Rīgas 64. vidusskola

Degvielas uzpildes staciju maršruta plānotājs

Pielaides darbs programmēšanā

Darba autori: Kristaps Reinis Strods, 12 DIT

Rīga, 2023

SATURS

1 IEVADS

1.1 Nolūks

Darbs ir sagatavots, kā pielaides darbs kursa Programmēšana II eksāmenam. Darbā aprakstīta programma, kas ir spēj aprēķināt kopējās degvielas izmaksas, kas ietver veiktās distances izmaksas un uzpildes izmaksas.

1.2 Darbības sfēra

Programma ir paredzēta ikvienam lietotājam, kurš to vēlas izmantot, šī programma ļaus lietotājiem vieglāk pieņemt lēmumu vai ir vērts doties pēc lētākas degvielas.

1.3 Definīcijas un akronīmi

Saīsinājums/akronīms	Skaidrojums
DUS	Degvielas uzpildes stacija
API	No angļu valodas[Application Programming Interface] - lietojumprogrammas saskarne
Lietotājs	Persona, kas izmanto programmu
Dictonary	Datu tips, kurā tiek glabāta informācija
Key results	Vārdnīcas atslēga ar nosaukumu results
Value position	Vērtības kuras glabājas zem position nosaukuma
Latitude	Kordinātu platums grādos
Longitude	Kordinātu garums grādos

1.4 Saistība ar citiem dokumentiem

Dokumenta PPS noformēšanā ievērotas standarta LVS 68:1996 prasības, kā arī kursa Programmēšana II pielaides darba prasības.

1.5 Pārskats

Dokumenta pirmajā nodaļā sniegts darba nolūka, mērķauditorijas apraksts, kā arī uzskaitīti dokumentā izmantotie saīsinājumi un akronīmi un sniegti to skaidrojumi.

Dokumenta otrajā nodaļā aprakstīta produkta perspektīva, funkcijas, sniegtas lietotāja raksturiezīmes, minēti vispārējie ierobežojumi, uzskaitīti pieņēmumi un atkarības.

Dokumenta trešajā daļā tiek aprakstītas visas programmas funkcijas, un kādā veidā tika iegūti dati.

Dokumenta ceturtajā daļā tiek aprakstīts, ar kādām metodēm tika panākts rezultāts.

Dokumenta piektajā daļā tiek sniegta informācija par programmas testēšanu.
Sestajā daļā tiek aprakstīta informācija, kā lietotājam rīkoties, lai iegūtu vēlamo rezultātu.
Dokumenta septītajā daļa tiek sniegta informācija par piemērojamo licenci.

2 PROBLĒMAS IZPĒTE UN ANALĪZE

2.1 Esošā stāvokļa apraksts

Ņemot vērā automašīnu skaita palielināšanos uz Latvijas ceļiem un ievērojamo degvielas cenu kāpumu Latvijā, cilvēki ar vien vairāk domā par veidiem, kā ieekonomēt pārvietojoties ar automašīnu.

Cilvēkiem nereti ir maldīgs uzskats, ka veicot garāku ceļu nekā nepieciešams līdz tuvākajai DUS, tiks ietaupīts ļoti daudz līdzekļu, bet bieži vien šīs ceļa izmaksas netiek aprēķinātas. Tās veidosies veicot liekus kilometrus uz DUS, tādējādi radot liekas CO2 emisijas.

Viens no svarīgākajiem uzdevumiem ir izveidot programatūru, kura spēs atrast tuvākās DUS konkrētā radiusā un parādīt to attālumus un izmaksas, un lietotājs pats varēs izlemt vai ir vērts doties pēc lētākas degvielas.

2.2 Mērķauditorija

Par risinājuma mērķauditoriju tiek uzskatīti visi tie Latvijas autovadītāji, kas interesējas par veidiem, kā ieekonomēt patērētos līdzekļus, pārvietojoties ar automašīnu, un vienlaikus nevēlas radīt tik lielas CO2 emisijas.

2.3 Programmas funkcionalitāte

1. Datu apstrāde.
2. Datu satura atpazīšana.
3. Datu izvade balstoties uz Datu saturu.

2.4 Vispārīgi ierobežojumi

- Programatūrai ir nepieciešams Internets, jo tiek ņemti dati no Interneta.
- Ierobežotais API pieprasījumu skaits, kas ir 2500 dienā.

