Sustav za izradu interaktivnog rasporeda

Glavaš, Dominik

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:373442

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-11-30**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

SUSTAV ZA IZRADU INTERAKTIVNOG RASPOREDA

Rijeka, rujan 2020.

Dominik Glavaš

0069083026

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

SUSTAV ZA IZRADU INTERAKTIVNOG RASPOREDA

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2020.

Dominik Glavaš

0069083026

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 5. ožujka 2020.

Zavod: **Zavod za računarstvo**
Predmet: **Uvod u objektno orijentirano programiranje**
Grana: **2.09.02 informacijski sustavi**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Dominik Glavaš (0069083026)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Sustav za izradu interaktivnog rasporeda / Interactive scheduling system**

Opis zadatka:

Izraditi programsko rješenje za izradu rasporeda djelatnika i strojeva poslovnog subjekta na osnovu unosa u pripadnoj bazi podataka te interaktivnu mogućnost dorade rasporeda i pronalazak prvog slobodnog termina za novi upit. Tehnički i grafički oblikovati korisničko sučelje traženog programskog sustava u vidu desktop aplikacije.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Dominik Glavaš

Zadatak uručen pristupniku: 16. ožujka 2020.

Mentor:

Doc. Goran Mauša, dipl. ing.

Predsjednik povjerenstva za završni ispit:

Izv. prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam završni rad „Sustav za izradu interaktivnog rasporeda“ izradio samostalno, koristeći literaturu i znanje stečeno na Tehničkom fakultetu Rijeka, uz pomoć mentora doc. dr. sc. Gorana Mauše

Rijeka, rujan 2020. _____

Dominik Glavaš

ZAHVALA

Zahvaljujem se mentoru doc. dr. sc. Goranu Mauši na pruženoj prilici i pomoći pri izradi završnog rada, te na svim smjernicama i savjetima za što bolje izvršavanje zadatka. Isto tako se iznimno zahvaljujem roditeljima i prijateljima na svoj pruženoj potpori.

Sadržaj

1. UVOD

Ovaj rad temelji se na, i opisuje izradu programskog rješenja za sustav izrade interaktivnog rasporeda. Problematika izrade rasporeda leži u optimizaciji korištenja raspoloživih resursa (npr. radnici, strojevi, materijal i slično) koji se raspoređuju te je uvelike uvjetovana samom količinom tih resursa. Optimiziranim rasporedom nastoji se koristiti što je manje resursa moguće za obavljanje određene količine posla i zadataka. Izrada aplikacije koja na temelju određenih podataka automatski generira interaktivan i optimiziran raspored upravo je bio cilj ovog projekta. Aplikacije radne površine (desktop aplikacije) vrlo je širok pojam koji obuhvaća sve aplikacije koje se mogu samostalno izvršavati na računalu. Tokom izrade desktop aplikacije potrebno je fokusirati se na više stvari istovremeno kao što su funkcionalnosti, intuitivnost korištenja same aplikacije, optimizacija, izgled korisničkog sučelja i drugo. Obim funkcionalnosti aplikacije određuje njene mogućnosti te samim time i fleksibilnost korištenja u raznim situacijama. Poznavanje svih funkcionalnosti daje korisniku bolju sliku o tome na koje se sve načine i u koje svrhe može koristiti aplikacija te koja su njena ograničenja. Vrlo bitan aspekt izrade desktop aplikacije je izrada grafičkog korisničkog sučelja. Korisničko sučelje je ono što korisnik vidi tokom korištenja aplikacije te ono uvelike uvjetuje jednostavnost i intuitivnost korištenja iste. Prema ISO/IEC 9126 standardu kvalitete, upotrebljivost je jedna od karakteristika internog/eksternog modela kvalitete. Upotrebljivost kao karakteristika dijeli se na podkarakteristike kao razumljivost, naučivost, operabilnost, atraktivnost i usklađenost upotrebljivosti[1]. Sklop svih navedenih značajki dovodi do uspješne i djelotvorne aplikacije. Cilj ovog projekta bio je, na temelju podataka o radnim nalozima i rokovima izvršavanja zadataka, napraviti aplikaciju sa interaktivnim „povuci i pusti“ (eng. drag and drop) sučeljem koje stvara prijedlog mogućeg rasporeda korištenja strojeva potrebnih za izradu određenog zadatka. Za izradu aplikacije koristio se programski jezik Python te neki od njegovih dostupnih modula i knjižnica potrebni za implementaciju funkcionalnosti kao što su već navedeni *drag and drop*, suradnja sa datotekama nastavka .xlsx i .csv. *Drag and drop* je grafički način prijenosa informacija gdje korisnik povlači grafičke komponente i ubacuje ih na odabrana mjesta. Implementacija te mogućnosti čini temelj aplikacije izrađene u sklopu ovog projekta.

2. SPECIFIKACIJA ZAHTJEVA

Zadatak projekta bio je izraditi programsko rješenje za izradu interaktivnog rasporeda korištenja strojeva.

Zahtjev Z1 - Raspored treba biti predstavljen u obliku grafičkog korisničkog sučelja kako bi se vizualno prikazali podatci i njihova podjela u rasporedu.

Zahtjev Z2 – Potrebno je omogućiti lagan i vizualan prijenos i zamjenu podataka pomoću *drag and drop* funkcionalnost u kojoj korisnik može povući grafički element sa sučelja i ispustiti ga na drugoj ili istoj lokaciji u svrhu zamjene i prijenosa podataka.

Zahtjev Z3 – Iz baze podataka treba dohvatiti potrebne podatke za smještanje zadataka u raspored na pravo mjesto.

Zahtjev Z4 – Popunjavanje rasporeda informacijama koje su dohvaćene iz baze.

Zahtjev Z5 – Mogućnost ručnog unosa zadataka u raspored.

Zahtjev Z6 - Također je potrebno implementirati mogućnost spremanja trenutnog stanja rasporeda kako se napravljene promjene ne bi izgubile pri izlasku iz aplikacije.

Zahtjev Z7 - Uvoz spremljenog stanja rasporeda kako bi se određeni raspored mogao ponovno koristiti i izmjenjivati.

Zahtjev Z8 – Posljednji zahtjev bio je uvesti mogućnost prebacivanja rasporeda u Excel tablicu primarno radi lakše izrade rasporeda u fizičkom formatu.

Sažetak svih zahtjeva ovog projekta može se vidjeti u tablici 2.1. Ovaj sustav mogao bi se dalje unaprijediti na način da se uvede funkcionalnost pronalaženja slobodnog termina za nekakav novi upit u slučaju gdje je raspored vrlo gust te za upit nema mjesta ukoliko se ne napravi neka veća izmjena. Podatci potrebni za realizaciju ovih mogućnosti pribavljali su se iz baze podataka popunjene sa primjerima mogućih radnih naloga. Baza podatka dio je ranije izrađenog sustava koji pospješuje organizaciju poslovanja u smislu pohrane podataka i automatizacije unosa istih, te izrade radnih naloga za obrt. Ona osim podataka potrebnih za izradu rasporeda sadrži sve druge bitne informacije o radnim nalozima, kao što su na primjer materijal i alat koji se koristi. Ovaj projekt nadovezuje se na postojeći sustav i koristi pogodnosti njegove baze za izradu rasporeda.

