Rīgas 64. vidusskola

# Datorkomponentu izvelnes programma

Pielaides darbs programmēšanā

**Darba autors:** Dāvis Karpovskis, 12 DIT

Rīga, 2023

# SATURS

1. [IEVADS 3](#_bookmark0)
   1. [Nolūks 3](#_bookmark1)
   2. [Darbības sfēra 3](#_bookmark2)
   3. [Definīcijas un akronīmi 3](#_bookmark3)
   4. [Pārskats 3](#_bookmark5)
2. [PROBLĒMAS IZPĒTE UN ANALĪZE 4](#_bookmark6)
   1. [Esošā stāvokļa apraksts 4](#_bookmark7)
   2. [Mērķauditorija 4](#_bookmark8)
   3. [Programmas funkcionalitāte 4](#_bookmark9)
   4. [Vispārīgi ierobežojumi 4](#_bookmark10)
   5. [Pieņēmumi un atkarības 4](#_bookmark11)
3. [PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA 5](#_bookmark12)
   1. [Datu plūsma 5](#_bookmark13)
   2. [Datu vākšana 5](#_bookmark14)
4. [PROGRAMATŪRAS IZSTRĀDES PLĀNS 7](#_bookmark15)
5. [ATKĻŪDOŠANAS UN AKCEPTTESTĒŠANAS PĀRSKATS 8](#_bookmark16)
6. [LIETOTĀJA CEĻVEDIS 10](#_bookmark17)
7. [PIEMĒROTĀS LICENCES 11](#_bookmark18)

[SECINĀJUMI 12](#_bookmark19)

# 1 IEVADS

## Nolūks

Darbs ir sagatavots, kā pielaides darbs kursa Programmēšana II eksāmenam. Darbā ir aprakstīta programma, kas atļauj vieglāk izvēlēties datoru komponentu sarakstu. Programma atrod visus datoru komponentus pasaulē un saliek tos aplikacijā, lai varētu veikt komponentu izvēli.

## Darbības sfēra

Programma ir paredzēta jebkuram cilvēkam, bet ja ir vismaz pamatzināšanas par datorkomponentiem, tad programma var tikt izmantota efektīvāk.

## Definīcijas un akronīmi

|  |  |
| --- | --- |
| **Saīsinājums/akronīms** | **Skaidrojums** |
| API | No angļu valodas[Application Programming  Interface] - lietojumprogrammas saskarne |
| RAM | No angļu valodas[Random Access Memory] – Operatīvā atmiņa |
| CPU | No angļu valodas[Central Processing Unit] - Procesors |
| GPU | No angļu valodas[Graphics Processing Unit] – Video/Grafikas karte |
| SSD | No angļu valodas[Solid State Drive] – Cietais disks |
| HDD | No angļu valodas[Hard Disc Drive] – Cietais disks |
| PSU | No angļu valodas[Power Supply Unit] – Barošanas bloks |
| JSON | Faila Formāts, kur cilvēkam ir vieglāk salasāmi datu objekti |
| GUI | No angļu valodas[Graphical User Interface] – Grafiskais interfeiss |
| Open-Source | “Atvērtais pirmkods”. Ir atļauja brīvi mainīt un izplatīt programmas pirmkodu. |

## Pārskats

Dokumenta pirmajā nodaļā sniegts darba nolūks, darbības sfēra un dokumentā izmantotie saīsinājumi un akronīmi kuriem ir doti skaidrojumi. Dokumenta otrajā nodaļā aprakstīta problēmas izpēte, mērķauditorija, programmas funkcionalitātes, vispārīgie ierobežojumi un pieņēmumi un atkarības. Dokumenta trešajā daļā tiek parādīta datu plūsma, kā tiek ievākti dati un funkciju paskaidrojumi.