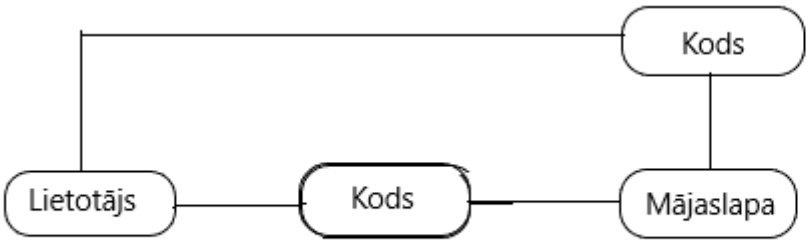
2.5 Pieņēmumi un atkarības

Darbā tiek pieņemts, ka cilvēku sniegtā informācijā mājas lapā https://gasprices.dna.lv/lv/ ir precīza, mājas lapa šos datus iegūst no aplikācijas “waze”, kur cilvēki var dalīties ar degvielas cenu konkrētajā DUS.

Izvadīto DUS skaits būs atkarīgs, no lietotāja ievadītā vēlamā rādiusa un degvielas veida.

3 PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

3.1 Datu plūsma



Attēls.3.1.0 **Programmas datu plūsmas diagramma**

3.2 Datu vākšana

Datu vākšana tika veikta no šādiem datu avotiem:

- https://gasprices.dna.lv/lv/
- API projektam tika iegūts no https://developer.tomtom.com

No API tika iegūta šāda informācija: vietu ģeogrāfiskās koordinātas, pieejamās DUS noteiktā radiusā, distances garums no konkrētās vietas. No mājas lapas https://gasprices.dna.lv/lv/ tika iegūtas konkrētas degvielas cenas konkrētām DUS, kuras tika atrastas ar API palīdzību.

Koordinātu noteikšanas funkcija

Tabula 1

Mērķis:
Iegūt koordinātas visām adresēm
Ievaddati:
Pilsētas un ielas nosaukums tiek ievietots specializētā API saites nosaukumā, kurš spēj atrast informāciju par šo vietu.
Apstrāde:

Nodrošinot savienojumu ar API tiek iegūts json fails, kurš tiek izmantots, kā dictonary, lai atrastu konkrēto key results un ar value position zem, kuras var atrast latitude and longitude
Izvaddati:
Konkrētās vietas ģeogrāfiskās kordinātas.

Tuvāko DUS noteikšanas funkcija

Tabula 2

Mērķis:
Atrast tuvumā esošās DUS konkrētā radiusā
Ievaddati:
Lietotājā izvēlētais rādiuss un iegūtās kordinātas no koordinātu noteikšanas funkcijas tiek ievietotas specializētā API saitē.
Apstrāde:
Nodrošinot savienojumu ar API tiek iegūts json fails, kurš tiks izmantots vēlāk funkcijā DUS
Izvadati:
Dictonary ar tikai results kā key vērtību

Attāluma noteikšanas funkcija

Tabula 3

Mērķis:
Noteikt distanci starp ievadīto adresi un atrastajām DUS
Ievaddati:
Abu objektu kordinātas tiek ievietotas specializētā API saitē
Apstrāde:
Nodrošinot savienojumu ar API tiek iegūts json fails, no kura tiek atrasts key routes un values un sumary un lenghtInMeters un tiek izdalīta ar 1000, jo atbilde ir metros
Izvaddati:
Distance no ievadītās adreses uz atrastajām DUS kilometros.

Mājaslapas funkcija

Tabula 4

Mērķis:
Iegūt datus no DUS konkrētajā pilsētā
Ievaddati:
Lietotāja ievietotā pilsēta tiek ielikta saitē https://gasprices.dna.lv/lv/?city=izvēlētā pilsēta
Apstrāde:
Ar bibliotēku Selenium tiek izvēlēts atrast elementus ar klases nosaukumu “dusitem”, jo zem šīs klases tiek glabāti datu kuri ir nepieciešami.
Izvaddati:
Visu DUS izvēlētajā pilsētā: nosaukumus, adreses un konkrētu degvielas veidu cenas

