Generovanie jednoduchej aplikácie pomocou nástrojov umelej inteligencie

Generating a simple application using artificial intelligence tools

Pavol Sojka[1]

Abstrakt

Článok sa zameriava na generovanie kódu jednoduchej aplikácie pre zistenie aktuálnej teploty podľa zadaného mesta v Slovenskej republike. Kód je plne generovaný pomocou nástrojov umelej inteligencie, a to konkrétne Gemini, Grok a ChatGPT četbotov. Predpokladá sa, že v blízkej budúcnosti by tieto nástroje mohli do veľkej miery prevziať programovanie/generovanie kódu a teda tým významne znížiť potrebu ľudských programátorov. Cieľom nášho článku je preskúmať aktuálny stav na tomto poli a či sú tri rôzne četboty schopné vygenerovať bez dodatočnej asistencie samostatný funkčný kód pre aplikáciu, ktorá má iba jeden účel, a to zistenie počasia z domény shmu.sk. V jednotlivých kapitolách popisujeme vytvorenie testovacieho prostredia a nasadenie vygenerovaného kódu umelou inteligenciou a porovnáme ho s nasadením kódu autorov.

Kľúčové slová

umelá inteligencia, Python, automatizácia, programovanie

Abstract

The article focuses on generating the code of a simple application for finding the current temperature according to a specified city in the Slovak Republic. The code is fully generated using artificial intelligence tools, namely Gemini, Grok and ChatGPT chatbots. It is assumed that in the near future these tools could largely take over programming/code generation and thus significantly reduce the need for human programmers. The aim of our article is to examine the current state of the art in this field and whether three different chatbots can generate, without additional assistance, a separate functional code for an application that has only one purpose, namely finding the weather from the shmu.sk domain. In individual chapters, we describe the creation of a test environment and the deployment of the code generated by artificial intelligence and compare it with the deployment of the authors' code.

Key words

artificial intelligence, Python, automation, programming

JEL classification

C8

1 Úvod

V posledných rokoch dochádza k výraznému pokroku v oblasti umelej inteligencie, pričom strojové učenie a hlboké učenie umožňujú riešiť čoraz komplexnejšie úlohy. Napriek tomu zostávajú oblasti, v ktorých ľudské programovanie vykazuje vyššiu efektivitu, najmä pri špecifických požiadavkách, ako je potreba vysokej interpretovateľnosti, adaptácie na neštandardné scenáre alebo prípadných etických obmedzení. Cieľom nášho článku je jednoduché porovnanie programovania umelej inteligencie a tradičného ľudského prístupu pri riešení špecializovanej úlohy, ktorou je príprava

[1] Ekonomická univerzita v Bratislave, Fakulta hospodárskej informatiky, Katedra aplikovanej informatiky, Dolnozemská cesta 1, 852 35 Bratislava, pavol.sojka@euba.sk.

aplikácie pre zistenie aktuálnej teploty zo stránok Slovenského hydrometeorologického ústavu (www.shmu.sk).

2 Príprava prostredia

Ako primárny operačný systém, na ktorom budú vykonávané experimenty, sme si vybrali na základe rešerše Linux Ubuntu 22.04 (Petersen, 2025). Naše prostredie bolo umiestnené na platforme Google Cloud na virtuálnom stroji s hardvérovými parametrami zahŕňajúcimi štyri CPU (centrálna procesorová jednotka, AuthenticAMD, AMD EPYC 7B12) a štyroch gigabajtoch RAM (operačná pamäť). Počet CPU a veľkosť operačnej pamäte sú na aktuálny stav vývoja oblasti hardvéru na nízkej úrovni, ale pre vykonateľnosť stanoveného experimentu sú plne dostačujúce.

Po úspešnom nasadení operačného systému sme nainštalovali jazyk Python (verzia Python 3.10.12) a manažér balíkov Pip (verzia 22.0.2) z repozitárov operačného systému. Spolu s Pythonom sme pomocou nástroja Pip nainštalovali aj odporúčané knižnice, ktoré boli potrené pre beh všetkých skriptov (Lubanovic, 2019).

