**Colegiul Național „Gheorghe Munteanu Murgoci ”**

**BRĂILA**

**LUCRARE PENTRU ATESTAREA COMPETENȚELOR PROFESIONALE LA INFORMATICĂ**

**Îndrumător: Prof. Marin Florina**

**Autor: elev Minea Ștefan Teodor**

**2024**

Cuprins

[Argument 2](#_Toc165934925)

[Aspecte teoretice 3](#_Toc165934926)

[Limbajul Python 3](#_Toc165934927)

[Elemente de HTML și CSS 4](#_Toc165934928)

[Biblioteca pandas 5](#_Toc165934929)

[Biblioteca matplotlib 6](#_Toc165934930)

[Biblioteca selectolax 7](#_Toc165934931)

[Bibliotecile tkinter si ttkbootstrap 8](#_Toc165934932)

[Resurse hardware si software necesare 10](#_Toc165934933)

[Structura aplicației 11](#_Toc165934934)

[Codul sursă 15](#_Toc165934935)

[Concluzie 27](#_Toc165934936)

[Sitografie 28](#_Toc165934937)

# Argument

Motivul alegerii temei a fost dorința de a ajuta oamenii în crearea unei viziuni de ansamblu asupra pieței automobilelor second-hand. Proiectul are ca scop informarea asupra unor aspecte generale referitoare la autoturismele ce aparțin unei anumite mărci. Astfel, relevanța sa educațională este dată de rolul în educația consumatorului.

Această aplicație se adresează potențialilor cumpărători, sau celor care doresc să fie informați cu privire la automobile.

Aplicația este utilă prin faptul că oferă posibilitatea vizualizării de date generale care nu sunt accesibile pe website-uri-le de anunțuri de vânzări automobile aflate la mâna a doua. Programul conceput adună date despre fiecare mașină a unei mărci, interpretând suma rezultatelor obținute și creând grafice în baza acestora. Informațiile sunt adunate de pe site-ul “https://www.autovit.ro/” întrucât acesta are cele mai multe anunțuri în domeniu din România.

# Aspecte teoretice

## Limbajul Python

Python este un limbaj de programare extrem de valoros și versatil datorită numeroaselor sale caracteristici și avantaje care îl fac preferat în diverse domenii.

Unul dintre punctele forte ale Pythonului este ușurința sa de utilizare. Sintaxa sa simplă și lizibilă face ca limbajul să fie accesibil atât pentru programatorii începători, cât și pentru cei experimentați. Această caracteristică permite scrierea de cod concis și clar, facilitând dezvoltarea rapidă a aplicațiilor și a scripturilor.

Un alt beneficiu al Pythonului este vasta sa colecție de biblioteci și framework-uri. Python dispune de o gamă largă de pachete specializate care acoperă diverse domenii, de la dezvoltarea web la inteligența artificială și analiza datelor. Printre cele mai populare biblioteci se numără Django pentru dezvoltarea web, TensorFlow pentru machine learning și Pygame pentru dezvoltarea de jocuri.

De asemenea, Pythonul este cunoscut pentru comunitatea sa activă și sprijinul continuu pentru dezvoltarea și îmbunătățirea limbajului. Există numeroase resurse, tutoriale și documentații disponibile online, ceea ce face ca învățarea și explorarea Pythonului să fie accesibilă și plăcută pentru toți cei interesați de programare.

Unul dintre punctele forte ale Pythonului este manipularea datelor, fiind disponibile biblioteci specializate precum pandas și NumPy. Aceste biblioteci oferă structuri de date puternice și instrumente avansate pentru manipularea și analiza datelor într-un mod eficient. De exemplu, pandas oferă DataFrame-uri, care permit organizarea și lucrul cu datele într-un format tabular, facilitând operațiuni precum filtrarea, sortarea, gruparea și agregarea datelor.

Datorită eficienței sale în manipularea datelor și a performanței ridicate a bibliotecilor sale, Python este utilizat frecvent în știința datelor, analiza financiară, analiza biologică și multe alte domenii în care lucrul cu volume mari de date este esențial.

De asemenea, Python dispune de biblioteci de vizualizare a datelor precum matplotlib și seaborn, care permit crearea de grafice și diagrame interactive pentru a vizualiza rezultatele analizelor de date. Aceste instrumente facilitează înțelegerea și interpretarea datelor, oferind o modalitate eficientă de comunicare a rezultatelor.

În concluzie, Pythonul este un limbaj de programare puternic și accesibil, potrivit pentru proiecte de toate dimensiunile și nivelurile de complexitate. Simplitatea sa, bibliotecile extinse și comunitatea activă îl fac o alegere ideală pentru programatori din diverse domenii și constituie baza pentru dezvoltarea unei game largi de aplicații și soluții software.

## Elemente de HTML și CSS

HTML (HyperText Markup Language) și CSS (Cascading Style Sheets) sunt două limbaje fundamentale utilizate în dezvoltarea web pentru structurarea și stilizarea paginilor web. Împreună, aceste limbaje permit crearea de pagini web interactive și estetice.

HTML este un limbaj de marcă pentru structurarea conținutului unei pagini web. Acesta utilizează o sintaxă bazată pe etichete (tags) pentru a defini diferite elemente ale unei pagini, cum ar fi titlul, paragrafele, imaginile, linkurile și multe altele. Elementele HTML sunt organizate sub forma unui arbore ierarhic, cu un element <html> ca rădăcină, urmat de <head> și <body>, care conțin informații despre pagină și conținutul vizibil pentru utilizatori.

Exemple de elemente HTML includ:

* <h1> până la <h6> pentru titluri de diferite niveluri
* <p> pentru paragrafe de text
* <a> pentru linkuri hipertext
* <img> pentru imagini
* <ul> și <ol> pentru liste neordonate și ordonate
* <table> pentru tabele
* <form> pentru formulare interactive

CSS este limbajul folosit pentru stilizarea și prezentarea elementelor HTML. Acesta permite definirea aspectului vizual al paginilor web, inclusiv culori, fonturi, dimensiuni, aliniere și layout-ul general al paginii. CSS utilizează selectori pentru a aplica stiluri la elementele HTML, oferind flexibilitate și control asupra aspectului și comportamentului paginilor web.

Exemple de proprietăți CSS includ:

* color pentru culoarea textului
* font-family pentru specificarea familiei de fonturi
* padding și margin pentru spațiul între elemente
* display pentru modul de afișare a elementelor (inline, block, flex etc.)
* position pentru controlul poziționării elementelor în layout-ul paginii

De asemenea, important în CSS este conceptul de cascading (cascadare), care se referă la modul în care stilurile sunt aplicate într-o pagină web. Stilurile pot fi definite local (inline), într-un bloc <style> în antetul paginii sau într-un fișier extern .css. Stilurile sunt apoi aplicate în funcție de specificitatea selectorilor și de ordinea în care sunt definite, oferind un control granular asupra aspectului final al paginii.

