Konfigurator sprzętu PC

Skład: Robert Baca, Arkadiusz Bodziony, Wiktor Ciskał, Mikołaj Śnieżko

Projekt wykonany w semestrze letnim roku akademickiego 2023/2024

1 Wprowadzenie

Celem tego projektu jest automatyczne pobieranie i analiza danych dotyczących cen, wydajności oraz kompatybilności ze sobą komponentów komputerowych takich jak CPU, GPU, HDD, RAM oraz płyty główne. Program pobiera dane z różnych stron internetowych, przetwarza je i prezentuje w formie tekstowej, dzięki czemu pozwala użytkownikowi wybrać konkretne komponenty oraz poznać ogólną ocenę zestawienia.

2 Przygotowanie danych

Dane są pobierane z następujących stron:

- `https://www.cpubenchmark.net/cpu_value_available.html`
- `https://www.videocardbenchmark.net/gpu_value.html`
- `https://www.harddrivebenchmark.net/hdd_value.html`
- `https://www.memorybenchmark.net/popular.html`
- `https://versus.com/en/motherboard`
- `https://www.money.pl/pieniadze/nbp/srednie/`

3 Importowanie niezbędnych modułów

```
1 import requests
2 from bs4 import BeautifulSoup
3 import tkinter as tk
4 from tkinter import ttk
5 from PIL import Image, ImageTk
6 import io
```

Ta sekcja importuje wszystkie niezbędne biblioteki:

- 'requests' do wysyłania zapytań HTTP
- 'BeautifulSoup' do parsowania HTML
- 'tkinter' do tworzenia GUI
- 'PIL' (Pillow) do manipulacji obrazkami

4 Funkcja 'scrapedata'

```
1 def scrape_data(url):
2     response = requests.get(url)
3     soup = BeautifulSoup(response.text, 'html.parser')
4     return soup
```

Funkcja scrapedata jest odpowiedzialna za pobieranie i parsowanie danych HTML z podanego URL. Używa requests do pobrania treści strony i BeautifulSoup do parsowania HTML.

5 Pobieranie danych o CPU

```
# Zapytanie dla CPU
soup = scrape_data('https://www.cpubenchmark.net/cpu_value_available.html')
cpu = [i.text for i in soup.find_all('span', class_="prdname")]
cpu_prices = [float(i.text.replace('$', '').replace(',', '')) for i in soup.find_all(
    'span', class_="price-neww")]
cpu_scores = [round(float(i.text.replace(',', '.')) / max([float(i.text.replace(',',
    '.')) for i in soup.find_all('span', class_="mark-neww")]), 2) for i in soup.
    find_all('span', class_="mark-neww")]
cpu_prices = dict(zip(cpu, cpu_prices))
cpu_scores = dict(zip(cpu, cpu_scores))
```

Ta sekcja pobiera i przetwarza dane o procesorach (CPU) z serwisu cpubenchmark.net. Zapisuje nazwy, ceny i oceny procesorów w odpowiednich strukturach danych (listach i słownikach).

6 Pobieranie danych o GPU

```
# Zapytanie dla GPU
soup = scrape_data('https://www.videocardbenchmark.net/gpu_value.html')
gpu = [i.text for i in soup.find_all('span', class_="prdname")]
gpu_prices = [float(i.text.replace('$', '').replace(',', '').replace('*', '')) for i
    in soup.find_all('span', class_="price-neww")]
gpu_scores = [round(float(i.text.replace(',', '.')) / max([float(i.text.replace(',',
    '.')) for i in soup.find_all('span', class_="mark-neww")]), 2) for i in soup.
    find_all('span', class_="mark-neww")]
gpu_prices = dict(zip(gpu, gpu_prices))
gpu_scores = dict(zip(gpu, gpu_scores))
```

Ta sekcja działa analogicznie do poprzedniej, ale dotyczy kart graficznych (GPU). Pobiera i przetwarza dane o nazwach, cenach i ocenach GPU.

7 Pobieranie danych o HDD/SSD

```
# Zapytanie dla HDD
soup = scrape_data('https://www.harddrivebenchmark.net/hdd_value.html')
hdd = [i.text for i in soup.find_all('span', class_="prdname")]
hdd_prices = [float(i.text.replace('$', '').replace(',', '').replace('*', '')) for i
    in soup.find_all('span', class_="price-neww")]
hdd_scores = [round(float(i.text.replace(',', '.')) / max([float(i.text.replace(',',
    '.')) for i in soup.find_all('span', class_="mark-neww")]), 2) for i in soup.
    find_all('span', class_="mark-neww")]
hdd_prices = dict(zip(hdd, hdd_prices))
hdd_scores = dict(zip(hdd, hdd_scores))
```

Sekcja pobiera i przetwarza dane o dyskach twardych (HDD) oraz dyskach półprzewodnikowych (SSD), podobnie jak w przypadku CPU i GPU. Zawiera nazwy, ceny i oceny dysków.