Naším cieľom bolo otestovať rýchlosť nášho naprogramovaného kódu v týchto krokoch:

- Vygenerovať Python kód pomocou nástrojov umelej inteligencie
- Otestovať vygenerovaný kód v praktickom nasadení
- Realizovať experiment pomocou nástroja naprogramovaného autormi
- Porovnať dosiahnuté výsledky a interpretovať ich

Autori do kódu vygenerovaného umelou inteligenciou vôbec nezasahovali, ale otestovali ho tak, ako bol vygenerovaný. Vzhľadom na relatívnu jednoduchosť úlohy bol na začiatku experimentu predpoklad, že kód bude plne funkčný, čo sa ale neskôr nepotvrdilo.

3 Výsledky

Každý kód bol spustený trikrát, aby boli obdržané výsledky verifikované. Kódy generované umelou inteligenciou boli prevzaté z četbotov ChatGPT, Gemini a Grok. Náhľady celých kódov spolu s výsledkami sú uvedené v nasledujúcich tabuľkách. Vygenerovaný kód bol inicializovaný otázkou: „*ahoj, vies urobit aplikaciu v pythone, ktora po zadani mesta vypise teplotu v stupnoch C zo stranky shmu.sk?*“. Všetky četboty otázke porozumeli aj napriek tomu, že bola položená bez použitia diakritiky, čo bolo aj zámerom na otestovanie, či umelá inteligencia rozumie aj takto položenému dopytu.

Poznámka: Výsledné ukážkové kódy môžu vyžadovať dodatočnú úpravu formou odrážok (tabulátorov) pre správne fungovanie, pretože mohli byť automaticky upravené textovým/grafickým softvérom. Kódy boli skopírované v pôvodnej podobe tak, ako boli vygenerované umelou inteligenciou.

Tab. 1: Kód generovaný chatGPT (verzia: GPT-4o)

```
import requests from bs4 import BeautifulSoup

def ziskaj_teplotu(mesto): # Mapovanie názvu mesta na URL parameter mesta = {
'bratislava': 'http://www.shmu.sk/sk/?page=1&id=meteo_mestske&station=BA',
'kosice': 'http://www.shmu.sk/sk/?page=1&id=meteo_mestske&station=KE', 'zilina':
'http://www.shmu.sk/sk/?page=1&id=meteo_mestske&station=ZA', # pridaj ďalšie
mestá podľa SHMÚ }

mesto_url = mesta.get(mesto.lower())
if not mesto_url:
    print(f"Mesto '{mesto}' nie je podporované.")
    return

response = requests.get(mesto_url)
if response.status_code != 200:
    print("Nepodarilo sa načítať dáta zo SHMÚ.")
    return

soup = BeautifulSoup(response.text, 'html.parser')

# Hľadanie elementu s teplotou - môže sa zmeniť podľa aktuálnej HTML štruktúry
teplota_element = soup.find('td', class_='td_temp')  # Pozor: treba overiť
aktuálnu triedu!

if teplota_element:
    print(f"Aktuálna teplota v {mesto.title()}: {teplota_element.text.strip()}
°C")
else:
    print("Teplotu sa nepodarilo nájsť.")


    1  #Príklad použitia

mesto = input("Zadaj mesto (napr. Bratislava, Kosice): ") ziskaj_teplotu(mesto)
```

Zdroj: vlastné spracovanie

Výsledok:
user@linux-ai-shmu:~/ai_shmu/chatGPT$ python3 shmu.py
Zadaj mesto (napr. Bratislava, Kosice): Bratislava
Teplotu sa nepodarilo nájsť.