Un alt aspect important în CSS este atributul class, folosit pentru a aplica stiluri la elemente HTML într-un mod flexibil și eficient. Acest atribut permite definirea unor grupuri de stiluri care pot fi apoi utilizate pe mai multe elemente fără a fi necesară repetarea stilurilor pentru fiecare element individual.

Prin definirea unei clase CSS cu anumite stiluri, cum ar fi culoarea textului, dimensiunea fontului sau alte proprietăți de design, putem aplica aceste stiluri la unul sau mai multe elemente HTML adăugând atributul class cu numele clasei respective. De exemplu, o clasă CSS numită .stil-text care definește un anumit aspect pentru text poate fi aplicată la diferite elemente <p>, <div>, sau orice alt element dorit, simplu prin adăugarea atributului class="stil-text" la aceste elemente.

Un alt beneficiu al atributului class este capacitatea de a aplica mai multe clase la același element HTML. Acest lucru permite combinarea diferitelor stiluri din mai multe clase pentru a obține un aspect specific. De exemplu, putem avea o clasă pentru culoarea textului și o altă clasă pentru fundal, și putem aplica ambele clase la același element pentru a obține un aspect definit de ambele stiluri.

Utilizarea atributului class în CSS ajută la organizarea și structurarea codului CSS, facilitând reutilizarea și gestionarea eficientă a stilurilor într-o aplicație web. Această practică duce la un cod mai curat, mai ușor de întreținut și mai eficient din punct de vedere al performanței în procesul de dezvoltare a paginilor web.

Combinația dintre HTML și CSS permite dezvoltatorilor să creeze pagini web interactiv și estetice, adaptate pentru diferite dispozitive și ecrane. Aceste limbaje formează fundația web-ului modern, oferind un cadru puternic pentru construirea experiențelor digitale și a interfețelor web interactive și atractive.

## Biblioteca pandas

Biblioteca pandas este un instrument esențial în limbajul Python pentru lucrul eficient cu date tabulare, permițând manipularea și analiza datelor într-un mod flexibil și puternic. Una dintre cele mai importante structuri de date oferite de pandas este DataFrame-ul, care poate fi comparat cu o tablă de bază de date sau o foaie de calcul, având rânduri și coloane etichetate.

Un DataFrame în pandas este o structură bidimensională, în care datele sunt aranjate în rânduri și coloane, fiecare coloană având un nume asociat. Această structură permite manipularea facilă a datelor, inclusiv indexare, selecție, filtrare și agregare.

Un prim pas important în lucrul cu pandas este importarea și exportarea datelor. Pentru a importa date într-un DataFrame, putem folosi diverse metode. De exemplu, pentru a importa date dintr-un fișier CSV (Comma Separated Values), putem utiliza funcția pd.read\_csv('nume\_fisier.csv'). Dacă avem date într-un fișier Excel, putem folosi pd.read\_excel('nume\_fisier.xlsx', sheet\_name='nume\_foaie') pentru a importa datele într-un DataFrame.

La exportarea datelor, pandas ne permite să salvăm DataFrame-ul în diverse formate. De exemplu, pentru a salva DataFrame-ul într-un fișier Excel, putem folosi df.to\_excel('nume\_fisier.xlsx', index=False) pentru a crea un fișier Excel fără a include indexul DataFrame-ului.

Pandas se integrează bine cu biblioteca matplotlib pentru vizualizare. Matplotlib oferă instrumente puternice pentru crearea de grafice și vizualizări din datele dintr-un DataFrame. De exemplu, putem utiliza matplotlib pentru a trasa grafice de linie sau grafice de bare pentru a reprezenta datele din DataFrame sub formă vizuală.

Integrarea dintre pandas și matplotlib oferă o varietate de opțiuni pentru vizualizarea și interpretarea datelor. Această combinație este extrem de utilă în analiza de date și explorarea datelor în Python, permitând utilizatorilor să efectueze manipulări complexe și să creeze vizualizări eficiente pentru a extrage informații importante din seturile de date.

## Biblioteca matplotlib

Matplotlib este o bibliotecă puternică pentru vizualizare de date în limbajul Python, oferind diverse funcționalități pentru crearea graficelor de bare, graficelor de linie și graficelor de proporție, printre altele.

Unul dintre cele mai utilizate tipuri de vizualizare oferite de matplotlib este diagrama de bare. Această diagramă este ideală pentru reprezentarea datelor categorice sau a datelor agregate în funcție de diferite categorii. Utilizând funcția plt.bar(), putem crea rapid o diagramă de bare în matplotlib, specificând valorile pentru axele x și y, precum și alte detalii cum ar fi etichetele și titlul graficului.

În ceea ce privește graficele de linie, acestea sunt utile pentru a ilustra tendințe sau evoluții în timpul datelor. Cu matplotlib, putem folosi plt.plot() pentru a trasa un grafic de linie, specificând valorile pentru axele x și y. Graficele de linie sunt eficiente pentru vizualizarea schimbărilor într-o serie de date în funcție de o variabilă independentă.

Pentru graficele de proporție (sau diagramele circulare), matplotlib oferă funcția plt.pie(), care este utilă pentru reprezentarea proporțiilor unei întregi în funcție de subcategorii sau segmente. Această funcție permite specificarea valorilor și etichetelor pentru fiecare segment al diagramei circulare, precum și alte detalii estetice.

Un alt aspect important al matplotlib este capacitatea sa de personalizare. Putem modifica aspectul graficelor prin adăugarea de titluri, etichete ale axelor, legende și diferite stiluri vizuale pentru a face graficele mai ușor de înțeles și mai atrăgătoare.

Matplotlib este adesea folosită împreună cu alte biblioteci de analiză de date, cum ar fi pandas și NumPy, pentru a crea vizualizări complexe și a interpreta datele într-un mod comprehensibil și interactiv. De la grafice simple la vizualizări avansate, matplotlib rămâne una dintre cele mai populare și puternice biblioteci pentru vizualizarea de date în Python, fiind folosită pe scară largă în domenii precum știința datelor, analiza financiară, vizualizarea geografică și multe altele.

