**Gabaldarbinieku algas kopotājs**

1. **Problēmas izpēte un analīze.**

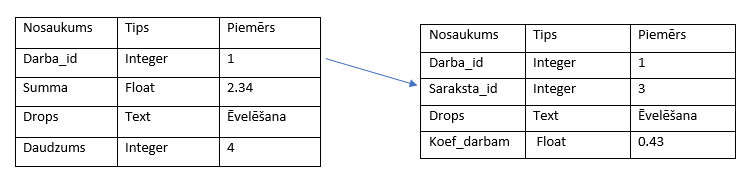
Izmantojot novērojumu izpētes metodi, strādājot vasarā kokapstrādes rūpnīcā veicu novērojumus, kā arī kontaktējoties ar kolēģiem novēroju, ka kokapstrādes rūpnīcu darbinieki, kuriem atalgojums par darbu tiek izmaksāts pēc padarītā darba daudzuma, līdz algas dienai grūti izsekot cik ir viņu nopelnītā summa, ja vien nekonsultējas ar grāmatvedi. Šādas konsultācijas aizņem lieku laiku, ko varētu izmantot produktīvāk gan ražotnes darbinieks, gan grāmatvedis. Protams, ka veicamās darbības, lai šo problēmu atrisinātu, ir vienādas un atkārtojas (aiziešana, apjautāšanās, rezultāta saņemšana un došanās prom).

1. **Programmatūras prasību specifikācija.**

Procesu var automatizēt, izveidojot programmu, kur darbinieki paši ievada apstrādāto materiālu daudzumu un, vadoties pēc iepriekš ievadīta koka apstrādes izcenojuma, ir iespējams apskatīt savu nopenīto bruto algu. Mērķauditorija ir jebkurš darbinieks, kura alga tiek rēķināta pēc padarītā darba, balstoties uz iepriekš aprakstīto problēmu, tikai pirms programmas lietošanas uzsākšanas ir vajadzība implementēt kodā darba tarifus un iespējamās veicamās darbības, piemēram, zāģēšana, ēvelēšana, saiņošana u.t.t.

Programmatūra iespējās ir izdarītā darba atlasīšana un daudzuma ievadīšana, darba algas summa, tāpēc tiks izveidota datubāze, lai varētu apstrādāt statistiku. Pieslēgšanās profilam notiek ar vienkāršu paroli. Dati tiek attēloti tabulā, kurus var sakārtot pēc jebkuras kolonas. Datus var izdzēst no datubāzes, kā arī mainīt summu. Darbu identificē ar ID numuru.

Programmatūra ir rakstīta python valodā, jo ir viegli saprotama un iespēja apstrādāt liela daudzuma datus. Lietotāja saskarne būvēta ar *PySimpleGUI* palīdzību, jo to ir viegli implementēt *python* kodā. Datu bāze nodaļa tiks veidota ar *sqlite3* bibliotēkas palīdzību. Paroles kodēšanu nodrošina *hashib* bibliotēka, bet sakārtošanu *operator* bibliotēka. Kods realizēts *Visual Studio Code* platformā, jo tā piedāvā pajomīgu funkciju daudzumu un paplašinājumus, kas palīdz programmatūras izstrāde un šo saturu piedāvā bezmaksas.



1. attēls datubāzes shēma

1. **Programmatūras izstrādes plāns**

Programmatūras izstrādē tiks izmantots Iteratīvais modelis. Pēc veiksmīgas problēmas izpētes, prasību specificēšanas un projektēšanas, var veidot nepārtrauktu ciklu ar izstrādi, testēšanu un izvērtēšanu. No sākuma tika atrasta problēma, uz kuras varētu balstīt programmu. Vēlāk izveidoja programmas detalizētāku aprakstu un viss tika dokumentēts sīkāk pa vajadzīgajām nodaļām programmas specifikācijas dokumentā. Uzsākot nebeidzama cikla darbību par turpmāko programmas veidošanu, no sākuma izstrādā konkurētu koda daļu, kas atbilstu par kādu no funkcijām (piemēram, datu ievade, tabulas izveide, datu rediģēšana u.t.t.), tālāk tiek veikta funkcijas testēšana paredzot visus iespējamos scenārijus, veikt visas darbības un pēcāk jau izvērtēt kļūdas un to iespējamās novēršanas. Kad šāds viens cikls ir veiksmīgi pabeigts, atkal sāk no sākuma ar koda izstrādi jau jaunai funkcijai.