Dokumenta ceturtajā daļā tiek apskatīts programmatūras iztrādes plāns Dokumenta peiktajā daļā tiek parādīts atkļūdošanas process. Dokumenta sestajā daļā tiek sniegta informācija kā lietotājam jārīkojas izmantojot programmu. Dokumenta septītajā daļā tiek apskatīta vispiemērotākā licence šij programmai

# 2 PROBLĒMAS IZPĒTE UN ANALĪZE

## Esošā stāvokļa apraksts

Ņemot vērā cik daudz datorkomponentu ir pasaulē, ir liela iespēja apjukt skatoties uz komponentu specifikāciju. Tā kā pašam taisīt datoru ir parasti lētāk nekā pirkt jau uztaisītu datoru, ar šo programmu būs iespējams atrast pareizos komponentus resursu prasībām.

Vissvarīgākais uzdevums programmai ir atrast komponentus pēc pieprasītā resursu daudzuma, un, ja lietotājs pats izvēlas komponentus, tad parbaudīt, vai izvēlētie komponenti strādās ar viensotru. Šī ir vislielākā problēma taisot datoru, jo, ja nokļūdās komponentu izvēlē, ir iespējams zaudēt laiku un naudu.

## Mērķauditorija

Par risinājuma mērķauditoriju tiek uzskatīi visi cilvēki, kas ir ieintresēti datorus vai arī vēlas ietaupīt naudu pērkot jaunu datoru.

## Programmas funkcionalitāte

* + 1. Datu iegūšana
    2. Datu apstrāde
    3. Datu sakopošana aplikācijā

## Vispārīgi ierobežojumi

* + 1. Programmai ir nepieciešams internets, jo datus iegūst no API.

## Pieņēmumi un atkarības

Darbā tiek pieņemts, ka komponentu specifikācija ir akurāti iegūta caur <https://github.com/JonathanVusich/pcpartpicker> bibliotēkas API. API datus iegūst no mājaslapas <https://pcpartpicker.com/>, kur tiek pieņemts, ka dati par specifikācijām ir akurāti.

# 3 PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

## Datu plūsma

**Diagram

Description automatically generated**

*Attēls.3.1.0* **Programmas datu plūsmas diagramma**

## Datu vākšana

Datu vākšana tika veikta no šādiem datu avotiem:

* + - <https://pcpartpicker.com/>
    - API: <https://github.com/JonathanVusich/pcpartpicker>

No mājaslapas API ieguva informāciju par datorkomponentu specifikācijām – zīmols, modelis, cena, kodolu skaits (CPU un GPU), atmiņas daudzums (RAM atmiņai, SSD un HDD) un jauda (prieks PSU).

### Datu iegūšanas funkcija(get\_data, get\_all\_data)

Tabula 1

|  |
| --- |
| **Mērķis:** |
| Iegūst datus no <https://pcpartpicker.com/> un parveido tos JSON faila formātā |
| **Ievaddati:** |
| Komponentu nosaukumi tiek ievadīti API, kas tālāk atrod datus par katru komponentu |
| **Apstrāde:** |

|  |
| --- |
| Iegūtie dati tiek parveidoti JSON faila formātā |
| **Izvaddati:** |
| Komponentu specifikācija JSON failu formātā |

### Datu ielādēšana(load\_components)

Tabula 2

|  |
| --- |
| **Mērķis:** |
| Ielādēt datus no iegūtajiem JSON failiem un parveidot pārlasāma veidā visu komponentu specifikācijas |
| **Ievaddati:** |
| Iegūtie JSON faili |
| **Apstrāde:** |
| Parveido komponentu specifikācijas no JSON faila formāta uz teksta formātu, lai lietotājs ir spējīgs izlasīt tos. |
| **Izvadati:** |
| Komponentu specifikācijas teksta formātā |

### GUI izveidošana(create\_widgets)

Tabula 3

|  |
| --- |
| **Mērķis:** |
| Izveidot GUI un ievietot tajā iegūtos datus |
| **Ievaddati:** |
| Iegūtie dati no datu ielādēšanas funkcijas |
| **Apstrāde:** |
| Tiek ievietoti katrā sekcijā ievadati |
| **Izvaddati:** |
| Programmatūras GUI |

# 4 PROGRAMATŪRAS IZSTRĀDES PLĀNS

Vispiemērotākais modelis programmas iztrādāšanai, izlasot darba kritērijus, ir ūdenskrituma(waterfall) modelis. Šādējādi ir iespējams pilnveidot visus darba prasītos kritērijus, soli pa solim. Kā arī ir iespējams turpināt šo darbu ar šo modeli, veicinot efektīvu un kvalitatīvu darba īstenošanu.