Tablica 2.1. Zahtjevi projekta

ID	**Zahtjev**
Z1	Izrada grafičkog korisničkog sučelja
Z2	Implementacija *drag and drop* mehanizma
Z3	Dohvaćanje informacija iz baze podataka
Z4	Popunjavanje rasporeda informacijama
Z5	Ručno unošenje zadataka
Z6	Spremanje stanja rasporeda
Z7	Učitavanje spremljenog rasporeda
Z8	Izvoz rasporeda u Excel tablicu

3. TEHNIČKO OBLIKOVANJE

Za uspješno izvršavanje zahtjeva iz prethodnog poglavlja bilo je potrebno upoznati se sa sustavom na koji se ovaj projekt nadovezuje, te vidjeti koje su sve informacije na raspolaganju. Uspostavljeno je da u bazi podataka postojećeg sustava nedostaje informacija o broju dana potrebnih za izvršavanje pojedinog zadatka te je u sklopu ovog projekta izvršena izmjena na bazi kako bi ona sadržavala i tu informaciju.

3.1. Rješenja pojedinih zahtjeva

Rješenje R1 - Temeljni zahtjev za stvaranje ove aplikacije je izrada grafičkog korisničkog sučelja te je prvi korak pri izradi projekta bio stvoriti sučelje. Za izradu grafičkog korisničkog sučelja u ovom projektu koristio se PyQt5 modul koji je za Python prilagođen API (Application Programming Interface) preuzet od GUI (Graphical User Interface) priručnika Qt. PyQt5 sadrži velik broj Python modula od kojih su za svrhu izrade ovog projekta korišteni: QtCore koji sadrži glavne klase koje nisu vezane za GUI, QtGui koji sadrži većinu klasa vezane za izradu GUI-a kao i 2d platno koje može spremiti tisuće drugih grafičkih komponenti i QtWidgets koji sadrži većinu GUI komponenti kao što su na primjer label (eng. naljepnica) i dugme[2][3]. Kako se radi o rasporedu, najpogodnija opcija podijele prostora sučelja je tablična podjela.

Rješenje R2 - Nakon izrade sučelja, potrebno je implementirati *drag and drop* mehanizam u svrhu lakog i intuitivnog prijenosa podataka kako bi se raspored mogao mijenjati po želji. PyQt5 modul koji se koristi u ovom projektu jedan je od modula koji podržava *drag and drop* što je bio i jedan od glavnih razloga odabira rada sa istim. Potrebno je omogućiti *drag and drop* na svakom objektu za koji želimo da posjeduje tu mogućnost. Za upravljanje detaljima o tome što će se desiti prilikom pojedine akcije (povlačenje ili ispuštanje) postoje metode u kojima se to definira. U ovom projektu to su bili detalji kao zamjena informacija grafičkih komponenti koje se povlači i na koju se ispušta.

Rješenje R3 - Kako bi sučelje prikazivalo smislene informacije potrebno je prvo te informacije dohvatiti iz baze podataka. Za podizanje baze koristi se XAMPP server koji se sastoji od Apache HTTP servera, MariaDB baze podataka i tumača skripti koje su napisane u PHP i Perl

programskim jezicima[4]. Za upravljanje bazom koristi se administracijski alat phpMyAdmin koji je otvorenog koda, pisan primarno u PHP programskom jeziku i jedan od najpopularnijih administracijskih alata. PhpMyAdmin pruža čitak uvid u bazu i njene tablice i elemente pohranjene u njoj što uvelike olakšava vizualizaciju rasporeda podataka unutar same baze[5]. Za komunikaciju sa bazom koristi se mysql modul koji ima sve potrebne metode za izvršavanje upita u bazu.

Rješenje R4 - Informacije dobivene iz baze potrebno je umetnuti na pravo mjesto u tablici rasporeda. Za to je potrebno saznati broj tjedna u kojem se pojedini zadatak nalazi i dane u tjednu u kojima će se zadatak izvršavati. Za postavljanje zadataka u raspored koristi se rekurzivna funkcija koja u suštini iterira po tablici unazad sve dok ne nađe dovoljno slobodnih mjesta (dana) koliko je potrebno za izvršavanje zadatka koji se trenutno smješta u raspored, te u ta slobodna mjesta upisuje taj isti zadatak.

Rješenje R5 - Pored automatski generiranog rasporeda na temelju podataka iz baze, dodatan zahtjev bio je implementirati mogućnost ručnog unosa zadataka. Potreba za tom mogućnosti proizlazi iz želje da, ukoliko se neki zadatak ne nalazi u bazi podataka i ne želi ga se unijeti u istu, se može jednostavno i brzo ručno unijeti željeni zadatak u željeno polje. Za implementaciju same funkcionalnosti koristi se priroda gumbova. Jednostavnim pritiskom na gumb otvara se prozor koji poziva korisnika da upiše tekst zadatka koji želi dodati. Pošto se lijevi klik miša već koristi za izvođenje *drag and drop* operacija, ručni unos zadatka implementiran je na desni klik miša.

Rješenje R6 - Kako bi promjene izvršene nad rasporedom mogle biti donekle trajne, uvedena je mogućnost spremanja trenutnog stanja rasporeda. Stanje se sprema u datoteku nastavka .csv. CSV (Comma-Separated Values) datoteka je tekstualna datoteka u kojoj su vrijednosti odvojene zarezom. Svaki redak u datoteci predstavlja jedan podatak u tekstualnom obliku. Podatak može biti sastavljen od više polja koja su sva odvojena zarezima jedna od drugih[6]. Ova funkcionalnost pokreće se klikom na gumb „Spremi“.

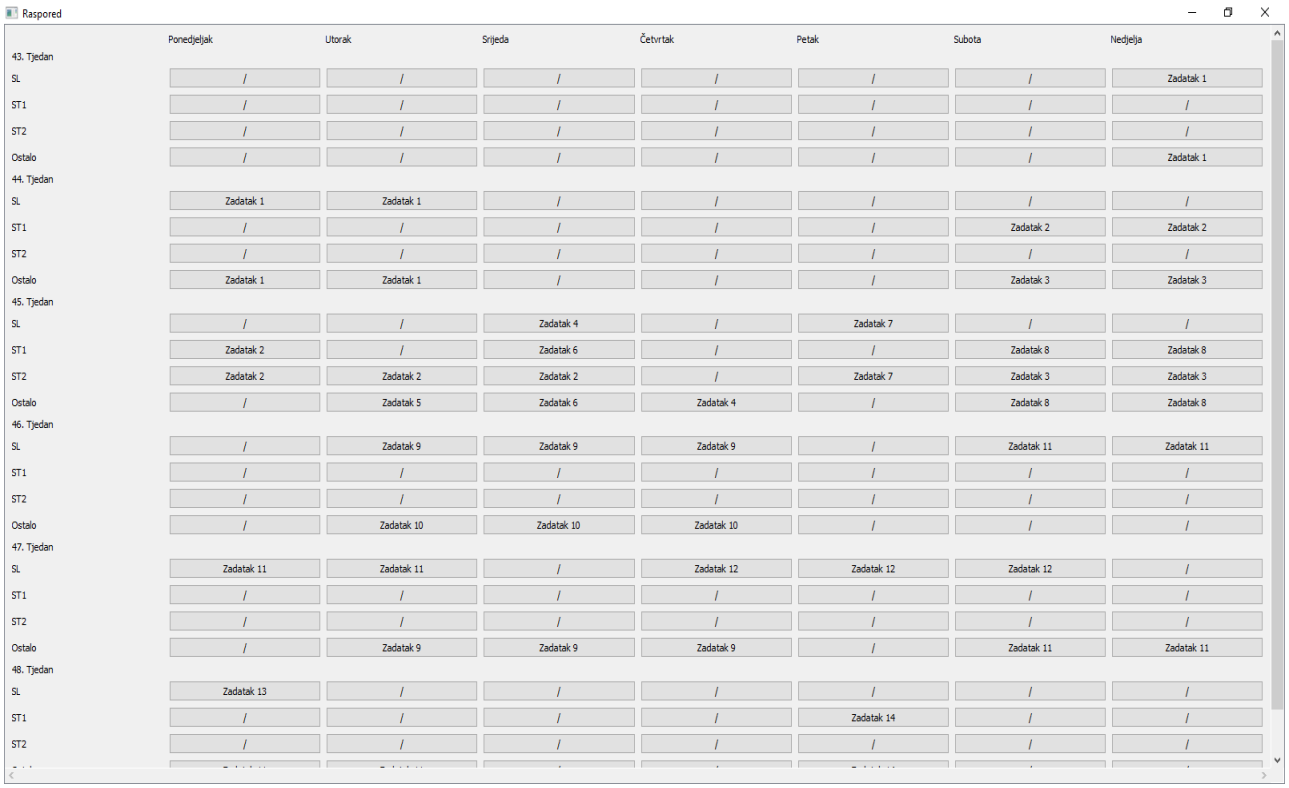
Rješenje R7 - Svrha spremanja stanja rasporeda je ta da se taj isti raspored u nekom budućem trenutku može ponovno koristiti kao predložak kako se ne bi morale vršiti sve promjene na novo. U tu svrhu također je implementirana funkcionalnost učitavanja prijašnje generiranog i spremljenog rasporeda u obliku .csv datoteke. Funkcionalnost je, slično onoj spremanja, implementirana preko dugmeta pod nazivom „Uvezi“.