Tab. 2: Kód generovaný Gemini (verzia: 2.5 Flash)

```
import requests from bs4 import BeautifulSoup

def
ziskaj_teploty_zo_shmu(url="https://www.shmu.sk/sk/?page=1&id=meteo_apocasie_sk"
): """ Získa aktuálne teploty zo stránky SHMÚ pomocou web scrapingu.

Args:
    url (str): URL adresa SHMÚ stránky s aktuálnym počasím.

Returns:
    dict: Slovník, kde kľúčom je názov stanice a hodnotou je teplota v °C.
          Vráti prázdny slovník, ak sa dáta nepodarí získať.
"""
teploty = {}
try:
    # Stiahnutie obsahu stránky
    response = requests.get(url, timeout=10) # Nastavenie timeoutu pre prípad,
že stránka neodpovedá
    response.raise_for_status() # Vyvolá HTTPError pre chybné status kódy (4xx,
5xx)

    # Parsovanie HTML obsahu
    soup = BeautifulSoup(response.text, 'html.parser')

    # SHMÚ zvyčajne zobrazuje dáta v tabuľkách.
    # Skúsime nájsť tabuľku s ID 'data-tabulka' alebo triedou, ktorá obsahuje
meteo dáta.
    # Táto časť kódu je najcitlivejšia na zmeny štruktúry SHMÚ stránky.
    tabulka = soup.find('table', {'class': 'dm_tabulka', 'id': 'data-tabulka'})

    if tabulka:
        # Prejdeme cez riadky tabuľky (tr)
        for riadok in tabulka.find_all('tr'):
            # Nájdenie všetkých dátových buniek (td) v riadku
            bunky = riadok.find_all('td')
            if len(bunky) >= 3: # Potrebujeme aspoň 3 bunky (stanica, teplota,
čas/ďalšie info)
                stanica = bunky[0].get_text(strip=True)
                teplota_str = bunky[2].get_text(strip=True) # Teplota je často v
treťom stĺpci

                try:
                    # Pokúsime sa prekonvertovať teplotu na float
                    teplota = float(teplota_str.replace(',', '.').replace(' °C',
''))
                    teploty[stanica] = teplota
                except ValueError:
                    # Ak sa nepodarí prekonvertovať, preskočíme
                    continue
    else:
        print("Chyba: Nenašla sa tabuľka s aktuálnymi dátami na stránke SHMÚ.")

except requests.exceptions.RequestException as e:
    print(f"Chyba pri sťahovaní stránky: {e}")
except Exception as e:
    print(f"Nastala neočakávaná chyba: {e}")
```

```
return teploty

if name == "main": aktualne_teploty = ziskaj_teploty_zo_shmu()

      if aktualne_teploty:
          print("Aktuálne teploty zo SHMÚ:")
          for stanica, temp in aktualne_teploty.items():
              print(f"- {stanica}: {temp}°C")
      else:
          print("Nepodarilo sa získať údaje o teplotách zo SHMÚ.")

      print("\n---")
      print("Pre konkrétne mesto musíte prejsť cez získaný slovník a nájsť
      zhodu.")
      print("Napríklad pre Bratislavu:")

      hladane_mesto = input("Zadajte mesto, pre ktoré chcete nájsť teplotu
      (napr. Bratislava): ")
      found = False
      for stanica, temp in aktualne_teploty.items():
          if hladane_mesto.lower() in stanica.lower():
              print(f"- {stanica}: {temp}°C")
              found = True
      if not found:
          print(f"Pre mesto '{hladane_mesto}' sa nenašli žiadne záznamy.")
```

Zdroj: vlastné spracovanie

Výsledok:

user@linux-ai-shmu:~/ai_shmu/gemini$ python3 shmu.py
Chyba: Nenašla sa tabuľka s aktuálnymi dátami na stránke SHMÚ.
Nepodarilo sa získať údaje o teplotách zo SHMÚ.

Pre konkrétne mesto musíte prejsť cez získaný slovník a nájsť zhodu. Napríklad pre Bratislavu: Zadajte mesto, pre ktoré chcete nájsť teplotu (napr. Bratislava): Bratislava Pre mesto 'Bratislava' sa nenašli žiadne záznamy.