8 Pobieranie danych o RAM

```
# Zapytania dla ram
soup = scrape_data('https://www.memorybenchmark.net/popular.html')  # Scrapowana strona
ram = soup.find_all('span', class_="prdname")
ram = [i.text for i in ram]  # zapisanie nazw ram do listy
ram_prices = soup.find_all('span', class_="price-neww")  # pobrane cen dla kazdego ramu
ram_prices = [float(i.text.replace('*', '').replace('NA', '0')) for i in ram_prices]
# oczyszczenie listy
ram_scores = soup.find_all('span', class_="count")  # pobrane danych dla ocen (przed konwersja)
ram_scores = [float(i.text.replace(' %', '')) for i in ram_scores]
# konwersja z procentow na wyniki
# Usunięcie rekordów z ceną równą zero i dostosowanie normalizacji
ram_data = list(zip(ram, ram_prices, ram_scores))
ram_data = [(name, price, score) for name, price, score in ram_data if price > 0]
# Obliczenie maksymalnej wartości ocen ze zmodyfikowanych danych - dla normalizacji
max_ram_score = max(ram_data, key=lambda x: x[2])[2]
# Normalizacja ocen RAM na podstawie maksymalnej wartości z danych po usunięciu cen r ównych zero
ram = [data[0] for data in ram_data]
ram_prices = {data[0]: data[1] for data in ram_data}
ram_scores = {data[0]: round(data[2] / max_ram_score, 2) for data in ram_data}
```

Ta sekcja pobiera i przetwarza dane o pamięciach RAM, w tym nazwy, ceny i oceny. Zawiera dodatkowe kroki przetwarzania, aby usunąć rekordy z ceną równą zero i znormalizować oceny.

9 Pobieranie danych o płytach głównych

```
# Zapytania dla płyt głównych
soup = scrape_data('https://versus.com/en/motherboard')
mother_board = soup.find_all('p', class_="BarsItem__name___3EC0w")
mother_board = [i.text for i in mother_board]
# Odnajdywanie cen płyt głównych i konwersja na pełne liczby
mother_board_prices = soup.find_all('div', class_="BarsItem__price___3dk0c")
mother_board_prices = [float(i.text[-5:].replace(',', '.')) * 100 if len(i.text) > 0 else None for i in mother_board_prices]
mother_board_scores = soup.find_all('span', class_="pointsText")
mother_board_scores = [float(i.text.replace('points', '')) for i in mother_board_scores]
mother_board_scores = [round(i/max(mother_board_scores), 2) for i in mother_board_scores]
mother_board_data = list(zip(mother_board, mother_board_prices, mother_board_scores))
mother_board_data = [(name, price, score) for name, price, score in mother_board_data]
mother_board = [data[0] for data in mother_board_data]
mother_board_prices = {data[0]: data[1] for data in mother_board_data}
mother_board_scores = {data[0]: data[2] for data in mother_board_data}
```

Ta sekcja pobiera i przetwarza dane o płytach głównych, w tym nazwy, ceny i oceny. Wartości cen są przeliczane na podstawie odpowiednich formatów.

10 Pobieranie danych walutowych

```
# Pobieranie kursów walut
soup = scrape_data('https://www.money.pl/pieniadze/nbp/srednie/')
site = soup.find_all('div', class_="rt-td")
currencies = []
values = []
for i in range(1, len(site), 5):
    currencies.append(site[i].text)
for i in range(2, len(site), 5):
    values.append(site[i].text)
values = [float(i.replace(',', '.')) for i in values]
currencies.insert(0, "PLN")
values.insert(0, 1)
exchange_rates = dict(zip(currencies, values))
```

Sekcja ta pobiera kursy walut z serwisu money.pl. Dodaje domyślny kurs dla PLN i tworzy słownik z kursami walut.