Degvielas cenas funkcija

Tabula 5

Mērķis:
Iegūt degvielas cenu
Ievaddati:
Lietotāja izvēlētais degvielas veids un Tuvākās DUS
Apstrāde:
Veicot pārveidojumus tiek atpazīts ievadītais degvielas veids un tiek veikti pārveidojumu lai iegūtu konkrētu cenu.
Izvaddati:
Visu tuvāko DUS cenas konkrētam degvielas veidam

Izmaksu funkcija

Tabula 6

Mērķis:
Noteikt izmaksas visām DUS
Ievaddati:
Lietotāja ievadīts degvielas patēriņš, plānotais uzpildes daudzums, degvielas veida cena, distance.
Apstrāde:

Pēc formulas (cena*distance*patēriņš)/100+(cena*plānotais uzpildes daudzums), tiek aprēķinātas kopējās izmaksas dodoties uz dažādām dus.
Izvaddati:
Izmaksas konkrētām DUS.

DUS funkcija

Tabula 7

Mērķis:
Atrast un apkopot informāciju par visām DUS
Ievaddati:
Tuvākās DUS, mājaslapas DUS, izmaksas, distance.
Apstrāde:
Attīra datus, no neeksistējošām DUS, atrod konkrētai DUS atbilstošo cenu un apkopo tos
Izvaddati:
Izvada vairākas DUS kuras lietotājs var salīdzināt

4 PROGRAMATŪRAS IZSTRĀDES PLĀNS

No visiem programatūras izstrādes modeļiem, vispiemērotākais būs ūdenskrituma modelis, jo pirmais posms, kas veikts pēc iepazīšanās ar darba kritērijiem, pēc kuriem varēs veikt tālākas darbības darba izstrādē, tika izdomāts temats un tad sekoja programmēšanas un testēšanas posms un šis ir dokumentācijas posms.

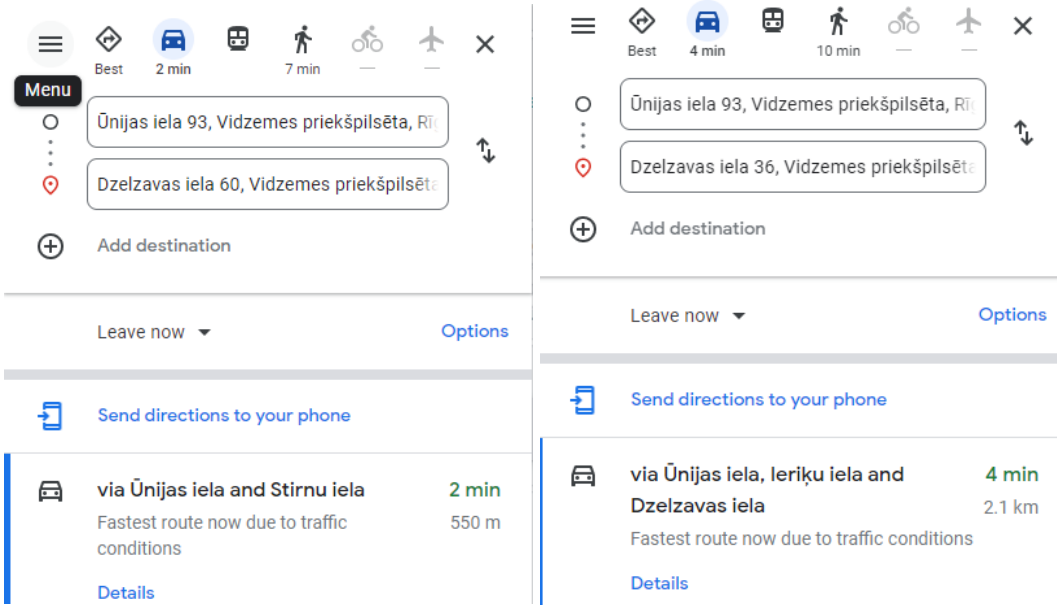
5 ATKĻŪDOŠANAS UN AKCEPTTESTĒŠANAS PĀRSKATS