Rješenje R8 – Posljednji zahtjev je bio izvoz rasporeda iz grafičkog korisničkog sučelja u Excel tablicu u .xlsx formatu. Za izvedbu ove funkcionalnosti potrebno je uvesti knjižnicu Workbook iz openpyxl modula i get_column_letter knjižnicu iz openpyxl.utils modula. Openpyxl je Python knjižnica za čiatanje/pisanje iz Pythona u Office Open XML format[7]. Funkcionalnost se kao i prijašnje realizira pritiskom na gumb pod nazivom „Izvezi u Excel“

Pojedine zahtjeve ovog projekta povezane sa njihovim rješenjima mogu se preglednije vidjeti u tablici 3.1. Za zahtjev pronalaska slobodnog termina za novi upit, ukoliko je raspored dovoljno gust da za taj upit nema mjesta, može se implementirati funkcionalnost proširenja rasporeda na veći broj tjedana te ponovna raspodjela već postavljenih zadataka kako bi se našlo dovoljno mjesta za upit. Slika 3.1. prikazuje izgled korisničkog sučelja nakon ispunjenih zahtjeva projekta.

Tablica 3.1. Matrica praćenja zahtjeva

ID_Zahtjev	**Zahtjev**	**ID_Rješenje**
Z1	Izrada grafičkog korisničkog sučelja	R1
Z2	Implementacija *drag and drop* mehanizma	R2
Z3	Dohvaćanje informacija iz baze podataka	R3
Z4	Popunjavanje rasporeda informacijama	R4
Z5	Ručno unošenje zadataka	R5
Z6	Spremanje stanja rasporeda	R6
Z7	Učitavanje spremljenog rasporeda	R7
Z8	Izvoz rasporeda u Excel tablicu	R8



Slika 3.1. Izgled grafičkog korisničkog sučelja

4. KORIŠTENE TEHNOLOGIJE I ALATI

Izbor alata i tehnologije prilikom izrade bilo kakve aplikacije vrlo je važna stvar. Bitno je izabrati optimalne alate i tehnologiju ovisno o namjeni aplikacije koja se radi, kako bi došlo do što manje problema i komplikacija prilikom pisanja koda.

4.1. Python

Za izradu projekta izabran je programski jezik Python zbog njegove široke uporabe i velikog izbora knjižnica koje olakšavaju izradu raznih klasa i funkcionalnosti za svakakve potrebe. Python je primarno objektno-orijentirani programski jezik, ali podržava i proceduralni i funkcijski način pisanja koda[8]. Objektno-orijentirana programska paradigma koristi klase koje služe kao predlošci po kojima se mogu instancirati objekti te iste klase koji zapravo zauzimaju memorijski prostor. Nadalje mogu se unutar klase definirati razne metode koje objekti klase mogu pozvati nad sobom. Metode se mogu koristiti u svrhu izmjene podataka, dohvaćanja podataka i raznih drugih funkcija koje može definirati programer. Također jedan od razloga zbog kojih se u ovom projektu koristio programski jezik Python je to što sadrži veliku podršku za izradu grafičkih korisničkih sučelja, što je jedan od glavnih aspekata ovog projekta i na čemu se isti temelji. Python je dizajniran na način da, umjesto da ima sve funkcionalnosti ugrađene u svoju jezgru, je jako proširiv u smislu svih nadodanih knjižnica i paketa koji se lako mogu uvesti po potrebi programa[8].

