Univerzitet u Beogradu

Fakultet organzacionih nauka

Katedra za elektronsko poslovanje

Tema seminarskog rada

Seminarski rad iz elektronskog poslovanja

Nastavnik: Titula, Ime nastavnika Student:Ime i prezime, br. Indeksa

Saradnik: Ime saradnika

Beograd, 2025.

**Sadržaj**

[1.Uvod 3](#_Toc193790633)

[2.Pregled literature 3](#_Toc193790634)

[3.Pregled postojećih rešenja 4](#_Toc193790635)

[4.Razvoj aplikacije 5](#_Toc193790636)

[5.Zakljucak 17](#_Toc193790637)

[6.Literatura 17](#_Toc193790638)

# 1.Uvod

U današnje vreme, automatizacija i efikasno upravljanje podacima su ključni faktori uspeha u mnogim organizacijama. Menadžment sistemi omogućavaju firmama da optimizuju svoje poslovne procese, smanje manuelni rad i povećaju produktivnost.

Python se pokazao kao idealan izbor za razvoj menadžment sistema zbog svoje jednostavne sintakse, bogate standardne biblioteke i širokog spektra dostupnih okvira i alata.

Ovaj rad istražuje proces kreiranja menadžment sistema koristeći Python, uz fokus na:

* Strukturu aplikacije i njene ključne komponente
* Implementaciju grafičkog korisničkog interfejsa (GUI)
* Sigurnosne aspekte i optimizaciju performansi

# 2.Pregled literature

Python je jedan od najpopularnijih programskih jezika današnjice. Karakteriše ga jednostavna i čitljiva sintaksa, što ga čini pogodnim za početnike, ali i za profesionalce u razvoju softvera. Python je višenamenski, interpretirani, objektno orijentisani jezik, koji se koristi u različitim oblastima, uključujući web programiranje, naučne proračune, razvoj veštačke inteligencije i automatizaciju.

Jedna od glavnih prednosti Pythona je njegova bogata kolekcija biblioteka i modula, koji omogućavaju jednostavno proširenje funkcionalnosti i ubrzavaju razvoj aplikacija. Python je takođe poznat po svojoj portabilnosti, jer omogućava razvoj aplikacija koje mogu raditi na različitim operativnim sistemima.

Python programi se pišu u jednostavnom tekstualnom formatu, a mogu se pokretati iz komandne linije ili putem interaktivnog okruženja kao što su **IDLE, Jupyter Notebook** ili **PyCharm**. [1]

CustomTkinter – Moderna alternativa Tkinter biblioteci CustomTkinter je napredna biblioteka zasnovana na Tkinter-u, koja omogućava moderniji i estetski prijatniji izgled grafičkog korisničkog interfejsa (GUI) u Python aplikacijama. Glavna prednost ove biblioteke u odnosu na klasični Tkinter je podrška za moderni dizajn, tamne i svetle teme, kao i poboljšane vidžete.[2]

Prednosti CustomTkinter biblioteke:

* Poboljšana estetika korisničkog interfejsa
* Jednostavna upotreba i kompatibilnost sa standardnim Tkinter-om
* Dinamičko prilagođavanje tamnim i svetlim temama
* Fleksibilnost u prilagođavanju izgleda elemenata aplikacije

Objektno orijentisano programiranje je [paradigma programiranja](https://sr.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D1%9A%D0%B0), koja koristi [objekte](https://sr.wikipedia.org/wiki/%D0%9E%D0%B1%D1%98%D0%B5%D0%BA%D0%B0%D1%82_(%D1%80%D0%B0%D1%87%D1%83%D0%BD%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE)) kao osnovu za projektovanje računarskih programa i različitih aplikacija [softvera](https://sr.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D0%B2%D0%B5%D1%80). Zasniva se na različitim tehnikama, kao što su [nasleđivanje](https://sr.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D1%92%D0%B8%D0%B2%D0%B0%D1%9A%D0%B5_(%D1%80%D0%B0%D1%87%D1%83%D0%BD%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE)), [apstrakcija](https://sr.wikipedia.org/wiki/%D0%90%D0%BF%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%98%D0%B0_(%D0%BE%D0%B1%D1%98%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D1%98%D0%B5%D0%BD%D1%82%D0%B8%D1%81%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D1%9A%D0%B5)), [polimorfizam](https://sr.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%B0%D0%BC_(%D1%80%D0%B0%D1%87%D1%83%D0%BD%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE)) i [enkapsulacija](https://sr.wikipedia.org/wiki/%D0%95%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D0%B0%D1%86%D0%B8%D1%98%D0%B0_(%D0%BE%D0%B1%D1%98%D0%B5%D0%BA%D1%82%D0%BD%D0%BE_%D0%BE%D1%80%D0%B8%D1%98%D0%B5%D0%BD%D1%82%D0%B8%D1%81%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D1%9A%D0%B5)).[3]

Pyinstaller (auto-py-to-exe) je moćan alat koji pojednostavljuje ovaj process pretvaranja Pythona skripti u samostalne izvršene datoteke (.exe) za Windows.[4]

G4F, skraćeno od GPT4Free, je moćan interfejs komandne linije za velike jezičke modele (LLM-CLI) sa misijom da decentralizuje i omogući besplatan pristup naprednim AI tehnologijama. G4F teži da demokratizuje veštačku inteligenciju pružajući korisnicima jednostavan i efikasan alat za interakciju sa najsavremenijim jezičkim modelima.[5]

# 3.Pregled postojećih rešenja

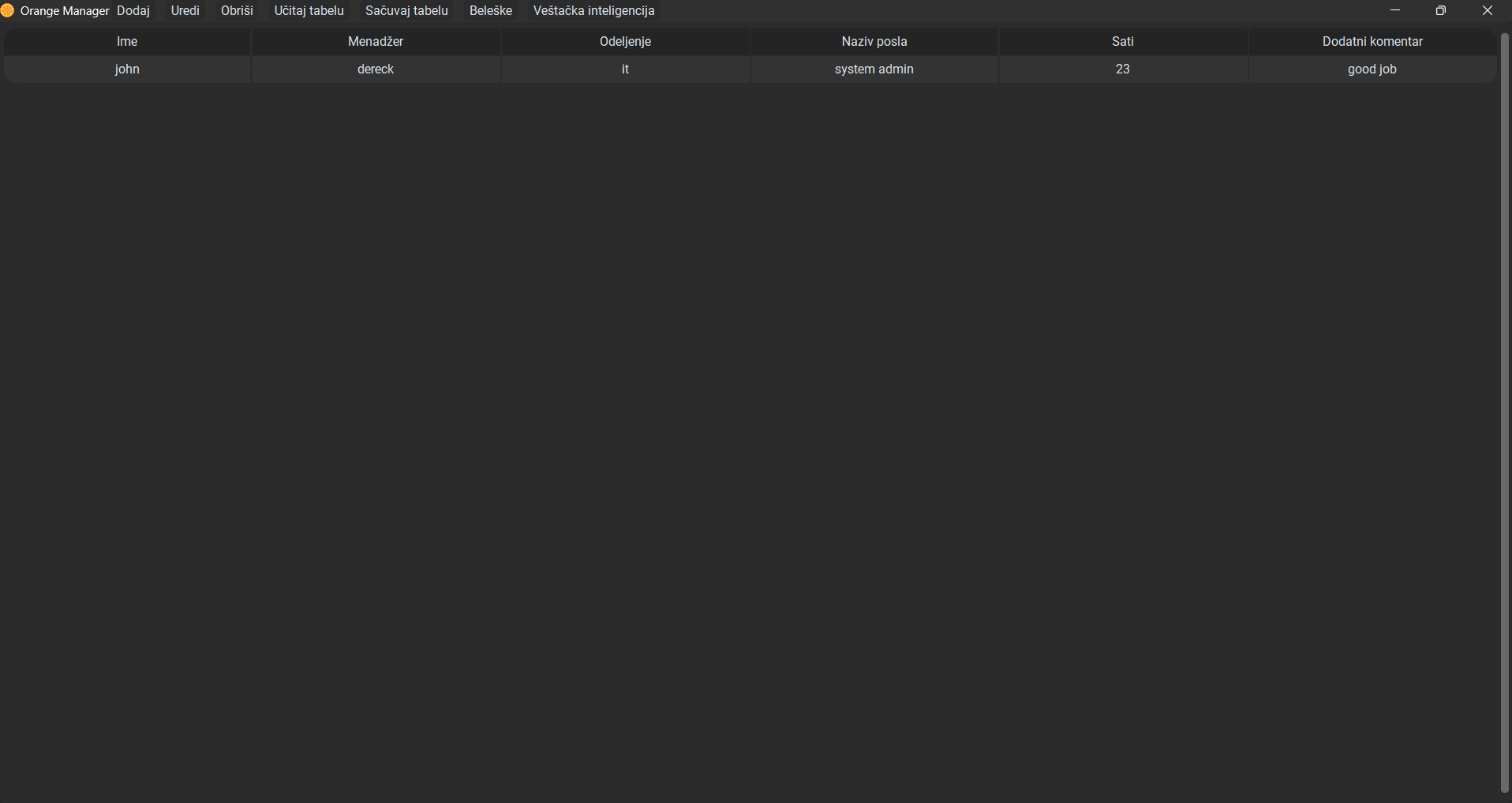
Postoje brojni menadžment sistemi razvijeni u Python-u koji se koriste u različitim domenima poslovanja. Neka od najpoznatijih rešenja uključuju:

* **ERPNext** – Open-source ERP sistem razvijen u Python-u koristeći framework Frappe. ERPNext pruža funkcionalnosti za upravljanje finansijama, zalihama, prodajom, ljudskim resursima i projektima. Njegova modularnost omogućava prilagođavanje specifičnim potrebama različitih industrija. Prednost ERPNext-a je jednostavno podešavanje i integracija sa drugim alatima.
* **Odoo** – Jedan od najpopularnijih ERP sistema razvijenih u Python-u, koji nudi širok spektar modula za upravljanje poslovnim procesima, uključujući prodaju, finansije, HR, logistiku i CRM. Odoo podržava prilagođavanje i integraciju sa drugim sistemima, što ga čini pogodnim za različite vrste organizacija – od malih preduzeća do velikih korporacija.
* **Tryton** – Fleksibilan i modularan poslovni sistem koji omogućava upravljanje finansijama, nabavkama, zalihama i prodajom. Tryton je razvijen kao skalabilna platforma pogodna za kompanije koje traže ERP rešenje sa otvorenim kodom (open source), lako prilagodljivo sopstvenim potrebama.
* **HospitalRun** – Specijalizovan menadžment sistem za zdravstvene ustanove, koji omogućava bolnicama i klinikama efikasno upravljanje podacima o pacijentima, zakazivanjem termina, farmaceutskim zalihama i finansijama. HospitalRun je razvijen sa ciljem da bude pristupačan i funkcionalan čak i u sredinama sa ograničenim internet resursima.

