API Implementation

Introduction

The API implementation is based on the specifications defined by our `openapi.json` file. All endpoints in the specifications should be implemented in the `develop` branch.

Our backend is based on 'falcon', a WSGI Server meant for making APIs in python. For our database we use 'diskcache' as it is embedded, fast and supports multiprocess access and is native to python.

File structure

Utility Files: * `base.py` * `db.py` * `utils.py`

Logic Files: * `tasks.py` * `realtime.py`

Endpoint Files: * `auth.py` * `backtesting.py` * `exchange.py` * `strategy.py`

The logic files run async to the API server itself.

Validation

Most endpoint validate the type of their input to ensure security of the server. If the validation fails the server throws an `AssertionException`, however when using it in production this will be caught by the exception handler as defined in `server.py` and will respond to the user/fontend a `401 Bad Request` error.

Managers

TTL Manager

Our server makes use of our TTL Manager, this manager keeps track of things that will expire soon. Once it expires arbitrary code can run, currently it's used to delete old backtesting results, and remove authentication tokens after they expire.

User Manager

This manager wraps a lot of user logic, this also validates certain operations on the database, like asserting a username is available when creating a user.

For passowrd hashing we use salted blake2b, which is a relatively secure hash; however it could be improved with multiple iterations to restrict brute-force attack if a leak is to occur.

Source code

Code is available at: * https://git.fhict.nl/I404788/trading-bot/-/blob/develop/routes * https://git.fhict.nl/I404788/trading-bot/-/blob/develop/server.py * https://git.fhict.nl/I404788/trading-bot/-/blob/develop/openapi.json