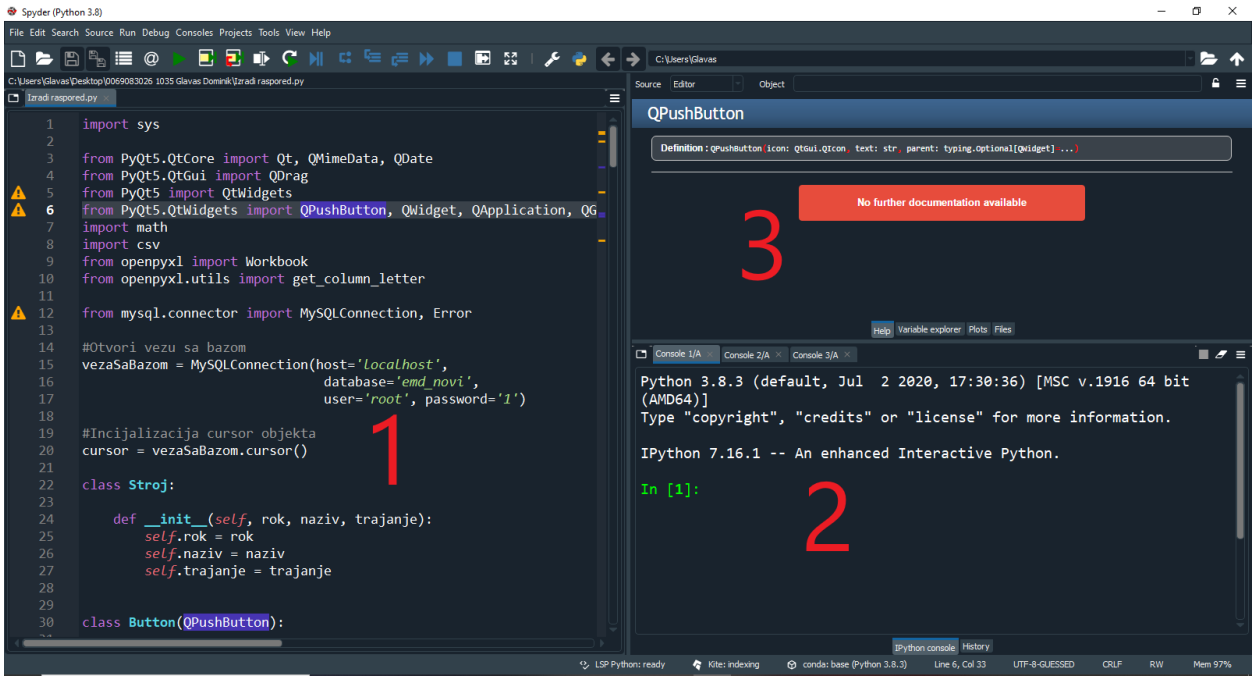
4.2. Anaconda

Pisanje programskog koda, prevađanje i provjera istoga mogu biti vrlo zahtjevni ukoliko se za to ne koriste programi koji pružaju podršku za razvoj. Razvojno okruženje pruža programeru pogodnosti prevoditelja i uređivača koda te programa za ispravljanje pogrešaka. Sve te stvari ključne su u uspješnom pisanju i prevađanju koda pa samim time i za dovršavanje aplikacije. Upravitelj paketa koji se koristi za izradu ovog projekta je Anaconda upravitelj paketa koji je ujedno i distribucija za programski jezik Python[9]. Također Anaconda je prenosiva na svim operativnim sustavima što olakšava rad na bilo kojem od operativnih sustava. Anaconda automatski pruža korisnicima pristup 250 različitim paketima i knjižnicama te daje opciju instaliranja dodatnih 7500 paketa i knjižnica. Također sadrži grafičko korisničko sučelje Anaconda

Navigator koje olakšava korištenje i pruža alternativu nad standardnom komandnom linijom koja se koristi. Preko tog sučelja moguće je izvoditi sve akcije kao i sa komandne linije bez unošenja komandi. Prema zadanim postavkama nakon instalacije Anaconda Navigator-a već su sadržane neke aplikacije kao što su JupyteLab, Jupyter Notebook, QtConsole, Spyder, Visual Studio Code i slično[10]. To su sve alati koji služe za olakšanje razvoja aplikacija i programa. U sklopu ovog projekta koristilo se razvojno okruženje Spyder.

4.3. Spyder

Spyder je razvojno okruženje otvorenog koda za znanstveno programiranje u programskom jeziku Python. U sklopu razvojnog okruženja Spyder integrirano je mnogo poznatih Python znanstvenih paketa kao što su NumPy, Matplotlib, Pandas, SciPy, IPython i drugi. Spyder kao razvojno okruženje ima mnoge prednosti i značajke koje uvelike olakšavaju izradu i dovršavanje programa i aplikacije. Pod te značajke spada: uređivač koda koji uključuje označavanje sintakse, introspekciju i dopunjavanje koda, podrška za višestruke konzole, mogućnost istraživanja i izmjene varijabli sa grafičkog korisničkog sučelja, prozor pomoći koji pruža detaljnije informacije o funkcijama, klasama i metodama automatski ili na zahtjev korisnika, program za ispravljanje koda koji omogućuje praćenje izvođenja koda korak po korak za detaljnu analizu izvedbe svake linije koda, integrirani istraživač datoteka za lakši rad sa datotekama, dnevnik povijesti u kojem su zapisane sve komande unesene u svakoj konzoli[11]. Slika 3.2. prikazuje izgled razvojnog okruženja Spyder gdje se u isto vrijeme prikazuju višestruke konzole, uređivač koda te prozor pomoći u razvojnom okruženju Spyder. Takav prikaz uvelike pojednostavljuje pisanje, vizualizaciju i provjeru ispravnosti programskog koda.



Slika 4.1. 1 – uređivač koda 2 – višestruke konzole 3 – prozor pomoći

5. IMPLEMENTACIJA

5.1. Grafičko korisničko sučelje

Cilj je bio izraditi grafičko sučelje u obliku tablice koje se generira dinamički ovisno o broju zadataka izvučenih iz baze podataka. Da bi se to postiglo potrebno je prvo inicijalizirati QGridLayout objekt koji predstavlja tablični raspored predmeta u ćelije indeksirane brojem stupca i retka. QGridLayout klasa pruža mnoge mogućnosti u vidu oblikovanja rasporeda predmeta od postavljanja dimenzija ćelija, pa do dohvaćanja samih predmeta u tim istim ćelijama, što se ispostavilo veoma korisnim za oblikovanje izgleda tablice u ovom projektu. Također je trebalo ponovno implementirati klasu QMainWindow koja postavlja glavni prozor koji se otvara prilikom pokretanja i stvaranja GUI-a. QMainWindow klasa također sadrži metode za oblikovanje samog prozora kao na primjer setWindowTitle(), pomoću koje postavljamo naziv samog prozora, ili setGeometry(), pomoću koje se postavljaju dimenzije prozora.

Za prikaz informacija, odnosno zadataka, ima puno mogućih opcija od kojih se u ovom projektu radi potrebnih funkcionalnosti koristi dugme. Dugme je prigodna opcija iz razloga što podržava *drag and drop,* koji je sastavni i fundamentalni dio ove aplikacije, i zato što je vrlo intuitivno i lagano povezati pritisak gumba sa određenom metodom. Za izradu dugmeta u PyQt5 modulu koristi se QPushButton klasa koju je potrebno ponovno implementirati kako bi se dodale mogućnosti prihvaćanja i izvođenja *drag and drop* akcija. Također se može definirati koje akcije će se poduzeti u slučaju pomicanja miša i u slučaju ispuštanja predmeta (Slika 4.1. Ponovna implementacija QPushButton klase). Za postavljanje dugmeta na poziciju u rešetkastom rasporedu iterira se kroz polje koje sadrži svako dugme i postavlja ga se na određenu poziciju pomoću addWidget() metode iz QGridLayout klase.

```
30  class Button(QPushButton):
31
32      def __init__(self, title, parent):
33          super().__init__(title, parent)
34
35      def mouseMoveEvent(self, e):
36
37          if e.buttons() != Qt.LeftButton:
38              return
39
40          mimeData = QMimeData()
41          mimeData.setText(self.text())
42
43          self.drag = QDrag(self)
44          self.drag.setMimeData(mimeData)
45          self.drag.setPixmap(self.grab())
46          self.drag.setHotSpot(self.rect().center())
47
48          dropAction = self.drag.exec_(Qt.MoveAction)
49
50      def mousePressEvent(self, e):
51
52          super().mousePressEvent(e)
53
54          if e.button() == Qt.RightButton:
55              text, result = QInputDialog.getText(self, 'Input Dialog', 'Upiši zadatak:')
56              if(result == True):
57                  self.setText(text)
58
```

Slika 5.1. Ponovna implementacija QPushButton klase

Za prikaz tekstualnih podataka koji se ne mijenjaju (dani u tjednu, broj tjedna, naziv stroja) koristila se QLabel klasa koja je najprigodnija za prikaz čistog teksta. Pojedini nazivi strojeva i brojevi tjedna u raspored su se također dodavali addWidget() metodom u istom iterativnom postupku kao i za dugmad (Slika 4.2. Iterativno postavljanje elemenata u rešetku).

```
269          for x in range(maximum - minimum + 1):
270              label = QLabel(str(minimum + x)+". Tjedan")
271              self.layout.addWidget(label, 1 + x * 5, 0)
272              label = (QLabel("SL"), QLabel("ST1"), QLabel("ST2"), QLabel("Ostalo"))
273              for y in range(4):
274                  self.layout.addWidget(label[y], 2 + y + x * 5, 0)
275                  for z in range(7):
276                      self.btns[self.i] = Button("/", self)
277                      self.layout.addWidget(self.btns[self.i], y + 2 + x * 5, z + 1)
278                      self.i += 1
```

Slika 5.2. Iterativno postavljanje elemenata u rešetku

5.2. *Drag and drop*

Drag and drop u grafičkim korisničkim sučeljima označava mogućnost, odnosno funkcionalnost uzimanja predmeta pritiskom tipke i držanje tipke na predmetu, povlačenja/pomicanja predmeta prilikom čega se tipka i dalje drži pritisnuta i ispuštanje predmeta na željenu lokaciju tako što se pusti tipka koja se držala za uzimanje i pomicanje predmeta. Takav način rada široko je rasprostranjen na skoro svim medijima, ali ne podržavaju ga svi programi i programski jezici. Rasprostranjen je iz razloga što se korištenjem tog načina rada može vizualno prikazati što se zapravo dešava sa podatcima i gdje se nalaze. Samim time može se uvelike olakšati intuitivnost i korištenje aplikacija i programa.

