

Rīgas 64. vidusskola

Koda fragmentu saglabāšanas un meklēšanas rīks

Pielaižu darbs programmēšanā

Darba autori: Sandis Dobelis, 12 DIT

Rīga, 2023

Saturs

1.	Ievads.....	3
1.1.	Nolūks.....	3
1.2.	Darbības sfēra	3
1.3.	Definīcijas.....	3
1.4.	Saistība ar citiem dokumentiem.....	3
1.5.	Pārskats.....	3
2.	Problēmas izpēte un analīze.....	4
2.1.	Problēmas apraksts	4
2.2.	Automatizācijas risinājumi.....	4
2.3.	Funkcionalitāte.....	4
2.4.	Vispārējie ierobežojumi	5
3.	Programmatūras prasību specifikācija	6
3.1.	Datu plūsma	6
3.2.	Mērķauditorija	6
3.3.	Programmatūras produkta apraksts	6
3.4.	Programmatūras produkta funkciju apraksts.....	7
4.	Programmatūras izstrādes plāns.....	9
4.1.	Metodes izvēle	9
4.2.	Izstrādes plāns.....	9
5.	Akcepttestēšanas pārskats	10
5.1	Testēšanas mērķis un izpilde.....	10
5.2	Akcepttestēšanas rezultāti	12
6.	Lietotāja ceļvedis	13
7.	Piemērotās licences pamatojums	14
7.1	Produkta licence.....	14
	Pielikumi.....	15

1. Ievads

1.1. Nolūks

Darbs ir veidots, kā Programmēšanas II kursa eksāmena pielāides darbs. Darbā tiek aprakstīta programma Koda Fragmentu Saglabāšanas un Meklēšanas Rīks, turpmāk KFSMR, kura spēj saglabāt un meklēt tekstu, kodu vai koda fragmentus, kā arī atrast koda atbilstošās repozitorijus no vietnes Github.

1.2. Darbības sfēra

Programma ir paredzēta lietotājiem, kuriem bieži nepieciešams saglabāt kodu vai fragmentus no tiem, programma atvieglo procesu, saglabājot vēlamu kodu un tekstu vienā vietā, kā arī atrast koda krātuves pēc nosaukumiem.

1.3. Definīcijas

Akronīms	Skaidrojums
GUI	Grafiskā saskarne (No angļu valodas “Graphical User Interface”)
API	Lietojumprogrammu saskarne (No angļu valodas “Application Program Interface”)
Repozitorijs	Datu glabātuve

1.4. Saistība ar citiem dokumentiem

Dokumenta PPS noformēšanā ievērotas standarta LVS 68:1996 prasības.

1.5. Pārskats

Pirmajā daļā ir apraksts par dokumenta nolūku, darbības sfēru, definīciju skaidrojumu, saistību ar citiem dokumentiem.

Otrajā daļā aprakstītas programmatūras problēmas izpēte un risinājumi.

Trešajā daļā aprakstītas programmatūras prasības un specifikācijas.

Ceturtajā daļā aprakstīts programmatūras izstrādes plāns.

Piektajā daļā aprakstīts programmatūras akcepttestēšanas pārskats un rezultāti.

Sestajā daļā aprakstīts lietotāja ceļvedis.

Septītajā daļā atrodas piemērotā licence un pamatojums.

2. Problēmas izpēte un analīze

2.1. Problēmas apraksts

Programmēšana ir ietekmīga joma, kas veido lielu daļu no mums apkārt esošajām tehnoloģijām, un, ņemot vērā arvien pieaugošo gan iesācēju, gan pieredzējušo programmētāju skaitu, tā pašlaik ir viena no izplatītākajām un ietekmīgākajām jomām. Programmēšanas laikā lietošanas ērtums un pieejamība var būtiski ietekmēt koda rakstīšanas ātrumu, kvalitāti un lasāmību. Liela daļa no pieejamības ir piekļuve centralizētām koda daļām un koda bibliotēkām, un bez tām kodēšanas laikā var tikt veltīgi tērēts laiks un pūles, kas var novest pie liekiem un apgrūtināšiem problēmu risinājumiem, atturēt programmētājus no jaunu koda valodu un bibliotēku izmēģināšanas.

