1. **Назначение программного продукта:**

CryptoMonkey является экспериментальным настраиваемым криптоботом с десктопным пользовательским интерфейсом. Программа предназначена для проведения анализа и исследования возможности торговли высоковолатильными перспективными криптовалютами, как в режиме эмуляции покупки и продажи, так и в режиме торговли реальными активами пользователя. Структурная схема построения программы приведена на рисунке 1.

1. **Принцип построения и состав**

2.1 Программа строиться по модульному принципу, где каждый модуль является отдельным файлом с раcширением “.py”. Управление осуществляется главным модулем CryptoMonkeyGui.py, содержащим класс пользовательского интерфейса, и осуществляющим взаимодействие с периферийными модулями в соответствии с пользовательскими настройками.

2.2 Состав и краткое описание модулей:

1. CryptoMonkeyGui.py – модуль пользовательского интерфейса. Предназначен для:

- настройки режимов работы программы;

- запуск модуля получения данных и анализа (CoinPriceAnalizer.py);

- отображение результатов проведенных операций и статистических данных из базы данных через взаимодействие с модулем БД (DataBase.py).

1. CoinPriceAnalizer.py – предназначен для:

* запуска парсера (модуль Parser.py);
* получения данных от парсера по выбранной пользователем валютной паре;
* формирование запроса и получение необходимых статистических данных по выбранной валютной паре из БД (DataBase.py) для учета при анализе;
* передачи полученных от парсера (модуль Parser.py) данных для отрисовки в GUI (рендеринг) и в модуль базы данных (DataBase.py) для формирования статистики;
* параллельного анализа на соответствие полученных данных критериям заданным пользователем с учетом статистических данных по выбранной валютной паре, полученных из БД;
* обработки (в случае соответствия заданным критериям) полученных от парсера данных и, при необходимости, передачи их на исполнение криптоброкеру (модуль CryptoBroker.py).

1. Parser.py – модуль для формирования стандартизированных запросов к программному интерфейсу криптобиржи (API) и получения данных в формате JSON по заданной пользователем валютной паре с последующей выдачей полученных данных в модуль анализа (CoinPriceAnalizer.py).
2. CryptoCoinBroker.py – предназначен для:

* запуска по команде модуля анализа (CoinPriceAnalizer.py) и получения (в виде параметров функции) данных для осуществления покупки/продажи криптовалюты из личного кабинета пользователя;
* осуществления покупки/продажи криптовалюты из личного кабинета пользователя через программный интерфейс криптобиржи (API) с использованием ключа-токена для доступа к личному кабинету;
* осуществления запросов через API криптобиржи для получения данных о состоянии счета пользователя;
* оценки результата проведенной операции и передачи данных о сделанной операции в модуль базы данных (DataBase.py) для записи в файл.

1. DataBase.py – модуль взаимодействия с файлом базы данных. Предназначен для:

* получения и записи данных парсинга от модуля анализа (CoinPriceAnalizer.py) в файл БД;
* получения и записи данных о проведенных операциях (транзакциях) и состоянии счета пользователя от криптоброкера (CryptoCoinBroker.py) в файл БД;
* получения и обработки запросов от графического интерфейса пользователя (CryptoMonkeyGui.py), осуществление взаимодействия с файлом БД в соответствии с этими запросами;
* выдача запрашиваемых данных из файла БД в GUI (CryptoMonkeyGui.py) для отрисовки.

1. **Технические требования**
   1. Общие требования

3.1.1 В программе укрупненно должны быть предусмотрены два режима работы:

* Эмуляция торговли на реальные активы;
* Торговля на реальные активы, находящиеся на счету в аккаунте пользователя.

Для первого варианта ввод токена от личного кабинета пользователя в GUI должен быть не обязателен. Должна быть предусмотрена передача критерия выбранного режима во входных параметрах функции запуска модуля анализа (**CoinPriceAnalizer.py**) с последующей трансляцией криптоброкеру для введения блокировки использования реальных активов пользователя при установке режима эмуляции.

Для второго варианта при выборе режима (а также в выпадающих вкладках меню), либо при попытке запуска, должен быть предусмотрен вывод окна для указания токена личного кабинета криптобиржи и блокировка запуска в случае отсутствия введенного токена пользователя.

Примечание: токен для подключения через API выдается криптобиржей в личном кабинете пользователя.

3.1.2 Во всех узлах программы с вероятностью возникновения сбоев и ошибок должна быть предусмотрена конструкция обработки исключений для избежания критического прекращения работы (конструкция типа try-except, см. пример в п.3.1.3).