Pored ovih, postoje i mnogi specijalizovani menadžment sistemi razvijeni u Python-u koji su prilagođeni različitim industrijama, kao što su sistemi za upravljanje školama, bibliotekama, restoranima i maloprodajnim lancima.

# 4.Razvoj aplikacije

Za razvoj aplikacije korišćen je programski jezik Python i framework Customtkinter. Aplikacija se zove Orange Manager i njena svrha je da olakša rad i upravljanje zaposlenima. Ima mogućnost direktnog pravljenja tabela za unos zaposlenih u kojoj ima mogućnost upisa imena i prezimena radnika, menadžera, odeljenja, naziv posla, broj radnih sati i unos dodatnih komentara. Pored samog unosa tabele postoje i mogućnosti beležnice i veštačke inteligencije(u vidu chat-bota)(GenAI).



( Slika 1 Ovo je osnovni izgled aplikacije)

Na slici mozemo da vidimo meni sa funkcijama: Dodaj, Unesi, Obriši, Učitaj tabelu, Sačuvaj tabelu, Beleške I Veštačka inteligencija. Da bi se napravio ovakav interfejs korisćen je sledeći kod.

import customtkinter, CTkMenuBar, CTkTable, tkinter, tkinter.filedialog, tkinter.messagebox, sys, csv # Uvoz potrebnih modula i biblioteka

class App(customtkinter.CTk): # Definisanje glavne aplikacione klase koja nasleđuje CTk klasu iz customtkinter biblioteke

def \_\_init\_\_(self, \*args, \*\*kwargs): # Konstruktor klase

super().\_\_init\_\_(\*args, \*\*kwargs) # Poziv konstruktora roditeljske klase

self.title("Orange Manager")

self.geometry("900x700")

self.iconbitmap("slike/Orange\_Manager.ico")

self.protocol("WM\_DELETE\_WINDOW", sys.exit) # Postavljanje funkcije koja se poziva pri zatvaranju prozora

self.appmenu = CTkMenuBar.CTkTitleMenu(self) # Kreiranje menija aplikacije

self.addbutton = self.appmenu.add\_cascade("dodaj", command=self.NewLine)

self.editbutton = self.appmenu.add\_cascade("uredi", command=self.EditLine)

self.deletebutton = self.appmenu.add\_cascade("obriši", command=self.DeleteLine)

self.loadbutton = self.appmenu.add\_cascade("ucitaj tabelu", command=self.LoadData)

self.savebutton = self.appmenu.add\_cascade("sacuvaj tabelu", command=self.SaveData)

self.notepadbutton = self.appmenu.add\_cascade("beleške", command=self.OpenNotepad)

self.aichatbot = self.appmenu.add\_cascade("veštačka inteligencija", command=self.OpenAIChantbot)

self.tableframe = customtkinter.CTkScrollableFrame(self)

self.tableframe.pack(fill="both", expand=True)

self.tablevalue = [["ime", "menadzer", "departman", "naziv posla", "sati", "dodatni komentar"]] # Inicijalizacija vrednosti tabele sa kolonama

self.table = CTkTable.CTkTable(self.tableframe, values=self.tablevalue) # Kreiranje okvira sa mogućnošću skrolovanja

self.table.pack(fill="both", expand=True) # Postavljanje tabele u prozor

def NewLine(self): # Definisanje metode za dodavanje novog reda u tabelu

self.nameinput = customtkinter.CTkInputDialog(title="ime", text="unesite ime", button\_fg\_color="orange").get\_input() # Prikaz dijaloga za unos imena

if self.nameinput == "" or self.nameinput == None: # Provera da li je ime uneseno

tkinter.messagebox.showerror("Greska", "Morate da popunite polje") # Prikaz poruke o grešci

return # Prekida rad

self.managerinput = customtkinter.CTkInputDialog(title="menadzer", text="unesite menadzera", button\_fg\_color="orange").get\_input()

if self.managerinput == "" or self.managerinput == None:

tkinter.messagebox.showerror("Greska", "Morate da popunite polje")

return

self.departmentinput = customtkinter.CTkInputDialog(title="departman", text="unesite departman", button\_fg\_color="orange").get\_input()

if self.departmentinput == "" or self.departmentinput == None:

tkinter.messagebox.showerror("Greska", "Morate da popunite polje")

return

self.jobtitleinput = customtkinter.CTkInputDialog(title="naziv\_posla", text="unesite naziv posla", button\_fg\_color="orange").get\_input()

if self.jobtitleinput == "" or self.jobtitleinput == None:

tkinter.messagebox.showerror("Greska", "Morate da popunite polje")

return

self.hoursinput = customtkinter.CTkInputDialog(title="sati", text="unesite radne sate", button\_fg\_color="orange").get\_input()

if self.hoursinput == "" or self.hoursinput == None:

tkinter.messagebox.showerror("Greska", "Morate da popunite polje")

return

self.additionalcommentinput = customtkinter.CTkInputDialog(title="dodatni komentar", text="unesite dodatni komentar", button\_fg\_color="orange").get\_input()

if self.additionalcommentinput == "" or self.additionalcommentinput == None:

tkinter.messagebox.showerror("Greska", "Morate da popunite polje")

return

self.tablevalue.append([f"{self.nameinput}", f"{self.managerinput}", f"{self.departmentinput}", f"{self.jobtitleinput}", f"{self.hoursinput}", f"{self.additionalcommentinput}"]) # Dodavanje novih vrednosti u tabelu

self.table.add\_row(values=[f"{self.nameinput}", f"{self.managerinput}", f"{self.departmentinput}", f"{self.jobtitleinput}", f"{self.hoursinput}", f"{self.additionalcommentinput}"]) # Dodavanje novog reda u tabelu

def EditLine(self): # Definisanje funkcije za uredjivanje reda

self.columnline = customtkinter.CTkInputDialog(title="kolona", text="unesite kolonu", button\_fg\_color="orange").get\_input() # Unos kolone

self.rowline = customtkinter.CTkInputDialog(title="red", text="unesite red", button\_fg\_color="orange").get\_input() # Unos reda

if self.rowline == "0" or self.rowline == None: # Provera da li je red zaglavlje

tkinter.messagebox.showerror("Greska", "Ne mozete da menjate zaglavlje") # Prikazivanje greske ako je red zaglavlje

else:

self.newvalue = customtkinter.CTkInputDialog(title="nova vrednost", text="unesite novu vrednost").get\_input() # Unos nove vrednosti

self.table.edit(row=int(self.rowline), column=int(self.columnline), textvariable=tkinter.StringVar(value=self.newvalue)) # Menjanje vrednosti u tabeli

self.tablevalue[int(self.rowline)][int(self.columnline)] = self.newvalue # Azuriranje vrednosti u tabeli

def DeleteLine(self): # Definisanje funkcije za brisanje reda

self.rowline = customtkinter.CTkInputDialog(title="red", text="unesite red", button\_fg\_color="orange").get\_input() # Unos reda

if self.rowline == "0" or self.rowline == None: # Provera da li je red zaglavlje

tkinter.messagebox.showerror("Greska", "Ne mozete da obrisete zaglavlje") # Prikazivanje greske ako je red zaglavlje

else:

self.table.delete\_row(int(self.rowline)) # Brisanje reda iz tabele

self.tablevalue.pop(int(self.rowline)) # Uklanjanje vrednosti iz liste

def LoadData(self): # Definisanje funkcije za ucitavanje podataka

self.tablevalue[:] = [] # Brisanje trenutnih vrednosti iz tabele

try:

with open(tkinter.filedialog.askopenfilename(filetypes=[("CSV Files", "\*.csv"), ("All Files", "\*.\*")]), "r+", newline="") as self.file:

self.reader = csv.reader(self.file) # Citanje podataka iz fajla

for self.row in self.reader: # Iteracija kroz redove u fajlu

self.tablevalue.append(self.row) # Dodavanje redova u tabelu

except FileNotFoundError:

pass

self.table.pack\_forget() # Skrivanje trenutne tabele

self.table = CTkTable.CTkTable(self.tableframe, values=self.tablevalue) # Kreiranje nove tabele sa ucitanim vrednostima

self.table.pack(fill="both", expand=True) # Postavljanje nove tabele u prozor

def SaveData(self): # Definisanje funkcije za cuvanje podataka

self.data = self.tablevalue[:] # Kopiranje trenutnih vrednosti iz tabele

with open(tkinter.filedialog.asksaveasfilename(filetypes=[("CSV Files", "\*.csv"), ("All Files", "\*.\*")]), "w+", newline="") as self.file: # Otvaranje CSV fajla za pisanje

self.writer = csv.writer(self.file)

self.writer.writerows(self.data)

def OpenNotepad(self): # Definisanje funkcije za otvaranje beleski

import notepad # Uvoz modula za beleske

notepad.Notes().mainloop()

