

Aplikacja do obsługi giełd kryptowalutowych - postępy

Adam Kasperowicz - 279046

Aplikacja w obecnym stanie osiągnęła następujące cele:

- Uzyskiwanie danych z giełd bez potrzeby wchodzenia na stronę giełdy.
- Zbieranie danych z giełdy w tle (np. Zapisywanie w bazie danych obecnych wartości danej pary walut co minutę przez nieokreślony czas).
- Jak najprostsze podłączanie nowych giełd do aplikacji.
- Możliwość wyświetlania danych z baz danych w postaci wykresów itp.

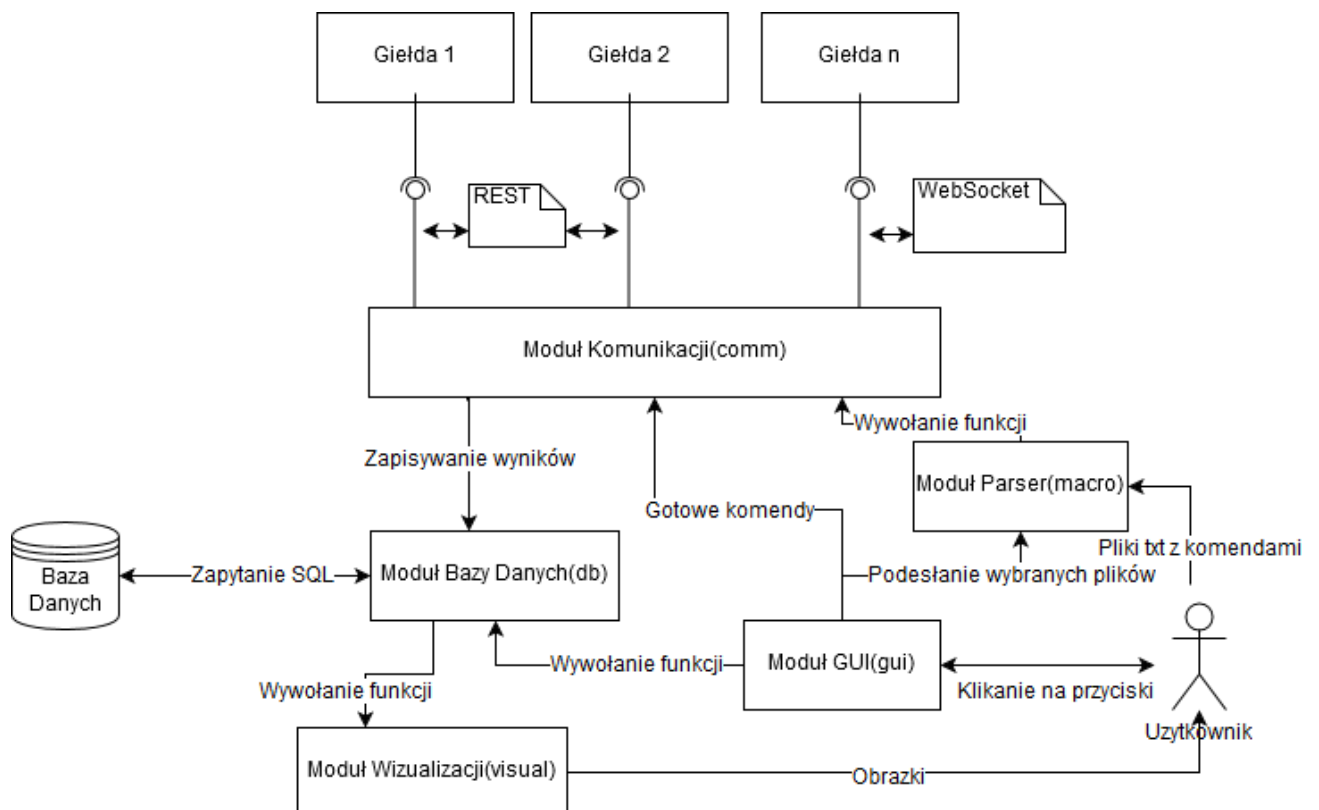
Do zaimplementowania zostały cele:

- Wykonywanie transakcji bez potrzeby wchodzenia na giełde.