2.2. Automatizācijas risinājumi

Viens risinājums šim būtu programmas vai papildinājumi, kuri papildina un iesaka kodu, ietaupot laiku un atvieglinot koda rakstīšanu. Vēl viens risinājums ir mašīnmācību un dabiskās valodas apstrādes programmu izveide, automātiski izstrādājot kodu atkarībā no vēlamā funkcionalitātes apraksta, līdzās šāda tipa programmai būtu skriptu rīki, lai automatizējot, piemēram, testēšanu un atklādošanu, ļaujot veltīt vairāk laika programmēšanai.

Šī darba izstrādei tika izvēlēta koda fragmentu un bibliotēku repozitoriju programmas izveide. Viegla piekļuve bieži izmantotam kodam sekmēs laika pavadīšanu noderīgi rakstot jaunu kodu, nevis atkārtoti pārrakstot vai meklējot jau uzrakstītu kodu.

2.3. Funkcionalitāte

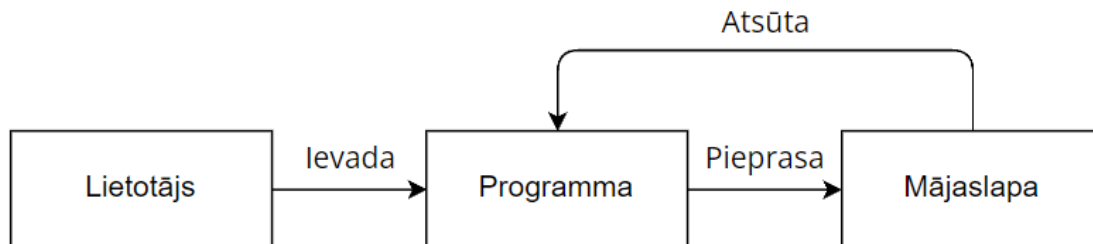
1. Teksta/koda ievade
2. Teksta/koda saglabāšana
3. Koda meklēšana

2.4. Vispārējie ierobežojumi

- Nepieciešama vismaz minimāla saprašana kodēšanā, lai pielāgotu programmu individuālām vajadzībām.
- Koda meklēšanas daļas darbība ar API balstās uz interneta savienojuma un API pieejamības.
- Repozitoriju saraksts ne vienmēr atbilst koda fragmentam.

3. Programmatūras prasību specifikācija

3.1. Datu plūsma



attēls 3.1 Datu plūsma

3.2. Mērķauditorija

Mērķauditorija ir jebkurš, kurš nodarbojas ar programmēšanu un vēlas rīku, ar kuru vienā vietā ir pieejams glabāt visbiežāk izmantoto kodu fragmentus, lai paātrinātu kodēšanu un samazinātu lieki tērēto laiku pārrakstot koda daļas.

3.3. Programmatūras produkta apraksts

KFSMR ir GUI programma, kurā ievada tekstu vai kodu, to nosaukumu, kas tad tiek saglabāts lokāli izveidotā datubāzē, vienlaikus arī tiek meklētas kodu repozitoriji no Github vietnes, kuri var atbilst ievadītā koda tematikai. Lietotājs varēs redzēt saglabātos kodus sarakstā, no kura var jebkurā brīdī nokopēt uzspiežot uz saraksta elementiem. Zem fragmentu saraksta arī būs repozitoriju saraksts, kurā uzspiežot uz saraksta elementiem var iegūt repozitoriju nosaukumus vai saites uz tiem.

Programmas izveidei nepieciešams dators, lai kodētu. Programmatūras valodai tiks izvēlēts Python, tā pieejamības un vieglās saprotamības dēļ. Tiks izmantotas requests, tkinter, sqlite3 Python bibliotēkas. Koda meklēšanai tiks izmantots Github REST API.

