Valuta- és kriptovaluta árfolyam-követő

Python demo projekt

Gaál Márk

Tartalom

1. Abstract

This project implements a currency and cryptocurrency rate tracker designed to periodically fetch, store, and update exchange rate data in a MySQL database. The application utilizes public APIs to gather real-time foreign exchange rates and cryptocurrency values. Exchange rates are retrieved from the Open Exchange Rates API, while cryptocurrency values are sourced from the CoinGecko API.

The system automatically organizes and stores the fetched data into dynamically created MySQL tables, ensuring seamless scalability for various currencies. Each table stores currency symbols, exchange rates, and corresponding timestamps, providing a robust dataset for future analysis and reference.

A scheduler built into the script ensures the application runs daily at midnight to fetch the latest rates. The script includes error handling to manage connectivity issues or API failures, ensuring reliability. The stored data can be used for financial analysis, trend prediction, or integration into larger applications requiring exchange rate information.

This project demonstrates the integration of APIs, Python, and MySQL to create an automated and efficient solution for real-time financial data tracking.

2. Kivonat

Ez a projekt egy valuta- és kriptovaluta-árfolyamkövetőt valósít meg, amely rendszeresen lekéri, tárolja és frissíti az árfolyam adatokat egy MySQL-adatbázisban. Az alkalmazás nyilvános API-kat használ a valós idejű devizaárfolyamok és kriptovaluta-értékek gyűjtésére. A devizaárfolyamokat az Open Exchange Rates API szolgáltatja, míg a kriptovaluta-értékeket a CoinGecko API biztosítja.

A rendszer automatikusan rendezi és tárolja a lekért adatokat dinamikusan létrehozott MySQL-táblákban, biztosítva a zökkenőmentes skálázhatóságot különféle pénznemekhez. Minden tábla tárolja a pénznem szimbólumát, az árfolyamot és az időbélyeget, így egy megbízható adathalmazt biztosít jövőbeli elemzésekhez és referencia célokra.

A szkriptbe épített ütemező gondoskodik arról, hogy az alkalmazás naponta éjfélkor lefusson, és lekérje a legfrissebb árfolyamokat. A szkript hibakezelést is tartalmaz, hogy kezelje a csatlakozási problémákat vagy az API-k hibáit, ezáltal megbízható működést garantál. A tárolt adatok felhasználhatók pénzügyi elemzésekhez, trendelőrejelzéshez, vagy más alkalmazásokba való integráláshoz, amelyek árfolyam-információkat igényelnek.

Ez a projekt bemutatja az API-k, a Python és az MySQL integrációját, hogy egy automatizált és hatékony megoldást hozzon létre a valós idejű pénzügyi adatkövetéshez.

3. Bevezetés

A globális pénzügyi piacok dinamikus változásai és a digitális valuták térnyerése szükségessé tették az árfolyamok folyamatos figyelemmel kísérését. Ez a projekt egy automatizált valuta- és kriptovaluta-árfolyamfigyelő rendszer kialakítását célozza meg, amely valós idejű adatokat gyűjt és tárol egy MySQL adatbázisban. Az alkalmazás két fő adattípust követ: a hagyományos pénznemek árfolyamait az Open Exchange Rates API segítségével, valamint a legnépszerűbb kriptovaluták értékeit a CoinGecko API használatával.

A rendszer célja, hogy pontos, naprakész információkat nyújtson a felhasználók számára, amelyek felhasználhatók pénzügyi elemzésekhez, döntéshozatalhoz vagy egyéb fejlesztési projektekhez. Az alkalmazás automatizált működésének köszönhetően naponta egyszer, éjfélkor frissíti az adatokat, amelyeket struktúrált formában ment el az adatbázisba. Ezzel biztosítja az adatok könnyű elérését és hosszú távú elemzési lehetőségeket.