1. **Atkļūdošanas un akcepttestēšanas pārskats.**

Datu ievades laukos, kur vajag ievadīt skaitļus, ja ievada nenaturālu skaitli vai burtu, pēc pogas “Ievadīt” nospiešanas parādās logs, ka ievads nav pareizi ievadīts. Ar pogas “Ievadīt” palīdzību pārbauda vai nav kāds no ievades laukiem atstāts tukšs, pretējā gadījumā izlec arī atbilstošs logs. Programma pārtrauc savu darbību, ja ievadītais ID jau ir sastopams datubāzē vai summas ievades logā ir ievadīts negatīvs skaitlis, burts vai citādi nepareizi ievadīts decimāldaļas skaitlis. Datubāzes rediģēšanas logos, kur prasa ievadīt ID, var ievadīt jebkādus ieejas datus, bet funkcija nostrādās tikai tad, ja būs ievadīts numurs, kas atbilst kādam ID datubāzē.

Testēšana autorizācijas logam:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testa numurs | Ieejas dati | Izejas dati | Iegūtais rezultāts | Piezīmes |
| 1 | ‘janis’, ‘kubki321’ | nav ievadīta pareiza parole vai lietotājvārds | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |
| 2 | ‘jnais’, ‘kubik123’ | nav ievadīta pareiza parole vai lietotājvārds | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |
| 3 | ‘janis’, ‘kubik123’ | Parādās datu ievades logs | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |

Testēšana darba numura, daudzuma ievades logam:

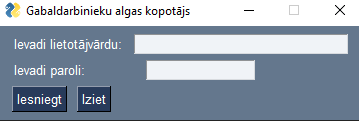
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testa numurs | Ieejas dati | Izejas dati | Iegūtais rezultāts | Piezīmes |
| 1 | -123 | ievads nav aizpildīts pareizi! | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |
| 2 | “” | ievadā ir tukšs lauks | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |
| 3 | 2.34 | ievads nav aizpildīts pareizi! | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |

Testēšana summas un maiņas summas logam:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testa numurs | Ieejas dati | Izejas dati | Iegūtais rezultāts | Piezīmes |
| 1 | -123 | ValueError: could not convert string to float: '-' | Atbilst | Programma pārtrauc savu darbību un konsolē izvada kļūdas paziņojumu |
| 2 | “” | ievadā ir tukšs lauks | Atbilst | Uzlecošais logs. Nospiežot pogu “Ok”, var vēlreiz mēģināt ievadīt datus |
| 3 | abc | ValueError: could not convert string to float: 'a' | Atbilst | Programma pārtrauc savu darbību un konsolē izvada kļūdas paziņojumu |
| 4 | 0.3.5 | ValueError: could not convert string to float: '0.3.' | Atbilst | Programma pārtrauc savu darbību un konsolē izvada kļūdas paziņojumu |

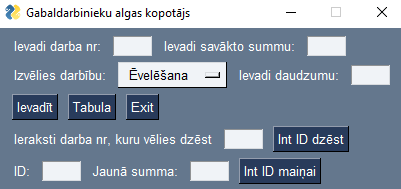
1. **Lietotāja ceļvedis.**

Programma tiek palaista, kad palaiž failu *main.py.* No sākuma parādās autorizēšanās logs, kur jāievada lietotājvārds (pieejamie lietotājvārdi ir “janis”, “kokle”, “kubiks”) un parole (vienīgā derīgā parole ir “kubik123”). Ja lietotājvārds un/vai parole ir nederīgi, tad parādās logs ar attiecīgu paziņojumu, bet pretējā gadījumā pieslēgšanās logs pazūd un var uzsākt darbu ar datiem. Poga “Iesniegt” aktivizē funkciju, kas pārbauda vai lietotājvārds un parole ir derīga, bet poga “Iziet” pārtrauc programmas darbību.