3.4. Programmatūras produkta funkciju apraksts

SnippetDatabase datubāzes izveides funkcijas apraksts

Mērķis:
Izveidot lokālu datubāzi, kur glabāt datus un tabulas.
Ievaddati:
Kodā ievadīti tabulu parametri.
Apstrāde:
Sqlite3 izveido datubāzi ar tabulām, ja tās vēl neeksistē.
Izvaddati:
Datubāze un tabulas.

Tabula 3.1

SnippetGUI grafiskās saskarnes funkcijas apraksts

Mērķis:
Tkinter grafiska saskarne ar pogām, kuras izpilda datu pievienošanas/kopēšanu.
Ievaddati:
Kodā ievadītie Tkinter GUI logrīki, ar pogu, teksta logu, un saskarnes parametriem un lielumiem.
Apstrāde:
Tkinter izveido GUI pēc parametriem, SnippetDatabase ievieto datus tabulās.
Izvaddati:
GUI saskarne, tabulu elementi, repozitoriju elementi.

Tabula 3.2

Search_repositories funkcijas apraksts

Mērķis:
Atrast repozitorijus Github.
Ievaddati:
Repozitoriju nosaukums; Kodā nepieciešamo repozitoriju skaits.
Apstrāde:

Github REST API apstrādā ievadīto nosaukumu un atgriež repozitoriju nosaukumus un saites.
Izvaddati:
4 repozitoriju nosaukumi un saites.

Tabula 3.3

Copy_selected_snippet funkcijas apraksts

Mērķis:
Nokopēt uzspiesto kodu tabulas elementu.
Ievaddati:
Kodu tabulas elements.
Apstrāde:
GUI saskarne attēlo sarakstu ar elementiem, kuri tiek glabāti datubāzē, nospiežot elements tiek kopēts un pievienots starpliktuvei (no angļu clipboard).
Izvaddati:
Izvēlēta elementa kods.

Tabula 3.4

Copy_repository_link funkcijas apraksts

Mērķis:
Nokopēt repozitorija nosaukumu vai saiti uz to.
Ievaddati:
Repozitoriju elementu saraksts.
Apstrāde:
Github REST API pievienotās saites attēlo uz GUI saraksta, nospiežot tiek nokopēts un pievienots starpliktuvē repozitorija nosaukums vai saite uz to, atkarībā uz kuru saraksta elementa nospieda.
Izvaddati:
Repozitorija nosaukums vai saite.

Tabula 3.5

4. Programmatūras izstrādes plāns

4.1. Metodes izvēle

Tika izvēlēta spirālveida metode, jo KFSMR programma ir veidota, lai to varētu katrs lietotājs pielāgot savām vēlmēm, tāpēc ir nepieciešams, lai izstrādes procesā varētu aiziet uz jebkuru no iepriekšējiem izstrādes soļiem, padarot testēšanu, uzlabošanu un programmatūras maiņu vieglāku un rezultatīvāku.

4.2. Izstrādes plāns

Izstrādes sākumā tika noskaidrotas pasūtītāju un lietotāju prasības programmatūras produktam. Svarīgākās no tām ir produkta funkcijas, nepieciešamības, ierobežojumi, kā arī programmatūras apraksti.

Projekta veidošanas solī tika izveidota programmatūras modeli un plānu atbilstoši pasūtītāja prasībām. Svarīgākais bija ieviest pamata funkcionalitātes, ar uzsvaru ieviest pamata GUI saskarni.

Programmēšanas solī tika izveidota sākotnēja pamata programma pēc iepriekšveidotajām prasībām, modeļa. Programmējot sadalot funkcijas un metodes saprotamos koda blokus, vieglākai turpmākai uzlabošanai un testēšanai.

Testēšanas solī, tika pārbaudītas programmatūras funkcijas, to ierobežojumi, kā arī programmatūras kļūdu paziņojumu efektivitāti, tādējādi pārbaudot funkciju atbilstību prasībām.