Ez a projekt nemcsak az API-k és adatbázisok integrációját demonstrálja, hanem egy hatékony és skálázható megoldást kínál a valós idejű pénzügyi adatkövetésra, míg számot tudok adni tudásomról, és munkámról Önök számára. Az egyszerű használat és a megbízhatóság érdekében a szkript beépített hibakezelést és dinamikus adattáblakezelést alkalmaz, amely támogatja a folyamatos és problémamentes működést.

A projekt a következő github repositoryban megtalálható: markgaal068/Currency-Tracker-Python: Python demo project

4. Projekt bemutatása

4.1 Architektúra

A Valuta- és Kriptovaluta-árfolyamfigyelő program architektúrája moduláris felépítésű, amely biztosítja a rendszer egyszerű bővíthetőségét és átláthatóságát. A rendszer három fő rétegre oszlik: az API réteg, az adatfeldolgozó réteg és az adatbázis réteg.

Az API réteg felelős a külső adatforrásokból, például az Open Exchange Rates API-ból és a CoinGecko API-ból származó valós idejű adatok begyűjtéséért. Az API-hívások HTTP protokollal valósulnak meg a Python requests könyvtár segítségével, és az adatok JSON formátumban érkeznek, amelyeket a rendszer feldolgoz. Az API réteg biztosítja a devizaárfolyamok és kriptovalutaárfolyamok pontos és naprakész lekérdezését.

Az adatfeldolgozó réteg a begyűjtött nyers adatokat előkészíti az adatbázisban történő tárolásra. Az árfolyamokat rendezett struktúrában dolgozza fel, pénznem (pl. USD, EUR, BTC, ETH), árfolyamérték és időbélyeg szerint. Ez a réteg tartalmazza a hibaellenőrzési mechanizmusokat is, amelyek kezelik az esetleges API-hibákat, például hálózati problémákat vagy érvénytelen válaszokat.

Az adatbázis réteg a tartós adattárolásért felelős, és egy MySQL adatbázist használ erre a célra. A rendszer dinamikusan hozza létre a táblákat: minden pénznemhez külön táblát (pl. exchange_usd, exchange_eur), míg a kriptovaluta árfolyamok számára egyetlen táblát (crypto_rates). Az adatokat SQL INSERT utasításokkal menti a megfelelő táblákba, és a mysql.connector könyvtárat használja az adatbázis-kezeléshez. A dinamikus tábla- és adattárolás biztosítja a skálázhatóságot, valamint a különböző devizák és kriptovaluták egyszerű kezelését.

Az egyes rétegek közötti integráció révén a rendszer valós idejű adatokat gyűjt, dolgoz fel és tárol. Ez az architektúra lehetővé teszi a megbízható, automatizált működést.

4.2 Futtatás és automatizálás: Időzítés

A Valuta- és Kriptovaluta-árfolyamfigyelő program egy önállóan futtatható Python-szkript formájában valósul meg, amely automatikusan, de előre meghatározott időpontban végezze el az árfolyamok lekérdezését és adatbázisba mentését. Az automatizálás biztosítja, hogy a rendszer emberi beavatkozások nélkül, folyamatosan friss adatokat tudjon tárolni.

4.2.1. Futtatási folyamat

A program indítása egyszerűen a Python-szkript futtatásával történik (python main.py). A futás során a program először kapcsolódik az API-khoz, majd az aktuális deviza- és kriptovaluta árfolyamokat lekéri. A begyűjtött adatokat feldolgozza, ellenőrzi, és elmenti a megfelelő MySQL táblákba. A futás végeztével az adatbázis a friss adatokkal bővül.

4.2.2. Automatizálás és időzítés

A program időzítéséért egy beépített időzítő funkció felel, amely a Python time és datetime könyvtárain alapul. Az időzítés úgy van beállítva, hogy a program minden nap éjfélkor automatikusan lefusson. A futtatási időpontot a kód következő részlete határozza meg:

```
if current_time.hour == 0 and current_time.minute == 0:
    run_tracker()
    time.sleep(86400)
```

Ez a logika biztosítja, hogy a program minden nap pontosan egyszer hajtsa végre a teljes adatgyűjtési és tárolási folyamatot. Az időzített működés miatt a programot elegendő egyszer elindítani, és az folyamatosan fut a háttérben. Fontos, hogy az eszköz amin a program fut, nem állhat le, mert akkor nem fog lefordulni a kód.

