Quick Trader

Dokument projektowy

Spis treści

1. Czym jest Quick Trader?

Quick Trader to proste narzędzie pozwalające wielu użytkownikom jednocześnie licytować przedmioty. Przedmioty są dodawane przez administratora serwera w bazie danych a następnie wystawiane na licytację ogólnodostępną dla użytkowników posiadających konta. Każda licytacja to szansa na zdobycie unikatów w dobrej cenie!

#wrzuć gdzieś tu screeny

1.1 Dlaczego Quick Trader?

Ponieważ cenimy bezpieczeństwo transakcji oraz naszych użytkowników. Każdy użytkownik musi zostać ręcznie zatwierdzony po weryfikacji tożsamości przez nasz zespół. Każde połączenie jest szyfrowane za pomocą klucza – na pierwszym miejscu stawiamy bezpieczeństwo naszych użytkowników.

1.2 Co wyróżnia Quick Trader na tle konkurencji?

Przede wszystkim prostota. Jeden ekran logowania, jeden ekran aukcji. Proste przyciski pozwalające na szybkie podbijanie wartości przedmiotu. Każdy przedmiot wystawiany jest na jedynie 60 sekund, natomiast każde podbicie wydłuża nieznacznie czas trwania licytacji. Przedmioty, które będą cieszyły się większym zainteresowaniem będą licytowane dłużej, natomiast przedmioty o mniejszym zainteresowaniu szybciej zostaną sprzedane – pozwoli to na płynny przepływ sprzedaży oraz zwiększy emocje związane z licytacjami!

2 Nasz zespół

Jesteśmy trzyosobową grupa pasjonatów, która dzięki kooperacji i zaangażowaniu postanowiła podjąć się wyzwania stworzenia aplikacji. Członkowie zespołu to:

- Emil Tomczyk - emil.tomczyk@skni.umcs.pl
- Marcin Szyszka - cinkowski07@gmail.com
- Łukasz Bochniak - lukasz.bochniak@skni.umcs.pl

Aplikacja powstała dzięki współpracy – nie byłoby jej, gdyby zabrakło któregokolwiek z członków zespołu. Postawiliśmy na zgranie, systematyczność, zaangażowanie i wymianę doświadczeń – inaczej Quick Trader by nie powstał.

3 Przebieg aukcji – jestem kupującym

Korzystanie z aplikacji jest proste. Należy podać jedynie swój login oraz unikalne hasło uzyskane podczas weryfikacji. Następnie po kliknięciu przycisku „Zaloguj” aplikacja zweryfikuje Twoją tożsamość i przeniesie Cię do ekranu aukcji. Tam jedyne, co musisz robić, to czekać na okazje i je łapać – Prosty interfejs pozwala na szybkie zrozumienie zasad działania programu. Aby licytować wystarczy klikać na dostępne przyciski aukcji, które automatycznie podbiją cenę. Nic prostrzego!

4 Przebieg aukcji – jestem administratorem

Po dodaniu licytujących do bazy należy dodać do niej również przedmioty wystawione na aukcję – wymagane jest, aby podać informacje takie jak nick sprzedającego, cena wywoławcza czy czas rozpoczęcia licytacji. Po uruchomieniu serwera wszystko dzieje się samo, wystarczy jedynie czekać na zakończenie licytowania wszystkich dostępnych w bazie przedmiotów. Przedmioty oraz kupujących

można dodawać w trakcie trwania sesji aukcyjnej aż do jej zakończenia – nie wpłynie to negatywnie na działanie serwera. O wszystko zadbaliśmy!

5 Instrukcja uruchomienia serwera oraz klienta

Do uruchomienia obu wymagany jest Python. Ze względu na użyte narzędzia jedynym obsługiwanym na ten moment systemem operacyjnym jest Linux. Aby uruchomić serwer wystarczy w katalogu serwera wydać następujące komendy:

```
$ python3.9 -m venv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
```

Powyższe komendy stworzą środowisko Pythona. Należy je wykonać tylko raz

```
$ source venv/bin/activate
$ python3.9 main.py
```

Z kolei tymi komendami uruchomimy demona.