3.1.3 В каждом модуле должно быть предусмотрено логирование в файл logs.txt и консоль через применение методов стандартной библиотеки **logging**.

Примечание: инициализация и настройка всех логгеров должна проводиться в главном модуле **CryptoMonkeyGui.py (код инициализации и настройки хэндлеров см в приложении 1)**.

Шаблон добавления логгера в каждом модуле:

logger = logging.getLogger('CryptoMonkey.{название модуля}')

**Образец добавления:**

logger = logging.getLogger('CryptoMonkey.CoinPriceAnalizer')

**Образец использования логгера с типом ошибки «error» в блоке обработки исключений и ошибок:**

try:

async with aiohttp.ClientSession() as session:

async with session.get(url) as response:

parsData = await response.json()

return parsData

except Exception as ex:

logger.error(ex)

3.2 Требования по взаимодействию модулей

3.2.1 Настройка алгоритма анализа должна производится из графического интерфейса пользователя (далее – GUI или модуль CryptoMonkeyGui.py).

3.2.2 После настройки, пользователем производится запуск **в отдельном потоке** алгоритма в бесконечном цикле (функции **startWorking** модуля **CoinPriceAnalizer.py**) с выбранным интервалом получения и обработки данных с криптобиржи.

3.2.3 В GUI должна быть реализована возможность остановки пользователем бесконечного цикла работы посредством нажатия кнопки.

3.2.4 Получение данных должно осуществляться через вызов **асинхронной (неблокирующей)** функции **startParsing** модуля **Parser.py** через программный интерфейс криптоплощадки (API).

3.2.5 Далее, полученные от API данные в формате JSON (он же dictionary в python) должны поступать на вход модуля **CoinPriceAnalizer.py**, где происходит их обработка в соответствии с заданными в GUI критериями, а также с учетом статистических данных по выбранной валютной паре, запрашиваемых из файла базы данных посредством вызова **асинхронной (неблокирующей)** функции **getCoinStatistics** модуля **DataBase.py**.

3.2.6 Одновременно с работой по п.3.2.5, данные, полученные модулем **CoinPriceAnalizer.py** от модуля **Parser.py,** должны транслироваться в неизменном виде (формат JSON) в модуль GUI (**CryptoMonkeyGui.py**), где должен производиться их рендеринг.

3.2.7 После обработки полученных от парсера и БД данных по выбранной валютной паре, при соответствии заданным пользователем критериям, должен производиться запуск криптоброкера (функции **startTrade** модуля **CryptoCoinBroker.py**) в **блокирующем поток** режиме: пока запущена функция **startTrade** модуля **CryptoCoinBroker.py** другие операции всех модулей **данного потока** не должны выполняться.

3.2.8 После совершения сделки криптоброкером, должен производиться автоматический запуск функции **confirmateTransaction** модуля **CryptoCoinBroker.py** для осуществления запроса и получения ответа через API криптобиржи о состоянии баланса пользователя.

3.2.9 После получения ответа по п.3.2.8, его результаты, а также параметры запуска функции **startTrade,** должны передаваться для записи в файл базы данных посредством запуска функции **writeTransactionData** модуля **DataBase.py.**

3.2.10 Модуль **DataBase.py** должен обеспечивать:

- обработку запроса от модуля **CoinPriceAnalizer.py** на выдачу из файла БД статистических данных по выбранной валютной паре посредством **асинхронной (неблокирующей)** функции **getCoinStatistics**;

- выдачу в модуль **CoinPriceAnalizer.py** статистических данных по выбранной валютной паре, посредством **асинхронной (неблокирующей)** функции **getCoinStatistics**;

- обработку запросов от модуля **CryptoMonkeyGui.py** на получение статистических сведений по выбранной валютной паре (функция **getCoinStatistics** модуля **DataBase.py**) и статистических сведений по выполненным криптоброкером транзакциям (функция **getBrokerTransStats** модуля **DataBase.py**);

- выдачу данных сведений из файла базы данных в GUI.

Спецификация на разрабатываемые модули и входные параметры вызываемых функций представлены в таблице 1.