Lai pārbaudītu vai kods strādā, tika ievadītas dažādas zināmas adreses, kuru datu patiesums var tikt novērtēts bez papildus analīzes, piemēram, tika izvēlēta mājas vai skolas adrese un salīdzināta ar google maps pieejamo informāciju.

```
Ievadi pilsētu: Rīga
Ievadi adresi: Ūnijas iela 93
ievadi mašīnas patēriņu (l/100km): 7
ievadi uzpildes daudzumu (l): 45
ievadi degvielas veidu (DD/95/98/LPG): 95
ievadi radiusu3.5
```

```
Dzelzavas iela 36 attālums līdz DUS 3.023 km un kopējās izmaksas būs 74.92 eiro
Dzelzavas iela 60 attālums līdz DUS 0.557 km un kopējās izmaksas būs 74.63 eiro
Gunāra Astras iela 7 attālums līdz DUS 2.232 km un kopējās izmaksas būs 74.82 eiro
Dzelzavas iela 14 attālums līdz DUS 2.618 km un kopējās izmaksas būs 71.12 eiro
Dzelzavas iela 3 attālums līdz DUS 2.481 km un kopējās izmaksas būs 74.85 eiro
Gunāra Astras iela 10 attālums līdz DUS 1.609 km un kopējās izmaksas būs 75.97 eiro
Augusta Deglava iela 51A attālums līdz DUS 2.427 km un kopējās izmaksas būs 74.85 eiro
```

Tika izvēlēta DUS Rīgā, Dzelzavas ielā 60 un Dzelzavas iela 36, jo viena atrodas vistuvāk un otra vistālāk.



Dzelzavas ielas 60 DUS programma ir aprēķinājusi diezgan precīzi, bet Dzelzavas ielas 36 attālums ir ļoti atšķirīgs. Šo iespējams varētu būt izraisijusi API attāluma aprēķināšana, kurā varētu nebūt jaunāko datu, jo tie netiek tik bieži atjaunināti, to izstrādātājs nevar ietekmēt.

Koda izstrādes sākumā tika novērots, ka API atrod tuvumā esošas DUS, kuras nav zināmas, tāpēc tās tika pārbaudītas internetā un tika izņemtas no datiem.

Lai pārliecinātos par aprēķinu patiesumu tika radīta formula, pēc kuras izstrādātāji manuāli aprēķināja rezultātus.

6 LIETOTĀJA CEĻVEDIS

Šī programma pagaidām nav pieejama plašākai publikai, bet, ja būtu, tad tā būtu telefona aplikācijas veidā, kuru varētu lejupielādēt uz jebkura telefona. Lietotājam vajadzētu tikai ievadīt pilsētu, adresi, kurā pašlaik atrodas, degvielas patēriņu (l/100) cik plāno uzpildīt (l), degvielas veidu, un rādiusu, kurā vēlas atrast savu DUS. Lietotājam tiks piedāvātas visas DUS, kas atrodas izvēlētajā rādiusā, un to attālums un izmaksas. Lietotājam pašam būs iespēja izvēlēties, starp šīm DUS uz kuru doties.

7 PIEMĒROTĀS LICENCES

Šī koda licence varētu būt GPL licence, jo izstrādātāji, vēlas, lai šī programatūra būtu pieejama plašākai publikai un dotu pienesumu mūsdienu sabiedrībai, un priecātos, ja kāds cits izstrādātājs uzlabotu šo programatūru, lai tā dotu vēl lielāku ieguldījumu sabiedrībā, kods ir pieejams saitē https://github.com/kakaoenjoyer/Degvielas-cenas-aprekinatajs.

SECINĀJUMI

Rakstot šo darbu no iegūtajiem rezultātiem tika secināts, ka veicot tālāku ceļu tiks ietaupīta neliela naudas summa un ka nav vērts tērēt laiku un lieki radīt vides piesārņojumu. Izstrādātāji apguva un pilnveidoja prasmes, API lietošanā, bibliotēkas "Selenium lietošanā", un funkciju veidošanā, visas šīs prasmes noderēs eksāmenam. Nākotnē izstrādātāji varētu apkopot visu DUS kordinātas datubāzē, kas atvieglotu kodu, jo nebūtu nepieciešams meklēt katras DUS kordinātas relācijas modeli skat. Pielikums 2.