Konfiguracja serwera odbywa się za pomocą pliku config. Jeśli plik ten nie istnieje, lub nie zawiera on wszystkich pozycji, to zostaną użyte wartości domyślne. Dostępne opcje konfiguracyjne to:

- ip – adres IP serwera, domyślnie ‘0.0.0.0’
- port – port sertera, domyślnie 55555
- dbpath – ścieżka do bazy danych, domyślnie ‘./db.sqlite3’
- tls_cert – ścieżka do certyfikatu TLS, domyślnie ‘./tls.cert’
- tls_key – ścieżka do klucza TLS, domyślnie ‘./tls.key’

Natomiast do uruchomienia klienta wymagana jest komenda:
#tu też XDDD

6 Protokół zastosowany do komunikacji klient – serwer

Słownik	Treść	Wyjaśnienie
	KLIENT	
type: **username:** **password:**	„auth” username password	Struktura wiadomości informującej serwer o próbie zalogowania ze strony klienta.
type: **price:** **token:** **username:**	„bet” price token username	Próba podbicia ceny aktualnie wystawionego przedmiotu przez klienta.
type: **username:** **token:**	„logout” username token	Informacja dla serwera o chęci wylogowania ze strony klienta
	SERWER	
type: **token:**	„auth” token	W przypadku poprawnego zalogowania odsyła token logowania, w przypadku niepoprawnego logowania – kod błędu 4
type: **successfully:**	„auth” True	W przypadku poprawnego wylogowania odsyła True – w przeiwnym razie kod błędu 5
type: **pong:**	„ping” True	W przypadku poprawnego wysłania sygnału aktywności odsyła true, jeśli nie – klient otrzymuje kod błędu 6
type: **current_price:** **name:** **end_time** **leader:** **started:**	„bet” current_price name end_time leader True	W przypadku podbicia ceny rozsyła broadcast’em do wszystkich klientów informacje o nowej cenie, nowym prowadzącym i uaktualnia czas zakończenia aukcji. Nowo podłączeniu kupujący otrzymują również informację o nazwie przedmiotu oraz o rozpoczęciu aukcji. W przypadku błędu kod 4

7. Kody błędów i ich obsługa

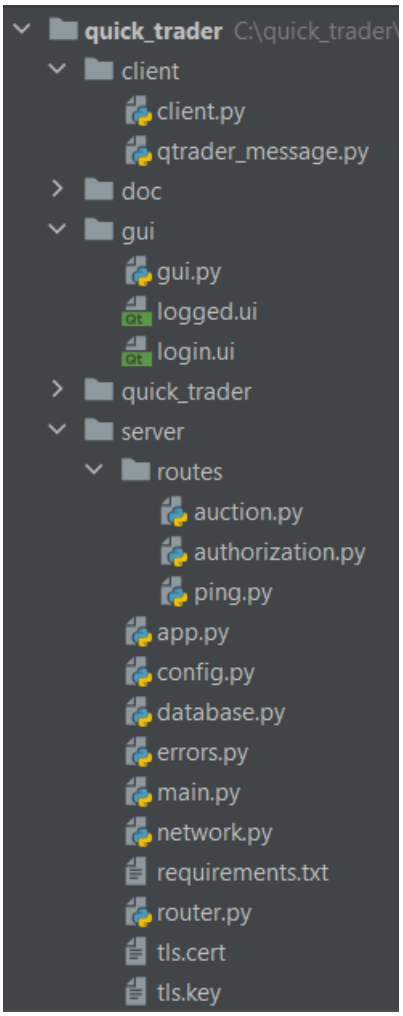
Wyróżniliśmy kilka rodzajów błędów, które z różnych przyczyn mogą się wydarzyć podczas korzystania z serwera tudzież klienta i zdefiniowaliśmy je w osobnym pliku errors.py. Błędy, które wyszczególniliśmy to:

Kod błędu	Nazwa błędu	Wyjaśnienie
1	ERROR_JSON_PARSE	Błąd parsowania danych JSON, niepoprawna struktura
2	ERROR_JSON_NO_TYPE	Pole ‘type’ nie jest zawarte w odebranych danych
3	ERROR_TYPE_NON_EXISTS	Pole ‘type’ zdefiniowane przez klienta nie istnieje na serwerze
4	ERROR_LOGIN_FAILED	Błąd podczas próby zalogowania się klienta na serwerze
5	ERROR_LOGOUT_FAILED	Błąd podczas próby wylogowania klienta z serwera
6	ERROR_AUTH_FAILED	Problem z autoryzacją – pojawia się, gdy token użytkownika jest niezgodny z jego nickiem.
7	ERROR_NO_AUCTION	

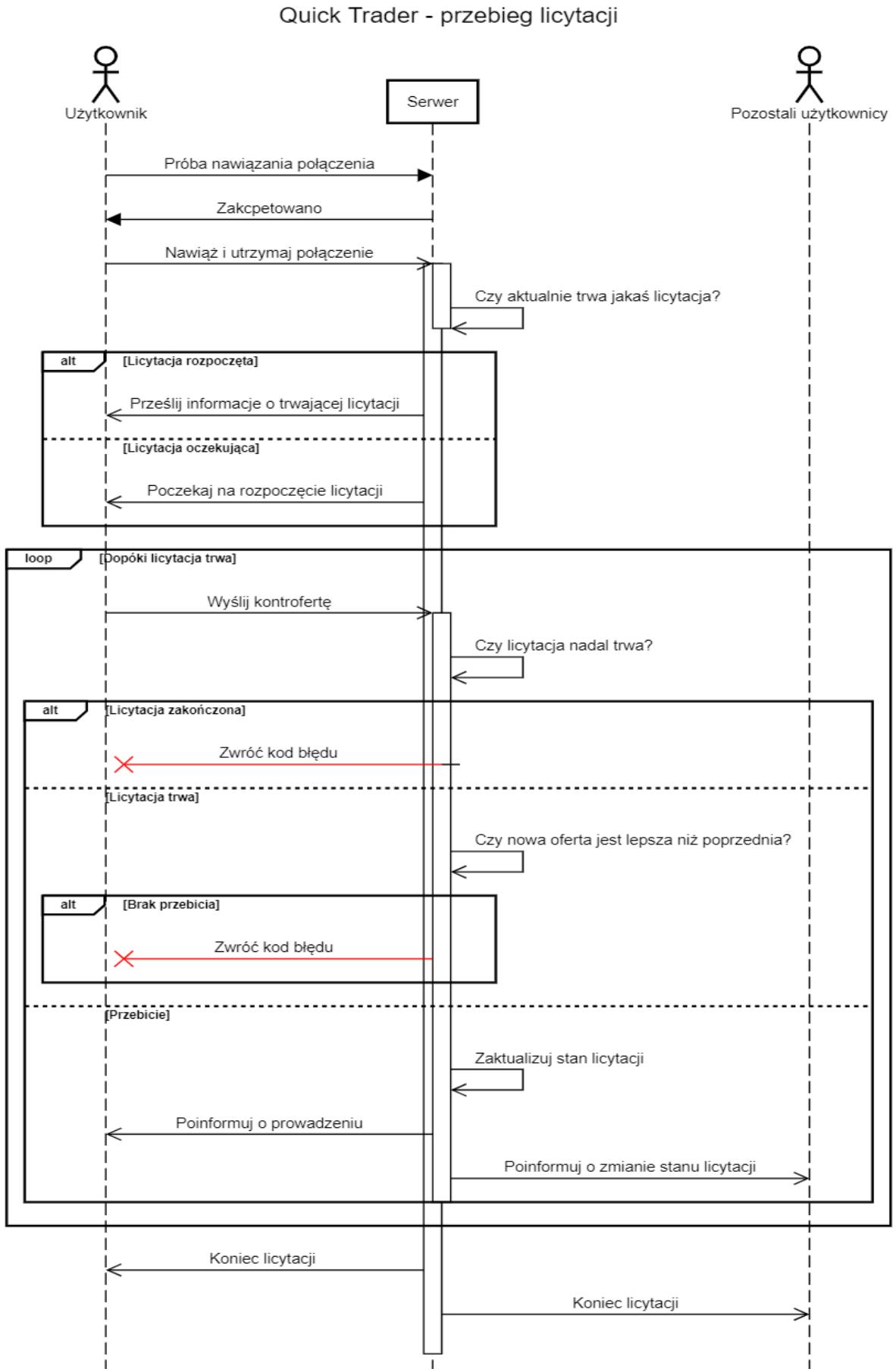
8. Struktura plików

Struktura plików w projekcie została podzielona na podkatalogi. W głównym katalogu „quick_trader” znajdują się podkatalogi:

- client – przechowujący pliki klienta aplikacji:
  - client.py
  - qtrader_message.py
- doc – zawierający dokumentację projektu
- gui – w którym znajdują się pliki interfejsu graficznego klienta:
  - gui.py
  - logged.ui
  - login.ui
- server – główny podkatalog z plikami serwera:
  - routes – kolejny podkatalog zawierający pliki obsługi
    - auction.py
    - authorization.py
    - ping.py
  - app.py
  - config.py
  - database.py
  - errors.py
  - main.py
  - network.py
  - requirements.txt
  - router.py
  - tls.cert
  - tls.key



9. Diagram sekwencyjny aplikacji



10. Serwer

10.1 Opis działania serwera

Serwer działa na dwóch wątkach. Jeden z nich służy do zdarzeniowej obsługi sieciowej, drugi do obsługi systemu oraz bazy danych. Dane między wątkami są przekazywane za pomocą dwóch kolejek queue.Queue(). Część sieciowa działa z użyciem selektora select.poll(). Dla każdego połączenia tworzony jest obiekt Client, który przechowuje dane w trakcie odbierania, wysyła dane itp. Obsługiwane są także wiadomości typu broadcast wysyłane do wszystkich klientów. Cała obsługa sieci