Таблица 1 - Спецификация на разрабатываемые модули

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер модуля** | **Наименование модуля** | **Назначение модуля** | **Название периферийного модуля** | |
| **Тип взаимодействия с периферийным модулем** | **Типы данных** |
| 1 | CryptoMonkeyGui.py | Графический интерфейс пользователя | **1.1 CoinPriceAnalizer.py:** | |
| Start working:  Запуск в отдельном потоке импортированной функции **startWorking** модуля **CoinPriceAnalizer**.  Момент выдачи:  При нажатии кнопки ЗАПУСК, циклически, через установленный в GUI интервал времени. Прекращение цикла по нажатию кнопки ОСТАНОВИТЬ. | Функция **startWorking** модуля **CoinPriceAnalizer**.  Входные параметры функции при запуске (атрибуты класса UI модуля GUI):  **- exchange: str** (выбранная криптобиржа);  **- baseCurrency: str** (выбранная базовая валюта);  **- quoteCurrency: str** (выбранная котируемая валюта);  **- buyPrice: float** (цена покупки базовой валюты);  **- sellPrice: float** (цена продажи);  **- userToken:str** (ключ-токен от личного кабинета пользователя). |
| Data for rendering:  Получение текущих данных по выбранной валютной паре до анализа для отрисовки в GUI.  Момент получения:  в каждом цикле работы функции **startWorking** модуля **CoinPriceAnalizer**. | Именованная переменная:  **currencyData: dict** (данные по валютной паре) |
| **1.2 DataBase.py:** | |
| Request for data  Асинхронный  (неблокирующий) запуск импортированной функции **getCoinStatistics** илифункции **getBrokerTransStats** модуля **DataBase**. | Функция **getCoinStatistics** илифункция **getBrokerTransStats** модуля **DataBase.py**.  Входные параметры функции **getCoinStatistics** при запуске:  **- exchange: str** (выбранная криптобиржа);  - **baseCurrency: str** (выбранная базовая валюта);  - **quoteCurrency: str** (выбранная котируемая валюта).  В функции **getBrokerTransStats** при запуске параметры отсутствуют. |
| DB data for rendering  Получение данных из БД в соответствии с сформированным запросом | Именованная переменная:  **coinsStats: dict** (данные по валютной паре) |
| 2 | CoinPriceAnalizer.py | Модуль анализа и реализации узлового взаимодействия GUI с остальной периферией | **2.1 CryptoMonkeyGui.py:** | |
| Start working:  См. п.1.1 |  |
| Data for rendering:  См. п.1.1 |  |
| **2.2 Parser.py:** | |
| Start parsing:  Асинхронный  (неблокирующий) запуск импортированной функции **startParsing** модуля **Parser**.  Момент выдачи:  Сразу при запуске главной функции модуля от GUI | Функция **startParsing** модуля **Parser**.  Входные параметры функции при запуске:  **- exchange: str** (выбранная криптобиржа);  **- baseCurrency: str** (выбранная базовая валюта);  **- quoteCurrency: str** (выбранная котируемая валюта). |
| Data for analisis:  Получение текущих данных по выбранной валютной паре от модуля **Parser**.  Момент получения:  При каждом запуске функции **startParsing** модуля **Parser**. | Именованная переменная:  **currencyData: dict** (данные по валютной паре) |
| **2.3 CryptoCoinBroker.py** | |
| Start buy/sell coin:  Блокирующий поток запуск импортированной функции **startTrade** модуля **CryptoCoinBroker**.  Момент выдачи:  После получения данных из модул**я Parser** и обработки в соответствии с алгоритмом анализа. | Функция **startTrade** модуля **CryptoCoinBroker**.  Входные параметры функции при запуске:  **- currencyData: dict** (данные по валютной паре);  - **exchange: str** (выбранная криптобиржа);  - **userToken:str** (ключ-токен от личного кабинета пользователя);  - **orderType: str** (тип сделки – **buy**(продажа)/**sell**(покупка)) |
| * 1. **DataBase.py** | |
| Request statistics data for analisis  Асинхронный  (неблокирующий) запуск импортированной функции **getCoinStatistics** модуля **DataBase**.  Момент выдачи: сразу при получении  данных из модул**я Parser.** | Функция **getCoinStatistics** модуля **DataBase.py**.  Входные параметры функции при запуске:  **- exchange: str** (выбранная криптобиржа);  - **baseCurrency: str** (выбранная базовая валюта);  - **quoteCurrency: str** (выбранная котируемая валюта). |
| Response data for analisis:  Получение статистических данных из модуля **DataBase** для анализа.  Момент получения:  При каждом запуске функции **getCoinStatistics** модуля **DataBase**. | Именованная переменная:  **coinsStats: dict** (данные по валютной паре) |
| Currency pare statistics data  Асинхронный  (неблокирующий) запуск импортированной функции **writeCoinStatistics** модуля **DataBase**.  Момент выдачи: сразу при получении  данных из модуля **Parser.** | Функция **writeCoinStatistics** модуля **DataBase**.  Входные параметры функции при запуске:  - **exchange: str** (выбранная криптобиржа):  **- currencyData: dict** (данные по валютной паре). |
| 3 | Parser.py | Модуль получения данных по выбранной валютной паре | **3.1 CoinPriceAnalizer.py** | |
| Start parsing:  См. п.2.2 |  |
| Data for analisis:  См. п.2.2 |  |
| **3.2 API выбранной криптобиржи** | |
| Request coin data:  Выдача get-запроса к API криптобиржи на получение данных выбранной валютной пары.  Момент выдачи:  сразу при запуске функции **startParsing** из модул**я CoinPriceAnalizer.** | url-адрес API и параметры get-запроса в соответствии с официальной документацией на API на сайте выбранной криптобиржи. |
| Response coin data  Получение ответа от API криптобиржи в формате JSON в соответствии с сделанным get-запросом.  Момент получения:  В момент выдачи от API сервера криптобиржи. | Именованная переменная:  **currencyData: dict** (данные по валютной паре) |
| 4 | CryptoCoinBroker.py | Модуль покупки/продажи активов в личном кабинете пользователя |  |  |
| 5 | DataBase.py | Модуль взаимодействия с файлом базы данных |  |  |