## Biblioteca selectolax

Biblioteca selectolax în limbajul Python, și în special modulul său parser, oferă funcționalități avansate pentru analiza și manipularea rapidă a codului HTML. Acest modul poate fi utilizat eficient în procesul de web scraping pentru extragerea informațiilor din pagini web.

În contextul integrării cu biblioteca playwright.sync\_api, care este o altă bibliotecă puternică pentru automatizarea browserelor web în Python, putem crea fluxuri de lucru complexe pentru a naviga pe pagini web și a extrage datele dorite.

Prin utilizarea selectolax.parser, putem analiza rapid și manipula conținutul HTML al paginilor web, extrăgând informații relevante și structurând datele într-un format util pentru analiză ulterioară. Acest modul oferă funcționalități avansate, similare cu cele din jQuery, pentru a selecta și accesa elementele HTML într-un mod flexibil și eficient.

Pe de altă parte, playwright.sync\_api ne permite să automatizăm interacțiunea cu browser-ul web, permițând navigarea pe pagini, completarea formularelor și accesarea conținutului din paginile web. Această bibliotecă oferă un control robust asupra browser-ului și a conținutului web, ceea ce facilitează procesul de extragere a datelor din surse online.

Pentru a combina selectolax cu playwright.sync\_api într-un proces de web scraping, putem urmări acești pași:

1. Obținerea conținutului HTML: Utilizăm playwright.sync\_api pentru a accesa și obține conținutul HTML al unei pagini web.
2. Utilizarea selectolax.parser pentru analiză: După ce obținem conținutul HTML, folosim selectolax.parser pentru a “parsa” acest conținut și a extrage informațiile relevante, cum ar fi textul din anumite taguri sau alte elemente HTML.
3. Manipularea și procesarea datelor: Folosind selectolax, putem procesa și extrage datele dorite din conținutul HTML obținut cu ajutorul playwright.sync\_api.
4. Extragerea informațiilor: Utilizând funcționalitățile oferite de ambele biblioteci, putem extrage și organiza informațiile din pagini web într-un format util și ușor de utilizat pentru analiză ulterioară.

În ansamblu, integrarea acestor biblioteci reprezintă o abordare eficientă pentru a manipula și extrage date din web într-un mod precis, controlat în limbajul Python.

## Bibliotecile tkinter si ttkbootstrap

Biblioteca tkinter și pachetul ttkbootstrap sunt două instrumente esențiale în limbajul Python în scopul creării interfețelor grafice pentru utilizatori (GUI). tkinter este o bibliotecă standard Python care oferă un set de instrumente pentru construirea interfețelor grafice simple, în timp ce ttkbootstrap este o extensie care adaugă aspecte moderne și stiluri îmbunătățite pentru componentele GUI.

Unul dintre cele mai simple elemente din tkinter este eticheta (label), utilizată pentru afișarea textului static sau a altor informații pe o fereastră GUI. Etichetele sunt create folosind clasa Label și pot fi plasate în diferite locații pe fereastră.

În cadrul dezvoltării interfețelor grafice cu tkinter, o altă componentă importantă este combobox-ul, care combină funcționalitatea unei casete de text cu lista derulantă (dropdown list). Combobox-ul permite utilizatorului să selecteze una sau mai multe opțiuni dintr-o listă predefinită.

În tkinter, combobox-urile pot fi create folosind clasa ttk.Combobox din pachetul ttk. Pentru a utiliza un combobox, mai întâi trebuie să definim opțiunile disponibile într-o listă sau tuplu, iar apoi să asociem aceste opțiuni cu combobox-ul. Utilizatorul poate selecta una dintre opțiuni din lista derulantă a combobox-ului sau poate introduce manual o valoare în caseta de text, în funcție de configurare.

Butonul (button) este un alt element fundamental al interfețelor GUI. În tkinter, putem crea butoane folosind clasa Button, oferindu-le funcționalitate prin legarea lor la funcții de callback pentru a gestiona evenimente precum clicul.

Butonul de comutare (toggle button) sau butonul cu comutator este o extensie a butonului obișnuit care poate fi activat și dezactivat alternativ. Această funcționalitate poate fi implementată în tkinter folosind clasa Checkbutton cu un aspect și comportament specific.

ttkbootstrap extinde funcționalitățile tkinter, oferind un set de stiluri îmbunătățite pentru componentele GUI. Acest pachet adaugă teme moderne și aspecte estetice care pot fi aplicate componentelor tkinter, inclusiv butoanelor, etichetelor și casetelor de selectare. Astfel, ttkbootstrap îmbunătățește experiența de proiectare a interfeței utilizator, oferind mai multe opțiuni de stilizare.

Integrarea dintre tkinter și ttkbootstrap se face prin importarea și utilizarea componentelor ttk (themed widgets) în cadrul unui proiect tkinter. Astfel, putem crea interfețe grafice moderne și estetice în Python, folosind o combinație între cele două biblioteci.

Împreună, tkinter și ttkbootstrap sunt instrumente puternice pentru dezvoltarea rapidă a interfețelor grafice în Python, permițând dezvoltatorilor să creeze aplicații interactive și prietenoase cu utilizatorul pentru o varietate de scopuri, de la aplicații desktop simple până la instrumente mai complexe cu interfețe sofisticate.

# Resurse hardware si software necesare

**Resurse minime hardware:**

* Procesor Intel Core i5 sau echivalent
* 4 GB RAM (de preferat 8 GB)
* 15 GB spațiu disponibil pe hard-disk
* Conexiune la Internet

**Resurse minime software:**

* Sistem de operare: Windows, Linux, macOS
* Biblioteci:

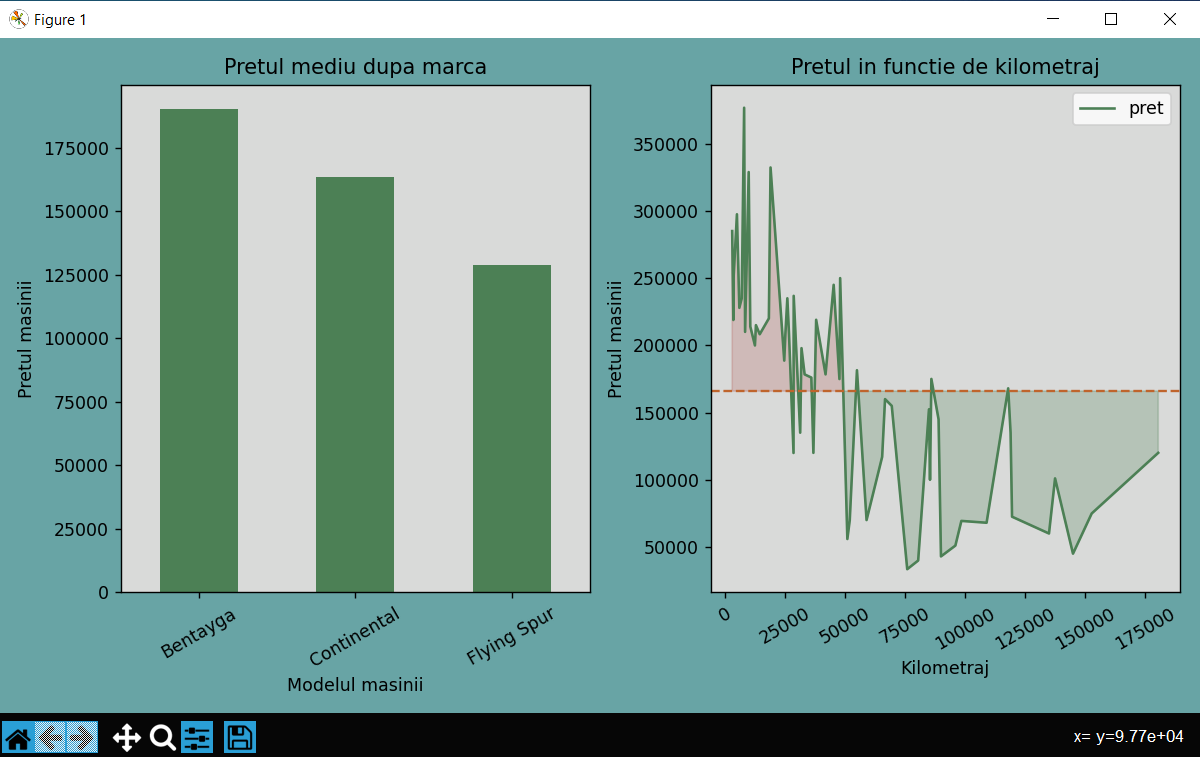
# Structura aplicației

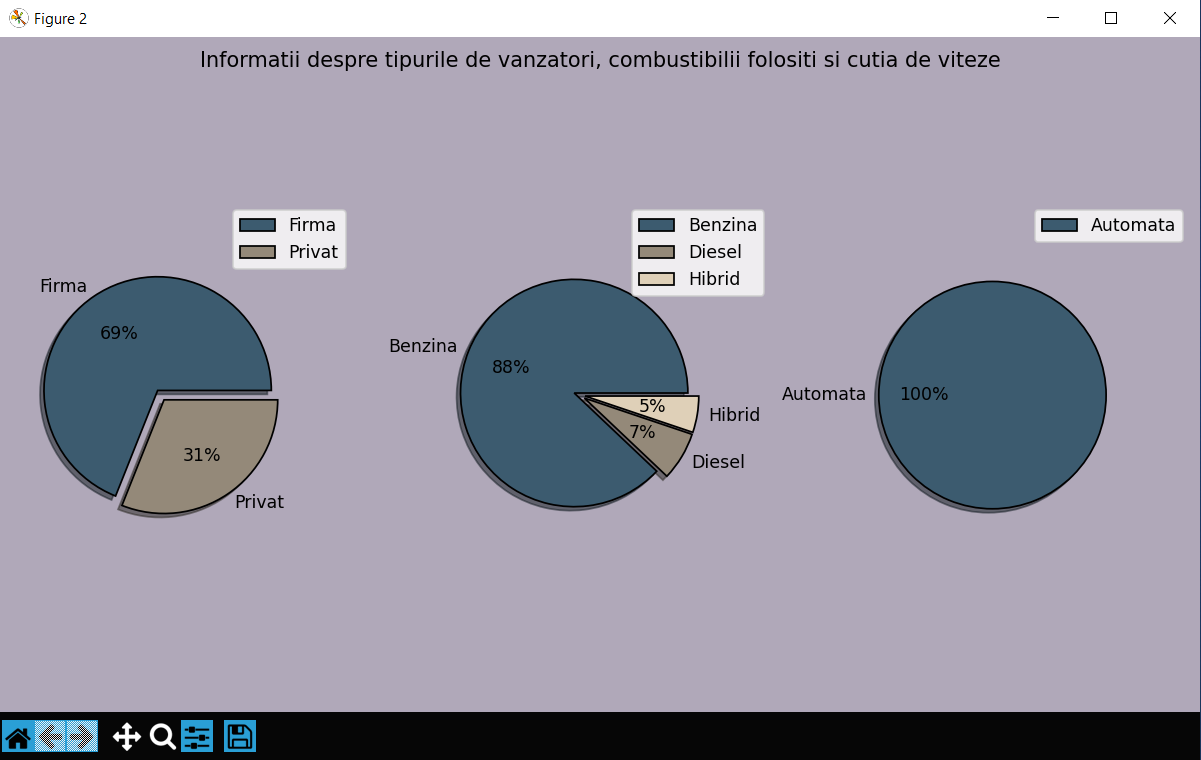
La lansarea aplicației se deschide ecranul principal. Am creat această interfață grafică cu ajutorul bibliotecii ttkbootstrap. Acesta este alcătuit din patru elemente pe care le voi prezenta pe rând.

Primul element este eticheta din partea superioară a ecranului care conține textul de culoare albastră. Acesta are scopul de a orienta utilizatorul cu privire la utilizarea aplicației. Eticheta se va modifica în funcție de context, afișând mesaje precum: “Nu există date deja salvate despre mașină” în cazul unei erori.