# 5 ATKĻŪDOŠANAS UN AKCEPTTESTĒŠANAS PĀRSKATS

Lai pārbaudītu vai programma strādā, tiek atrastas specifikācijas dažiem komponentiem, kuriem ir zināmas specifikācijas, tādējādi var pārliecināties, ka iegūtie dati ir patiesi.

AMD Ryzen 7 3700X CPU

{

            "brand": "AMD",

            "model": "Ryzen 7 3700X",

            "cores": 8,

            "base\_clock": {

                "cycles": 3600000000

            },

            "boost\_clock": {

                "cycles": 4400000000

            },

            "tdp": 65,

            "integrated\_graphics": null,

            "multithreading": true,

            "price": [

                "EUR",

                "255.99"

            ]

        }

Graphical user interface, text, application

Description automatically generated

Programma ir precīzi ieguvusi datus no <https://pcpartpicker.com/> mājaslapas par AMD Ryzen 7 3700X procesoru.

Zotac Geforce RTX 2060 SUPER Videokarte

{

            "brand": "Zotac",

            "model": "AMP",

            "chipset": "GeForce RTX 2060 SUPER",

            "vram": {

                "total": 8000000000

            },

            "core\_clock": {

                "cycles": 1470000000

            },

            "boost\_clock": {

                "cycles": 1680000000

            },

            "color": "Black",

            "length": 268.0,

            "price": [

                "EUR",

                "882.01"

            ]

        }

A picture containing graphical user interface

Description automatically generated

Dati ir akurāti iegūti no <https://pcpartpicker.com/> mājaslapas par Zotac Geforce RTX 2060 SUPER Videokarti

# 6 LIETOTĀJA CEĻVEDIS

Šī programma nav pieejama publikai, bet, ja būtu, programma būtu pieejama mājaslapas veidā, jo nav vajadzības instalējāmai aplikācijai, ja to parasts lietotājs izmantos dažas reizes dzīvē. Tā būtu ērti izmantojama mājaslapa, kuru ievadati būtu tikai datorkomponenti. Lietotājam tiks doti internetveikali, no kuriem var iegādāties izvēlētos komponentus.

# 7 PIEMĒROTĀS LICENCES

Licence šij programmai varētu būt GNU Ģenerālā publiskā licence. Programma būtu Open-Source, jo izstrādātājs vēlas, lai šī programma tiek attīstīta un uzlabota vietās par kuru izstrādātājs nav padomājis. Ar katru ieguldījumu ir iespējams atvieglot lietotāja pieredzi izmantojot programmu.

# SECINĀJUMI

Rakstot programmu tika iegūtas jaunas zināšanas par API lietošanu, failu veidošanu un apstrādāšanu, kā arī tika iegūtas jaunas prasmes veidot efektīvāku kodu, kas nenoliedzami noderēs Programmēšanas II kursa eksāmena veikšanā. Lai iegūtu vēlamo rezultātu, tika secināts, ka programmā būtu jāiegulda daudz vairāk laiks un zināšanas. Iespējams nākotnē izstrādātājs varēs pielietot iegūtās zināšanas, lai izveidotu vēlamo rezultātu darbam. Kā ari iespējams pārveidot kodu, tā, lai tas aizņemtu mazāk sistēmas resursus, kā arī attīstīt datubāzi(skat. Pielikums 2), lai vispārīgi programma strādātu ātrāk un efektīvāk.