**Рисунок 1 - Структурная схема построения программы**

Response data for analisis

Request statistics data for analisis

Currency pare statistics data

API

DB Data for rendering

Request for data

**Crypto**

**exchange**

Data for analisis

Data for

rendering

Start working

**Data base**

**GUI**

User data after operation

Response

coin data Request

Request coin data

User data response

User data request

Buy/sell request

Start buy/sell coin

Start parsing

**Coin price analizer**

|  |
| --- |
| **Class cryptoBroker** |
| **getAccountId** |
| **getCurrencyInfo** |
| **getBalance** |
| **startTrade** |
| **cancelOrder** |

**list: userBalances**

**async def getBalance**

**CryptoCoin broker**

**def startTrade**

**dict: currencyData**

**async def getCurrenciesInfo**

**Приложение 1 - Код инициализации и настройки хэндлеров**

**в главном модуле для всех логгеров приложения**

import logging

def init\_logger(name):  
 logger = logging.getLogger(name)  
 FORMAT = "%(asctime)s -- %(name)s -- %(lineno)s -- %(levelname)s -- %(message)s"  
 logger.setLevel(logging.DEBUG)  
 sh = logging.StreamHandler()  
 sh.setFormatter(logging.Formatter(FORMAT))  
 sh.setLevel(logging.DEBUG)  
 fh = logging.FileHandler(filename="Logs.txt", encoding="UTF-8")  
 fh.setFormatter(logging.Formatter(FORMAT))  
 fh.setLevel(logging.DEBUG)  
 logger.addHandler(sh)  
 logger.addHandler(fh)  
 logger.debug('logger was initialized')  
  
init\_logger('CryptoMonkey')

logger = logging.getLogger('CryptoMonkey.CryptoMonkeyGui')

**Приложение 2 - Алгоритм**

T = ∞

T = ∞

T = 4 часа

T = 4 часа

Нет Да

Падение цены на >=5% от зафиксированной

Нет Да

Падение цены на >=5% от зафиксированной

**Фиксация цены**.

Продажа 50% оставшихся монет

**Фиксация цены**.

Продажа 50% монет

Нет

Рост цены на >=10% от зафиксированной

Да

Уход в долгосрок

Нет

Падение цены на >=5% от зафиксированной Да

**Фиксация цены**.

Покупка дополнительно на 25% установленной суммы

Нет

Рост цены на >=10% от зафиксированной

Да

Нет

Падение цены на >=5% от зафиксированной

Да

**Фиксация цены**.

Покупка дополнительно на 25% установленной суммы

Да

Рост цены на >=10% от закупочной

Нет

T = ∞

Запуск функции

**startWorking**

модуля **CoinPriceAnalizer.py**

1.**Фиксация цены** выбранной базовой валюты

2.Мониторинг цены выбранной базовой валюты (**baseCurrency**).

Да

Падение цены на >=5%

Нет

Да

Падение цены на >=5%

Нет

Нет

Покупка на установленную пользователем сумму

Да

Рост цены на >=10%

Нет

Да

Рост цены на >=10%

Нет

Продажа всех оставшихся монет