се поставити и неке границе као што су претрага тог бар кода у случају да исти већ постоји у бази. Ово се ради како не би долазило до грешака и исто тако како би администратора лишило могућности да прави грешке или пак злоупотреби софтвер. У опцији додавања радника, спадају подаци неопходни за једног радника, које такође можемо видети на слици 2.2.2.2., а они су:

- ИД запосленог, ова ставка је већ попуњена бројем од 10 насумично одабраних цифара;
- Име запосленог;
- Презиме запосленог;
- Датум рођења;
- Позиција запосленог;
- Фотографија.



Dodavanje zaposlenog

ID zaposlenog 2703733690

Ime zaposlenog

Prezime zaposlenog

Datum rođenja

Pozicija zaposlenog

Fotografija

Prilozi fotografiju

Nazad Dodaj

Слика 2.2.2.2 Додавање запосленог

У продужетку следи код функције за додавање новог запосленог у којој се уједно и генерише бар код односно идентификациони број запосленог – најважнији атрибут запосленог. Поред тога, поједине податке који нису видљиви администратору, програм сам генерише и уписује их у базу података. Разлог овоме јесте тај што ти подаци нису битни за интерфејс нити су ту да би се приказали већ имају друге функције, опширније је у коду.



## dodaj\_zaposlenog.py

- Функција за додавање запосленог

```
def dodajZaposlenog():  
    global broj # globalna promenljiva  
    global putanja # globalna promenljiva  
    id = e1.get()  
    ime = e2.get()  
    prezime = e3.get()  
    datumRodjenja = e4.get()  
    pozicija = e5.get() #Hvatamo unose tekstualnih polja (entry box) i  
                        smestamo ih u promenljive  
    radniSati = 0  
    naPoslu = "Ne" #Podrazumevan atribut za radnika da nije na poslu  
    izvor_putanje = r""+putanja  
    destinacija_putanje = r"C:\Users\Rile\PycharmProjects\Diplomski\zaposleni"+ "\\" +ime + "-" +  
    prezime + ".jpg"  
    shutil.copy(izvor_putanje, destinacija_putanje) #kopiranje fotografije u direktorijum i  
                                                    formatiranje naziva slike  
    conn = sqlite3.connect("zaposleni.db")  
    sekunde = 0  
    conn.execute("insert into Zaposleni (id, ime, prezime, datumRodjenja, pozicija, radniSati,  
    sekunde, naPoslu, slika) values (?, ?, ?, ?, ?, ?, ?, ?, ?)",  
                (id, ime, prezime, datumRodjenja, pozicija, radniSati, sekunde, naPoslu,  
    destinacija_putanje)) # Upisivanje svih podataka u bazu podataka, ukljucujuci I one koji  
                        nisu vidljivi administratoru  
    print("Uspesno upisivanje u bazu.")  
    conn.commit()  
    conn.close()  
    messagebox.showinfo(title=None, message="Uspesno ste dodali radnika!")
```

Након анализе претходног кода који служи за додавање запосленог и поред свих ових опција од којих су неке мање а неке више битне, ту је опција за гледање свих запослених, којом администратор има увид у графички приказ сваког запосленог, његове податке и уопштену проверу да ли је запослени на послу. Један од важнијих података јесте број радних сати, као што видимо на слици 2.2.2.3. у продужетку. Овај податак је изузетно битан за целокупну фирму и слободно можемо рећи да је ово један од најважнијих података било које фирме, с обзиром да на основу њега (радних сати) се израчунавају

примања једног запосленог. Начин на који је осмишљен систем сабирања радних сати и уписивања истих јесте тај да се време пријаве уписује у базу под колоном start\_time па се иста та вредност након одјаве корисника учитава и сабира проведено време. Систем је тестиран више пута са различитим начинима и увек је давао успешни резултат. Опширније такође можемо видети у коду који следи.