Svaka klasa vezana za GUI u PyQt5 modulu podržava *drag and drop* funkcionalnost, ali za primjenu i prihvaćanje takvih akcija potrebno je inicijalizirati objekt QDrag klase u svakoj od tih klasa. QDrag klasa pruža podršku za prijenos podataka zasnovan na MIME podatcima i pruža metode za izgled i rad sa objektima te klase. Za točan rad objekta koji podržava *drag and drop* potrebno je definirati metodu mouseMoveEvent() unutar implementacije klase widgeta gdje se ujedno i inicijalizira QDrag objekt i postavlja akcija koja će se poduzeti prilikom puštanja widgeta. To mogu biti akcije kopiranja podataka, pri čemu podatci ostaju na izvornoj poziciji kao i na novoj poziciji, i akcija pomicanja podatka, pri čemu se podatci sa prvotne pozicije premještaju na lokaciju ispuštanja operacije povlačenja. Također treba definirati metode dragEnterEvent() gdje se prihvaća akcija povlačenja, dragMoveEvent() gdje se također prihvaća akcija povlačenja za elemente koji su već u stanju povlačenja i pomiču se te metoda dropEvent() u kojoj se definira što će se desiti prilikom puštanja widgeta koji je u stanju povlačenja.

Metoda dragMoveEvent() u ovom projektu implementirana je na način da vrši provjeru je li widget kojeg pomičemo i preko kojeg pomičemo tipa Button i vrši se provjera radi li se kojim slučajem o istom widgetu. Ukoliko jedan od widgeta nije tipa Button ili se radi o istom widgetu odbija se akcija tako da se između ta dva widgeta ne može izvršiti *drag and drop* akcija. Metoda dropEvent() definirana je tako da se u trenutku ispuštanja widgeta dobavlja pozicija istog te pozicija widgeta na koji se ispušta. Nakon toga slijedi provjera opisana u metodi dragMoveEvent() te ukoliko provjera prođe dobavljaju se indeks i pozicija oba widgeta preko funkcija indexOf() i

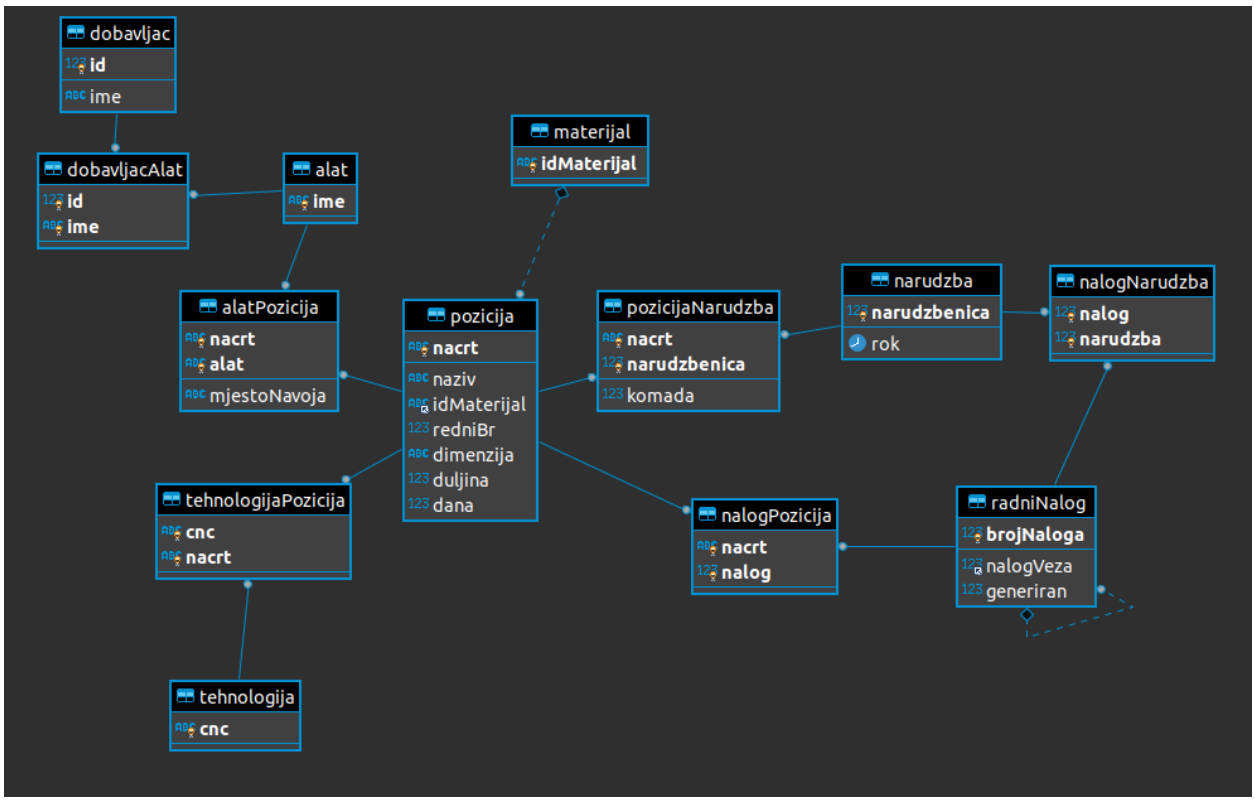
getItemPosition() iz QGridLayout klase. Pomoću indeksa i pozicije vraćaju se widgeti na njihove prvotne pozicije, međutim zamjenjuje im se tekst (Slika 4.3. Implementacija „drag“ metoda).

```
307     def dragEnterEvent(self, e):
308         e.accept()
309 
310     def dragMoveEvent(self, e):
311         src = e.source()
312         trg = QApplication.widgetAt(self.mapToGlobal(e.pos()))
313         if (isinstance(e.source(), Button) and isinstance(trg, Button) and trg != src):
314             e.accept()
315         else:
316             e.ignore()
317 
318     def dropEvent(self, e):
319         src = e.source()
320         trg = QApplication.widgetAt(self.mapToGlobal(e.pos()))
321         if (not isinstance(src, Button) or not isinstance(trg, Button)
322             or trg == src):
323                 return
324         layout = self.widget.layout()
325 
326         sourceIndex = layout.indexOf(src)
327         sourcePos = layout.getItemPosition(sourceIndex)
328 
329         targetIndex = layout.indexOf(trg)
330         targetPos = layout.getItemPosition(targetIndex)
331 
332         layout.addWidget(src, *sourcePos)
333         layout.addWidget(trg, *targetPos)
334 
335         tekst = src.text()
336         src.setText(trg.text())
337         trg.setText(tekst)
338 
339         e.accept()
```

Slika 5.3. Implementacija „drag“ metoda

5.3. Baza podataka

Svaka baza podataka sastoji se od entiteta i atributa. Entiteti su bilo što o čemu se spremaju podatci. Na primjer, u pojednostavljenoj bazi nekakve online trgovine entitet bi mogao biti „Proizvod“, te bi se za pojedine proizvode spremali podatci o istom. Ti podatci koji opisuju entitete su atributi i oni su skupa sa entitetima glavne sastavnice baze podataka. U prijašnjem primjeru pojednostavljene baze online trgovine atributi koji bi mogli pripadati entitetu „Proizvod“ mogli bi biti cijena, naziv, šifra i slično. U bazi još postoje veze između entiteta koje određuju njihovu povezanost. U bazi podataka entiteti i njihovi atributi spojeno su prikazani kao tablice gdje su atributi stupci tablice (Slika 4.4. ER (Entity Relationship) dijagram).