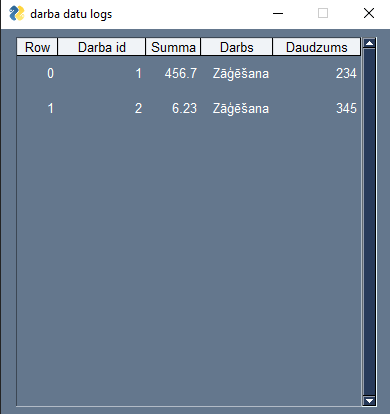
2. attēls autorizēšanās logs

Ievades loga pirmā daļa ir paredzēta datu ievadei (jāaizpilda logi ar darba numuru, summu, darba tipu un daudzumu). Pēc veiksmīgas logu aizpildīšanas, ko pārbauda atkļūdošana, kas ir aprakstīta šī dokumenta 4. punktā, nospiežot pogu “Ievadīt”, parādās logs par veiksmīgu datu ievadi. Poga “Exit” pārtrauc programmas darbību. Nākama loga rinda, gadījumā, ja ir vēlme izdzēst datus, tad, ievadot konkurētu darba numuru, tiek no datubāzes izdzēsts darba numurs ar tā attiecīgajiem datiem. Loga pēdējā rinda spēj samainīt summu ierakstot konkurētu darba numuru un summu uz ko lietotājs vēlas mainīt. Šīs abas darbības tiek izpildītas nospiežot attiecīgo pogu rindas beigās un datu validāciju arī pārbauda pēc līdzīga principa kā tas ir aprakstīts šī dokumenta 4. punktā.



3. attēls datu ievades un rediģēšanas logs

Nospiežot pogu “Tabula”, parādās logs ar izveidotu tabulu, kur ir attēloti visi datubāzē pieejamie dati. Virs katras kolonas ir to nosaukumi, pēc kuriem arī ir iespēja kārtot datus augošā secībā. Ja tiek veiktas datubāzes datu rediģēšanas darbības, kamēr tabula ir atvērta, tad, lai atjaunotu tabulas datus, tā ir jāaizver un jāatver vēlreiz.



4. attēls tabulas logs

1. **Piemērotās licences pamatojums.**

MIT licence ir pieļaujama licence, kas ļauj atkārtoti izmantot un modificēt programmatūru bez daudziem ierobežojumiem, padarot to par labu izvēli izglītības vai pētniecības nolūkos. MIT licence ir atļauta atvērtā pirmkoda programmatūras licence. Izmantojot MIT licenci datu bāzes projektam, citiem var būt vieglāk izmantot un sniegt ieguldījumu, piemēram, ja ir vajadzība pēc kāda laika, mainīt kādu funkciju, līdz ar to var veiksmīgi rediģēt kodu.

1. **Programmatūras kods**

**Faila *main.py* kods:**

# importē bibliotēkas un failus

import PySimpleGUI as sg

import datu\_inters, paroles\_logs, tabula

import sqlite3

conn = sqlite3.connect('data.db')

cur = conn.cursor()

# izveiodo tabulas datubāzē, ja tādas jau neeksistē

cur.execute("""CREATE TABLE IF NOT EXISTS dati(

darba\_id INTEGER PRIMARY KEY,

summa float,

drops text,

daudzums integer

)""")

cur.execute("""CREATE TABLE IF NOT EXISTS darba\_koef(

id INTEGER PRIMARY KEY,

saraksta\_id INTEGER,

drops text,

koef\_darbam float,

FOREIGN KEY (saraksta\_id) REFERENCES users(darba\_id)

)""")