Aplikacja powinna posiadać następujące cechy:

- Pełne pokrycie testami (Test Driven Development wykorzystując PyUnit).
- Wykorzystywanie zawsze najbardziej efektywnego protokołu komunikacji z giełdą(jeśli giełda oferuje zarówno REST'a jak i WebSocket'a to korzystamy z drugiej opcji).
- Jak najbardziej przyjazna i jak najprostsza dla użytkownika.

Obecnie struktura aplikacji przedstawia się następująco



Moduł GUI (gui) :

Podstawowe funkcjonalności interfejsu graficznego zostały poprawnie zaimplementowane. Kod znajduje się w pakiecie „src/gui”. Szata graficzna została utworzona z pomocą języka QML zaś funkcje kontrolne zaimplementowane zostały w Pythonie przy pomocy PyQt5.

Niestety biblioteka okazała się znacznie mniej przyjazna niż można by było sądzić po pierwszym wrażeniu. Duże nieścisłości w dokumentacji, nieaktualne informacje oraz duże różnice pomiędzy wersjami sprawiły, że nawet najmniejsze elementy wymagały ogromnego nakładu czasowego.

Poniżej widać obecną prowizoryczną wersję interfejsu.

Obecny interfejs graficzny



W związku z problemami jakie powodowała biblioteka podjęta została decyzja o tymczasowym pominięciu całkowitej implementacji algorytmu obsługi interfejsu. Brakuje również testów, których możliwość poprawnego utworzenia jest bardzo utrudniana.

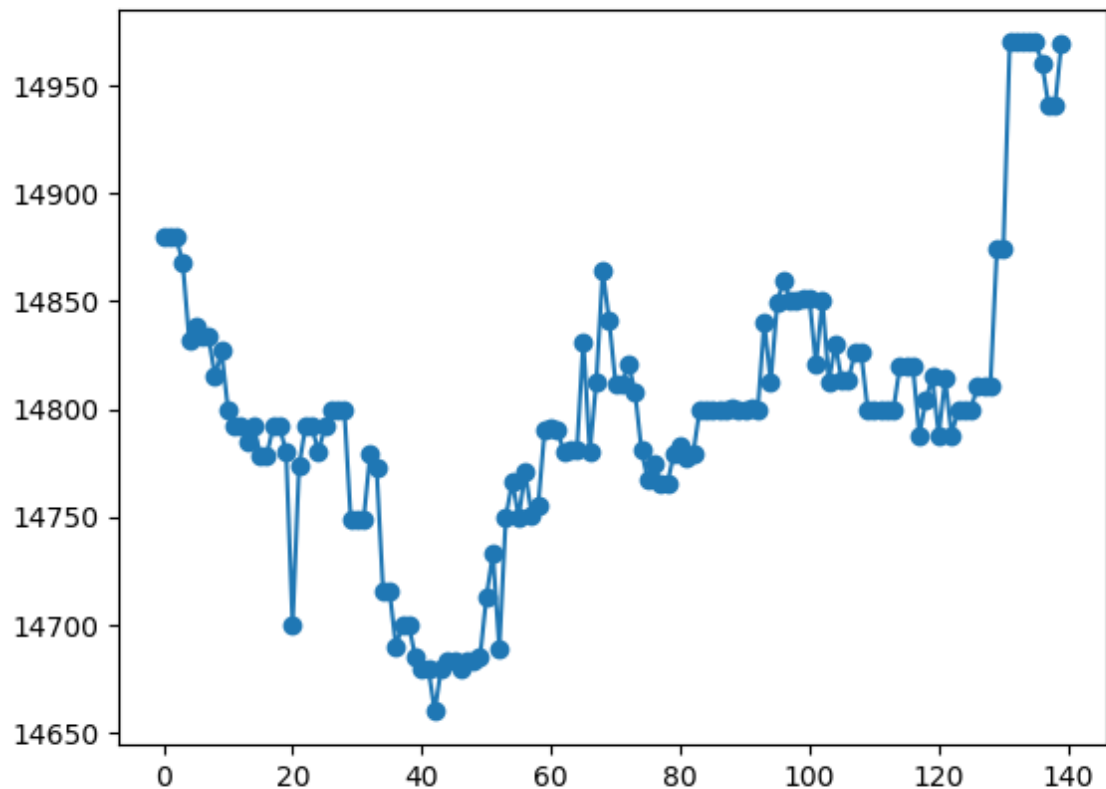
Moduł Parser(macro):

W związku ze zmianami architektonicznymi spowodowanymi poznawaniem działania wykorzystywanych bibliotek, zaplanowany został Moduł Parser, którego zadaniem jest tylko i wyłącznie parsowanie makr napisanych w prostym języku i wywoływanie odpowiednich funkcji modułu komunikacyjnego.