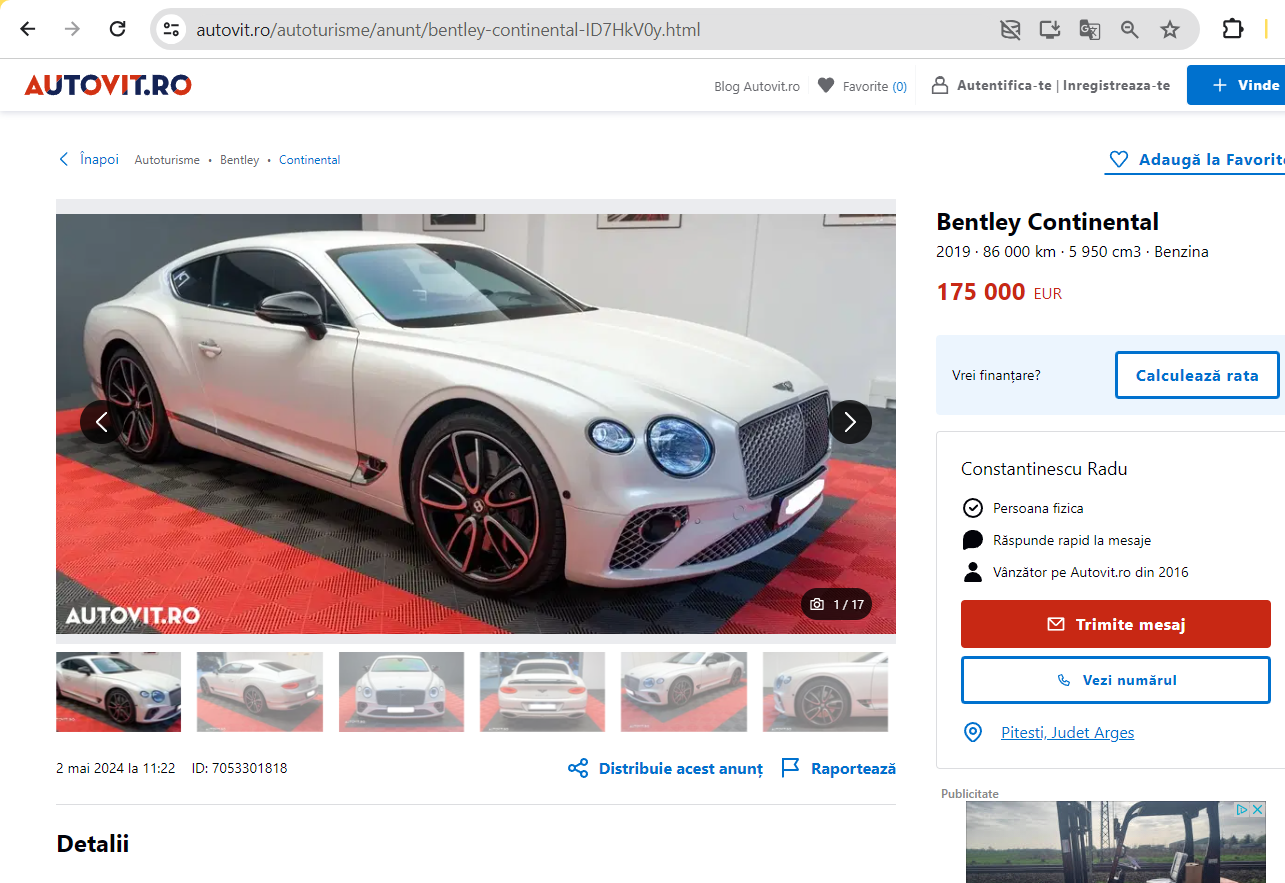
Cel de-al doi-lea element este caseta de selectare de culoare mov. Aceasta permite utilizatorului să aleagă marca de mașină despre care dorește să extragă date. În momentul în care utilizatorul va apăsa pe săgeata din colțul din dreapta jos, va apărea o listă de unde va putea alege marca de mașini dorită

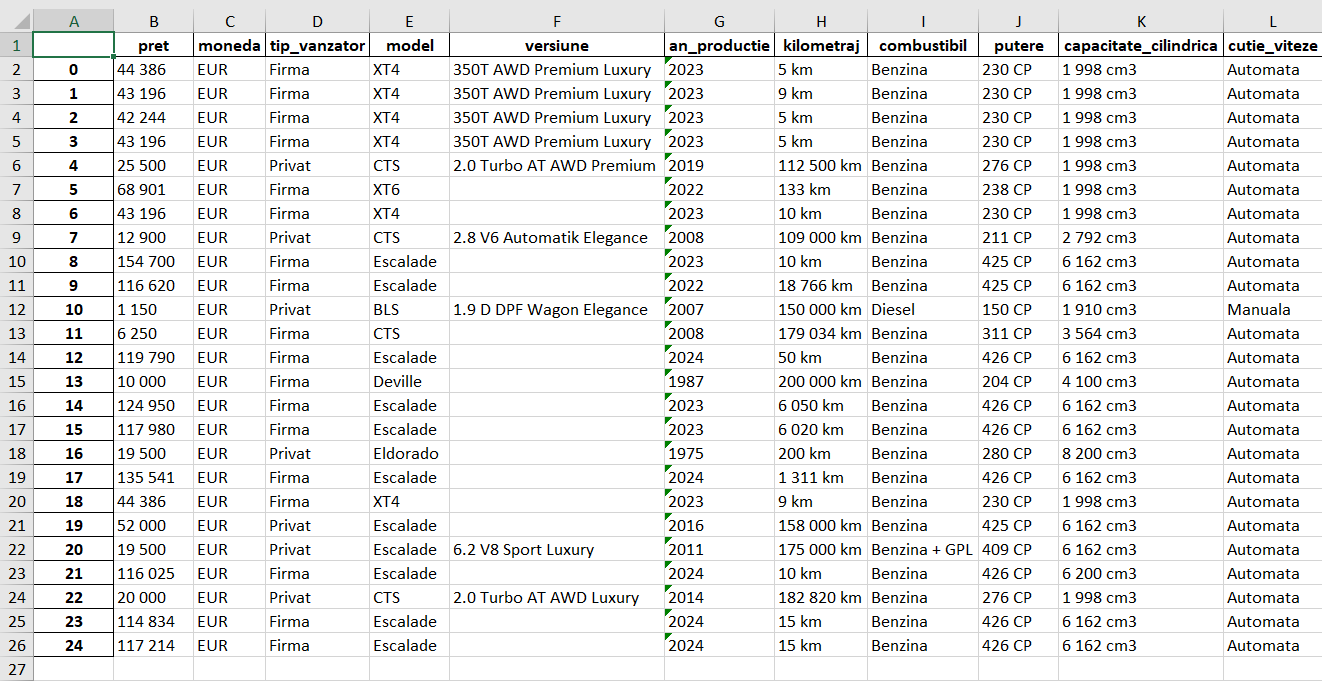
Al trei-lea element este un buton de comutare căruia îi este asociat un text ce îi explică întrebuințarea. În cazul în care este activ, programul înțelege că au mai fost generate anterior statistici ale mărcii respective și va încerca să acceseze datele deja colectate pentru afișarea statisticilor. În cazul în care utilizatorul cere folosirea unor date deja existente, însă programul nu le găsește, atunci va genera un mesaj de eroare în ecranul principal.

Ultimul element este butonul roșu din partea de jos a ferestrei. Acesta odată apăsat va genera graficele cerute de utilizator, conform cerințelor specificate prin caseta de selectare și butonul de comutare.