# izkārtojums logam

layout = [

[sg.Text("Ievadi darba nr:"), sg.I(key = '-ID-',enable\_events=True,size=(5, 1)),sg.Text("Ievadi savākto summu:"), sg.I(key = '-SUMMA-',enable\_events=True,size=(5, 1))],

[sg.Text("Izvēlies darbību:"),sg.OptionMenu(values = ['Ēvelēšana', 'Garināšana', 'Zāģēšana'],default\_value = 'Ēvelēšana', key = "-DROPS-"),sg.Text("Ievadi daudzumu:"),sg.I(key = '-DAUDZUMS-', size = (5,1))],

[sg.B('Ievadīt'),sg.B('Tabula'),sg.Exit()],

[sg.T('Ieraksti darba nr, kuru vēlies dzēst'),sg.I(key = '-DZEST\_ID-',enable\_events=True,size=(5, 1)),sg.B('Int ID dzēst')],

[sg.Text("ID:"),sg.I(key = '-MAINIT\_ID-',enable\_events=True,size=(5, 1)),sg.Text("Jaunā summa:"),sg.I(key = '-UZ\_KO-',enable\_events=True,size=(5, 1)),sg.B('Int ID maiņai')]

]

window = sg.Window("Gabaldarbinieku algas kopotājs", layout)

vajag\_autorizeties=0

# cikls loga nepārtrauktas darbības nodrošināšanai

while True:

# pārbauda autorizāciju

if vajag\_autorizeties ==0:

paroles\_logs.protect()

vajag\_autorizeties=1

event, values = window.read()

# iziet no programmas

if event == sg.WIN\_CLOSED or event == 'Exit':

break

# ievada datus un pārbauda pareizos datu tipus ievades logiem

if event == 'Ievadīt':

if values['-DAUDZUMS-'] == "" or values['-ID-'] == "" or values['-SUMMA-'] == "":

sg.popup("ievadā ir tukšs lauks")

if values['-ID-'].isdigit() == False or values['-DAUDZUMS-'].isdigit() == False:

sg.popup("ievads nav aizpildīts pareizi!")

else:

datu\_inters.ievieto\_darbu(values['-ID-'],values['-SUMMA-'],values['-DROPS-'],values['-DAUDZUMS-'])

sg.popup("darbs ievietots!")

# nodrošina summas pareizo datus ievades tipu

if event == '-SUMMA-' and values['-SUMMA-']:

in\_as\_float = float(values['-SUMMA-'])

if event == '-UZ\_KO-' and values['-UZ\_KO-']:

in\_as\_float = float(values['-UZ\_KO-'])

# izveido tabulu

if event == 'Tabula':

tabula.create()

# izdzēš datus pēc darba ID

if event == 'Int ID dzēst':

if values['-DZEST\_ID-'] == "":

sg.popup("ievadā ir tukšs lauks")

else:

dzest\_id=values['-DZEST\_ID-']

datu\_inters.izdzest\_darbu(dzest\_id)

# maina summu konkurētam darba ID

if event == 'Int ID maiņai':

if values['-MAINIT\_ID-'] == "" or values['-UZ\_KO-']=="":

sg.popup("ievadā ir tukšs lauks")

else:

mainit\_id=values['-MAINIT\_ID-']

uz\_ko=values['-UZ\_KO-']

datu\_inters.izmaina\_summu(mainit\_id,uz\_ko)

# aizver logu un beidz datubāzes procesus

conn.commit()

conn.close()

window.close()

**Faila *tabula.py* kods:**

# importē bibliotēkas un failus

import PySimpleGUI as sg

import datu\_inters

import operator

# iegūst datus no datubāzes

def dabu\_datus():

darba\_dati = datu\_inters.parbaude()

return darba\_dati

# sakārto tabulas datus pēc noteiktas kolonas

def sort\_table(table, col\_clicked):

table = sorted(table, key=operator.itemgetter(col\_clicked))

return table

# izveido tabulas logu

def create():

darbu\_masivs = dabu\_datus()

headings = ['Darba id', 'Summa', 'Darbs', 'Daudzums']

darba\_datu\_layout = [

[sg.Table(values=darbu\_masivs, headings=headings, max\_col\_width=35,

auto\_size\_columns=True,

display\_row\_numbers=True,

justification='right',

num\_rows=10,

enable\_events=True,

enable\_click\_events=True,

key='-TABULA-',

row\_height=35,

tooltip='darbu tabula'