11 Funkcja 'calculate()'

```
# Funkcja obliczająca wyniki po naciśnięciu odpowiedniego przycisku w GUI
def calculate():
    try:
        selected_cpu = cpu_combobox.get()
        selected_gpu = gpu_combobox.get()
        selected_hdd = hdd_combobox.get()
        selected_ram = ram_combobox.get()
        selected_currency = currency_combobox.get()
        selected_mother = mother_board_combobox.get()
        exchange_rate = exchange_rates[selected_currency]

        total_price = round((cpu_prices[selected_cpu] + gpu_prices[selected_gpu] +
            hdd_prices[selected_hdd] +
                             ram_prices[selected_ram] + mother_board_prices[
                                 selected_mother])*exchange_rates["USD"], 2)
        total_price = round(total_price / exchange_rate, 2)
        total_score = round(cpu_scores[selected_cpu] + gpu_scores[selected_gpu] +
            hdd_scores[selected_hdd] +
                            ram_scores[selected_ram] + mother_board_scores[
                                selected_mother], 2)

        if total_score < 0.9:
            total_score_label.config(text=f"Łączny wynik: {total_score} (słaby)")
            total_score_label.config(foreground="red")
            img_url = "https://media.makeameme.org/created/oh-dude-thats-5c689f.jpg"
        elif 0.9 <= total_score < 1.15:
            total_score_label.config(text=f"Łączny wynik: {total_score} (średni)")
            total_score_label.config(foreground="yellow")
            img_url = "https://us-tuna-sounds-images.voicemod.net/64c2bcda-a203-47cd-
                a81a-68bf07397033-1701636231104.jpeg"
        else:
            total_score_label.config(text=f"Łączny wynik: {total_score} (świetny!)")
            total_score_label.config(foreground="green")
            img_url = "https://melmagazine.com/wp-content/uploads/2021/01/66f-1.jpg"

        total_price_label.config(text=f"Łączna cena: {total_price} {selected_currency
            }")

        response = requests.get(img_url)
        img_data = response.content
        img = Image.open(io.BytesIO(img_data))
        img = img.resize((150, 100), Image.LANCZOS)
        img = ImageTk.PhotoImage(img)
        image_label.config(image=img)
        image_label.image = img
        image_label.grid(row=7, column=1, rowspan=2)


    except KeyError:
        total_score_label.config(text="Wybierz każdy komponent!")
        total_price_label.config(text="")
        image_label.config(image='')
```

Funkcja calculate oblicza łączną cenę i wynik na podstawie wybranych komponentów oraz aktualnego kursu walut. Wyświetla wynik i obrazek w zależności od uzyskanego wyniku. W przypadku błędu wyświetla odpowiedni komunikat.

12 Funkcja 'showprices()'

```
def show_prices():
    selected_cpu = cpu_combobox.get()
    selected_gpu = gpu_combobox.get()
    selected_hdd = hdd_combobox.get()
    selected_ram = ram_combobox.get()
    selected_mother_board = mother_board_combobox.get()
    selected_currency = currency_combobox.get()
    exchange_rate = exchange_rates[selected_currency]

    cpu_price = round(cpu_prices[selected_cpu] * exchange_rates["USD"] /
        exchange_rate, 2)
    gpu_price = round(gpu_prices[selected_gpu] * exchange_rates["USD"] /
        exchange_rate, 2)
    hdd_price = round(hdd_prices[selected_hdd] * exchange_rates["USD"] /
        exchange_rate, 2)
    ram_price = round(ram_prices[selected_ram] * exchange_rates["USD"] /
        exchange_rate, 2)
    mother_board_price = round(mother_board_prices[selected_mother_board] *
        exchange_rates["USD"] / exchange_rate, 2)

    prices_cpu_label.config(text=f"Cena CPU: {cpu_price} {selected_currency}")
    prices_gpu_label.config(text=f"Cena GPU: {gpu_price} {selected_currency}")
    prices_hdd_label.config(text=f"Cena HDD: {hdd_price} {selected_currency}")
    prices_ram_label.config(text=f"Cena RAM: {ram_price} {selected_currency}")
    prices_mother_label.config(text=f"Cena płyty głównej: {mother_board_price} {
        selected_currency}")

    root.geometry(f"{window_width + 170}x{window_height}+{position_right - 170}+{
        position_top}")
```

Funkcja showprices wyświetla szczegółowe ceny dla każdego wybranego komponentu, przeliczone na wybraną walutę. Aktualizuje również rozmiar okna GUI.