Dokumentēšanas solī tika dokumentētas prasības, funkcijas, tā izstrādi un nepieciešamības, lai jebkurš lietotājs varētu pārzināt, kā izmantot, testēt vai modificēt KFSMR programmatūru.

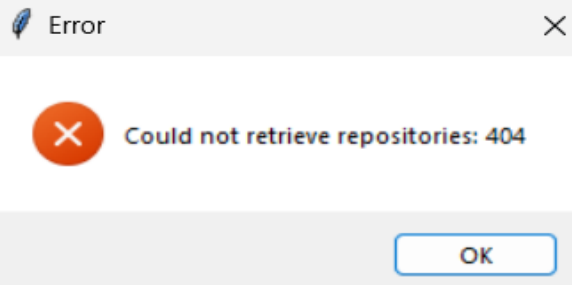
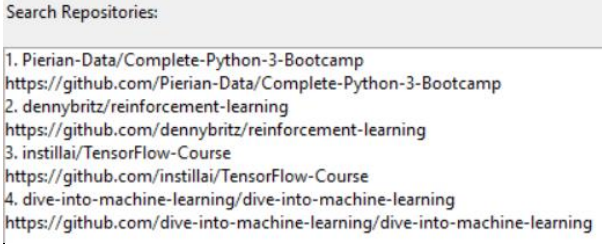
Izplatīšanas solis, sekmīgi ieviest programmu tīmeklī vai mājaslapā, lai programmatūru varētu ekspluatēt lietotāji.

5. Akcepttestēšanas pārskats

5.1 Testēšanas mērķis un izpilde

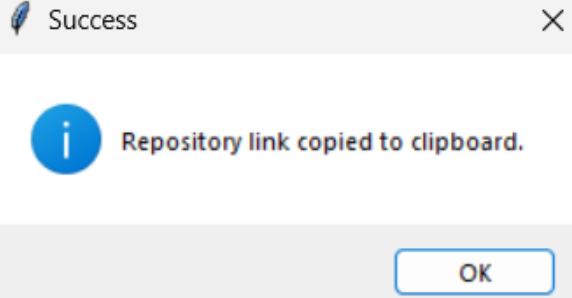
Testēšanā tiks pārbaudīti programmatūras kļūdu paziņojumi vai korekti izvaddati.

Repozitoriju meklēšanas funkcijas testēšana

Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts	Rezultāta atspoguļojums
Nav repozitorijs	Programma uzrāda kļūdas uznirstošu logu ar tekstu “Could not retrieve repositories: 404”	Izpildās	
Ir repozitorijs	Programma uzrāda repozitorijus tabulā	Izpildās	

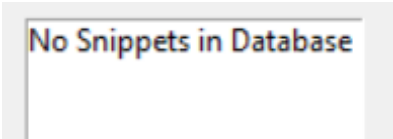
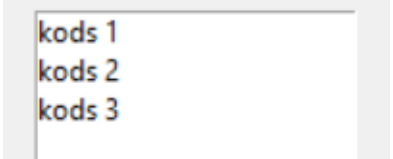
Tabula 5.1

Copy_repository funkcijas testēšana

Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts	Rezultāta atspoguļojums
Repozitoriju tabulas Elements	Programma uzrāda uznirstošu logu ar tekstu “Repository link copied to clipboard”	Izpildās	

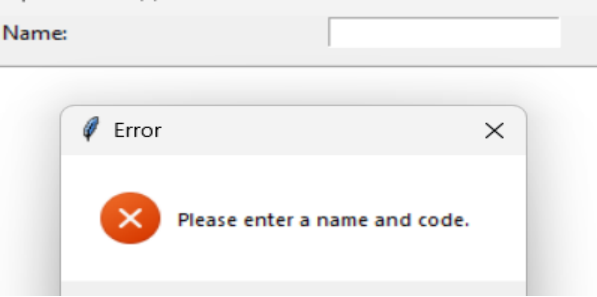
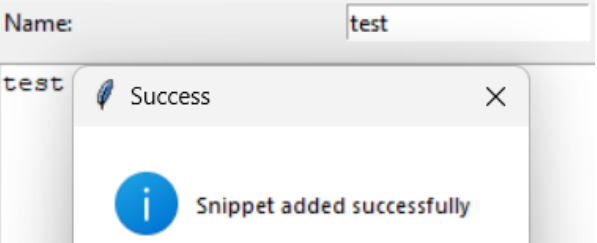
Tabula 5.2