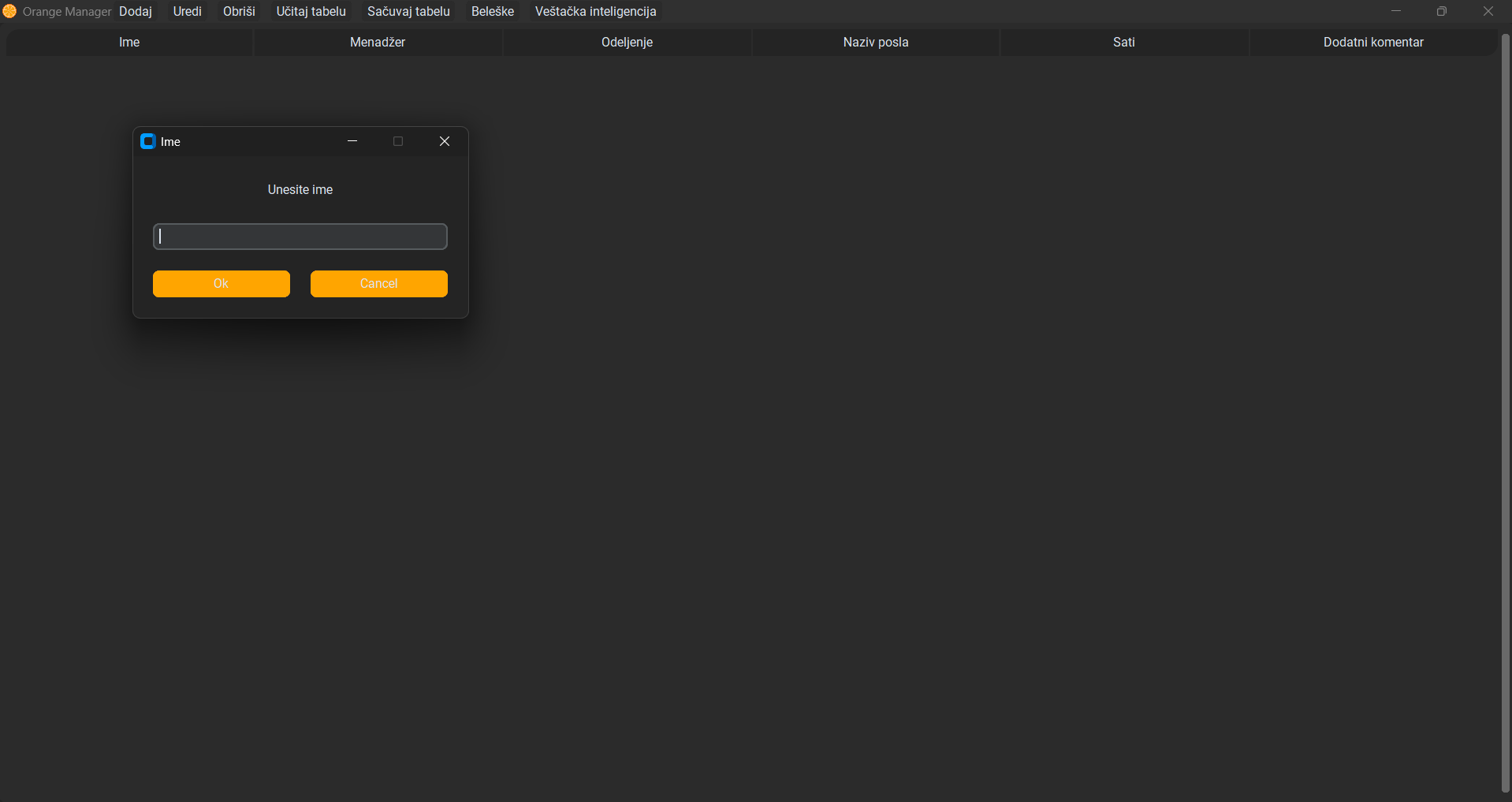
def OpenAIChantbot(self): # Definisanje funkcije za otvaranje AI chatbot-a

import AI # Uvoz modula za vestacku inteligenciju

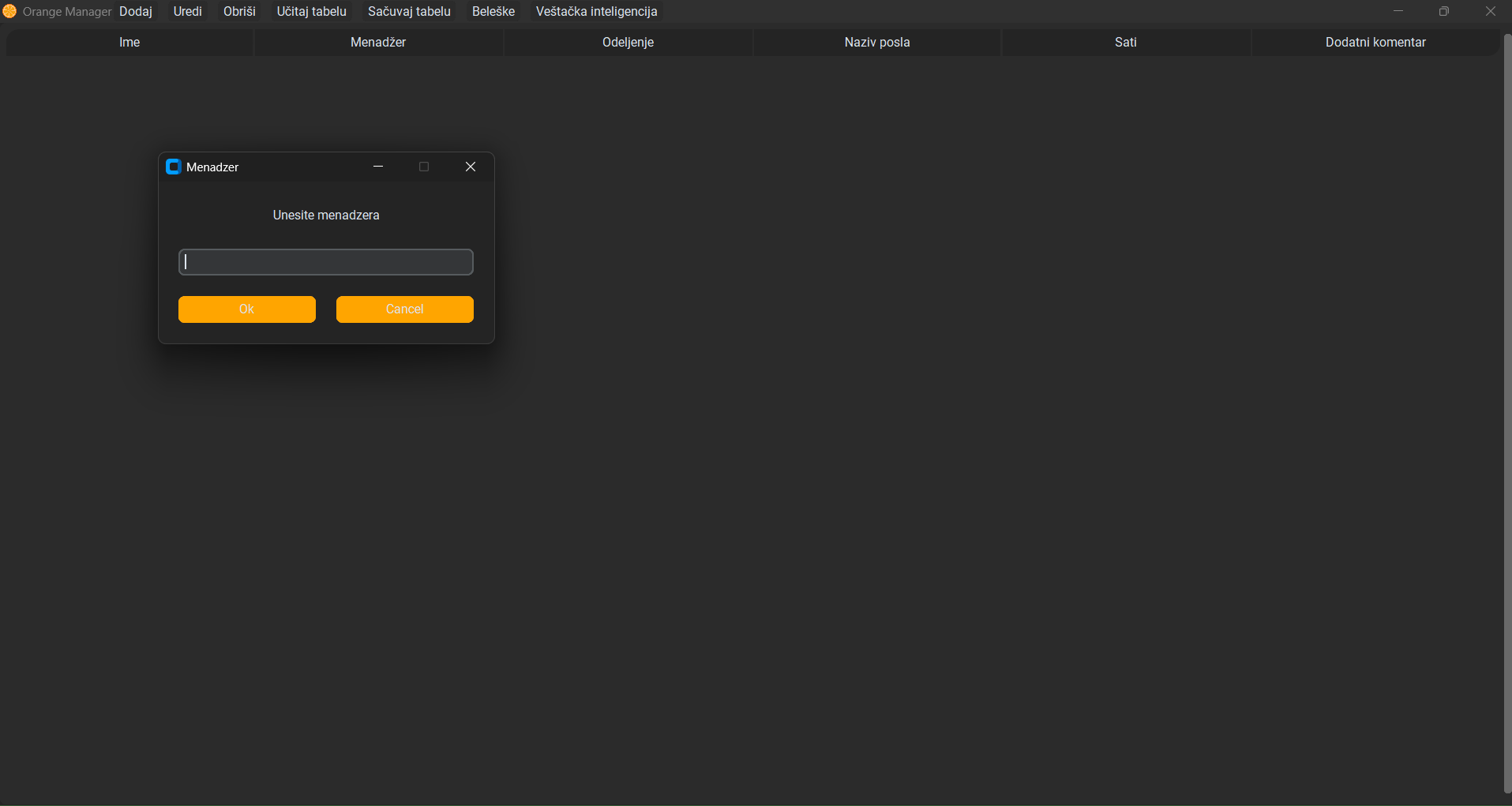
AI.AI\_Window().mainloop() # Pokretanje glavne petlje AI chatbot-a

if \_\_name\_\_ == "\_\_main\_\_":

