

## Техническое задание

на разработку программы для автоматической торговли по сигналам на языке программирования PHP.

Telegram RUS

<https://t.me/joinchat/F16v6U3VVb6oCc10di4DVQ>

## Требования

PHP

MySQL

PHP-cURL (В базовом комплекте php не поставляется, устанавливается отдельно)

## Назначение программы

Торговля по сигналам на криптовалютной бирже binance в автоматическом режиме.

## Логика программы

- 1 Получение сигналов об открытии позиций
- 2 Сравнение сигналов с данными открытых позиций в хранилище  
Если такой позиции не открыто - далее п.3. Если позиция открыта - п.7.
- 3 Получение данных о котировках
- 4 Получение данных о отправленных ордерах  
Если ордера не исполнены - отменяем ордера. Далее п.5.  
Если ордера отсутствуют, далее п.5  
Если ордера исполнены - записывание данных об открытии позиции в хранилище, удаление данных об отправленных ордерах из хранилища, далее п.7.
- 5 Сравнение данных о балансах необходимых для открытия позиции  
Если средств хватает - далее п.6. Иначе п.7.
- 6 Отправка ордера на покупку, запись в хранилище данных об отправленных ордерах
- 7 Сравнение котировок и вычисление профита по открытым позициям  
Если профит растет, обновляем данные по открытой позиции в хранилище  
Если профит падает, проверяем настройки выхода  
Если условие выхода выполнено, закрываем позицию, отправляем ордер продажи, удаляем запись по открытой позиции из хранилища, записываем в хранилище данные по получению профита.  
... Далее п.1

## Состав программы

- Основной код  
*Обработка:*
  - получение данных о балансах пользователя биржи через API
  - получение данных котировках биржи через API
  - получение данных о выполненных ордерах пользователя биржи через API
  - получение данных о неисполненных ордерах пользователя биржи через API
  - получение данных о сигналах через API
  - вычисление данных об открытых позициях
  - сравнение правил биржи и настроек пользователя
  - вычисление контрольных правил для открытия/закрытия позиций
  - отправка ордеров через API*Хранение:*
  - хранение настроек биржи в MySQL
  - хранение настроек пользователя в MySQL
  - хранение данных об открытых ордерах в MySQL
  - хранение данных об открытых ордерах в MySQL
- Вспомогательный код  
*Обработка:*
  - получение данных о настройках пользователя из MySQL
  - запись данных о настройках пользователя в MySQL
  - получение данных настройках биржи через API

## Описание функций

### 1 Хранение настроек пользователя MySQL

Таблица **settings**

| <u>Поле</u> | <u>Тип</u>     | <u>Обязательный</u> | <u>Описание</u>   |
|-------------|----------------|---------------------|---|
| id          | Целое число    | Да                  | Номер комплекта настроек                                  |
| api_key     | Строка         | Да                  | API ключ для торговли                                     |
| api_secret  | Строка         | Да                  | API секретный ключ для торговли                           |
| open_url    | Строка         | Да                  | Ссылка для получения сигналов открытия позиции            |
| amount      | Число с точкой | Да                  | Размер позиции для открытия позиции. Эквивалентное в USDT |

|            |                |     |   |
|------------|----------------|-----|---|
| fee        | Число с точкой | Да  | Размер комиссии биржи за совершение сделки  |
| delta      | Число с точкой | Да  | Размер погрешности, необходимый для вычисления соответствия ордера  |
| pause      | Целое число    | Да  | Время паузы в секундах, которое будет выдержано после открытия позиции до начала действия проверок выхода. Также время паузы необходимо для работы с сигналами. |
| close_url  | Строка         | Нет | Ссылка для получения сигналов закрытия позиции  |
| close_time | Целое число    | Нет | Время в секундах, после которого позиция закрывается  |
| close_lose | Число с точкой | Нет | Значение падения профита от максимального значения профита в процентах, ниже которого позиция закрывается   |
| close_exit | Число с точкой | Да  | Значение падения цены котировки в процентах, ниже которого позиция закрывается  |

## 2 Хранение ордеров

### MySQL

Таблица **orders**

| <u>Поле</u> | <u>Тип</u>     | <u>Описание</u>                           |
|-------------|----------------|---|
| id          | Целое число    | Номер по порядку                          |
| symbol      | Строка         | Обозначение котировки (BTCUSDT)           |
| order_id    | Целое число    | Номер ордера на бирже                     |
| price       | Число с точкой | Цена                                      |
| qty         | Число с точкой | Количество                                |
| side        | Строка         | Направление сделки (BUY или SELL)         |
| open_time   | Целое          | Время отправки ордера в формате timestamp |

|           |             |   |
|-----------|-------------|---|
|           | число       |   |
| position  | Целое число | При открытии позиции = 0<br>При закрытии позиции = номер позиции, которую закрывает ордер |
| close_res | Строка      | Причина закрытия (URL, TIME, LOSE, EXIT)  |
| set_id    | Целое число | Номер настройки пользователя по которой работает торговая программа                       |