Slika 5.4. ER (Entity Relationship) dijagram

Podatci iz baze koji su korišteni u ovom projektu su podatci o nazivu zadatka, roku izrade zadatka te vremenskom trajanju izrade zadatka u danima. Kako bi se iz Pythona mogla uspostaviti veza i komunicirati sa bazom podataka potrebno je uvesti mysql modul koji ima sve potrebne metode za te potrebe. Veza se uspostavlja sa mySQLConnection() metodom a za dobavljanje podataka i izvršavanje upita nad bazom inicijalizira se kursor objekt pozivom metode cursor().

Radi lakšeg korištenja podataka dobavljenih iz baze konstruirana je klasa pod nazivom „Stroj“ koja ima tri varijable klase (Slika 4.5. Klasa „Stroj“). Iz te se klase nadalje koriste informacije o nazivu, roku i trajanju pojedinih zadataka. Također radi lakše raspodjele zadataka u tablici grafičkog sučelja kreirana su 4 polja koja sadrže vrijednosti tipa „Stroj“, jedno za svaki stroj koji se koristi. Time su podatci odmah u startu podijeljeni po strojevima koji ih izvode, što u kombinaciji sa brojem tjedna u kojem se zadatak izvodi definira redak u tablici gdje će se smjestiti pojedini zadatak.

```
22  class Stroj:
23
24      def __init__(self, rok, naziv, trajanje):
25          self.rok = rok
26          self.naziv = naziv
27          self.trajanje = trajanje
28
```

Slika 5.5. Klasa „Stroj“

5.4. Popunjavanje rasporeda

Informacije dobivene iz baze potrebno je umetnuti na pravo mjesto u tablici rasporeda. Za to je potrebno saznati broj tjedna u kojem se pojedini zadatak nalazi i dane u tjednu u kojima će se zadatak izvršavati. Kako bi se saznao broj tjedna potrebno je varijablu „rok“ koju sadrži objekt klase „Stroj“ pretvoriti u QDate oblik. QDate je klasa iz PyQt5 modula koja omogućuje rad sa datumima. Kako bi iz objekta tipa string dobili QDate objekt koristi se fromString() metoda QDate klase. FromString() metoda prima datum u string obliku i format u kojem je datum zapisan kako bi string datum pretvorila u QDate datum. Nadalje se dobiveni QDate datum koristi za dobivanje broja tjedna u kojem se određeni rok nalazi pomoću weekNumber() metode koja vraća polje u kojem se nalaze broj tjedna i godina tog datuma. Dan u tjednu kojem pripada određeni datum može se izračunati pomoću dayOfWeek() metode koja vraća vrijednosti od 1 do 7, gdje 1 predstavlja Ponedjeljak, a 7 Nedjelju.

Kako aplikacija ne bi ispisivala prazne tjedne, prilikom inicijalizacije grafičkog sučelja provjerava se broj tjedna za svaki zadatak te se sprema najveći i najmanji broj kako bi se znalo od kojeg do kojeg tjedna je potrebno napraviti raspored. Nakon određivanja najvećeg i najmanjeg broja tjedna poziva se metoda ubaci() (Slika 4.6. Metoda ubaci()) koja prima parametre kao što su stroj na kojem se izvodi zadatak, najmanji broj tjedna, rok do kojeg zadatak mora biti izvršen i redak koji označuje redak stroja u tjednoj tablici. Metoda ubaci() na temelju tih podataka pronalazi točan broj tjedna za određeni zadatak te poziva rekurzivnu metodu smjesti().

```
120     def ubaci(self, stroj, minimum, date, row):
121         for x in range(len(stroj)):
122             date = date.fromString(str(stroj[x].rok), "yyyy-MM-dd")
123             #date = date.addDays(-stroj[x].trajanje)
124             tmp1 = date.weekNumber()
125             tmp2 = tmp1[0]
126             tmp = date.dayOfWeek()
127             self.smjesti(0, row, stroj[x].trajanje, tmp2, minimum, tmp, stroj[x].naziv)
128         return
```

Slika 5.6. Metoda ubaci()

Rekurzivne metode su metode koje pozivaju same sebe. Sastoje se od osnovnog slučaja i napretka. Osnovni slučaj je slučaj u kojem rekurzivna metoda prestaje pozivati samu sebe jer su zadovoljeni uvjeti određeni od programera. Napredak je slučaj u kojem rekurzivna metoda ponovno poziva samu sebe pritom mijenjajući jedan ili više ulaznih parametara kako metoda ne bi zapela u beskonačnoj petlji. Dakle rekurzija ne može postojati bez prisutnosti i osnovnog slučaja i slučaja napretka.

Rekurzivna metoda smjesti() (Slika 4.7. Rekurzivna metoda smjesti()) kao parametre prima broj slobodnih dana, koji je pri prvom pozivu metode uvijek nula, redak stroja u tjednoj tablici, broj uzastopnih dana potrebnih za izvršavanje zadatka, broj tjedna u kojem se trenutno nalazi zadatak, najmanji broj tjedna za koji postoji zadatak, dan u tjednu u kojem se trenutno nalazi zadatak i naziv zadatka.

```
130     def smjesti(self, x, row, dana, tmp2, minimum, tmp, naziv):
131         if(x == dana and self.layout.itemAtPosition(row + (tmp2 - minimum) * 5,
132                                                     tmp).widget().text() == "/"):
133             for z in range(dana):
134                 if((tmp + z) > 7):
135                     tmp = 1 - z
136                     tmp2 = tmp2 + 1
137                 self.layout.itemAtPosition(row + (tmp2 - minimum) * 5,
138                                            tmp + z).widget().setText(str(naziv))
139 
140             return
141 
142         if(self.layout.itemAtPosition(row + (tmp2 - minimum) * 5, tmp).widget().text() == "/"):
143             if(tmp == 1):
144                 tmp = 8
145                 tmp2 = tmp2 - 1
146 
147             if(tmp2 < minimum):
148                 print("Zadatak "+str(naziv)+" ne stane!")
149                 return
150             return self.smjesti(x + 1, row, dana, tmp2, minimum, tmp - 1, naziv)
151         else:
152             if(tmp == 1):
153                 tmp = 8
154                 tmp2 -= 1
155 
156             if(tmp2 < minimum):
157                 print("Zadatak "+str(naziv)+" ne stane!")
158                 return
159             return self.smjesti(0, row, dana, tmp2, minimum, tmp - 1, naziv)
160 
```

Slika 5.7. Rekurzivna metoda smjesti()