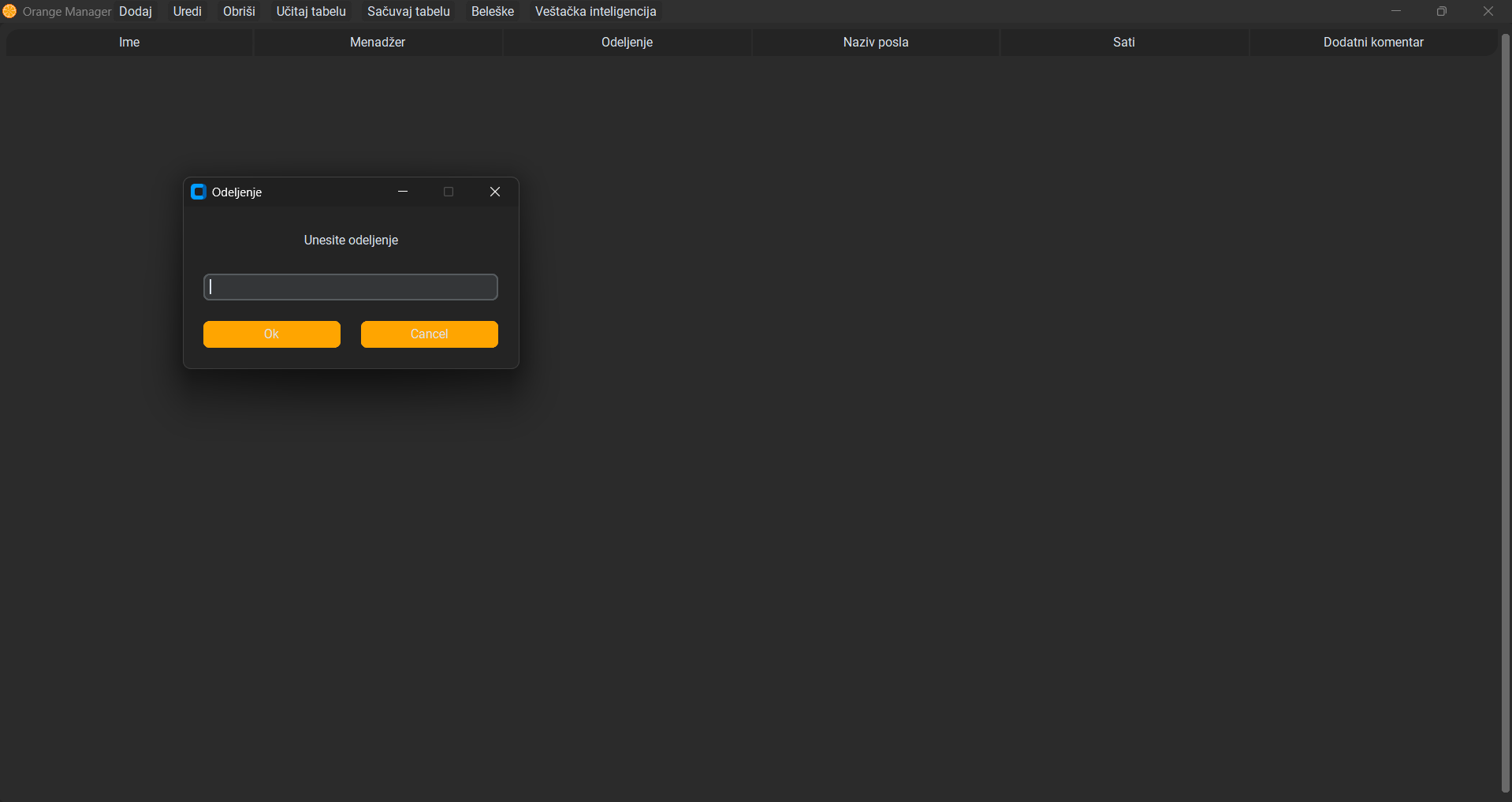
App().mainloop() # Pokretanje glavne petlje aplikacije



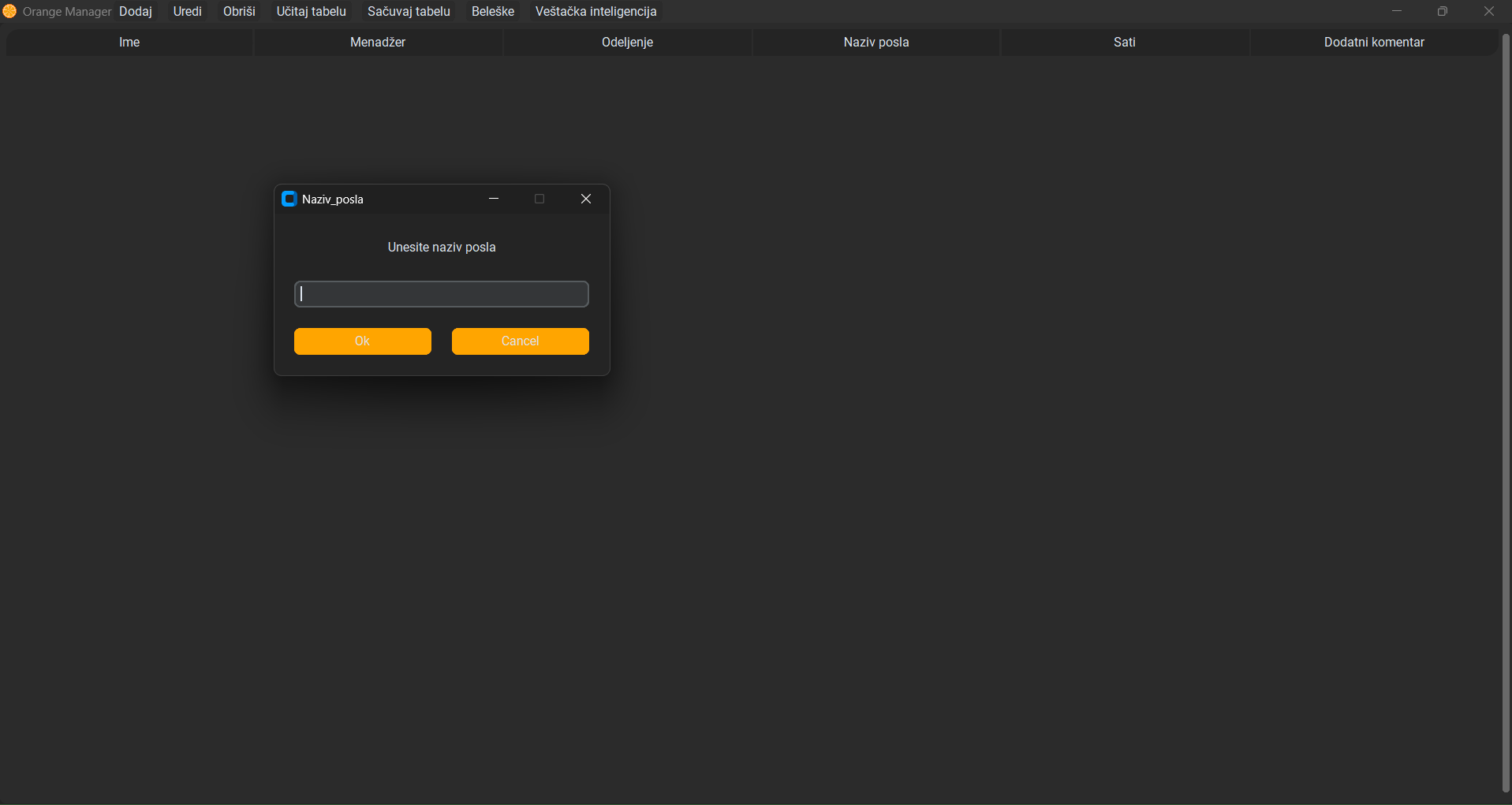
(Slika 2 Unos imena I prezimena radnika)



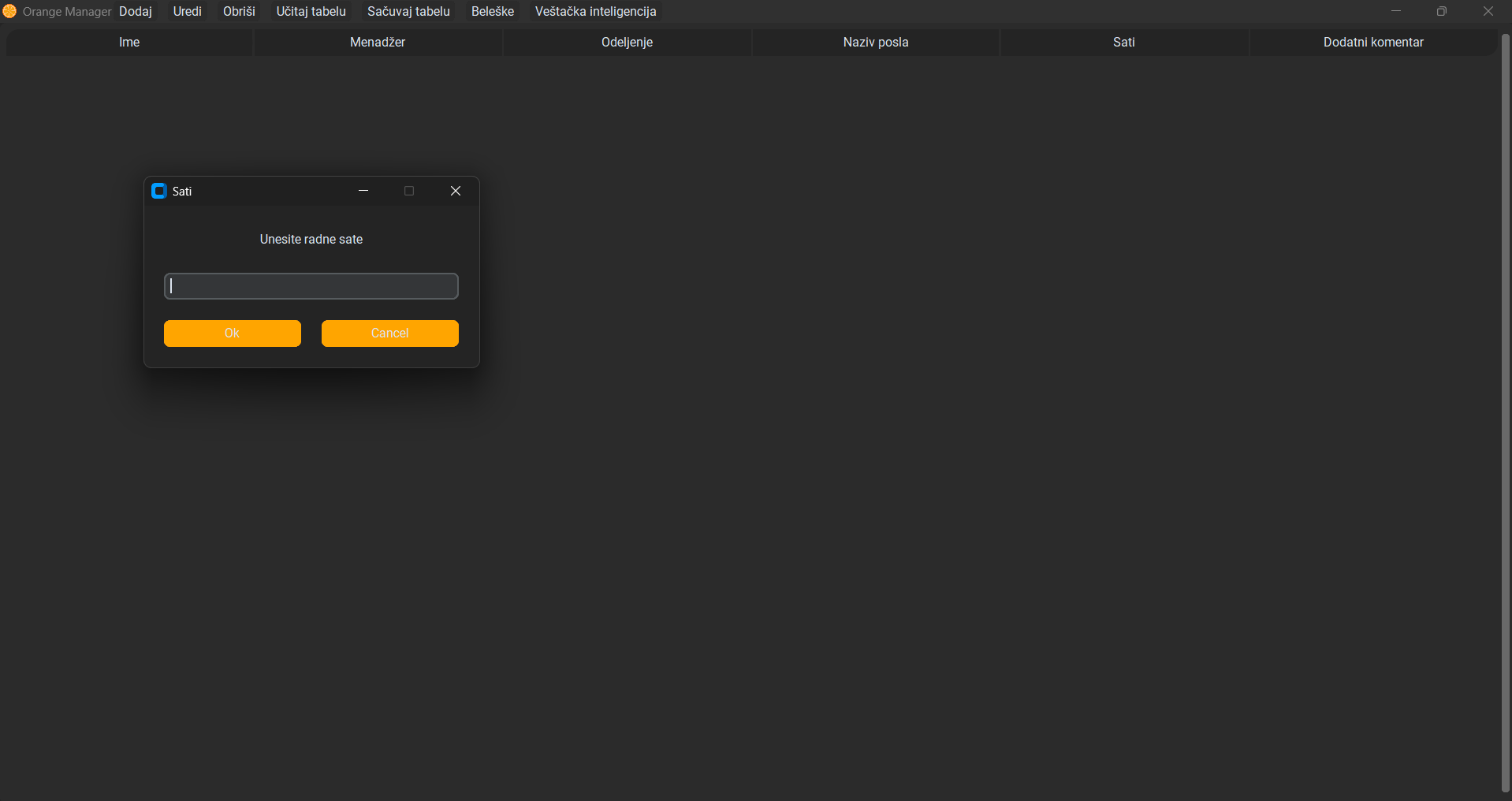
(Slika 3 Unos menadžera)