4.2.3. Hibaelhárítás és újra próbálkozás

A rendszer tartalmaz egy automatikus újrapróbálkozási mechanizmust is. Ha az időzített futás során az API-hívások nem járnak sikerrel (például hálózati hiba miatt), a program 30 másodpercenként újrapróbálja az adatgyűjtést. Ez biztosítja, hogy az időzítés ellenére se maradjon el az adatfrissítés, és a rendszer megbízhatóan működjön:

```
else:
    time.sleep(30)
```

4.2.4. Automatikus működés integrálása

A futtatást és időzítést szükség esetén külső eszközökkel, például Linux esetén cron vagy Windows esetén Task Scheduler segítségével is lehet automatizálni, ha a szkriptet rendszeres időközönként különálló futtatásokkal szeretnénk végrehajtani. Példa cron bejegyzésre:

```
0 0 * * * python /path/to/main.py
```

4.2.5. Előnyök

Az automatizált futtatás és időzítés révén a rendszer emberi beavatkozás nélkül is folyamatosan működik, biztosítva a naprakész adatgyűjtést. Ez különösen hasznos olyan helyzetekben, ahol a valós idejű árfolyamok követése vagy azok rendszeres frissítése elengedhetetlen. Az újrapróbálkozás és hibakezelés további megbízhatóságot biztosít.

4.2.6. Hátrányok és nehézségek

Bár a Valuta- és Kriptovaluta-árfolyamfigyelő „rendszer” számos előnnyel rendelkezik, bizonyos hátrányokat is figyelembe kell venni a használata és karbantartása, esetleges bővítése, vagy integrálása során. Ezek az aspektusok különösen akkor jelentkezhetnek, ha a rendszert nagyobb léptékben vagy speciális környezetekben kívánjuk alkalmazni.

4.2.7. API-korlátozások

A rendszer függ az API-k elérhetőségétől és stabilitásától. Ha az árfolyamadatokat szolgáltató API-k, jelen esetben Open Exchange Rates vagy CoinGecko ideiglenesen elérhetetlenné válnak, a program nem tud adatokat gyűjteni, és az adatbázis frissítése elmarad. Továbbá, sok API használata korlátozott, például napi hívások maximális száma vagy előfizetési költségek formájában, ami hosszú távon növelheti a fenntartási költségeket.

4.2.8. Időzítés és folyamatos futás

A program folyamatos futása magasabb erőforrás-használattal járhat, különösen olyan rendszereken, ahol korlátozott számú futó szál vagy alacsony energiahatékonyságú processzor áll rendelkezésre. Az időzítési logika jelenlegi implementációja nem

alkalmaz külső időzítő szolgáltatást, így a Python-szkript folyamatos futása szükséges, ami hosszú távon terhelheti a rendszert. Ez problémát jelenthet alacsony teljesítményű vagy erőforrás-korlátozott környezetekben.

4.2.9. Adatbázis-kezelés és skálázás

Noha a program képes dinamikusan létrehozni táblákat és kezelni az adatokat, a nagy mennyiségű adat hosszú távon problémákhoz vezethet. Az adatbázis növekedése a tárolási költségek növekedésével, valamint az adatok elérésének lassulásával járhat. Továbbá a rendszer jelenlegi formájában nem rendelkezik adatarchiválási vagy régebbi adatok törlésére vonatkozó funkcióval, ami idővel redundanciához vezethet.