Metoda smjesti() svakom iteracijom prolazi unazad kroz jednu ćeliju tablice, a kreće od ćelije koja predstavlja rok izvršenja zadatka. Metoda će napredovati u slučaju kada je ćelija slobodna odnosno sadrži tekst „/“, a u pozivu će povećati broj uzastopnih slobodnih dana i smanjiti dan u tjednu za 1. Ako ćelija nije slobodna, odnosno tekst ćelije je različit od „/“, metoda se ponovno poziva ali se broj uzastopnih dana vraća na 0, a dan u tjednu se smanjuje za 1. Prije svakog ponovnog poziva funkcije provjerava se koji je dan u tjednu i je li došlo do promjene tjedna. Također se provjerava ukoliko je broj tjedna manji od najmanjeg broja tjedna, pri čemu se metoda prekida. Do osnovnog slučaja doći će kada je broj uzastopnih slobodnih dana jednak broju dana potreban za izvršavanje zadatka te ukoliko je trenutna ćelija na kojoj se metoda nalazi slobodna, odnosno tekst ćelije je „/“. Kada dođe do osnovnog slučaja mijenja se tekst, u onoliko ćelija koliko je dana potrebno za izvršavanje, u naziv zadatka. Dakle metoda prolazi unazad ćelijama i kada ustanovi dovoljno uzastopnih slobodnih ćelija smjesti zadatak u te ćelije. Ta funkcionalnost uvedena je kako bi se izbjeglo preklapanje datuma i gubitak zadataka sa rasporeda.

5.5. Ručno unošenje zadataka

Pritiskom desne tipke miša na željeni gumb stvara se QInputDialog objekt. QInputDialog klasa u PyQt5 modulu koristi se za obradu i rad sa unosima korisnika. Metoda getText() stvara novi prozor sa prostorom za korisnika da upiše željeni tekst i gumbovima za predaju i otkazivanje akcije. Pritiskom na gumb „OK“ metoda vraća uneseni tekst u obliku string-a te taj tekst mijenja prijašnji tekst zadatka na odabranom gumbu. Osim getText() metode QInputDialog klasa još sadrži metode getDouble() koja od korisnika traži decimalni broj, getInt() koja zahtijeva cijeli broj, getItem() koja zahtijeva od korisnika da izabere znak iz string-a i getMultiLineText() gdje korisnik mora unijeti string od više redaka[12].

5.6. Spremanje stanja rasporeda

Za spremanje podataka koristi se gumb „Spremi“, koji kada se pritisne poziva metodu toCSV() (Slika 4.8. toCSV() metoda) koja otvara novu .csv datoteku i u nju sprema tekst svakog gumba u zaseban red. Ukoliko se tekst sastoji od više nego jednog znaka svaki znak je odvojen zarezom. Za čitanje i pisanje .csv datoteka u Pythonu koristi se csv knjižnica. Metoda writer() stvara objekt pisač i prima kao parametar datoteku u koju će se pisati. Objekt pisač koristi se writerows() metodom kojom zapisuje u datoteku parametar koji mu je proslijeđen. Također, potrebno je koristiti „encoding='utf-8'“ parametar pri otvaraju .csv datoteke kako bi se uspješno zapisali strani i inače nepoznati znakovi. Nakon završetka pisanja u datoteku potrebno je istu zatvoriti kako bi se oslobodila memorija.

```
104     def toCSV(self):
105         txt = []
106         today = QDate().currentDate()
107         file = open('Raspored_'+today.toString()+'.csv', 'w+', newline ='', encoding="utf-8")
108         for x in range(len(self.btns)):
109             txt.append(0)
110 
111         for x in range(len(self.btns)):
112             txt[x] = self.btns[x].text()
113 
114         with file:
115             write = csv.writer(file)
116             write.writerows(txt)
117 
118         file.close()
```

Slika 5.8. toCSV() metoda

5.7. Učitavanje spremljenog rasporeda

Pritiskom na dugme „Uvezi“ poziva se metoda uvoz() (Slika 4.9. uvoz() metoda) u kojoj se stvara QFileDialog objekt koji se koristi za potrebe biranja datoteke sa računala. QFileDialog je klasa u PyQt5 modulu koja je pogodna za korištenje u slučajevima gdje je potrebno da korisnik bira datoteku ili mapu sa računala. Pomoću getOpenFileName() metode iz QFileDialog klase dohvaća se naziv odabrane datoteke. Taj naziv koristi se kako bi se otvorila ta ista datoteka te bi se iz nje iščitali podatci zadataka. Čitanje se vrši preko objekta čitača koji ima suprotnu ulogu prije spomenutog objekta pisača. Kako su vrijednosti u .csv datoteci odvojene zarezom, vrijednosti tipa string razdvojene su na pojedinačne znakove koje je pri čitanju potrebno ponovno spojiti u jedan string. To se vrši iteracijom kroz datoteku te spajanje znakova iz istog reda[13].

```
89      def uvoz(self):
90          fileName = QFileDialog.getOpenFileName(self, "Odaberi postojeći raspored", "/",
91                                                 "CSV Files (*.csv)")
92          data = []
93          for x in range(len(self.btns)):
94              data.append(0)
95
96          with open(fileName[0], newline='', encoding='utf-8') as csvfile:
97              data = csv.reader(csvfile)
98              for row in data:
99                  new = ""
100                 for x in row:
101                     new += x
102                 self.btns[self.j].setText(new)
103                 self.j += 1
```

Slika 5.9. uvoz() metoda

5.8. Izvoz rasporeda u Excel tablicu

Funkcionalnost prebacivanja podataka iz grafičkog sučelja se kao i prijašnje realizira pritiskom na gumb pod nazivom „Izvezi u Excel“ pri čemu se pokreće metoda izvoz() (Slika 4.10. izvoz() metoda). Prvo se inicijalizira Workbook objekt iz kojeg se metodom „active“ stvara objekt aktivnog lista iz Excel datoteke. Nadalje se iterira po rešetkastom rasporedu i provjerava se ukoliko postoji predmet u određenoj ćeliji. Ako predmet postoji njegov sadržaj zapisuje se u Excel tablicu pomoću cell() metode. Dodatno se podešava širina stupaca kako bi stao sav tekst u njih radi lakšeg čitanja. U konačnici se datoteka sprema u radni direktorij sa provedenim izmjenama.

```
69      def izvoz(self):
70          wb = Workbook()
71          ws = wb.active
72          today = QDate.currentDate()
73
74          for x in range(self.layout.rowCount() - 1):
75              for y in range(self.layout.columnCount()):
76                  check = type(self.layout.itemAtPosition(0, 0))
77                  tp = type(self.layout.itemAtPosition(x, y))
78                  if(tp == check):
79                      continue
80                  else:
81                      txt = self.layout.itemAtPosition(x, y).widget().text()
82                      d = ws.cell(row=x + 1, column=y + 1, value=txt)
83
84          for i in range(8):
85              ws.column_dimensions[get_column_letter(i+1)].width = 25
86
87          wb.save("Raspored "+today.toString()+".xlsx")
```