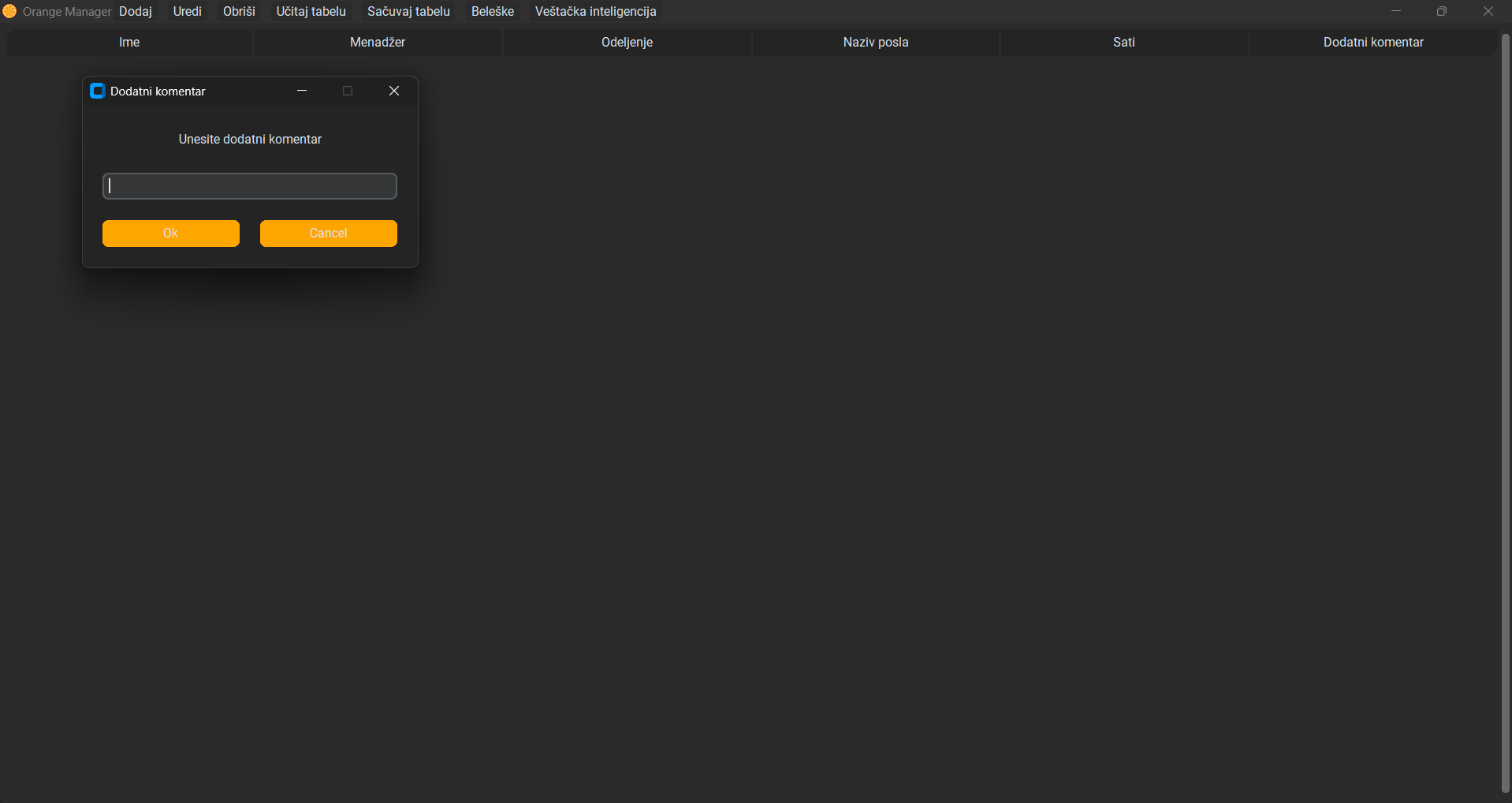
(Slika 4 Unos odeljenja)



(Slika 5 Unos naziva posla)

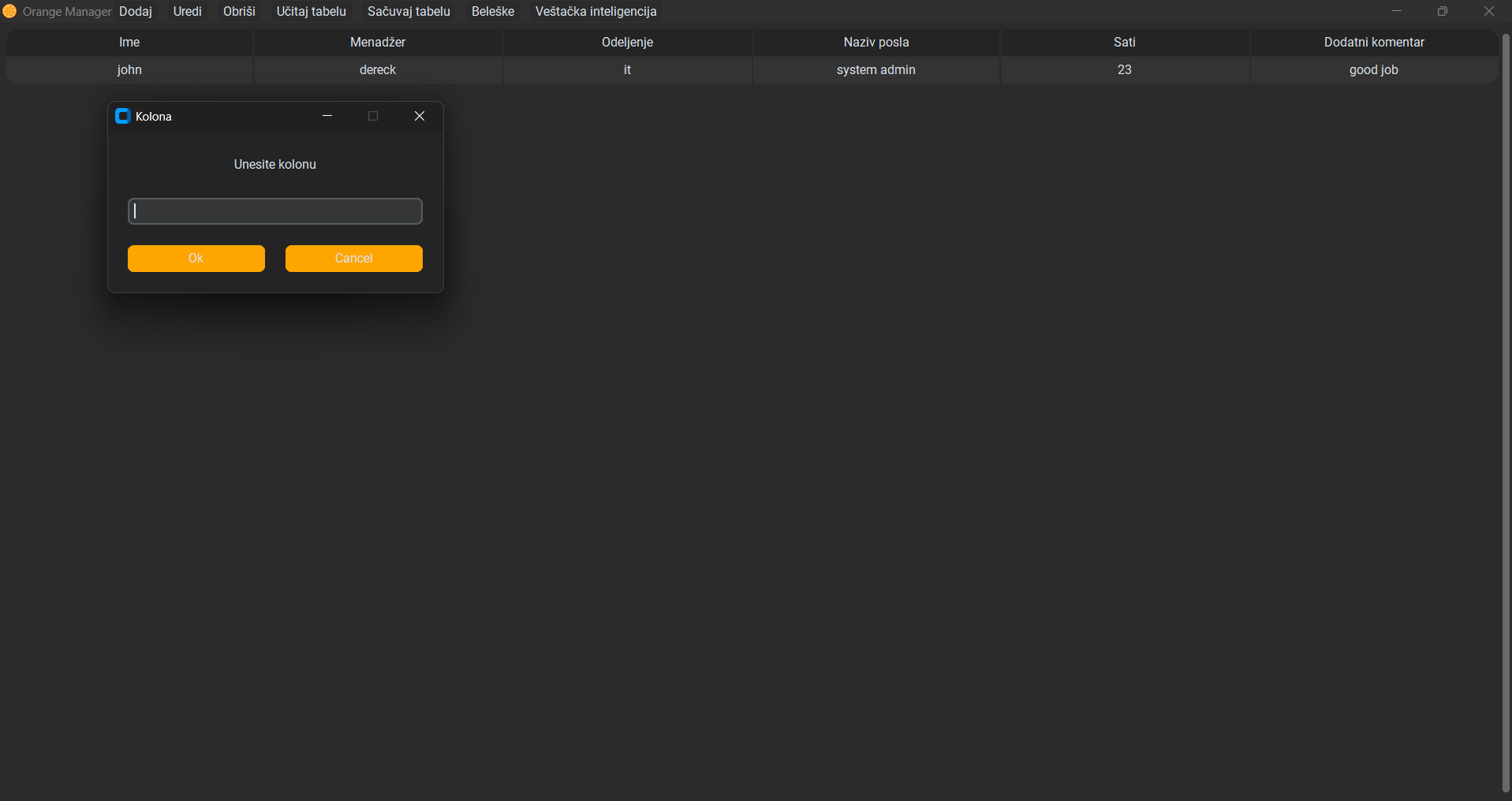


(Slika 6 Unos radnih sati)