## prikaz.py

- Функција за графички приказ запослених

```
root=tkinter.Tk() # објекат у који сместамo главни прозор tkinter библиотеке

root.title("Zaposleni koji su trenutno na poslu")

root.geometry("475x520")

rows = 0

def azuriraj_scroll (event):

    photoCanvas.configure(scrollregion=photoCanvas.bbox("all")) # funkcija za azuriranje
                                                                    pozicije scrollbara

photoFrame = tkinter.Frame(root, bg="#EBEBEB", width="500", height="400")

photoFrame.grid()

photoFrame.rowconfigure(0, weight=1)

photoFrame.columnconfigure(0, weight=1)

photoCanvas = tkinter.Canvas(photoFrame, bg="#EBEBEB", height="500", width="450")

photoCanvas.grid(row=0, column=0, sticky="nsew")

canvasFrame = tkinter.Frame(photoCanvas, bg="#EBEBEB", width="500")

photoCanvas.create_window(0, 0, window=canvasFrame, anchor='nw')

# pravimo frejm I kanvas u koji smestamo taj frejm

def kreiraj_detaljan_frejm(): # funkcija za pravljenje frejma u koji se kasnije smestaju svi
                             podaci zaposlenih

    conn = sqlite3.connect("zaposleni.db") # povezivanje sa bazom podataka

    cursor = conn.cursor() # pravimo objekat u koji smestamo kursor uz pomocu kojeg radimo
                             upite
```

```

upit = f""" # pravimo upit na osnovu kojeg uzimamo samo korisnike koji su na poslu trenutno

        SELECT *

        FROM zaposleni

        WHERE naPoslu = 'Da'

        """

cursor.execute(upit) # izvršavamo upit
rezultat = cursor.fetchall()
conn.close()

for i in rezultat:

    step = tkinter.LabelFrame(canvasFrame, text="Detalji:", width="400")
    step.grid(row=rows, columnspan=7, sticky='W', padx=5, pady=5, ipadx=5, ipady=5)
    tkinter.Label(step, text="ID", font = "Arial 8 bold").grid(row=0, sticky='E', padx=5,
pady=2)
    tkinter.Label(step, text="Ime", font = "Arial 8 bold").grid(row=1, sticky='E', padx=5,
pady=2)
    tkinter.Label(step, text="Prezime", font = "Arial 8 bold").grid(row=2, sticky='E',
padx=5, pady=2)
    tkinter.Label(step, text="Datum rođenja", font="Arial 8 bold").grid(row=3,
sticky='E', padx=5, pady=2)
    tkinter.Label(step, text="Pozicija", font="Arial 8 bold").grid(row=4, sticky='E',
padx=5, pady=2)
    tkinter.Label(step, text="Radni sati", font="Arial 8 bold").grid(row=5, sticky='E',
padx=5, pady=2)
    tkinter.Label(step, text="Na poslu", font="Arial 8 bold").grid(row=6, sticky='E',
padx=5, pady=2)

    for x in range(13):

        tkinter.Label(step, text="", font="Arial 8 bold italic").grid(row=5, column=x,
sticky='E', padx=5, pady=2)

        canvas = tkinter.Canvas(step, bg="red", height="100", width="100")
        canvas.grid(row=9, column=3, sticky='E', padx=0, pady=0)

        slika = f"{i[9]}"

        img = (Image.open("" + slika))

        resized_image = img.resize((150, 150), Image.ANTIALIAS)

        new_image = ImageTk.PhotoImage(resized_image)