Jako, że większość zasobów czasowych została poświęcona modułowi gui, moduł parser pozostał w fazie planowania. W folderze „src/macro” znaleźć możemy przykładowe makro, które zawierało by praktycznie wszystkie zamarzone funkcjonalności.

Moduł Wizualizacji:

Moduł, który został zaimplementowany w bardzo prostej postaci pozwala nam wyświetlić wykres utworzony z cen zapisanych w naszej bazie danych. Poniżej widzimy przykładowy wykres dla około 140 rekordów cen Bitcoina w dolarach.

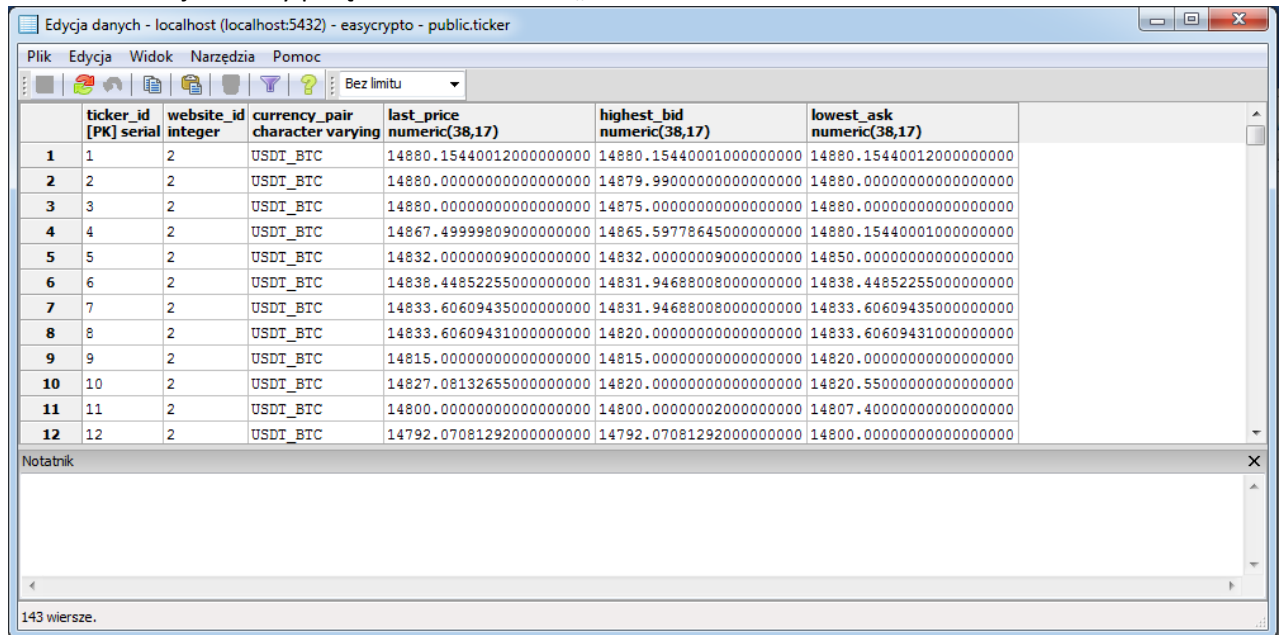


Moduł Bazy Danych:

Wykorzystywaną technologią bazy danych jest PostgreSQL. Do obsługi zapytań wykorzystywana jest biblioteka SQLAlchemy. Bardzo miły w obsłudze zestaw narzędzi pozwolił na skuteczne utworzenie interfejsu pozwalającego innym modułom na dostęp do bazy danych.

W celu uruchomienia programu wymagana jest baza danych PostgreSQL obecna na komputerze. Baza powinna się nazywać „easycrypto”. Hasło do bazy umieszczamy w pliku „pass.txt”. Plik należy wcześniej utworzyć w folderze „src/db”. Oczywiście rozwiązanie te jest chwilowe.