Pielikumi

Pielikums 1

```
import requests
import json
from selenium import webdriver
from selenium.webdriver.common.by import By

def kordinatas(address, city, api_key):#funkcija lai atrastu visu vietu
kordinātas
    url =
f"https://api.tomtom.com/search/2/geocode/{address},{city}.json?key={api_key}"
    response = requests.get(url)
    data = response.json()

    return (data["results"][0]["position"]["lat"],
data["results"][0]["position"]["lon"])


def DUS_tuvuma(latitude, longitude, radiuss, api_key):#pēc ievadītās adreses
kordinātām tiek atrastas tuvejas DUS
    url = f"https://api.tomtom.com/search/2/categorySearch/petrol
station.json?key={api_key}&lat={latitude}&lon={longitude}&radius={radiuss*1000
}"
    response = requests.get(url)
    data = response.json()

    return data["results"]


def attalums(start_lat, start_lon, end_lat, end_lon, api_key):#pēc kordinātām
atrod attālumu starp A un B punktiem
    url =
f'https://api.tomtom.com/routing/1/calculateRoute/{start_lat},{start_lon}:{end
_lat},{end_lon}/json?key={api_key}&traffic=true'
    response = requests.get(url)
    data = response.json()

    return data["routes"][0]["summary"]["lengthInMeters"] / 1000


def majaslapa(city):# noskraipo datus no mājas lapas
    driver = webdriver.Chrome()
    driver.get(f"https://gasprices.dna.lv/lv/?city={city}")

    return driver.find_elements(By.CLASS_NAME,('dusitem'))


def degvielas_cena(gas_station, fuel_type):
```

```
    prices = gas_station.text.replace('\n','EUR',2)# parveido pirmos divus uz
EUR
    prices = prices.split('EUR')# sadala pa EUR
    spec = '\n'+fuel_type

    for i in range (1, len(prices)): # pārbauda DD, jo viņš vienigais ir bez
\n
        if fuel_type =='DD':
            if fuel_type in prices[i]:
                return float(prices[i].replace(fuel_type + '\n',''))#noņem
nost degvielas nosaukumu, lai paliktu tikai skaitlis
        elif spec in prices[i]:
            return float(prices[i].replace(fuel_type + '\n',''))

    return None

def izmaksas(distance, fuel_consumption, fuel_price, fuel_amount):

    return (fuel_price*fuel_consumption*distance)/100+(fuel_price*fuel_amount)

def letaka_DUS(address, city, fuel_type, fuel_consumption, fuel_amount,
radiuss, api_key):
    gas_price_data = majaslapa(city)
    coordinates = kordinatas(address,city, api_key)
    gas_stations = DUS_tuvuma(*coordinates, radiuss, api_key)
    printed_addresses = [] #lists priekš adresēm

    for gas_station in gas_stations:
        if gas_station['poi']['name'] not in ['Lukoil', 'Trest']:
            gas_station_address = gas_station['address']['freeformAddress']
            k = gas_station_address.find(",")
            gas_station_address = gas_station_address[:k]

            if gas_station_address not in printed_addresses: # pārbbauda vai
adrese jau ir tikusi izkopēta
                printed_addresses.append(gas_station_address)
                gas_price = None
                for gas_price_element in gas_price_data:
                    if gas_station_address in gas_price_element.text:
                        gas_price = degvielas_cena(gas_price_element,
fuel_type)
                        break

                if gas_price is not None:
                    distance = attalums(*coordinates,
*kordinatas(gas_station_address,city,api_key), api_key)
                    fuel_cost = izmaksas(distance, fuel_consumption,
gas_price, fuel_amount)
```

```
                    fuel_cost = round(fuel_cost,2) #izmaksas noapaļo lidz 2
cipariem aiz komata
                    atbilde = (str(gas_station_address)+' attālums līdz DUS
'+str(distance)+' km un kopējās izmaksas būs '+str(fuel_cost)+' eiro
').encode()
                    atb = atbilde.decode()
                    print(atb)

api_key = '63xr4Tv54wXI8xyOYAXeKlTrtIpWS5yJ'
city = input('Ievadi pilsētu: ')
address = input('Ievadi adresi: ')
fuel_consumption = float(input('ievadi mašīnas patēriņu (l/100km): '))
fuel_amount = float(input('ievadi uzpildes daudzumu (l): '))
fuel_type = input('ievadi degvielas veidu (DD/95/98/LPG): ')
radiuss = float(input('ievadi radiusu'))

letaka_DUS(address, city, fuel_type, fuel_consumption,
fuel_amount,radiuss,api_key)
```

Lai labāk pārredzētu kodu, kods ir pieejams arī saitē “https://github.com/kakaoenjoyer/Degvielas-cenas-aprekinatajs”

Pielikums 2