4.2.10. Biztonsági aggályok

Az API-kból érkező adatok kezelése során figyelmet kell fordítani az adatok integritására és biztonságára. Ha az API-k válasza manipulált vagy hibás adatokat tartalmaz, azok az adatbázisba kerülve megbízhatatlan eredményekhez vezethetnek. Továbbá a MySQL-adatbázis védelme is kiemelt fontosságú, mivel a nem megfelelő konfiguráció potenciális adatbiztonsági rést jelenthet.

4.2.11. Fenntarthatóság

A program hosszú távú fenntartása fejlesztői erőforrásokat igényelhet. Az API-változások, új pénznemek vagy kriptovaluták hozzáadása, valamint a technológiai környezet változása (például Python vagy MySQL verziófrissítések) mind további munkát jelenthetnek. Ezek megoldására a kód folyamatos karbantartása szükséges, ami erőforrásokat emészthet fel.

5. Programkód

5.1 Könyvtárak és szerepük

5.1.1 Requests

A requests egy rendkívül népszerű Python-könyvtár, amely egyszerűsíti a HTTP-kérelmek kezelését. Lehetővé teszi, hogy a program könnyedén kommunikáljon webes API-kkal, például GET vagy POST kérések küldésével és fogadásával. A program a request könyvtárral lekérdezi az árfolyamokat a külső API-któl. A lekérdezés során egyszerű JSON formátumú adatokat kér le, amelyek könnyen feldolgozhatók. Funkciója közé tartozik a request.get() függvény, ami egy HTTP GET kérést ad számunkra, valamint a response.json(), ami a választ JSON struktúrává alakítja. Példa a programkódból:

```
def fetch_exchange_rates():
    response = requests.get(EXCHANGE_RATES_API)
    if response.status_code == 200:
        return response.json()["rates"]
    else:
        raise Exception("Nem sikerült lekérni a
devizaárfolyamokat.")
```

5.1.2. Mysql.connector

A mysql.connector a MySQL adatbázis-kezelő rendszer hivatalos Python-illesztőprogramja, amely lehetővé teszi a Python-alkalmazások számára, hogy közvetlenül kommunikáljanak egy MySQL adatbázissal. Az árfolyamok és azok időbélyegének tárolásához szükség van egy jól strukturált adatbázisra, amihez kapcsolatot a könyvtár használatával biztosíthatunk. Jelen projektben a feladatai közé tartozik a kapcsolat létrehozása, táblák készítése, valamint azok rekordokkal való feltöltése. A következő kódsnipettben az adatbázis táblák létrehozásának kódja látható:

```
def create_table(cursor, table_name):
    cursor.execute(
        f"""
        CREATE TABLE IF NOT EXISTS {table_name} (
            id INT AUTO_INCREMENT PRIMARY KEY,
            currency VARCHAR(10),
            rate FLOAT,
            timestamp DATETIME
        )
        """
    )
```

5.1.3. Datetime

A datetime python könyvtár a dátumok és időpontok kezelésére szolgáló könyvtár. A kódunkban elengedhetetlen, hiszen szükséges az árfolyamok pontos időbélyegének (timestamp) rögzítéséhez. Főbb funkciói közé tartozik az aktuális dátum és idő lekérése, valamint időbélyeg generálása az adatbázis-bejegyzésekhez. Példa a használatra:

```
exchange_rates = fetch_exchange_rates()
        timestamp = datetime.now()
        for currency, rate in exchange_rates.items():
            table_name = f"exchange_{currency.lower()}"
            create_table(cursor, table_name)
            store_rates(cursor, table_name, [{"symbol":
currency, "rate": rate, "timestamp": timestamp}])
```

5.1.4. Time

A time könyvtár alacsony szintű időkezelési funkcióját biztosítja, például késleltetéseket (sleep), amelyek gyakran előfordulnak – pl időzítések. A program a time.sleep() metódust használja arra, hogy bizonyos időközönként újraindítsa a lekérdezést, valamint minimalizálja az erőforrások túlzott használatát azzal, hogy várakozási időt állít be futások között. Példa a kódból:

```
if __name__ == "__main__":
    print("Indul az árfolyamkövető program.")
    while True:
        current_time = datetime.now()
        if current_time.hour == 0 and current_time.minute ==
0:
            run_tracker()
            print("Futtatás befejezve. 24 óra várakozás.")
            time.sleep(86400)
        else:
            time.sleep(30)
```