Imaginea de mai sus este primul ecran afișat după executarea programului și reprezintă primul set de grafice. Partea de jos a ferestrei conține o serie de butoane în partea din stânga. Acestea permit mărirea imaginii, micșorarea ei, navigarea prin graficul afișat, modificarea modului de vizualizare, întoarcerea la modul inițial de vizualizare sau salvarea imaginii.

Acesta este cel de-al doilea set de date, afișat într-o fereastră diferită de primele seturi de date. Din nou se pot observa butoanele menționate în partea din stânga-jos, fiind posibile aceleași operațiuni ca pentru primul set de grafice.

 În cele din urmă, atunci când sunt colectate date despre autoturismele cerute de utilizator, va apărea un ecran în care putem observa modul în care programul colectează date despre marca cerută.

În momentul în care este apăsat butonul roșu din ecranul principal pe care scrie “lansează grafice”, în cazul în care nu sunt păstrate datele deja colectate, programul va începe procesul de adunare a datelor de pe site. Programul lansează automat un browser și accesează link-ul care duce către lista de mașini din marca selectată de utilizator. Se poate observa cum aplicația parcurge pe rând paginile cu anunțuri despre marcă. Mai apoi se accesează link-ul fiecărui anunț și se adună datele obținute de pe paginile fiecărui anunț. În cele din urmă informațiile se vor încărca într-un fișier intitulat “masini.xlsx”, unde se va crea automat o nouă foaie cu numele mașinii.

# Codul sursă

**gui.py:**

import tkinter as tk

import ttkbootstrap as tb

import services.autovit.principal as p

import pandas as pd

from services.autovit.analysis.grafice import start\_grafice

def ascunde\_butoane():

update\_toggle.forget()

buton\_lanseaza\_statistici.forget()

def afiseaza\_butoane():

update\_toggle.pack(pady=100)

buton\_lanseaza\_statistici.pack(pady=100)

def clicker():

# statistica = statistica\_marca\_auto()

if on\_off.get()==0:

up\_page\_label.config(text="Asteptati cat timp se incarca datele si graficele")

#label\_asteptare.pack(pady=100)

marca\_selectata = combo\_alege\_masini.get()

url\_masina\_ceruta = f"https://www.autovit.ro/autoturisme/{marca\_selectata}"

p.incarcare\_date\_masina\_selectata(url\_masina\_ceruta, marca\_selectata)

start\_grafice(marca\_selectata)

else:

marca\_selectata = combo\_alege\_masini.get()

try:

df = pd.read\_excel('masini.xlsx', f'{marca\_selectata}')

except:

up\_page\_label.config(text="Nu exista date deja salvate despre masina")

start\_grafice(marca\_selectata)

#creere fereastra

root = tb.Window(themename="cyborg")

root.title("Selectare configuratie statistici pentru masini")

root.geometry("1000x800")

#primul label

up\_page\_label = tb.Label(root, bootstyle="primary", text="Alege marca de masina pentru care ai vrea sa aflii informatii", font=("Arial", 18), anchor="center")

up\_page\_label.pack(fill="both")

#label\_asteptare = tb.Label(root, text="Asteptati cat timp se incarca datele si graficele", bootstyle = "inverse-primary")

#meniu alegere masina

marci\_masini = ["bentley", "opel", "ford", "audi", "bmw", "cupra", "tesla", "cadillac"]

combo\_alege\_masini = tb.Combobox(root, bootstyle="info", values=marci\_masini, style="")

combo\_alege\_masini.pack(pady=100)

combo\_alege\_masini.current(0)

#checkbox daca facem update la informatii sau le luam pe cele deja existente din tabel

on\_off = tb.IntVar()

update\_toggle = tb.Checkbutton(bootstyle="warning square-toggle",

text="Pastreaza datele deja colectate despre masinile din aceasta marca",

variable=on\_off,

onvalue=1,

offvalue=0)

update\_toggle.pack(pady=100)

#buton pentru lansarea statisticilor

buton\_lanseaza\_statistici = tb.Button(root, bootstyle="danger", text="creeaza statisticie", command=clicker)

buton\_lanseaza\_statistici.pack(pady=100)

root.mainloop()

**principal.py:**

import services.autovit.raw\_data as autovit\_raw\_data

import services.autovit.lista\_auto as lista\_containere\_autovit

from services.autovit.data\_from\_containers import date\_linkuri\_masini

import services.autovit.analysis.import\_export as impexp

from services.autovit.analysis.grafice import start\_grafice

def main():

date\_toate\_masinile = date\_linkuri\_masini("https://www.autovit.ro/autoturisme/anunt/cadillac-xt6-ID7HiDX2.html")

return 0

def mainy():

val="pret"

impexp.import\_from\_excel(val,'bentley')

return 0

def incarcare\_date\_masina\_selectata(base\_url\_autovit, marca\_selectata):

#base\_url\_autovit = "https://www.autovit.ro/autoturisme/bentley"

links\_masini\_autovit = parcurgere\_pagini\_site(base\_url\_autovit)

#links\_masini\_autovit=[]

#links\_masini\_autovit.append("https://www.autovit.ro/autoturisme/anunt/cadillac-xt6-ID7HiDX2.html")

date\_toate\_masinile = date\_linkuri\_masini(links\_masini\_autovit)

impexp.export\_to\_excel(date\_toate\_masinile, marca\_selectata)

#start\_grafice(marca\_selectata)

def mainx():

base\_url\_autovit = "https://www.autovit.ro/autoturisme/bentley"

links\_masini\_autovit = parcurgere\_pagini\_site(base\_url\_autovit)

#print(links\_masini\_autovit)

date\_toate\_masinile = date\_linkuri\_masini(links\_masini\_autovit)

#print(date\_toate\_masinile)

impexp.export\_to\_excel(date\_toate\_masinile,'bentley')

#date\_toate\_masinile = date\_linkuri\_masini(pagina)

#parcurgere\_o\_pagina(base\_url\_autovit)

return 0

def parcurgere\_o\_pagina(url):

pagina\_parsata = autovit\_raw\_data.extract(url, 3)

print(lista\_containere\_autovit.parse\_pagina\_auto(pagina\_parsata))

def parcurgere\_pagini\_site(base\_url):

nr\_pagina = 1

links\_from\_all\_pages = []

while True:

result\_page = autovit\_raw\_data.extract(base\_url, nr\_pagina)

if result\_page == None:

break

else:

# vector cu link de la masinie din pagina

# links\_from\_all\_pages = extend.#functie

links\_from\_all\_pages.extend(lista\_containere\_autovit.parse\_pagina\_auto(result\_page))

