

Kriptós példa projekt

A projekt egy rövid példa kriptós adatok feldolgozására. Lényegében az adatokat egy exchange-ről leszedi meghatározott időtartamokban, eltárolja, majd ezek alapján lehet például ahogyan a kód is mutatja, szűrni a kereskedéshez érdekesebb tokeneket. Sokféleképpen felhasználható kisebb-nagyobb bővítésekkel, például figyelje a történéseket a piacon és figyelmeztetést adjon, ha az engem érdekelt szűrő feltételek igazak egy tokenre, vagy éppen egy trading bot felépítéséhez.

A microservice architektúra úgy jelenik meg, hogy két részből áll fel a projekt. Van egy fő program, ami az adatokat lekérdezi, majd eltárolja az adatbázisban, ezt az app mappában található `app.py` fájl tartalmazza. Valamint egy api gateway szerű program ami továbbítja az adatokat az adatbázisból a külvilág felé, ezt az api mappában az `api.py` fájl tartalmazza.

A megoldás pythonban a flask framework segítségével készült, adatbázishoz postgreszt használtam, valamint dockerizáltam a projektet.. Az exchange-től való adatok lekérdezéshez a `ccxt` libraryt használtam fel, jelen példában a Binance-től kérek le adatot, de a library segítségével könnyedén meg lehetne oldani, hogy másik exchange-et használjon fel a program vagy hogy több helyről szerezze be az adatokat egyszerre. A lekérések ütemezéséhez az `apscheduler` libraryt használok fel, ennek a segítségével ütemezni tudom a különféle lekérdezéseket percre pontosan mikor történjenek. Ez fontos mivel a példakód a micro timeframe-ben a 15 perces változások alapján működik, így minden óra egész, 15, 30 és 45ik percében kell, hogy megtörténjen a lekérdezés. Itt a végső megoldás ezeknek az időzítéseknek a minimálisan előbbre eltolása bizonyult, hogy az exchange biztosan befejezze az adatok frissítését és már az aktuális adatot küldje. Ez azt is eredményezi, hogy így később frissül az eredmény 1-2 perccel, de ez alapvetően nem számottevő probléma egyik use case-nél sem. Az adatbázishoz való összeköttetésben az SQLAlchemy libraryt használok, ennek a segítségével történik a postgres adatbázishoz a csatlakozás és az adatok tárolása és lehívása. Ezeket a technológiákat azért választottam, mert így könnyedén bővíthető, skálázható a projekt több exchange támogatásával, többféle adat feldolgozásával. Az adatbázis sql alapú, mivel fix., strukturázott adatokat használok fel és így jobban kezelhető mint egy nosql adatbázissal.

A `models.py` fájl egy a két fő rész között megosztott fájl, ezt docker segítségével osztom meg a kettejük között. Ez tartalmazza az orm-hez táblánként lebontva az adatok felépítését. Lényegében van egy *Coin* nevű tábla, amiben a szimbólumok vannak eltárolva, ehhez egy a több kapcsolatban csatlakozik a *Price*, *OpenInterest* és a *BaseVolume*.

Az `app.py` fájlban elsőként lekérdezem az összes future marketben található coin szimbólumát, hogy a program tudja, milyen szimbólumok léteznek, ezt a `fetch_all_coins()` függvény hajtja végre. Ezekből leszűröm a top 20 tokenet a napi kereskedési volumen alapján, ezt a `fetch_top_symbols()` függvény hajtja végre. Ezt minden induláskor első lépésként végre kell hajtani, hogy a scheduler-be időzített függvények később ezt már fel tudják használni. Ezután időzitem a price, volume, open interest és a top 20 szimbólumot lekérdező függvényt, és innentől kezdve elindul az adatgyűjtés. Az adatokat lekérdező függvények hasonló logika alapján épülnek fel. Ha nincsen egy adatra batch lekérdezési opció a szimbólum alapján, akkor szimbólumként felveszek egy-egy taszkot, majd lefutatom

az aszinkron lekérdező függvényt és bevárom az összes adatot. Ezután az adatokat felviszem az adatbázisba, valamint ha szükséges még további processinget hajtok végre előtte, mint a százalékos változások kiszámítása.

Az api.py fájlban a végpontok szerepelnek, az adatbázisban tárolt adatokhoz két-két GET, egy ami az összes adatot visszaadja és egy ami szimbólumra szűrve. Ezeken felül található még egy a *get_interesting_coins()* néven, ami különböző feltételek alapján leszűrve ad vissza egy tokent. A példában az open interest és a volumen változása alapján történik a szűrés és a minden feltételnek megfelelt szimbólumot adja vissza. A price adatokat habár sehol sem használom itt fel közvetlenül, mégis tárolom, mivel egy frontend felületen potenciálisan hasznos lehet feltüntetni, pluszban kiegészítve százalékos változással 24 óra alapján, egyébként elhagyható.