```

```

label2 = tkinter.Label(canvas, image=new_image)

label2.image = new_image

label2.grid()


e1 = tkinter.Entry(step)
e2 = tkinter.Entry(step)
e3 = tkinter.Entry(step)
e4 = tkinter.Entry(step)
e5 = tkinter.Entry(step)
e6 = tkinter.Entry(step)
e7 = tkinter.Entry(step)
e1.insert(0, f"{i[0]}")
e2.insert(0, f"{i[1]}")
e3.insert(0, f"{i[2]}")
e4.insert(0, f"{i[3]}")
e5.insert(0, f"{i[4]}")
e6.insert(0, f"{i[5]}")
e7.insert(0, f"{i[7]}")


e1.grid(row=0,column=1,columnspan=7, sticky="WE")
e2.grid(row=1,column=1,columnspan=7, sticky="WE")
e3.grid(row=2,column=1,columnspan=7, sticky="WE")
e4.grid(row=3, column=1, columnspan=7, sticky="WE")
e5.grid(row=4, column=1, columnspan=7, sticky="WE")
e6.grid(row=5, column=1, columnspan=7, sticky="WE")
e7.grid(row=6, column=1, columnspan=7, sticky="WE")

rows += 1

```

```

photoScroll = tkinter.Scrollbar(photoFrame, orient=tkinter.VERTICAL)
photoScroll.config(command=photoCanvas.yview)
photoCanvas.config(yscrollcommand=photoScroll.set)
photoScroll.grid(row=0, column=1, sticky="ns")

```

```
canvasFrame.bind("<Configure>", update_scrollregion)

# ubacujemo scrollbar u canvas koji smo ranije kreirali


create_detail_frame()
```

Након анализе претходног кода који служи за графички приказ запослених, где се сваки онлајн радник представља у фрејму са свим подацима, долазимо до закључка да овакав тип прегледа информација доноси најбоље резултате будући да је прегледан и лак за употребу.

Svi zaposleni

Detalji:

ID	6449716892
Ime	Ana
Prezime	Petrovic
Datum rođenja	13.09.1999
Pozicija	assistant
Radni sati	41:59
Na poslu	Da



Detalji:

ID	6925674320
Ime	Milos
Prezime	Mitrovic
Datum rođenja	25.01.2000
Pozicija	director

Слика 2.2.2.3 Списак запослених

Ако погледамо претходну фотографију 2.2.2.3, можемо видети да за сваког радника постоји детаљан фрејм са свим подацима, почев од основног идентификационог броја па све до тога да ли је запослени на послу. Још један начин да администратор увиди да ли је запослени присутан или не.

### 3. ПОСТУПАК ИЗРАДЕ

Поступак израде је опис свих идеја и проблема приликом израде софтвера. Конкретно у случају Софтвера за идентификацију запослених, поступак израде је сложени процес који се састоји из посебно одабраних сегмената где сваки има свој део имплементације. Процес израде по поступцима се може видети у продужетку, са детаљним описима.

#### 3.1. Идеја софтвера

Сама идеја је настала као потреба данашњих фирми да имају увид у информације о радницима и да лакше руководе истима. Софтвер за идентификацију запослених је настао услед идеје да постоји глобални систем на који се радници пријављују и одјављују. Софтвер захтева константу комуникацију сервера и клијента па исто тако захтева постојање локалне мреже (енг. LAN). Без сервера, софтвер не би радио из разлога што се све информације уписују у базу преко сервера што због сигурности то због једноставности. Приступ серверу имају само надређени док са друге стране, приступ клијенту могу добити сви радници. Када говоримо о клијенту, то није ништа друго него обичан уређај (или апарат) који физички постоји испред врата уз помоћу којег радници читавају своје бар код бројеве односно картице. Потом се све те информације преко мреже шаљу серверу и на тај начин уписују.