Populate_snippets_listbox funkcijas testēšana

Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts	Rezultāta atspoguļojums
Tukša koda elementu tabula	Tabulā teksts “No snippets in Database”	Izpildās	
Koda elementi tabulā	Tabulas saraksts	Izpildās	

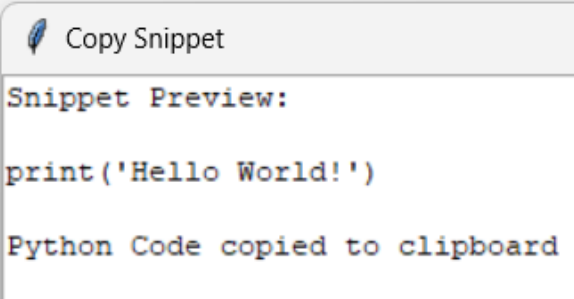
Tabula 5.3

add_snippet funkcijas testēšana

Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts	Rezultāta atspoguļojums
Nav ievadīts kods un/vai nosaukums	Programma uzrāda uznirstošu logu ar tekstu “Please enter a name and code”	Izpildās	
Ievadīts kods un nosaukums	Programma uzrāda uznirstošu logu ar tekstu “Snippet added successfully”	Izpildās	

Tabula 5.4

Get_code_from_database funkcijas testēšana

Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts	Rezultāta atspoguļojums
Koda tabulas elements	Programma uzrāda uznirstošu logu ar “Snippet Preview: {code} {name} copied to clipboard”	Izpildās	

Tabula 5.5

5.2 Akcepttestēšanas rezultāti

Visi kļūdas paziņojumi nostrādāja un programmatūras pamata funkcijas strādā.

6. Lietotāja ceļvedis

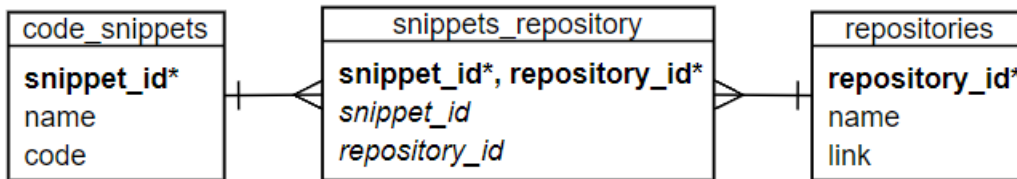
- Ievadiet koda/teksta nosaukumu lodziņā pa labi no “name:”
- Ievadiet kodu/tekstu teksta logā zem “name:”
- Nospiediet “Add snippet”, kods un tā nosaukums tika pievienots sarakstā pa labi no teksta loga. Kā arī automātiski koda nosaukums tika ievadīts “Search:” lodziņā, kas uzrāda apakšējā sarakstā četrus augstāk vērtētos repozitorijus, ar līdzīgu nosaukumu ievadītajam.
- Uzrādīto repozitoriju skaits maināms kodā zem funkcijas `search_repositories` - `for i, result in enumerate(results[:X])`, X vietā ievadot skaitli (Pamata vērtība: 4).
- Iespējams arī izmantot manuāli meklēšanas logu, ierakstot vēlamo terminu logā pa labi no “Search:”
- Nospiežot uz koda nosaukuma, Repozitorija nosaukuma vai Repozitorija saites tā tiks nokopēta un pievienota starpliktuvei.