Tab. 3: Kód generovaný Grok (verzia: Grok 3)

```
import requests from bs4 import BeautifulSoup import re

def get_temperature(city): # Normalizácia názvu mesta (odstránenie diakritiky
pre URL) city = city.lower().strip() # Nahrádzame diakritiku pre jednoduchšie
vyhľadávanie diacritics = { 'á': 'a', 'ä': 'a', 'č': 'c', 'ď': 'd', 'é': 'e',
'í': 'i', 'ľ': 'l', 'ň': 'n', 'ó': 'o', 'ô': 'o', 'ŕ': 'r', 'š': 's', 'ť': 't',
'ú': 'u', 'ý': 'y', 'ž': 'z' } for char, repl in diacritics.items(): city =
city.replace(char, repl)

# Vytvorenie URL pre vyhľadávanie na SHMÚ
base_url = "https://www.shmu.sk/sk/?page=1&subpage=3&name="
url = base_url + city

try:
```

```
    # Odoslanie HTTP požiadavky
    response = requests.get(url, headers={'User-Agent': 'Mozilla/5.0'})
    response.raise_for_status()  # Overenie, či požiadavka bola úspešná

    # Parsovanie HTML obsahu
    soup = BeautifulSoup(response.text, 'html.parser')

    # Hľadanie elementu s teplotou (napr. v tabuľke alebo meteograme)
    # SHMÚ zvyčajne zobrazuje teplotu v div s triedou 'actual' alebo podobnej
    temp_element = soup.find('div', class_='actual')
    if temp_element:
        # Extrahovanie textu s teplotou
        temp_text = temp_element.get_text(strip=True)
        # Predpokladáme formát ako "12 °C" alebo "12°C"
        match = re.search(r'(\d+\.?\d*)\s*°C', temp_text)
        if match:
            temperature = match.group(1)
            return f"Aktuálna teplota v meste {city.title()} je {temperature}
°C"
        else:
            return f"Teplota pre mesto {city.title()} nebola nájdená."
    else:
        return f"Údaje o teplote pre mesto {city.title()} neboli nájdené."

except requests.exceptions.RequestException as e:
    return f"Chyba pri získavaní údajov: {e}"

def main():
    while True:
        city = input("Zadaj názov mesta (alebo 'koniec' pre ukončenie): ")
        if city.lower() == 'koniec':
            print("Ukončujem program.")
            break
        if not city.strip():
            print("Prosím, zadaj platný názov mesta.")
            continue
        result = get_temperature(city)
        print(result)
if name == "main":
       main()
```

Zdroj: vlastné spracovanie

Výsledok:

user@linux-ai-shmu:~/ai_shmu/grok$ python3 shmu.py
Zadaj názov mesta (alebo 'koniec' pre ukončenie): Bratislava
Údaje o teplote pre mesto Bratislava neboli nájdené.
Zadaj názov mesta (alebo 'koniec' pre ukončenie):

Nasleduje kód autorov, ktorý je krátený (Tab. 4) a uvedený iba ako ukážka, nakoľko do budúcna je záujem autorov zistiť, či umelá inteligencia zlepšila schopnosť vytvoriť kód pre tento typ aplikácie tak, aby bol funkčný. Autori preto nechcú vytvoriť návod pre umelú inteligenciu, ako kód naprogramovať. Ak si chce čitateľ overiť funkčnosť kódu autorov, tak na poskytnutom odkaze je možné overiť si, či kód vracia správne dáta v reálnom čase (ak nebola medzitým funkčne pozmenená hlavná stránka SHMÚ, tak kód funguje správne) http://rsrch.ddns.net/rsrch/shmu/ a porovnať výsledné dáta so stránkou www.shmu.sk – údaje o teplote v meste Bratislava (alebo v inom zvolenom meste podľa mapy na úvodnej stránke SHMÚ) sa musia zhodovať.