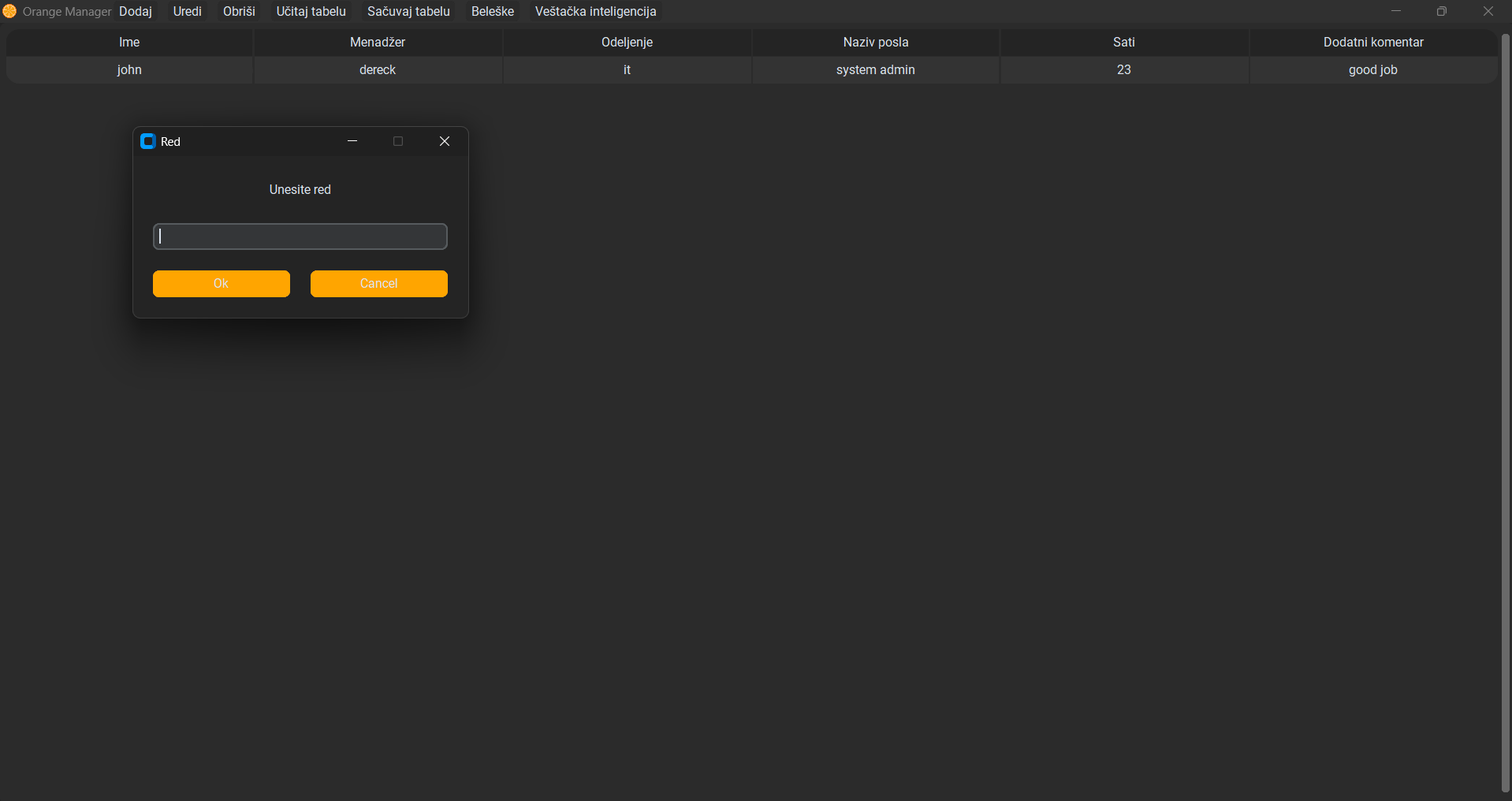


(Slika 7 Unos dodatnog komentara)

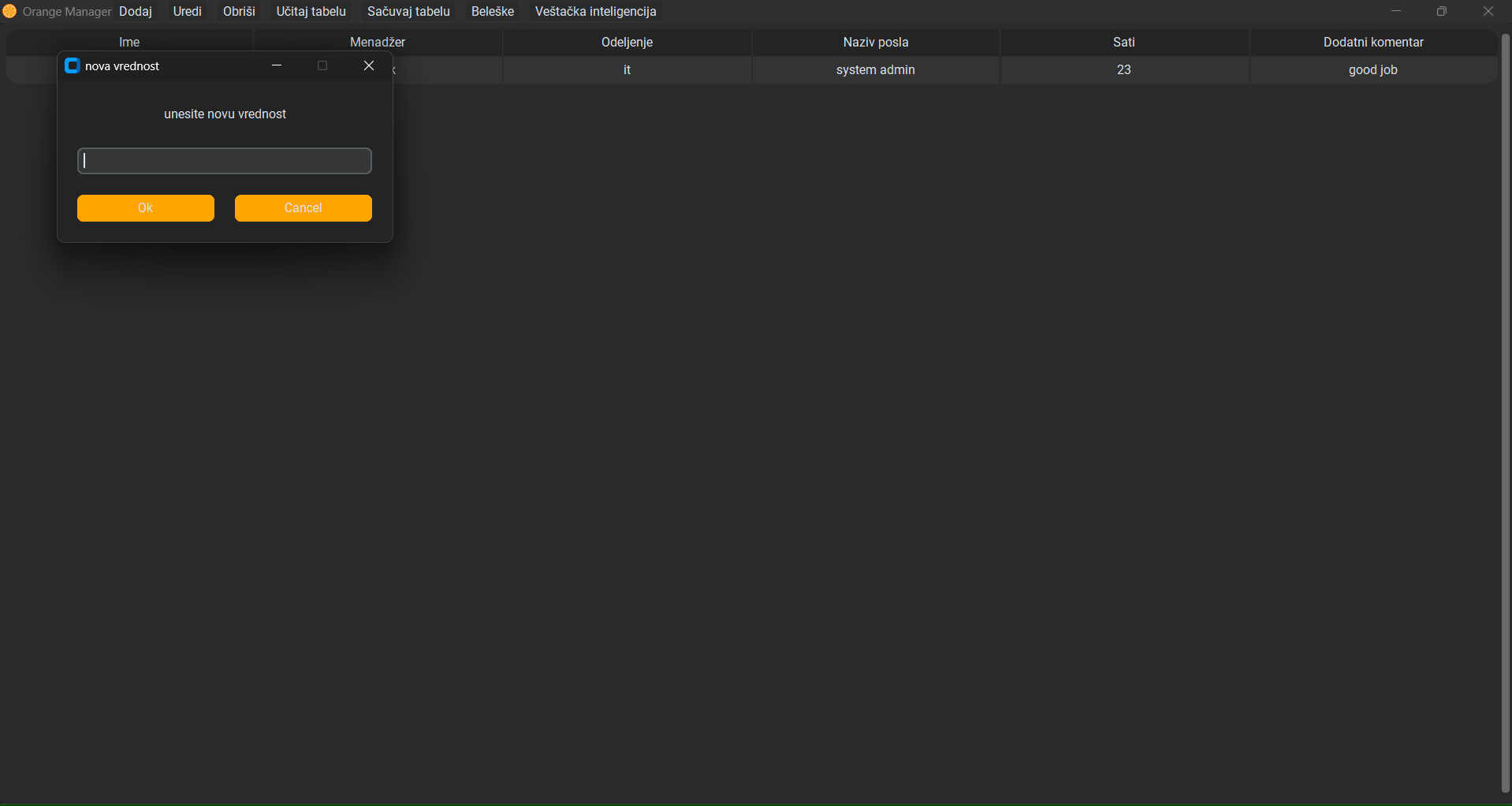
Kada koristimo meni imamo opciju uredi sa kojom se može dodati kolona ili red. Nijedno polje ne sme ostati prazno jer se izbacuje poruka da dolazi do greške.



(Slika 8 Primer za unošenje kolone)

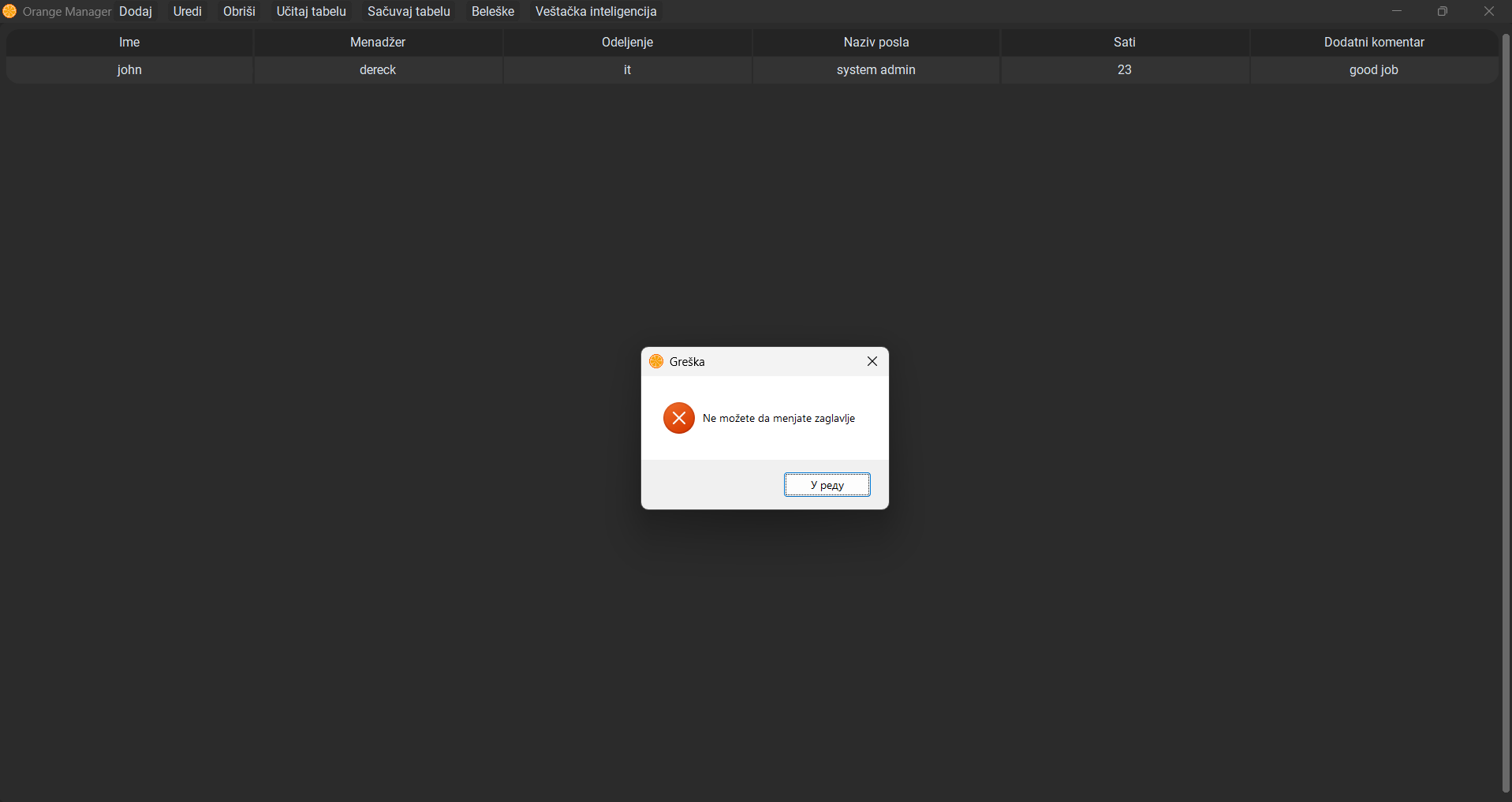


(Slika 9 Primer za unošenja reda)