7. Piemērotās licences pamatojums

7.1 Produkta licence

Programma KFSMR, tiktu licencēta ar GNU GPL licenci jeb GNU Vispārējo publisko licenci, jo KFSMR pamatā ir lietotāja spējā modificēt, personalizēt programmu pēc savām vēlmēm, vajadzībām, KFSMR pamatā arī tiek izmantotas programmatūras bibliotēkas ar “GNU Lesser General Public” licenci. KFSMR nav veidota, kā komerciāla, bet gan kā lietotāja draudzīga programmatūra.

Pielikumi



Pielikums 1. Relāciju tabula

```
1 import sqlite3
2 from tkinter import messagebox, Toplevel, Text, Scrollbar
3 import tkinter as tk
4 import requests
5
6 class SnippetDatabase:
7     def __init__(self):
8         self.conn = sqlite3.connect("Snippets.db")
9         self.c = self.conn.cursor()
10
11         self.create_table()
12
13     def create_table(self):
14         self.c.execute("""CREATE TABLE IF NOT EXISTS code_snippets (
15             snippet_id INTEGER PRIMARY KEY AUTOINCREMENT,
16             name TEXT,
17             code TEXT
18         )""")
19
20         self.c.execute('''CREATE TABLE IF NOT EXISTS repositories
21             (repository_id INTEGER PRIMARY KEY,
22             name TEXT,
23             link TEXT)''')
24
25         self.c.execute('''CREATE TABLE IF NOT EXISTS snippets_repository (
26             snippet_id INTEGER,
27             repository_id INTEGER,
28             FOREIGN KEY (snippet_id) REFERENCES code_snippets (id),
29             FOREIGN KEY (repository_id) REFERENCES repositories (id),
30             PRIMARY KEY (snippet_id, repository_id)
31         )''')
32
33         self.conn.commit()
34
35     def add_snippet(self, name, code):
36         self.c.execute("INSERT INTO code_snippets (name, code) VALUES (?, ?)", (name, code))
37         self.conn.commit()
38
39     def view_snippets(self):
40         self.c.execute("SELECT name, code FROM code_snippets")
41         snippets = self.c.fetchall()
42
43         return snippets
44
45     def __del__(self):
46         self.conn.close()
47
48 class SnippetGUI:
49     def __init__(self):
50         self.database = SnippetDatabase()
51
52         self.root = tk.Tk()
53         self.root.title("Code Snippets")
54
55         self.create_gui()
56
57
```

```

58 #Create the GUI
59 def create_gui(self):
60     tk.Label(self.root, text="Name:").grid(row=0, column=0, sticky=tk.W)
61     self.name_entry = tk.Entry(self.root)
62     self.name_entry.grid(row=0, column=1)
63
64     self.code_text = tk.Text(self.root, height=10, width=40)
65     self.code_text.grid(row=1, column=0, columnspan=2, pady=(10, 0))
66
67     add_button = tk.Button(self.root, text="Add Snippet", command=self.add_snippet)
68     add_button.grid(row=2, column=0, pady=(10, 0))
69
70     self.snippets_listbox = tk.Listbox(self.root, height=10, width=22)
71     self.snippets_listbox.grid(row=1, column=2, padx=(10, 15), pady=(10, 0), rowspan=1)
72
73     self.populate_snippets_listbox()
74
75     self.snippets_listbox.bind("<<ListboxSelect>>", self.copy_selected_snippet)
76
77     tk.Label(self.root, text="Search Repositories:").grid(row=3, column=0, sticky=tk.W)
78     self.search_entry = tk.Entry(self.root)
79     self.search_entry.grid(row=3, column=1)
80
81     search_button = tk.Button(self.root, text="Search", command=self.search_repositories)
82     search_button.grid(row=3, column=2)
83
84     self.repositories_listbox = tk.Listbox(self.root, height=10, width=40)
85     self.repositories_listbox.grid(row=4, column=0, columnspan=2, padx=(0, 10), pady=(10, 0))
86
87     self.repositories_listbox.bind("<<ListboxSelect>>", self.copy_repository_link)
88
89 def search_repositories(self):
90
91     conn = sqlite3.connect('repositories.db')
92     c = conn.cursor()
93
94     search_term = self.search_entry.get()
95     if not search_term:
96         return
97
98     url = f"https://api.github.com/search/repositories?q={search_term}&sort=stars&order=desc"
99
100     try:
101         response = requests.get(url, verify=True)
102         response.raise_for_status()
103     except requests.exceptions.HTTPError as err:
104         messagebox.showerror("Error", f"Could not retrieve repositories: {err}")
105         return
106
107     results = response.json().get("items", [])
108
109     self.repositories_listbox.delete(0, tk.END)
110