)

]

]

tabula = sg.Window("darba datu logs",

darba\_datu\_layout, modal=True)

# cikls loga nepārtrauktas darbības nodrošināšanai

while True:

event, values = tabula.read()

# beidz programmas darbību

if event == "Exit" or event == sg.WIN\_CLOSED:

break

# nolasa kura kolona ir noklikšķināta, lai varētu sakārtot datus

if event[2][0] == -1 and event[2][1] != -1:

col\_num\_clicked = event[2][1]

new\_table = sort\_table(darbu\_masivs, col\_num\_clicked)

tabula['-TABULA-'].update(new\_table)

# aiver tabulas logu

tabula.close()

Faila *datu\_inters.py* kods:

# importē bibliotēku

import sqlite3

# ievieto datus datubāzē

def ievieto\_darbu(darba\_id,summa,drops,daudzums):

conn = sqlite3.connect('data.db')

conn.execute("""INSERT INTO dati (darba\_id,summa,drops,daudzums) VALUES(?,?,?,?);""",(darba\_id, summa,drops,daudzums))

conn.commit()

conn.close()

# ievāc datus un datubāzes un ieliek tos listā

def parbaude():

results = []

conn = sqlite3.connect('data.db')

cursor = conn.execute("SELECT \* from dati")

for row in cursor:

results.append(list(row))

return results

# izdzēš datus pēc darba ID

def izdzest\_darbu(dzest\_id):

conn = sqlite3.connect('data.db')

conn.execute("DELETE FROM dati WHERE darba\_id = ?", (dzest\_id))

conn.commit()

conn.close()

# maina summu konkurētam darba ID

def izmaina\_summu(mainit\_id,uz\_ko):

conn = sqlite3.connect('data.db')

conn.execute("UPDATE dati set summa = ? where darba\_id = ?", (uz\_ko,mainit\_id))

conn.commit()

conn.close()

Faila *paroles\_logs.py* kods:

# importē bibliotēkas

import PySimpleGUI as sg

import hashlib

def protect():

# izkārtojums logam

layout = [

[sg.Text("Ievadi lietotājvārdu:"), sg.Input(key='-EMAIL-', do\_not\_clear=True, size=(30, 1))],

[sg.Text('Ievadi paroli:', size=(15, 1)), sg.InputText('', key='-PASSWORD-', password\_char='\*', size=(15, 1))],

[sg.Button("Iesniegt"),sg.Button("Iziet")]

]

password\_window = sg.Window('Gabaldarbinieku algas kopotājs', layout, modal=True)

# pārbauda paroli

def verify\_password(password):

hash = '8b15670ea8fb9647e783263380bbbadf0cfa543a11dd7140fecf34cc4b710266'

password\_utf = password.encode('utf-8')

password\_hash = hashlib.sha256(password\_utf).hexdigest()

if hash == password\_hash:

return True

return False

# pārbauda e-pasta adresi

def verify\_email\_address(email\_address):

user\_email\_addresses = ['janis', 'kokle', 'kubiks']

if email\_address in user\_email\_addresses:

return True

return False

# cikls loga nepārtrauktas darbības nodrošināšanai

while True:

event, values = password\_window.read()

# beidz programmas darbību

if event == "Iziet" or event == sg.WIN\_CLOSED:

exit()

# pārbauda iesniegtos autorizācijas datus

if event == 'Iesniegt':

email\_input\_value = values['-EMAIL-']

password\_input\_value = values['-PASSWORD-']

if verify\_password(password\_input\_value) and verify\_email\_address(email\_input\_value):

break

else:

sg.popup("nav ievadīta pareiza parole vai lietotājvārds")

continue

# aizver autorizācijas logu

password\_window.close()