#### 3.2. Реализација идеје

Реализација идеје се састојала из пар сегмената и првобитних питања на које би требало дати једноставан одговор а притом у исто време задовољити све потребне критеријуме оваквог софтвера. Реализација идеје по етапама и фазама:

- 1) Пројектовање прототипа интерфејса
- 2) Креирање приступачног интерфејса на основу прототипа
- 3) Идеја чврсте повезаности клијента и сервера
- 4) Испитивање и решавање свих проблема у комуникацији
- 5) Оптимизација протока података преко мреже ради лакшег упис у базу
- 6) Тестирање свих могућности софтвера и спровођење стрес тестова
- 7) Додавање нових опција, одржавање и ажурирање система

##### 1) Пројектовање прототипа интерфејса

Сам прототип интерфејса односно идеја изгледа софтвера је од почетка реализације па до завршног дела изгледала потпуно исто, без неких већих промена и обликовања. Идеја је да је интерфејс приступачан и лак за употребу што је посебно од значаја приликом ангажовања запосленог на месту администратора, а то значи да не би била потребна посебна обука тог кадра. Ово доприноси лакшем руковођењу софтвера и лакши приступ истом.

##### 2) Креирање приступачног интерфејса на основу прототипа

Претапање идеје у изглед и касније целокупан софтвер је био лакши део израде, будући да је софтвер лаган за употребу, као што смо и претходно наводили. Приликом израде интерфејса и самог софтвера, доста времена сам издвојио на саме боје текстова и контрола, као и фонтове и величине контрола. Такав приступ изради софтверу, где се сваки детаљ посебно израђује и сваки сегмент исто тако поступно креира, је сам по себи обећавајући.

### 3) Идеја чврсте повезаности клијента и сервера

Када говоримо о повезаности сервера и клијента, најидеалније окружење софтвера би било постојање локалне мреже у којој би се размењивали подаци а поред тога сервер са великим простором и брзим протоком мреже и наравно адекватног стручног кадра. Ова три својства чине идеални софтвер и идеално окружење за непрекорно и тачно размењивање података. У принципу, од велике је важности увек предвидети све могуће потребе софтвера као и оставити простора и начина за унапређивањем у будућности.

### 4) Испитивање и решавање свих проблема у комуникацији

Код комуникације између два рачунара или сервера и клијента, увек треба предвидети неке од могућих случајева грешака и редослед дешавања, посебно када је у питању софтвер који користи мрежу као стални алат за комуникацију. Заиста је неминовно да постоје грешке у комуникацији и зато је неопходно предвидети исте а потом их и решити или једноставно поставити услове за њихова дешавања.

Први проблем на који сам ја наишао приликом имплементирања библиотеке **сокет** (енг. Socket, мрежна комуникација) у софтвер јесте била константна потребна сервера да прима податке без обзира што увек мора бити спреман и за друге радње као што су претрага, брисање и додавање запосленог. Значи у суштини администратор софтвера (сервера) не би смео да буде ометен приликом извршавања тих операција у случају да сервер прима конекције и податке од стране клијента – лаички речено, примање података не би требало да му смета у свом послу. Решење за ово јесте било коришћење опције **нити** у програмском језику **пајтон** (енг. Python), на овај начин софтвер може извршавати истовремено више радњи без да те радње зависе једне од других. У принципу, овај однос између функција и радњи је независан и заправо је могуће креирати безброј нити које би извршавале свој део посла не обзирајући се на рад програма. То би могло да се посматра као да се у позадини програма извршава неки програмски алгоритам док је на површини све и даље исто и спремно за било какав рад. Функције које пружају нити су изузетно битне и од посебног значаја и посебно се треба користити приликом мрежних операција јер је важно да сам сокет буде независан.

Поменули смо енглеску реч **socket**. Сокет није само мрежна комуникација већ означава пар ип адресе и порта, односно двосмерна комуникација два рачунара на мрежи, детаљније је на слици 3.2.1.



Слика 3.2.1 Формат сокет адресе

У нашем случају софтвера и самог тестирања, сервер користи ИП адресу локалног хоста **127.0.0.1**, која представља **локалну** адресу сваког рачунара. Ову адресу је могуће користити само у случају нашег рачунара а са друге стране у случају да софтвер комерцијализујемо било би потребно да се обезбеди статична ИП адреса која би била „хостинг адреса“ (Нпр. **68.148.05.15**) и преко које би се вршио сав интернет саобраћај.