6. Adatbázis

6.1 MySQL általános bemutatása

A MySQL egy nyílt forráskódú, relációs adatbázis-kezelő rendszer (RDBMS), amely az SQL (Structured Query Language) nyelvet használja az adatok kezelésére és lekérdezésére. Széles körben elterjedt mind kisebb alkalmazások, mind nagyobb, komplex rendszerek esetében. A MySQL rendkívül népszerű webalapú alkalmazások adatbázisaként, például a WordPress, a Drupal vagy az e-kereskedelmi rendszerek alapjaként.

6.2 Előnyök

A MySQL szabadon használható, módosítható és bővíthető, így költséghatékony választás. Könnyű integrációt kínál programozási nyelvekkel, például Python, PHP vagy Java, és a LAMP stack alapvető eleme. Nagy teljesítményű, hatékonyan kezeli a nagy mennyiségű adatot és az egyidejű lekérdezéseket, miközben skálázható kisebb alkalmazásoktól nagy, terheléselosztott rendszerekig. Kiterjedt dokumentációval és aktív közösséggel rendelkezik, amelyek támogatják a tanulást és a hibakezelést. Mindezek miatt népszerű választás fejlesztők és vállalkozások körében.

6.3 Hátrányok

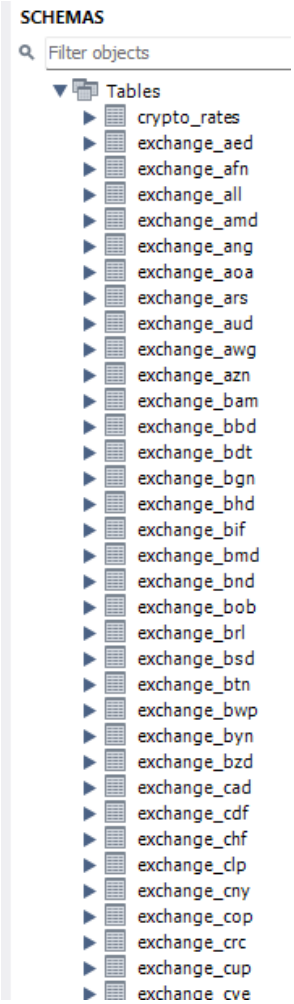
A MySQL ámbár népszerű és széles körben használt adatbáziskezelő, vannak bizonyos korlátai. Nagyvállalati környezetben a nyílt forráskódú verzió nem kínál minden speciális funkciót, például bizonyos replikációs technikákat vagy audit eszközöket. Az adatbiztonság terén is vannak hiányosságai, mivel titkosítási és biztonsági funkciói elmaradhatnak más rendszerektől, mint például az Oracle Database vagy a PostgreSQL. Nagy terhelésű környezetekben vagy óriási adatbázisok kezelésekor teljesítménybeli problémák léphetnek fel. Emellett régebbi verzióknál a tranzakciókezelés és más haladó funkcionalitások támogatása nem volt kielégítő, ami korlátozhatta a használhatóságot bizonyos projektekben.

6.4 Kezelhetőség

A MySQL könnyen telepíthető és használható különböző operációs rendszereken, mint a Windows, Linux és macOS, és egyszerű kezelést biztosít például a MySQL Workbench segítségével. Az SQL szabványosított nyelv támogatása megkönnyíti a használatát azok számára, akik már ismerik az adatbázis-kezelés alapjait. Erős adatbiztonsági és jogosultságkezelési funkciói lehetővé teszik az érzékeny adatok védelmét. Emellett egyszerűen végezhető adatmentés és visszaállítás, minimalizálva az adatvesztés kockázatát, miközben a replikációs támogatás javítja a teljesítményt és biztosítja az adatok magas rendelkezésre állását.