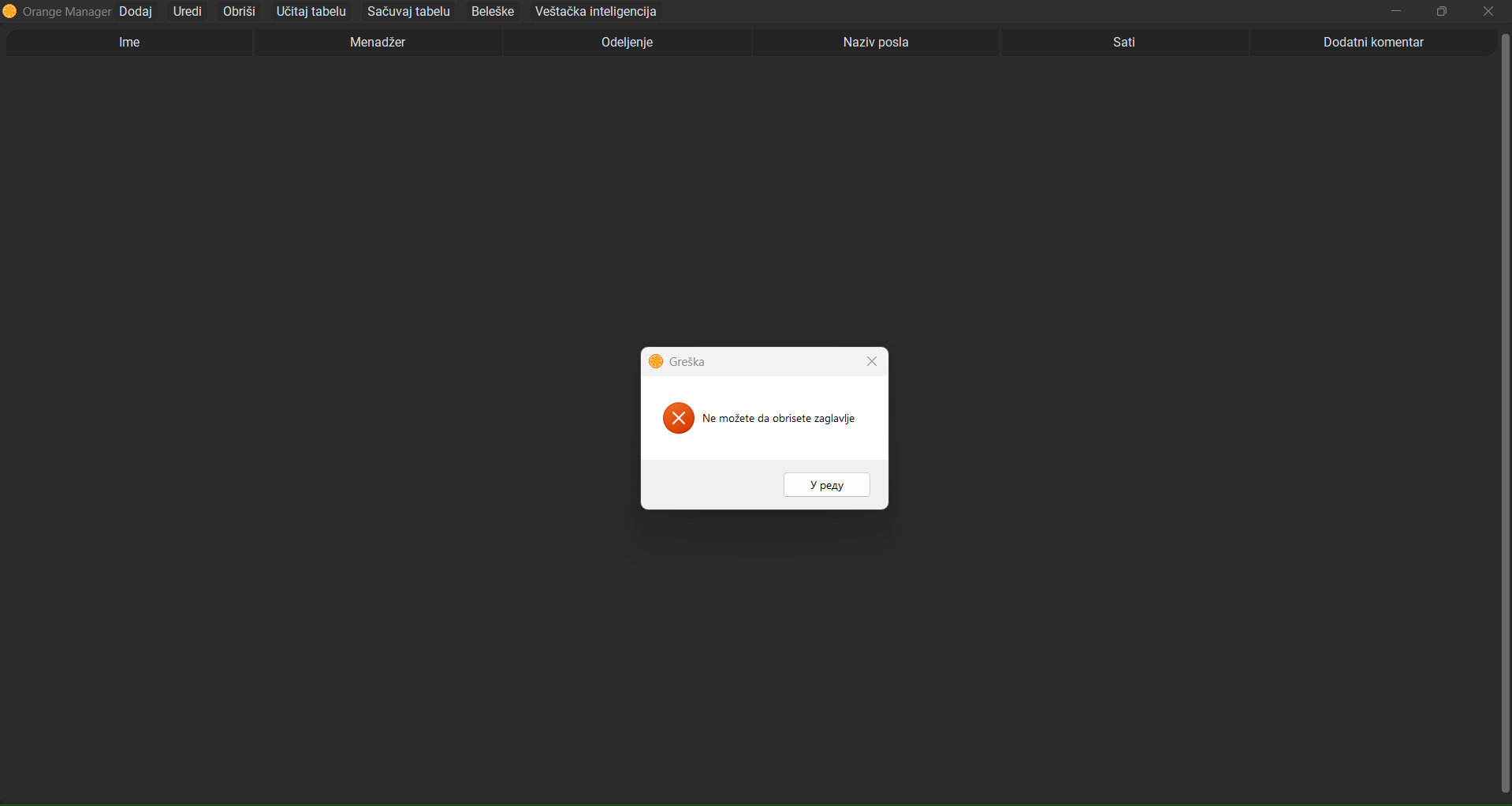


(Slika 10 Unos nove vrednosti)

Samo zaglavlje korisnici ne mogu da menjaju ili obrišu. Kada to pokušaju dolazi do izbacivanja prozora za grešku koji ih zaustavlja.



(Slika 11 Izbacivanje prozora za grešku prilikom pokušaja da se promeni zaglavlje)



(Slika 12 Izbacivanje prozora za grešku prilikom pokušaja da se izbriše zaglavlje)

Klikom na beleške ili na veštačku inteligencija otvaraju nam se 2 nova prozora. Jedan koji se koristi za beleške o radnicima I ima osnovne opcije za čuvanje I brisanje. Drugi je prozor za chat-bota koji je tu da pruži asistenciju u slučaju da je potrebna neka pomoć.

Da bi se napravio notes korišćen je sledeći kod:

import customtkinter, CTkMenuBar, tkinter, tkinter.messagebox, tkinter.messagebox #importovanje modula I biblioteka

class Notes(customtkinter.CTk): #definisanke klase notes koja predstavlja aplikaciju

def \_\_init\_\_(self): #inicijalizacija aplikacije

super().\_\_init\_\_()

customtkinter.set\_appearance\_mode("Dark") #podešava tamni način rada

customtkinter.set\_default\_color\_theme("blue") #podešava plavu temu kao podrazumevanu

self.title("Notes") # postavlja naslov prozora

self.geometry("600x500") #definiše veličinu prozora

self.iconbitmap("slike/Orange\_Manager.ico") #postavlja ikonicu aplikacije

self.menu\_bar = CTkMenuBar.CTkTitleMenu(self) #glavni meni apkikacije

self.config(menu=self.menu\_bar) #povezuje se sa glavnim prozorom

self.menu\_bar.add\_cascade("Otvori", command=self.open\_file) #dodaje komandu Otvori koja poziva open\_file()

self.menu\_bar.add\_cascade("Sacuvaj", command=self.save\_file) #dodaje komandu Sačuvaj koja poziva save\_file()

self.menu\_bar.add\_cascade("Obrisi", command=self.clear\_text) #dodaje komandu Obriši koja poziva clear\_text()

self.text\_area = customtkinter.CTkTextbox(self, wrap="word", font=("Arial", 14))

self.text\_area.pack(fill="both", expand=True) #rasteže tekstualno polje da zauzme ceo prozor

def save\_file(self): #čuvanje fajla

self.file\_path = tkinter.filedialog.asksaveasfilename(filetypes=[("Text Files", "\*.txt"), ("All Files", "\*.\*")]) #korisnik može uneti ime fajla I odabrati ekstenziju

if self.file\_path:

try:

with open(self.file\_path, "w+", encoding="utf-8") as self.file: # fajl se otvara u režimu wt

self.file.write(self.text\_area.get("1.0", tkinter.END)) # tekst se ispisuje od prvpog reda do kraja

tkinter.messagebox.showinfo("Sacuvano", "Beleska je uspesno sacuvana.") # prikazuje obaveštenje o uspešno čuvanju

except FileNotFoundError as e:

tkinter.messagebox.showerror("Greska", f"Doslo je do greske pri cuvanju: {e}") #ako dodje do greške prikazuje se prozor sa greškom

def open\_file(self):

self.file\_path = tkinter.filedialog.askopenfilename(filetypes=[("Text Files", "\*.txt"), ("All Files", "\*.\*")]) #otvaranje dijaloga za biranje fajla

if self.file\_path:

try:

with open(self.file\_path, "r+", encoding="utf-8") as self.file: #fajl se otvara u režimu rt

self.content = self.file.read()

self.text\_area.delete("1.0", "end")

self.text\_area.insert("1.0", self.content) #tekst fajl se učitava I ubacije u text\_area

except FileNotFoundError("Fajl ne postoji") as e:

tkinter.messagebox.showerror("Greska", f"Doslo je do greske pri otvaranju: {e}")

def clear\_text(self):

self.text\_area.delete("1.0", "end") #briše sav tekst iz text\_area

if \_\_name\_\_ == "\_\_main\_\_":

app = Notes().mainloop() #Ako se fajl pokrene direktno (\_\_name\_\_ == "\_\_main\_\_"), kreira se instanca Notes() i poziva mainloop(), koji pokreće Tkinter petlju.