Slika 5.10. izvoz() metoda

6. ZAKLJUČAK

Cilj ovog rada bio je izrada interaktivnog i automatskog generiranja rasporeda poslova i zadataka u tvrtki usmjerenoj na proizvodnju korištenjem strojne obrade materijala koji se temelji na podacima iz baze podataka, kao što su: rok izrade zadatka, trajanje izrade zadatka u danima i naziv zadatka. Baza podataka iz koje su se uzimali podatci dio je već postojećeg sustava na koji se ovaj rad nadovezuje. Sustav je razvijen do mjere da se unos podataka u bazu vrši automatski iz dokumenata o radnim nalozima i organizacije poslova. Podatci kojima se u ovom radu opisani sustav koristio bili su krucijalni za izradu istog, no baza podataka sadrži još mnogo više podataka vezanih za radne naloge i njihovo izvršavanje, iz čega se da zaključiti da se ovaj sustav može svakako još nadograđivati i prenamijeniti u razne svrhe, kao na primjer za izradu rasporeda djelatnika ili za automatizirano izvršavanje narudžbi ukoliko je nedovoljno materijala. Za izradu rada bilo je potrebno istražiti puno o temi *drag and drop* te granicama njenih mogućnosti i implementirati to stečeno znanje na način da se uklapa u dizajn i namjenu aplikacije. Također je bilo potrebno upoznati se sa radom sa Excel tablicama u programskom jeziku Python te kako to uvrstiti u aplikaciju kao korisnu značajku i bitnu funkcionalnost u praktičnoj primjeni aplikacije. U načelu rad se može podijeliti na 4 glavna dijela: izrada grafičkog korisničkog sučelja, implementacija *drag and drop* mehanizma u svrhu lakšeg i intuitivnijeg prijenosa podataka, dohvaćanje i korištenje informacija o zadatcima iz baze, te implementacija potrebnih funkcionalnosti za praktičan rad aplikacije. Svaki od tih dijelova predstavljao je izazov za sebe. Primjer jednog izazova bio bi izrada grafičkog sučelja gdje je bilo potrebno upoznati se sa PyQt5 alatom za izradu sučelja te ostvariti smisleni raspored grafičkih komponenti na sučelju. To je ostvareno uz pomoć QGridLayout klase za izradu tabličnog rasporeda komponenti. Python kao izbor programskog jezika pokazao se vrlo uspješnim i pogodnim za izradu ovakve vrste aplikacije. Tome uvelike pridonosi njegova proširivost u smislu silnih knjižnica koje se mogu uvesti i koje pružaju podršku za skoro sve potrebe. Također, Python je lako prenosiv te stoga i pogodan za rad na svim operativnim sustavima. Rad na ovom projektu ukazao je na neke od prednosti korištenih alata, kao što je velika podrška za *drag and drop* mehanizam u PyQt5 modulu. Ta prednost je omogućila veliku fleksibilnost pri izboru grafičkog elementa (u ovom slučaju izabran je gumb) za prikaz podataka u rasporedu. PyQt5 modul izvrstan je alat za izradu kompaktnog i funkcionalnog grafičkog korisničkog sučelja i pruža velik obim mogućnosti za izradu sučelja raznih namjena. Velik doprinos uspješnom dovršenju rada imale su i csv knjižnica, koja svojim metodama rješava probleme čitanja i pisanja .csv datoteka, te openpyxl knjižnica koja omogućuje komunikaciju sa Excel tablicama direktno iz Pythona. U ovom projektu upotrijebile su se praktične mogućnosti

navedenih alata i paketa za izradu desktop aplikacije praktične namjene koja služi za unaprjeđenje modela poslovanja, odnosno jednostavniju organizaciju poslovanja proizvodnog obrta.

7. LITERATURA

[1] Botella P. i dr.: „ISO/IEC 9126 in practice: what do we need to know?“, s Interneta, https://www.researchgate.net/publication/254891819_ISOIEC_9126_in_practice_what_do_we_need_to_know, Siječanj 2004.

[2] Fitzpatrick M.: „Create Simple GUI Applications with Python & Qt5“, 11. kolovoz 2019.

[3] „PyQt“, s Interneta, https://en.wikipedia.org/wiki/PyQt, 16. srpanj 2020.

[4] „XAMPP“, s Interneta, https://en.wikipedia.org/wiki/XAMPP, 20. kolovoz 2020.

[5] „phpMyAdmin“, s Interneta, https://en.wikipedia.org/wiki/PhpMyAdmin, 07. rujan 2020.

[6] „Comma-separated values“, s Interneta, https://en.wikipedia.org/wiki/Comma-separated_values, 29. kolovoz 2020.

[7] Gazoni E.; Clark C.: „openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files“, s Interneta, https://openpyxl.readthedocs.io/en/stable/, 21. kolovz 2020.

[8] „Python (programming language)“, s Interneta, https://en.wikipedia.org/wiki/Python_(programming_language), 10. rujan. 2020.

[9] „Anaconda“, s Interneta, https://docs.anaconda.com/anaconda/, 09. rujan 2020.

[10] „Anaconda (Python distribution), s Interneta, https://en.wikipedia.org/wiki/Anaconda_(Python_distribution), 03. kolovoz 2020.

[11] „Spyder (software)“, s Interneta, https://en.wikipedia.org/wiki/Spyder_(software), 08. rujan 2020.

[12] „QInputDialog Class“, s Interneta, https://doc.qt.io/qt-5/qinputdialog.html, 10. rujan 2020.

[13] „QFileDialog Class“, s Interneta, https://doc.qt.io/qt-5/qfiledialog.html, 10. rujan 2020.

8. POPIS OZNAKA I KRATICA

API	Application Programming Interface
GUI	Graphical User Interface
MIME	Multipurpose Internet Mail Extensions
XAMPP	XAMPP Apache + MariaDB + PHP + Perl
ER	Entity Relationship
CSV	Comma-Separated Values
XML	EXtensible Markup Language

9. SAŽETAK

Ovim radom stvorena je desktop aplikacija kao dio sustava za izradu interaktivnog rasporeda u programskom jeziku Python. U glavne značajke aplikacije ubrajaju se grafičko korisničko sučelje, *drag and drop* mehanizam , povezivanje sa bazom podataka te pisanje u Excel tablice direktno iz Pythona. U radu je opisan proces stvaranja aplikacije te neke od glavnih metoda i značajki potrebne za praktičan rad aplikacije. Aplikacija je u stanju generirati grafičko sučelje na temelju podataka o zadatcima iz baze te implementira mogućnosti izmjene podataka putem *drag and drop* mehanizma ili direktno dodavanje zadatka desnim klikom miša. Dodatne funkcionalnosti su uvedene u smislu mogućnosti spremanja stanja rasporeda i korištenja istog kao predložak u CSV formatu te izvoz podataka u Excel tablicu radi preglednosti i mogućnosti ispisivanja rasporeda na papir.

Ključne riječi:

Grafičko korisničko sučelje, *drag and drop*, povezivanje sa bazom podataka, CSV format, izvoz podataka u Excel tablicu.

SUMMARY

This paper describes the making of a desktop application as part of a sysem for creating interactive schedules in the Python programming language. The main features of this application are its graphical user interface, the drag and drop mechanism, connection to database and writing into Excel sheets directly from Python. This paper also describes the process of making an application and some of the main methods and features neccessary for its practical use. The application is capable of generating a graphical interface based on the information about the tasks from the database and implement the possibility of switching tasks via the drag and drop mechanism or by directly entering tasks by right-clicking on a slot. Additional funcionalities are added in a way of saving the current state of the schedule and using it later on as a template in the CSV format, aswell as exporting the data to an Excel spreadsheet for transparency and the possibility of printing the schedule.

10. DODATAK A

Ovom radu (digitalna verzija rada) priložen je dodatak A koji sadrži kod desktop aplikacije za izradu interaktivnog rasporeda.