```



```

110
111     conn = sqlite3.connect('snippets.db')
112     c = conn.cursor()
113
114     for i, result in enumerate(results[:4]):
115         name = result["full_name"]
116         link = result["html_url"]
117         self.repositories_listbox.insert(tk.END, f"{i+1}. {name}")
118         self.repositories_listbox.insert(tk.END, link)
119     c.execute("INSERT OR IGNORE INTO repositories (id, name, link) VALUES (?, ?, ?)", (i+1, na
me, link))
120
121     conn.commit()
122     conn.close()
123
124     def copy_repository_link(self, event):
125         selection = self.repositories_listbox.curselection()
126         if not selection:
127             return
128
129         link = self.repositories_listbox.get(selection)
130         self.root.clipboard_clear()
131         self.root.clipboard_append(link)
132
133         messagebox.showinfo("Success", "Repository link copied to clipboard")
134
135     def populate_snippets_listbox(self):
136         self.snippets_listbox.delete(0, tk.END)
137
138         snippets = self.database.view_snippets()
139
140         if not snippets:
141             self.snippets_listbox.insert(tk.END, "No Snippets in Database")
142             return
143
144         for name, code in snippets:
145             self.snippets_listbox.insert(tk.END, name)
146
147     def add_snippet(self):
148         name = self.name_entry.get()
149         code = self.code_text.get("1.0", tk.END).strip()
150
151         if not name or not code:
152             messagebox.showerror("Error", "Please enter a name and code")
153             return
154
155         self.database.add_snippet(name, code)
156
157         messagebox.showinfo("Success", "Snippet added successfully")
158
159         self.populate_snippets_listbox()
160
161         self.name_entry.delete(0, tk.END)
162         self.code_text.delete("1.0", tk.END)

```

```

160
161     self.name_entry.delete(0, tk.END)
162     self.code_text.delete("1.0", tk.END)
163
164     # Update the search bar with the name of the new snippet
165     self.search_entry.delete(0, tk.END)
166     self.search_entry.insert(0, name)
167     self.search_repositories()
168
169     def copy_selected_snippet(self, event):
170         selection = self.snippets_listbox.curselection()
171         if not selection:
172             return
173
174         name = self.snippets_listbox.get(selection)
175         code = self.get_code_from_database(name)
176
177         self.root.clipboard_clear()
178         self.root.clipboard_append(code)
179
180         popup = Toplevel(self.root)
181         popup.title("Copy Snippet")
182         popup.geometry("800x600")
183
184         scrollbar = Scrollbar(popup)
185         scrollbar.pack(side="right", fill="y")
186
187         textbox = Text(popup, wrap="none", yscrollcommand=scrollbar.set)
188         textbox.insert("1.0", f"Snippet Preview:\n\n{code}\n\n{name} copied to clipboard")
189         textbox.config(state="disabled", height=10, width=50)
190         textbox.pack(expand=True, fill="both")
191
192         scrollbar.config(command=textbox.yview)
193
194     def get_code_from_database(self, name):
195         self.database.c.execute("SELECT code FROM code_snippets WHERE name = ?", (name,))
196         result = self.database.c.fetchone()
197         return result[0]
198
199     def run(self):
200         self.root.mainloop()
201
202 if __name__ == "__main__":
203     gui = SnippetGUI()
204     gui.run()

```

Pielikums 2 KFSMR programmatūras kods