# PIELIKUMI

Pielikums 1

import json

from pcpartpicker import API

import tkinter as tk

import os

class GetData():      # \_\_init\_\_ ir konstruktors, kas uzstāda API reģionu uz "de"(Vācija)

    def \_\_init\_\_(self):

        self.api = API()

        self.api.set\_region("de")

    def get\_data(self, data\_type):  # get\_data ir funkcija, kas iegūst datus no pcpartpicker un saglabā tos json failā

        data = self.api.retrieve(data\_type)

        data = data.to\_json()

        with open(f"{data\_type}\_data.json", 'w') as outfile:

            outfile.write(data)

    def get\_all\_data(self): # get\_all\_data ir funkcija, kas izsauc get\_data funkciju un iegūst datus no pcpartpicker par visiem komponentiem

        data\_types = ['cpu', 'video-card', 'motherboard', 'memory', 'power-supply', 'internal-hard-drive'] # komponentu saraksts

        for data\_type in data\_types:

            try:

                self.get\_data(data\_type)

            except Exception as e:

                print(f"An exception occurred while retrieving {data\_type} data: {e}") # izdrukā tieši kuram komponentu neizdevās iegūt datus, ja tas notiek

GetData().get\_all\_data() # izsauc get\_all\_data funkciju, lai iegūtu datus

class ComponentSelector(tk.Frame):

    def \_\_init\_\_(self, master=None): # \_\_init\_\_ ir konstruktors, kas uzstāda komponentu sarakstus

        super().\_\_init\_\_(master)

        self.master = master

        self.cpu\_var = tk.StringVar()

        self.gpu\_var = tk.StringVar()

        self.motherboard\_var = tk.StringVar()

        self.memory\_var = tk.StringVar()

        self.psu\_var = tk.StringVar()

        self.hdd\_var = tk.StringVar()

        self.components = {}

        self.load\_components()

        self.create\_widgets()

        self.pack()

    def load\_components(self): # load\_components ir funkcija, kas ielādē datus no json failiem

        component\_files = {

            'cpu': 'cpu\_data.json',

            'video-card': 'video-card\_data.json',

            'motherboard': 'motherboard\_data.json',

            'memory': 'memory\_data.json',

            'power-supply': 'power-supply\_data.json',

            'internal-hard-drive': 'internal-hard-drive\_data.json'

        }

        for component, filename in component\_files.items():

            try:

                filepath = os.path.join(os.path.dirname(\_\_file\_\_), filename)

                filepath = filepath.encode() # Hašo faila ceļu

                filepath = filepath.decode() # Decodē faila ceļu

                with open(filepath, 'r') as f:

                    data = json.load(f)

                    self.components[component] = [(c['brand'], c['model']) for c in data[component]]

            except FileNotFoundError:

                print(f"{filename} not found")

    def create\_widgets(self): # create\_widgets ir funkcija, kas, izmantojot tkinter, izveido izvēlnes, kurās ir komponentu saraksti

        cpu\_label = tk.Label(self, text="CPU")

        cpu\_label.pack()

        cpu\_menu = tk.OptionMenu(self, self.cpu\_var, \*self.components['cpu'])

        cpu\_menu.pack()

        gpu\_label = tk.Label(self, text="GPU")

        gpu\_label.pack()

        gpu\_menu = tk.OptionMenu(self, self.gpu\_var, \*self.components['video-card'])

        gpu\_menu.pack()

        motherboard\_label = tk.Label(self, text="Motherboard")

        motherboard\_label.pack()

        motherboard\_menu = tk.OptionMenu(self, self.motherboard\_var, \*self.components['motherboard'])

        motherboard\_menu.pack()

        memory\_label = tk.Label(self, text="Memory")

        memory\_label.pack()

        memory\_menu = tk.OptionMenu(self, self.memory\_var, \*self.components['memory'])

        memory\_menu.pack()

        psu\_label = tk.Label(self, text="Power Supply")

        psu\_label.pack()

        psu\_menu = tk.OptionMenu(self, self.psu\_var, \*self.components['power-supply'])

        psu\_menu.pack()

        hdd\_label = tk.Label(self, text="Internal Hard Drive")

        hdd\_label.pack()

        hdd\_menu = tk.OptionMenu(self, self.hdd\_var, \*self.components['internal-hard-drive'])

        hdd\_menu.pack()

if \_\_name\_\_ == '\_\_main\_\_':

    selector = ComponentSelector()

    selector.mainloop()

Pielikums 2

Diagram

Description automatically generated