# adaug pagina la setul de pagini

nr\_pagina += 1

return links\_from\_all\_pages

if \_\_name\_\_ == "\_\_main\_\_":

main()

**lista\_auto.py:**

from selectolax.parser import HTMLParser

def parse\_pagina\_auto(page\_content):

results = []

htmlpage = HTMLParser(page\_content)

container\_masina = htmlpage.css("article.ooa-yca59n")

for item in container\_masina:

try:

link\_masina = item.css\_first("h1.e1i3khom9").css\_first("a").attributes["href"]

except AttributeError:

link\_masina = None

if link\_masina != None:

results.append(link\_masina)

return results

**raw\_data.py:**

from playwright.sync\_api import sync\_playwright

def extract(base\_url, page\_nr):

url = base\_url + "?page=" + str(page\_nr)

with sync\_playwright() as pw:

browser = pw.chromium.launch(headless=False)

page = browser.new\_page()

page.goto(url, wait\_until="networkidle")

if (page.url == url) | (page\_nr == 1):

return page.content()

else:

return None

**exctract\_css\_data.py:**

from selectolax.parser import HTMLParser

def extract\_text(html,sel):

try:

return html.css\_first(sel).text()

except AttributeError:

return None

**data\_from\_containers.py:**

from selectolax.parser import HTMLParser

from playwright.sync\_api import sync\_playwright

from services.autovit.extract\_css\_data import extract\_text

def date\_linkuri\_masini(links\_masini\_autovit):

date\_toate\_masinile=[]

for link in links\_masini\_autovit:

date\_o\_masina={}

with sync\_playwright() as pw:

browser=pw.chromium.launch(headless=False)

page = browser.new\_page()

page.goto(link, wait\_until="networkidle")

raw\_page=page.content()

parsed\_page=HTMLParser(raw\_page)

data=parsed\_page.css("div.ooa-162vy3d.e18eslyg3")

if data!=None:

date\_o\_masina['pret']=extract\_text(parsed\_page.css\_first("div.ooa-1xhj18k.eqdspoq2"), "h3")

date\_o\_masina['moneda']=extract\_text(parsed\_page.css\_first("div.ooa-1xhj18k.eqdspoq2"), "p")

for item in data:

#identificare taguri proprietati vehicul

#print(item.html)

tip\_camp\_item=extract\_text(item, "p.e18eslyg4")

match tip\_camp\_item:

case "Oferit de":

date\_o\_masina['tip\_vanzator']=extract\_text(item, "p.e16lfxpc0")

case "Model":

date\_o\_masina['model']=extract\_text(item, "a.e16lfxpc1")

case "Versiune":

date\_o\_masina['versiune']=extract\_text(item, "a.e16lfxpc1")

case "Anul producției":

date\_o\_masina['an\_productie']=extract\_text(item, "p.e16lfxpc0")

case "Km":

date\_o\_masina['kilometraj']=extract\_text(item, "p.e16lfxpc0")

case "Combustibil":

date\_o\_masina['combustibil']=extract\_text(item, "a.e16lfxpc1")

case "Putere":

date\_o\_masina['putere']=extract\_text(item, "p.e16lfxpc0")

case "Capacitate cilindrica":

date\_o\_masina['capacitate\_cilindrica']=extract\_text(item, "p.e16lfxpc0")

case "Cutie de viteze":

date\_o\_masina['cutie\_viteze']=extract\_text(item, "a.e16lfxpc1")

case "Consum Urban":

date\_o\_masina['consum\_urban']=extract\_text(item, "p.e16lfxpc0")

case "Culoare":

date\_o\_masina['culoare']=extract\_text(item, "a.e16lfxpc1")

case "Numar locuri":

date\_o\_masina['numar\_locuri']=extract\_text(item, "p.e16lfxpc0")

if(date\_o\_masina["pret"]!=None):

date\_toate\_masinile.append(date\_o\_masina)

return date\_toate\_masinile

**import\_export.py:**

import pandas as pd

def export\_to\_excel(date\_toate\_masinile, marca\_selectata):

df = pd.DataFrame(date\_toate\_masinile)

excel\_file = pd.ExcelFile('masini.xlsx')

sheet\_names=excel\_file.sheet\_names

replace=0

with pd.ExcelWriter('masini.xlsx', engine='openpyxl', mode='a') as writer:

workBook = writer.book

try:

workBook.remove(workBook[f'{marca\_selectata}'])

except:

print("Worksheet does not exist")

finally:

df.to\_excel(writer, sheet\_name=f'{marca\_selectata}')

def import\_from\_excel(valoare\_ceruta, marca\_selectata):

df = pd.read\_excel('masini.xlsx', f'{marca\_selectata}')

lista\_valori = df[f'{valoare\_ceruta}'].values.tolist()

return lista\_valori

**grafice.py:**

import pandas as pd

from matplotlib import pyplot as plt

from services.autovit.analysis.import\_export import import\_from\_excel

import pyautogui

import numpy as np

def screen\_size():

width, height= pyautogui.size()

return (width/100,height/100)

def start\_grafice(marca\_selectata):

df = pd.read\_excel('masini.xlsx', f'{marca\_selectata}')

w,h=screen\_size()

fig1, (g1, g2)= plt.subplots(nrows=1, ncols=2)

fig2, (pc1, pc2, pc3)=plt.subplots(nrows=1, ncols=3)

fig1.set\_size\_inches(w/2, h/2)

fig2.set\_size\_inches(w/2, h/2)

df['pret'] = df['pret'].apply(lambda x: int(x.replace(" ", "")))

df['kilometraj'] = df['kilometraj'].apply(lambda x: int(x.replace(" ","").replace("km","")))

df['tip\_vanzator'] = df['tip\_vanzator'].apply(lambda x: str(x.split(' ',1)[0]))

df['combustibil'] = df['combustibil'].apply(lambda x: str(x.split(' ',1)[0]))

df['cutie\_viteze'] = df['cutie\_viteze'].apply(lambda x: str(x.split(' ',1)[0]))

df = df.astype({"pret": 'int32', "kilometraj": 'int32', "tip\_vanzator":'string', "combustibil":'string', "cutie\_viteze":'string'})

pret\_model(df, g1)

kilometraj\_pret(df, g2)

fig1.set\_facecolor("#68A4A5")

fig1.tight\_layout()

fig1.show()

tip\_vanzator(df, pc1)

combustibil(df,pc2)

cutie\_viteze(df, pc3)

fig2.suptitle('Informatii despre tipurile de vanzatori, combustibilii folositi si cutia de viteze')

fig2.set\_facecolor("#b0a8b9")

fig2.tight\_layout()

fig2.show()

def pret\_model(df, g):