znajduje się w pliku `server/network.py`. Obsługa aplikacji zaczyna się w pliku 'server/app.py' . Znajduje się tutaj nieskończona pętla obsługująca cały program oraz funkcja wysyłająca wiadomość `ping` do wszystkich klientów. Służy ona także części sieciowej do stwierdzenia przerwania połączenia z klientem. Zapytania od klientów są obsługiwane na zasadzie endpointów, określanych przez parametr `type` wiadomości. W zależności od niego serwer wykonuje konkretną akcje i odsyła dane do serwera. Za trasowanie odpowiada plik `server/router.py`, a za implementację poszczególnych funkcji odpowiadają pliki w katalogu `server/routes/`. Znajduje się tam między innymi logowanie czy podbijanie aukcji. Do obsługi bazy danych został wybrany ORM `peewee`. Modele oraz inicjalizacja jej znajduje się w pliku `server/database.py`. Start całego serwera jest zdefiniowany w pliku `server/main.py`. Wywoływana jest tam też inicjalizacja konfiguracji, która jest obsługiwana w pliku `server/config.py`, a sama konfiguracja jest zapisana w `server/config`. Do tego jest jeszcze plik `server/error.py` w którym są zdefiniowane kody błędów.

10.2 Dodawanie użytkowników oraz aukcji

Aby dodać użytkownika do bazy danych, należy w tabeli `user` dodać nowy wiersz, określić ID oraz nazwę użytkownika. Pozostałe pola można ustawić na 0. W tabeli `auth` należy dodać wiersz z takim samym ID jak w tabeli `user`, hasło zakodowane w sha512 oraz pusty ciąg znaków w login_token.

10.3 Dodawanie aukcji

Aby dodać aukcję należy w tabeli `auction` dodać wiersz. Należy ustawić nazwę, cenę wywoławczą, datę startową w formacie `YYYY-MM-DD HH:mm:ss`, ID sprzedawcy oraz flagę ended ustawić na 0.

Licytacja rozpocznie się o zadanej godzinie i potrwa 60 sekund plus ilość podbić razy 10 sekund (każde podbicie wydłuża czas aukcji o 10 sekund).