Kod za povezivanje sa provajderom za chat-bota:

import g4f #g4f – Python biblioteka koja omogućava interakciju sa besplatnim API-jem za OpenAI modele, g4f.Provider omogućava izbor provajdera za AI odgovore

class LargeLanguageModel: #kreira klasu koja omogućava dobijanjw odgovora od AI modela

def ResponseFromAI(self, prompt): #slanje upita modelu

self.user\_query = [{f"role": f"user", f"content": prompt}] #prima prompt kao korisnički unos

self.response = g4f.ChatCompletion.create(model=f"gpt-4o", messages=self.user\_query) #g4f.ChatCompletion.create()-šalje upit AI modelu, model=”gpt-4o”-koristi GPT-4o, messages=self.user\_query-prosleđuje korisnički upit

return self.response #vraća generisani odgovor

Kod za chat-bota:

import customtkinter, tkinter, typing, LLM, speech\_recognition #importovanje biblioteka: typing omogućava korišćenje naprednih tipova podataka(self za anotaciju metoda unutar klase),LLM prilagođena biblioteka za rad sa velikim jezičkim modelima, speech recognition omogućava prepoznavanje govora pomoću mikrofona

class AI\_Window(customtkinter.CTk): # Klasa AI\_Window nasleđuje customtkinter.CTk, što znači da je glavni prozor aplikacije baziran na customtkinter

def \_\_init\_\_(self: typing.Self, \*args, \*\*kwargs) -> None:

super().\_\_init\_\_(\*args, \*\*kwargs) #args,kwargs omogućava prosleđivanje dodatnih argument+ata prilikom instanciranja

self.title("ai chatbot") #postavlja naslova prozora

self.geometry(f"525x300") #postavlja veličinu prozora

self.resizable(False, False) #onemogućava promenu veličine prozora

self.iconbitmap("slike/Orange\_Manager.ico") #postavlja ikonicu prozora

self.ai\_window\_textbox = customtkinter.CTkTextbox(master=self, height=265, width=524, corner\_radius=0, fg\_color=f"transparent", text\_color=(f"black", f"white")) # CTkTextbox služi za prikaz AI odgovora i korisničkog unosa, corner\_radius=0 uklanja zaobljenje uglova, fg\_color="transparent" – postavlja pozadinu kao providnu, text\_color=("black", "white") – crni tekst u svetlom modu, beli u tamnom.

self.ai\_window\_textbox.place(x=0, y=0) #postavlja textbox na vrh prozora

self.ai\_window\_textbox.configure(state=f"disabled") #onemogućava ručno menjane teksta

self.ai\_window\_entry = customtkinter.CTkEntry(master=self, height=30, width=465, border\_width=0, fg\_color=f"transparent", placeholder\_text=f"...") #CTjEntry omogućava korisniku da unese poruku, placeholder\_text=”…”-postavlja podrazumevani tekst(koji nestaje kad korisnik počne da kuca

self.ai\_window\_entry.place(x=0, y=269)

self.ai\_window\_microphone\_button = customtkinter.CTkButton(master=self, height=30, width=30, border\_width=0, fg\_color=f"transparent", text=f"🎤", command=self.AudioInput) #dodavanje mikrofonskog dugmeta, command=self.AudioInput-klik na dugme pokreće AudioInput()

self.ai\_window\_microphone\_button.place(x=465, y=269)

self.ai\_window\_send\_request\_button = customtkinter.CTkButton(master=self, height=30, width=30, border\_width=0, fg\_color=f"transparent", text=f"->", command=self.Response) #dugme za slanje poruka, klik poziva funkciju Responese()koja obrađuje korisnički unos I generiše odgovor

self.ai\_window\_send\_request\_button.place(x=495, y=269)

self.ai\_window\_entry.bind(f"<Return>", self.Response) #Kada korisnik pritisne Enter, poziva se funkcija Response()

def Response(self, configure):

self.ai\_window\_entry\_data = self.ai\_window\_entry.get() # Uzima tekst iz Entry polja (self.ai\_window\_entry\_data)

self.ai\_window\_textbox.configure(state=f"normal") #omogućava unos teksta u taxtbox

self.query = LLM.LargeLanguageModel().ResponseFromAI(self.ai\_window\_entry\_data) #poziva funkciju ResponseFromAI da generiše odgovor

self.ai\_window\_textbox.insert(tkinter.END, f"USER:\n{self.ai\_window\_entry\_data}\nGPT-4o:\n{self.query}\n", f"-1.0") #dodaje korisnički unos I AI odgovor u Textbox

self.ai\_window\_textbox.configure(state=f"disabled") #onemogućava unos u Textbox

self.ai\_window\_entry.delete(f"-1", tkinter.END) #briše Entry polje nakon slanja poruke

def AudioInput(self):

self.recognizer = speech\_recognition.Recognizer() #kreira objekat Recognizer() iz speech\_recognition

with speech\_recognition.Microphone() as self.source:

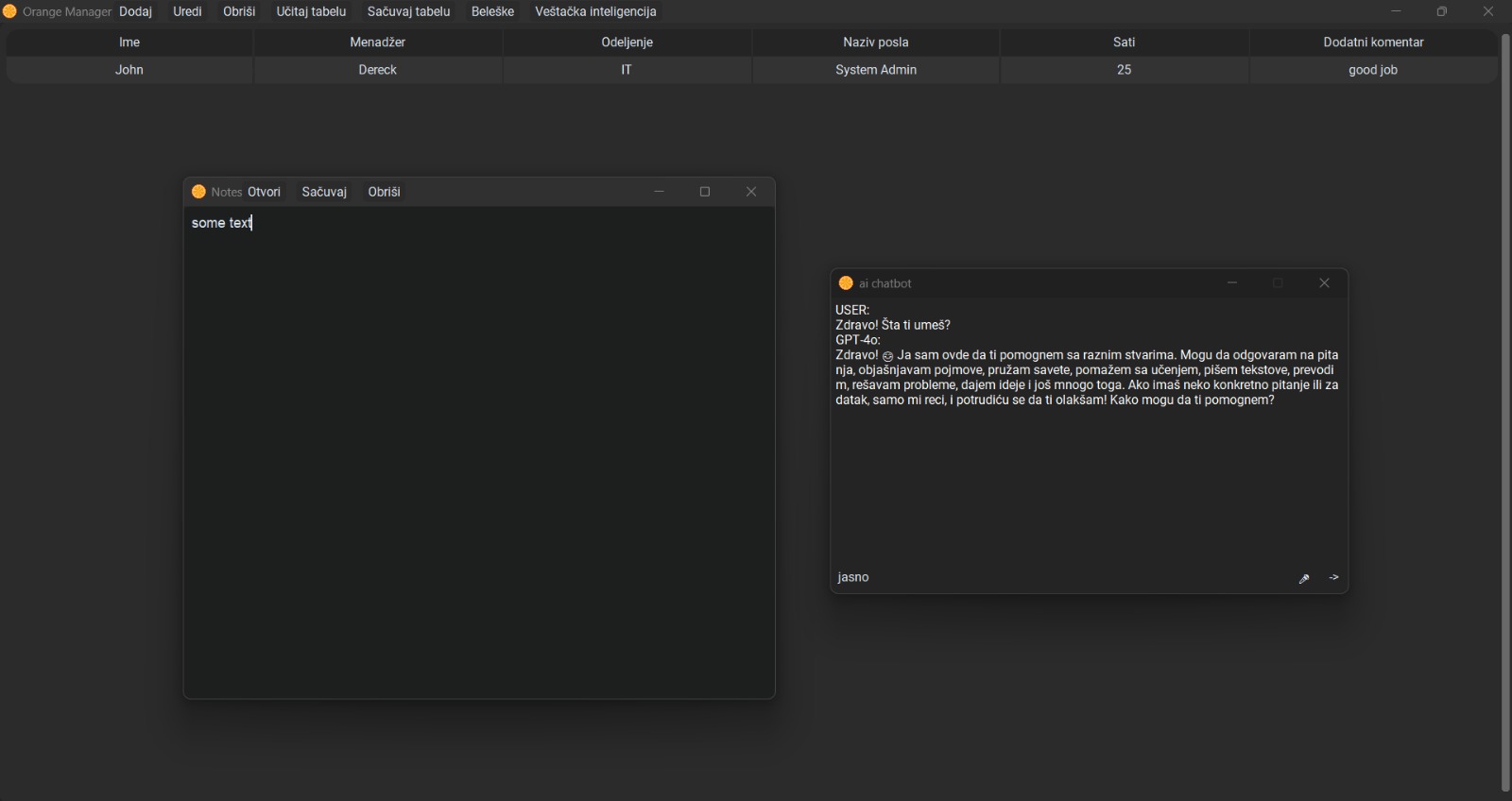
self.audio\_data = self.recognizer.record(self.source, duration=5) #koristi mikrofon da snimi glas(trajanje 5 sekundi)

self.text = self.recognizer.recognize\_google(self.audio\_data) #koristi Googleov API za prepoznavanje govora I konvertuje zvuk u tekst

self.ai\_window\_entry.insert(f"0", self.text) #ubacuje prepoznati tekst u Entry polje

if \_\_name\_\_ == f"\_\_main\_\_":

AI\_Window().mainloop() # Ako se fajl pokrene direktno (\_\_name\_\_ == "\_\_main\_\_"), instancira se AI\_Window() i pokreće mainloop()



(Slika 13 Beleške I Veštačka inteligencija)

Radu lakše upotrebe korišćen je Pyinstaller (auto-py-to-exe) koji omogućava da se aplikacija otvori bez obzira na to da li korisnik Python nsatliran ili ne.

# 5.Zakljucak

Korišćenje Python-a za razvoj menadžment sistema donosi mnoge prednosti, uključujući jednostavnost implementacije, bogat ekosistem biblioteka i fleksibilnost u prilagođavanju poslovnim potrebama. Kombinacijom različitih tehnologija, uključujući baze podataka, web frameworke i GUI biblioteke, moguće je razviti kompletna softverska rešenja koja odgovaraju specifičnim zahtevima korisnika. Takođe, zahvaljujući open-source zajednici i dostupnim resursima, razvoj menadžment sistema u Python-u postaje pristupačniji i brži.

U budućnosti, dalji razvoj menadžment sistema u Python-u može se očekivati u pravcu veće integracije sa veštačkom inteligencijom, analitikom podataka i cloud tehnologijama, čime bi se dodatno unapredila njihova efikasnost i prilagodljivost potrebama modernog poslovanja.

# 6.Literatura

[1] <https://developer.mozilla.org/en-US/docs/Glossary/Python>

[2] <https://customtkinter.tomschimansky.com/documentation/>

[3] <https://cubes.edu.rs/programiranje/sta-je-objektno-orijentisano-programiranje/>

[4] <https://www.geeksforgeeks.org/introduction-to-auto-py-to-exe-module/>

[5] <https://g4f.mintlify.app/docs/get-started/introduction>