13 Tworzenie GUI

```
root = tk.Tk()
root.title("Podsumowanie podzespołów")
root.configure(bg="#2e2e2e")

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

window_width = 500
window_height = 700
position_top = int(screen_height/2 - window_height/2)
position_right = int(screen_width/2 - window_width/2)
root.geometry(f"{window_width}x{window_height}+{position_right}+{position_top}")

style = ttk.Style()
style.configure("TLabel", background="#2e2e2e", foreground="white", font=("Arial",
    14))
style.configure("TCombobox", font=("Arial", 12))
style.configure("TButton", font=("Arial", 12, "bold"), background="black", foreground
    ="white")

style.map("TButton",
          foreground=[('!active', 'black'), ('pressed', 'white'), ('active', 'black')
              ],
          background=[('!active', 'black'), ('pressed', 'black'), ('active', 'white')
              ])

cpu.sort()
```

```
gpu.sort()
hdd.sort()
ram.sort()
mother_board.sort()

cpu_label = ttk.Label(root, text="Wybierz CPU:")
cpu_label.grid(row=0, column=0, padx=50, pady=10, sticky='w')
cpu_combobox = ttk.Combobox(root, values=cpu)
cpu_combobox.grid(row=0, column=1, padx=20, pady=10, sticky='e')

gpu_label = ttk.Label(root, text="Wybierz GPU:")
gpu_label.grid(row=1, column=0, padx=50, pady=10, sticky='w')
gpu_combobox = ttk.Combobox(root, values=gpu)
gpu_combobox.grid(row=1, column=1, padx=20, pady=10, sticky='e')

hdd_label = ttk.Label(root, text="Wybierz HDD/SSD:")
hdd_label.grid(row=2, column=0, padx=50, pady=10, sticky='w')
hdd_combobox = ttk.Combobox(root, values=hdd)
hdd_combobox.grid(row=2, column=1, padx=20, pady=10, sticky='e')

ram_label = ttk.Label(root, text="Wybierz RAM:")
ram_label.grid(row=3, column=0, padx=50, pady=10, sticky='w')
ram_combobox = ttk.Combobox(root, values=ram)
ram_combobox.grid(row=3, column=1, padx=20, pady=10, sticky='e')

mother_board_label = ttk.Label(root, text="Wybierz płytę główną:")
mother_board_label.grid(row=4, column=0, padx=50, pady=10, sticky='w')
mother_board_combobox = ttk.Combobox(root, values=mother_board)
mother_board_combobox.grid(row=4, column=1, padx=20, pady=10, sticky='e')

currency_label = ttk.Label(root, text="Wybierz walutę:")
currency_label.grid(row=5, column=0, padx=50, pady=10, sticky='w')
currency_combobox = ttk.Combobox(root, values=list(currencies))
currency_combobox.grid(row=5, column=1, padx=20, pady=10, sticky='e')
currency_combobox.current(0)

calculate_button = ttk.Button(root, text="Oblicz", command=calculate)
calculate_button.grid(row=6, column=0, columnspan=2, pady=20)

total_score_label = ttk.Label(root, text="Łączny wynik:")
total_score_label.grid(row=7, column=0, columnspan=1, padx=(80,0), pady=5, sticky='w'
    )

total_price_label = ttk.Label(root, text="Łączna cena:")
total_price_label.grid(row=8, column=0, columnspan=1, padx=(80,0), pady=5, sticky='w'
    )

price_button = ttk.Button(root, text="Pokaż ceny szczegółowe", command=show_prices)
price_button.grid(row=9, column=0, columnspan=2, pady=20)

prices_cpu_label = ttk.Label(root, text="")
prices_cpu_label.grid(row=10, column=0, columnspan=2, pady=5, padx=50, sticky='w')

prices_hdd_label = ttk.Label(root, text="")
prices_hdd_label.grid(row=10, column=1, columnspan=2, pady=5, padx=45, sticky='w')

prices_gpu_label = ttk.Label(root, text="")
prices_gpu_label.grid(row=11, column=0, columnspan=2, pady=5, padx=50, sticky='w')

prices_ram_label = ttk.Label(root, text="")
prices_ram_label.grid(row=11, column=1, columnspan=2, pady=5, padx=45, sticky='w')

prices_mother_label = ttk.Label(root, text="")
prices_mother_label.grid(row=12, column=1, columnspan=2, pady=5, padx=45, sticky='w')

image_label = ttk.Label(root)
```

```
image_label.grid(row=7, column=1, rowspan=2, padx=5, sticky='e')

wyjscie = tk.Button(root,
                    text='Wyjscie',
                    width=10,
                    bg='tomato',
                    command=root.destroy,
                    bd=4)
wyjscie.grid(column=1, row=13, padx=20, pady=20, sticky=tk.SE)

root.mainloop()
```

Ta sekcja tworzy graficzny interfejs użytkownika (GUI) przy użyciu tkinter. Składa się z etykiet, rozwijanych list (combobox), przycisków oraz etykiet do wyświetlania wyników i obrazków. GUI pozwala użytkownikowi wybrać komponenty, wyświetlić szczegółowe ceny oraz obliczyć łączną cenę i wynik.

14 Podsumowanie

Użyteczność programu

- **Dla entuzjastów komputerowych:** Umożliwia szybkie porównanie cen i wydajności różnych komponentów komputerowych.
- **Dla osób składających komputer:** Pomaga w wyborze najlepszych komponentów w ramach określonego budżetu i preferencji wydajnościowych.
- **Dla sprzedawców i doradców IT:** Może służyć jako narzędzie do szybkiego przedstawienia klientowi możliwych konfiguracji i ich kosztów w różnych walutach.