Tab. 4: Stručný náhľad na kód autorov

```
import requests
from bs4 import BeautifulSoup
import csv
def scrape_page(soup, quotes):
    cdata = soup.find(string=re.compile("CDATA"))
    import string
    cdata2=cdata.split(",")
   global m
    m = "Bratislava"
    m = input("Zadaj mesto (default=%s): " % m) or m
    found=False
    for mesto in cdata2:
          if m.capitalize() in mesto:
                quotes.append(mesto)
                found=True
                break
   if not found:
          print("Mesto nenajdene")
```

Zdroj: vlastné spracovanie

Výsledok:

user@linux-ai-shmu:~/ai_shmu/moj_kod$ python3 shmu.py
Zadaj mesto (default=Bratislava):
Teplota pre mesto Bratislava je 22.5 °C

Z uvedených príkladov vidíme, že ani jeden z použitých četbotov nebol schopný vygenerovať požadovaný kód vracajúci správny výsledok, hoci údaje sa nachádzajú na hlavnej stránke www.shmu.sk a netreba sa preklikávať hlbšie do kódu podstránok. Autor pripravil pomocou jazyka Python kód, ktorý rovnako ako kódy umelej inteligencie, stiahol úvodnú web stránku shmu.sk do lokálneho počítača a potom analýzou získaných dát zistil príslušnú teplotu v danom meste. Z tohto experimentu vyplýva, že umelá inteligencia síce vie riešiť rôzne komplexné úlohy, keď je implementovaných mnoho skúmaných existujúcich riešení, z ktorých sa učí, ale niekedy aj banálne zadanie (ako v tomto článku) vedie k nezdaru pri všetkých skúmaných četbotoch.

4 Diskusia

Verzie četbotov boli testované v období jún-júl 2025 a teda do budúcna by bolo vhodné test zopakovať pre novšie verzie týchto nástrojov. Je predpoklad, že v ďalších rokoch umelá inteligencia pokročí do takej úrovne, že tento jednoduchý test bude úspešne zvládnutý a rovnako aj zložitejšie implementácie iných aplikácií. Test na stránke SHMÚ s autorovou verziou kódu bol realizovaný v rovnakom období ako pri nástrojoch umelej inteligencie, a to v mesiacoch jún-júl 2025. Autorský kód nemusí v budúcich testoch fungovať, nakoľko je založený na technológii tzv. Webscrapingu (Clark, 2025), kde je dôležité prehliadať štruktúry

hlavnej web stránky SHMÚ. Ak sa tieto štruktúry zmenia, napríklad sa výrazne zmení vzhľad hlavnej stránky, tak aplikácia môže prestať úplne fungovať. Tento prístup (webscraping) je len čisto experimentálny a určený len pre výskumné účely, nakoľko môže vyvolávať otázky o legálnosti takto získaných dát.

5 Záver

V článku sme popísali postup, ako sme navrhli experiment na získavanie relevantných dát z hlavnej stránky SHMÚ pomocou technológie webscrapingu implementovanej autormi článku a následne sme otestovali aplikácie, ktoré vygenerovali tri rôzne četboty na báze technológie LLM (Large Language Model), ktoré mali naprogramovať aplikáciu na základe zadaných parametrov (Campesato, 2024). Experiment ukázal, že aktuálne verzie četbotov túto úlohu nezvládli a ani jeden z nich nevedel poskytnúť požadované údaje zo stánky SHMÚ.

Literatúra

1. Campesato, O. (2024*). Large Language Models for Developers: A Prompt-based Exploration of LLMs*. United States: De Gruyter.
2. Clark, W. (2025). *Web Scraping with Python Step by Step: A Practical Guide with Examples*. (n.p.): Walzone Press.
3. Lubanovic, B. (2019). *Introducing Python: Modern Computing in Simple Packages*. United States: O'Reilly Media.
4. Petersen, R. (2025). *Ubuntu 24.04 LTS Server: Administration and Reference*. (n.p.): Surfing Turtle Press.