## 5) Оптимизација протока података преко мреже ради лакшег уписа у базу

Још једна кључна ствар у креирању софтвера, који као алат уписивања користи мрежну комуникацију, јесте оптимизовање преноса података и уписа истих. Замислите базу података од преко 10 хиљада радника у којој се свакодневно бар 90% тог броја упише и испише са посла. Прво и основно, огромна база података би била потребна за чување тих информација, такође, била би потребна и константна употреба мреже без загушивања саобраћаја. Ова количина протока захтева проток без грешака и без додатних проблема у саобраћају јер је већ сама по себи оптерећујућа. Из тог разлога, програмери широм света осмишљају и пројектују најбоље начине за упис и читање података из базе без да се иста оптерети.

У конкретном случају Софтвера за идентификацију запослених, није дошло до проблема загушивања саобраћаја нити је било проблема меморијског типа јер једноставно софтвер није коришћен за велики број људи нити је планиран за тако нешто велико. Али свакако, на софтвер сам применио неке од типова оптимизација које су опште познате и којих се сваки програмер треба придржавати како не би стварао додатне проблеме у раду софтвера.

Оптимизација и добар начин писања кодова се врше на више начина, а само неки од њих су:

- Коришћење адекватних типова података (нпр. тип података интегер за потребе уписивање броја а не тип података стринг или текст за потребе коришћења броја);
- Уписивање само неопходних података, без оптерећивања базе података непотребним уписивањем и исписивањем које додатно троши меморију и место;
- Омогућити серверу да има довољно простора за чување података, односно предвидети обим базе и на основу тога припремити све могуће сценарије;
- Као и много других могућности, потребно је пазити на сваки део кода.

## 6) Тестирање свих могућности софтвера и спровођење стрес тестова

Поред ситних тистрања свих функција у току израде програма, које сваки програмер ради приликом имплементирања, неопходно је вршити и поједина тежа оптерећења односно стрес тестове. Тзв. стрес тестови се врше при завршној фази израде софтвера, када је доста функција већ одрађено и када је једноставно програм спреман за комерцијализацију и пуштање у рад. Обично постоји тим који врши ова стрес тестирања а она се базирају на комплетан тест свих функција и могућности софтвера на све могуће начине, што и говори назив овог тестирања. У доста случајева у овој фази тестирања се пронађу скривене грешке које се касније враћају програмерима да их поправе.

У нашем случају Софтвера за идентификацију запослених, стрес тест се јесте спроводио на крају али само од стране мене лично. Приликом таквог тестирања нисам наишао ни на једну грешку и то је вероватно зато што сам ја већ добро тестирао програм приликом писања кода и програмирања. Али свакако је важно спровести овај корак јер увек може дати неке повратне информације које могу помоћи приликом решавања појединих проблема.



## 7) Додавање нових опција, одржавање и ажурирање система

Као и сваки пројекат у било којој сфери посла, важно је оставити простора за усавршавање и додавање функција као и одржавање истог. Када се говори о Софтверу за идентификацију запослених, сматрам да постоји огроман простор за усавршавање и додавање могућности. Разлог томе јесте што када је у питању овакав тип софтвера, који помаже у руковођењу фирме, увек је могуће допуњавање и креирање нових додатних опција. Неке од таквих идеја јесу:

- Креирање видео надзора и имплементирање система за праћења у сам софтвер
- Прављење простора за нове податке сваког радника који би додатно унапредили практичност
- Креирање могућности за директним штампањем картица са подацима запослених
- Будући да већ користимо мрежну комуникацију, имплементирање система комуникације између администратора, односно неки систем дописивања
- Креирање система корисника администратора, где сваки администратор има своју лозинку и корисничко име са којима се пријављује на систем
- Опција за израчунавање радних сати за одређени месец или период радних дана
- Могућност корисника да провери своје радне сате на самом терминалу за пријаву
- Чување свих података који се тренутно уписују у лог.тхт у базу података намењену само тој потреби и на основу тога имати приступ свим подацима о пријавама и одјавама
- као и многе друге опције.