6.5 Az adatbázis a projektből

6.5.1. Schemas



1. ábra - Sémák, a feladat adatbázi táblái

A kép egy adatbázis sémát ábrázol, amely MySQL-ben van definiálva. A "Tables" rész alatt különböző táblák találhatók, amelyek az árfolyamokkal kapcsolatos adatokat tárolják.

- A "crypto_rates" tábla a kriptovaluták, például Bitcoin és Ethereum árfolyamadatait tartalmazza.
- Az "exchange_" előtaggal kezdődő táblák egy-egy adott valuta árfolyamát tárolják (például "exchange_usd" az amerikai dollárhoz, "exchange_eur" az euróhoz tartozik).
- A táblák külön-külön rendszerezik az egyes valutákhoz kapcsolódó adatokat, mint például az árfolyamértékek és időbélyegek.

6.5.2. Rekordok

id	currency	rate	timestamp
1	AED	3.6725	2025-01-19 20:54:03
2	AED	3.6725	2025-01-20 09:30:24
3	AED	3.6725	2025-01-20 09:40:49
4	AED	3.6725	2025-01-20 12:14:05
5	AED	3.6725	2025-01-21 00:00:27
6	AED	3.6725	2025-01-22 00:00:13
NULL	NULL	NULL	NULL

2. ábra - példa a tárolásra

A rekordok az alábbiak szerint mentődnek el az adatbázisunkba: az ábra az "exchange_aed" adatbázis táblát mutatja (az Egyesült Arab Emírségek dirhamjára vonatkozó árfolyamokkal), amely az AED (Arab Emirátusi Dirham) árfolyamadatait tartalmazza.

A táblában az alábbi oszlopok szerepelnek:

- id: Egyedi azonosító, amely minden sorhoz egyedi értéket rendel. Ez általában automatikusan növekvő egész szám.
- currency: Az adott valuta kódja (ISO 4217 szabvány alapján). Ebben az esetben mindig "AED".
- rate: A valuta árfolyama az alapvalutához viszonyítva, jelen esetben az amerikai dollárhoz (USD). Az érték az itt látható sorokban konstans: 3.6725.
- timestamp: Az adott árfolyam rögzítésének időpontja, amely pontos dátumot és időt is tartalmaz. Ez segíti az árfolyamok időbeli változásának nyomon követését.

7. Lezárás, vélemény

Ez a feladat egy remek lehetőség volt számomra, hogy megmutassam a technikai tudásomat, különösen az automatizálás, az API-k kezelése és az adatbázis-kezelés terén. A célom az volt, hogy egy olyan rendszert, valamint dokumentációt hozzak létre, amely által megfelelően bemutatkozhatok a munkámon keresztül Önök számára.

A Python könyvtárak, mint a requests, mysql.connector, datetime, és time jól illeszkedtek a feladathoz, és úgy érzem, hogy sikerült teljes mértékben kihasználni a lehetőségeket, amelyeket biztosítanak.

A hibakezelés megfelelően működik, és igyekeztem úgy megoldani a problémákat, hogy a rendszer ne álljon le váratlan hibák miatt. Azonban azt is látom, hogy van még tér a fejlődésre. A kódot például még jobban optimalizálhatnám, és a hibakezelést is finomíthatnám, hogy többféle lehetséges hibát kezelni, esetleg külön fileba naplózni tudjon.

Elmondhatom, hogy elégedett vagyok a végeredménnyel, az elvégzett munkával, mivel sikerült egy stabil és működő kódot létrehoznom. Az ilyen típusú feladatok segítenek abban, hogy még jobban megismerjem a különböző technológiai eszközöket és, hogy hatékonyan oldjam meg a valós problémákat, hogy jobb szakember lehessek.

Gaál Márk

2025. 01. 22.