# df1=df.groupby(['model'])['pret'].mean().round(0)

df1=df.groupby(['model'])['pret'].mean().astype(int)

df1.plot(kind='bar', x='model', y='pret', color="#4C8055", ax=g, rot=30)

g.set\_facecolor("#D9DAD9")

g.set\_xlabel("Modelul masinii")

g.set\_ylabel("Pretul masinii")

g.set\_title("Pretul mediu dupa marca")

def kilometraj\_pret(df, g):

dfg=df.groupby(['kilometraj'])['pret'].mean().astype(int)

df1=dfg.reset\_index()

df1.plot(kind='line', x='kilometraj', y='pret', color="#4C8055", ax=g, rot=30)

g.set\_facecolor("#D9DAD9")

g.set\_xlabel("Kilometraj")

g.set\_ylabel("Pretul masinii")

g.set\_title("Pretul in functie de kilometraj")

suma\_preturi=0

count\_masini=0

for x in df['pret']:

suma\_preturi=suma\_preturi+x

count\_masini += 1

pret\_mediu\_masina=suma\_preturi/count\_masini

g.fill\_between(x=df1['kilometraj'], y1=df1['pret'], y2=pret\_mediu\_masina, where=(df1['pret']>pret\_mediu\_masina),

interpolate=True, color='#B25E57', alpha=0.25, label='pret peste medie')

g.fill\_between(x=df1['kilometraj'], y1=df1['pret'], y2=pret\_mediu\_masina, where=(df1['pret']<=pret\_mediu\_masina),

interpolate=True, color='#4C8055', alpha=0.25, label='pret sub medie')

g.axhline(y=pret\_mediu\_masina, color='#BF642C', linestyle='--', linewidth=1.35, label='Media pretului unei masini')

def tip\_vanzator(df, g):

dfg = df.groupby(['tip\_vanzator'])['pret'].count().astype(int)

df1 = dfg.reset\_index()

explode\_tip\_vanzator=[0.05 for x in df1['tip\_vanzator']]

labels\_tip\_vanzator=[x for x in df1['tip\_vanzator']]

colors\_tip\_vanzator=['#3C5B6F','#948979','#DFD0B8','#4b4453','#845ec2']

df1.plot(kind='pie', y='pret', ax=g, autopct='%1.0f%%', labels=labels\_tip\_vanzator, colors=colors\_tip\_vanzator,

shadow=True, explode=explode\_tip\_vanzator, wedgeprops={'edgecolor':'black'})

g.set\_ylabel("")

g.legend(loc='upper right', bbox\_to\_anchor=(1.15,1.15), borderaxespad=0)

def combustibil(df, g):

dfg = df.groupby(['combustibil'])['pret'].count().astype(int)

df1 = dfg.reset\_index()

explode\_combustibil=[0.05 for x in df1['combustibil']]

labels\_combustibil=[x for x in df1['combustibil']]

colors\_combustibil=['#3C5B6F','#948979','#DFD0B8','#4b4453','#845ec2']

df1.plot(kind='pie', y='pret', ax=g, autopct='%1.0f%%', labels=labels\_combustibil, colors=colors\_combustibil,

shadow=True, explode=explode\_combustibil, wedgeprops={'edgecolor':'black'})

g.set\_ylabel("")

g.legend(bbox\_to\_anchor=(1.15,1.15), loc='upper right', borderaxespad=0)

def cutie\_viteze(df, g):

dfg = df.groupby(df['cutie\_viteze'])['pret'].count().astype(int)

df1 = dfg.reset\_index()

explode\_cutie\_viteze=[0.05 for x in df1['cutie\_viteze']]

labels\_cutie\_viteze=[x for x in df1['cutie\_viteze']]

colors\_cutie\_viteze=['#3C5B6F','#948979','#DFD0B8','#4b4453','#845ec2']

df1.plot(kind='pie', y='pret', ax=g, autopct='%1.0f%%', labels=labels\_cutie\_viteze, colors=colors\_cutie\_viteze,

shadow=True, explode=explode\_cutie\_viteze, wedgeprops={'edgecolor':'black'})

g.set\_ylabel("")

g.legend(bbox\_to\_anchor=(1.15,1.15), loc='upper right', borderaxespad=0)

# Concluzie

Realizarea aplicației m-a ajutat în dezvoltarea cunoștințelor practice de programare, dar și în extinderea noțiunilor teoretice învățate pe parcursul liceului în orele de informatică. Prin îmbinarea noțiunilor deja învățate și necesitatea căutării de informații noi în scopul creării proiectului am observat un progres semnificativ în evoluția cunoștințelor mele în domeniu.

Totodată, prin folosirea unui limbaj de programare nou, python, am putut fundamenta noțiunile de C++ predate, folosindu-le într-un context diferit de cel academic. Am descoperit și aprofundat interoperabilitatea conceptelor învățate, precum funcția, variabila, lista, clasa, obiectul sau structura repetitivă prin realizarea unei aplicații cu utilitate practică. La acestea s-au adăugat noțiuni noi, precum elemente de sintaxă HTML și CSS, menționate în capitolul referitor la aspecte teoretice.

Programul creat are o utilitate practică extinsă, mai ales în contextul în care în România există o cerere consistentă pentru mașini la mâna a doua. Cum mașinile sunt o achiziție importantă pentru majoritatea persoanelor, informarea consumatorilor este esențială.

Concluzionând, mă bucur că pot contribui, prin aplicația realizată, la acest proces de informare a consumatorilor și că pot ajuta în timp ce la rândul meu îmi dezvolt cunoștințele în programare.

# Sitografie

<https://ro.wikipedia.org/wiki/Python>

<https://ro.wikipedia.org/wiki/HyperText_Markup_Language>

<https://ro.wikipedia.org/wiki/Cascading_Style_Sheets>

<https://www.w3schools.com/>

<https://stackoverflow.com/questions>

<https://selectolax.readthedocs.io/en/latest/parser.html>

<https://playwright.dev/python/docs/api/class-playwright>

<https://pandas.pydata.org/docs/>

<https://matplotlib.org/stable/api/pyplot_summary.html>

<https://docs.python.org/3/library/tk.html>

<https://ttkbootstrap.readthedocs.io/en/version-0.5/>

<https://pyautogui.readthedocs.io/en/latest/>

<https://realpython.com/>

<https://www.geeksforgeeks.org/python-programming-language/?ref=home-articlecards>

<https://saturncloud.io/blog/>

<https://www.pythonisti.ro/ds-03-data-science-introducere-in-pandas.php>