Ово су само неке опције које су заправо идеје, док са друге стране имам доста других идеја које би овај софтвер унапредиле и поставиле на следећи ниво. Такође сматрам да сваки софтвер у свету има простора за унапређивањем и додавањем нових функција.

### 3.3. Проблеми настали током имплементације

Како су проблеми у коду и имплементацији неизбежан део сваког пројекта, тако се и у мом случају јавило пар проблема. Неки су били мањи и безазлени које сам успео да решим врло брзо док је пар проблема захтевало дуже решавање. Један од таквих проблема јесте било то што се приликом рестартовања сервера, време пријављивања радника увек ресетовало. Овакав тип проблема је нешто што увек треба да се има у виду и да се увек софтвер припреми унапред на овакве случајеве, јер увек може доћи до кvara и до рестартовања сервера. У овом случају, проблем сам решио тако што сам уместо слања почетног времена радника директно серверу, уписивао то време у базу података и приликом каснијег одјављивања радника, читао исто то време и на тај начин правио радне сате. Решење је заправо заиста било једноставно и није било потребно превише времена да се имплементира. Након тога, у току тестирања сам рестартовао сервер и више пута и након детаљног израчунавања времена, добио повратне информације да је проблем решен.

Такође је битно напоменути да се старо почетно време увек „гази“ и мења новим почетним временом, тако да база није оптерећена. Овим системом сам обезбедио динамично уписивање без оптерећивања сервера и базе података уопштено, што је од великог значаја.

## 4. ПРАКТИЧНА ПРИМЕНА СОФТВЕРА

Када говоримо о практичној примени једног софтвера у свету, његова ширина може бити велика будући да се софтвери данас праве као решење и одговор за свакакве типове проблема. У нашем случају Софтвера за идентификацију запослених, употреба заиста може бити широка. Разлог томе јесте то што увек можемо пронаћи неку фирму којој треба овакав тип софтвера за одржавање, циљајући на омање компаније односно фирме јер ове веће већ вероватно имају такве софтвере креиране. Овакав тип софтвера је посебно пожељан у фирмама које руководе великом подацима и на основу тог захтева им се посао може олакшати. Софтвер за идентификацију запослених у себи садржи више корисних функција уз помоћу којих се без проблема може руководити мањом базом података и системом. Поседује систем за праћење радника односно његов доласка и одласка из фирме, систем за додавање, брисање и претрагу запослених као и систем за проверу запослених који су тренутно на послу. Ових неколико система имплементираних у софтвер омогућавају брзо и лако руковођење софтвером, па је из тог разлога он и сам по себи приступачан и подложен за комерцијализацију.

### 3.1. Метода употребе

Метода односно начин употребе софтвера за идентификацију запослених јесте широк. Може се користити у више сврха а циљ сваке сврхе јесте успешно руковођење једне организације.

За коришћење овог софтвера са администраторске стране, није потребна специјална обука стручног кадра будући да је софтвер лак и приступачан за употребу, бар у овој тренутној верзији (в. 0.1.). Софтвер за идентификацију запослених је сам по себи лак за коришћење јер је графички представљен тако да свако може разумети функције и сам систем. Такође све ово важи и за кориснике односно запослене у тој фирми који требају свакодневно користити овај систем и уз помоћу њега се пријављивати и одјављивати. Функције пријава и одјава су по истом принципу направљене па је због тога њихова употреба разумљива и прилично лака за употребу. Радник приласком на уређај читава свој бар код уз два клика и на тај начин себи обезбеђује улазак односно излазак из фирме.