Poniżej widzimy parę rekordów z tabeli „ticker”.



	ticker_id [PK] serial	website_id integer	currency_pair character varying	last_price numeric(38,17)	highest_bid numeric(38,17)	lowest_ask numeric(38,17)
1	1	2	USDT_BTC	14880.154400120000000000	14880.154400010000000000	14880.154400120000000000
2	2	2	USDT_BTC	14880.000000000000000000	14879.990000000000000000	14880.000000000000000000
3	3	2	USDT_BTC	14880.000000000000000000	14875.000000000000000000	14880.000000000000000000
4	4	2	USDT_BTC	14867.499998090000000000	14865.597786450000000000	14880.154400010000000000
5	5	2	USDT_BTC	14832.000000090000000000	14832.000000090000000000	14850.000000000000000000
6	6	2	USDT_BTC	14838.448522550000000000	14831.946880080000000000	14838.448522550000000000
7	7	2	USDT_BTC	14833.606094350000000000	14831.946880080000000000	14833.606094350000000000
8	8	2	USDT_BTC	14833.606094310000000000	14820.000000000000000000	14833.606094310000000000
9	9	2	USDT_BTC	14815.000000000000000000	14815.000000000000000000	14820.000000000000000000
10	10	2	USDT_BTC	14827.081326550000000000	14820.000000000000000000	14820.550000000000000000
11	11	2	USDT_BTC	14800.000000000000000000	14800.000000200000000000	14807.400000000000000000
12	12	2	USDT_BTC	14792.070812920000000000	14792.070812920000000000	14800.000000000000000000

Struktura tabel definiowana jest formatem JSON. W folderze „src/db/data” znajdują się pliki JSON zawierające informacje o podstawowej strukturze tabel. Pliki te są czytane przez DBControl a następnie tworzone są tabele w bazie danych na ich wzór.

Moduł komunikacji:

Ostatni z modułów wykorzystuje w swoim rdzeniu bibliotekę Twisted. Praca z tym oprogramowaniem okazała się zdecydowanie najprzyjemniejszą częścią pracy. Obecne działanie polega na przyjęciu komendy którą należy wykonać. Następnie, komponujemy instancje klasy Bot wraz z komunikatorem odpowiadającym protokołowi oraz parserem odpowiadającym giełdzie by ostatecznie pobierać dane z giełd i ewentualnie zapisywać je w bazie danych.

Zaimplementowane zostały 3 połączenia z giełdami

- Poloniex REST – pozwala na periodyczne pobieranie wartości danej pary walut z giełdy.
- Bittrex REST – pozwala pobrać dane na temat par walut dostępnych na giełdzie.
- Poloniex WAMP – wykorzystujący bibliotekę Autobahn, powinien dawać nam możliwości pobierania tych samych danych co REST'owy odpowiednik.

Przy próbie implementacji protokołu WAMP ujawnił się duży problem. Zarówno Twisted jak i PyQT korzystają pętli zdarzeń jako rdzenia działania. Niestety by obie biblioteki działały jednocześnie wymagana jest obejście tego problemu. Podjęta została próba wykorzystania biblioteki pyqt5reactor. Biblioteka spełnia swoją rolę dla połączenia REST. Niestety, Autobahn nie trawi tej mieszanki i nie pozwala nam na użycie protokołu WAMP.

Problem ten z pewnością jest również rozwiązywalny. Niestety, jest również czasochłonny.

Fazy implementacji

1. Zapis obecnej wartości pary walut w bazie danych dla jednej giełdy.
2. Możliwość zapisywania obecnej wartości pary walut przez cały dzień i wizualizacja tej wartości w postaci wykresu.
3. Dołączenie dwóch kolejnych giełd.
4. Zaimplementowanie wszystkich ważniejszych danych możliwych do pobrania z giełdy
5. Możliwość zdalnego handlowania na giełdzie.
6. Prosty bot przyjmujący strategię średnich kroczących.

Powyżej zaznaczone zostały fazy **zaimplementowane**, **częściowo zaimplementowane** i **nieruszone**.

Obecny kurs działań przewiduje powstrzymanie się przed implementacją kolejnych funkcjonalności. Znacznie ważniejszą rzeczą jest naprawa wszelkich błędów oraz napisanie testów pokrywających większość programu.