Табела 2.1 – Приказ улога у систему

	Улога	Степен хијерархије	Овлашћење
Запослени	Пријава и одјава са система	I	Без овлашћења
Администратор	Руковођење системом и базом података	II	Пуно овлашћење над системом

## 5. ЗАКЉУЧАК

Користећи све претходно наведене технологије и узимајући у обзир све проблеме настале током имплементације, са лакоћом би се могло рећи да софтвер за идентификацију запослених има све предиспозиције да буде комерцијализован у сваком смислу те речи. Главна коришћена технологија је пајтон (енг. **Python**), уз помоћу којег је креиран цео интерфејс и написан цео код. Пројекат је сам по себи довољно приступачан и једноставан за употребу, па је одатле и његово коришћење практично за све кадрове, као нпр што је кадар за људске ресурсе. Искуство стечено приликом развоја овог софтвера је значајно јер је изузетно унапредило моје знање и разумевање коришћених технологија, па се такође овај пројекат може користити и као део портфолиа или референца за посао у будућности. Поред тога, исте те технологије су у данашњици ИТ света јако тражене па је самим тим и овај пројекат један велики плус и предност сам по себи. Идеја за неко будуће усавршавање пројекта јесте да се додају многе опције као што су имплементација видео надзора или опција за израчунавање радних сати за одређени месец или период радних дана као и многе друге могућности које су заправо наведене у ранијим главама.

Сматрам да је непрестано усавршавање вештина најбољи начин за постизање успеха, па је тако и у овом софтверу остављено простора за даљим усавршавањем и унапређивањем како би исти био савршен.

- **ИНДЕКС ПОЈМОВА**

## **А**

администратор 5, 6, 9, 12, 14, 16, 19, 20, 21, 24, 25, 26, 28, 29

## **Б**

база 3 5 7 8 14 15 16 19 20 21 25 26 27 28 29

број 2, 5, 6, 14, 15, 16, 20, 21, 24, 25, 26, 27

## **З**

запослени 5, 6, 8, 9, 12, 13, 14, 15, 16, 21, 22, 23, 24, 25, 27, 28, 29, 30

## **И**

интерфејс 2, 4, 5, 6, 9, 12, 14, 17, 20, 25

интернет 26

информација 3, 5, 7, 8, 12, 14, 15, 23, 25, 27, 28

## **К**

корисник 6, 21, 28, 29

код 5, 6, 7, 8, 14, 16, 17, 20, 21, 23, 25, 26, 27, 28, 29

## **М**

мрежа 25, 26, 27, 28

## **П**

пајтон 26, 30

податак 3, 5, 8, 9, 14, 16, 20, 21, 25, 26, 27, 28, 29

пројекат 25, 27, 28, 30

порука 8, 10

## **Р**

радник 3, 5, 6, 8, 12, 14, 15, 16, 20, 23, 24, 27, 28, 29

## **С**

саобраћај 16, 26, 27

сервер 2, 5, 6, 7, 8, 9, 25, 26, 27, 28

софтвер 1, 2, 3, 5, 6, 9, 20, 25, 26, 27, 28, 29, 30

систем 5, 7, 22, 26, 29, 30

## **Ф**

фотографија 3, 5, 12, 14, 20, 24

- **ЛИТЕРАТУРА**

- [1] Python Software Foundation, <https://docs.python.org/3/>, преузето: ноембар 2021.
- [2] SQLite Open-Source, <https://www.sqlite.org/index.html>, преузето: ноембар 2021.
- [3] socket — Low-level networking interface, <https://docs.python.org/3/library/socket.html>, преузето: ноембар 2021.
- [4] tkinter — Python interface to Tcl/Tk, <https://docs.python.org/3/library/tkinter.html>, преузето: ноембар 2021.

- **ПРИЛОЗИ**

• **ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

**ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

<b>Студент (име, име једног родитеља и презиме):</b>	Никола Радиша Рилак
<b>Број индекса:</b>	НРТ – 73/18

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

- резултат сопственог истраживачког рада;
- да овај рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
- да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
- да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
- да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
- 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;
- 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;
- да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, \_\_. \_\_. 201\_\_ године

Својеручни потпис студента

\_\_\_\_\_