### 3 Хранение открытых позиций

MySQL

Таблица **positions**

| <u>Поле</u> | <u>Тип</u>     | <u>Описание</u>  |
|-------------|----------------|--|
| id          | Целое число    | Номер по порядку   |
| symbol      | Строка         | Обозначение котировки (BTCUSDT)  |
| price       | Число с точкой | Цена открытия позиции  |
| qty         | Число с точкой | Количество   |
| open_time   | Целое число    | Время открытия позиции в формате timestamp   |
| max_price   | Число с точкой | Максимальная цена.<br>Значение максимальной цены обновляется в таблице если текущее значение больше предыдущего. |
| set_id      | Целое число    | Номер настройки пользователя по которой работает торговая программа  |

### 4 Хранение закрытых позиций

MySQL

Таблица **history**

| <u>Поле</u> | <u>Тип</u>  | <u>Описание</u>  |
|-------------|-------------|------------------|
| id          | Целое число | Номер по порядку |

| symbol      | Строка         | Обозначение котировки (BTCUSDT)   |
|-------------|----------------|---|
| open_price  | Число с точкой | Цена открытия позиции   |
| close_price | Число с точкой | Цена закрытия позиции   |
| qty         | Число с точкой | Количество  |
| open_time   | Целое число    | Время открытия позиции в формате timestamp                                    |
| close_time  | Целое число    | Время закрытия позиции в формате timestamp                                    |
| close_res   | Строка         | Причина закрытия позиции (URL, TIME, LOSE, EXIT)                              |
| set_id      | Целое число    | Номер настройки пользователя по которой торговая программа отработала позицию |

## 5 Хранение параметров торгуемых пар биржи MySQL

Таблица **pairs**

| <u>Поле</u> | <u>Тип</u>     | <u>Описание</u>            |
|-------------|----------------|----------------------------|
| id          | Целое число    | Номер по порядку           |
| symbol      | Строка         | Обозначение торгуемой пары |
| base        | Строка         | Основная валюта            |
| quote       | Строка         | Котируемая валюта          |
| min_price   | Число с точкой | Минимальная цена           |
| max_price   | Число с точкой | Максимальная цена          |
| step_price  | Число с точкой | Шаг цены                   |
| min_qty     | Число с        | Минимальный объем          |

| точкой   |                |                    |
|----------|----------------|--------------------|
| max_qty  | Число с точкой | Максимальный объем |
| step_qty | Число с точкой | Шаг объема         |

## 6 Получение сигналов

### 6.1 Получение сигналов через API биржи и вывод в API торговой программы

RНР

#### screener\_api.php

Получение сигналов происходит при обращении к странице и передаче параметров поиска. В результате страница выдает ответ в формате JSON.

Запрос\*

```
GET http://funnymay.com/api/screener_api.php
```

Параметры

| Параметр | Тип    | Обязательный | Описание                                    |
|----------|--------|--------------|---|
| bs       | Строка | Нет          | Базовая валюта                              |
| qt       | Строка | Нет          | Котируемая валюта                           |
| vr       | Строка | Нет          | Относительный объем                         |
| ch       | Строка | Нет          | Изменение в %                               |
| qr       | Строка | Нет          | Эквивалент в USDT                           |
| p10      | Строка | Нет          | Положение цены относительно SMA10           |
| p20      | Строка | Нет          | Положение цены относительно SMA20           |
| p50      | Строка | Нет          | Положение цены относительно SMA50           |
| p100     | Строка | Нет          | Положение цены относительно SMA100          |
| p200     | Строка | Нет          | Положение цены относительно SMA200          |
| hl50d    | Строка | Нет          | Положение цены относительно Hi-Lo 50 дней   |
| hl52w    | Строка | Нет          | Положение цены относительно Hi-Lo 52 недель |
| m        | Строка | Нет          | Соотношение майера                          |

| d | Строка | Нет | Дни подряд |
|---|--------|-----|------------|
|---|--------|-----|------------|

Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1,
  "data": [
    {
      "symbol":"EOSETH",
      "price":0.32914,
      "change":5.745,
      "eq_usdt":45.0123
    },
    { ... }
  ]
}
```

*Примечание:* Настройку строки запроса можно осуществить на сайте <http://funnymay.com/screener.php>  
API-ссылка генерируется внизу страницы.

## 6.2 Получение данных от API торговой программы и расшифровка PHP

**main.php**

метод **get\_signals**

Функция основного кода передает запрос к странице **screener\_api.php**, получает данные в формате JSON и расшифровывает их и сохраняет на время цикла в памяти.

## 7 Получение балансов пользователя

7.1 Получение балансов пользователя через API биржи и вывод в API торговой программы

PHP

**binance\_api.php**

Получение балансов происходит при обращении к странице и передаче необходимых параметров. В результате страница выдает ответ в формате JSON.

## Запрос

GET /api/binance\_api.php

## Параметры

| Параметр   | Тип    | Обязательный | Описание                        |
|------------|--------|--------------|---------------------------------|
| mode       | Строка | Да           | getBalances                     |
| api_key    | Строка | Да           | API ключ для торговли           |
| api_secret | Строка | Да           | API секретный ключ для торговли |

## Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1,
  "data": {
    "BTC":0.01,
    "LTC":1.2345,
    "ETH":0.0,
    ...
  }
}
```

## 7.2 Получение данных от API торговой программы и расшифровка

PHP

**main.php**метод **get\_balances**

Функция основного кода передает запрос к странице **binance\_api.php**, получает данные в формате JSON и расшифровывает их и сохраняет на время цикла в памяти.

**8 Получение активных ордеров пользователя**

8.1 Получение активных ордеров пользователя через API биржи и вывод в API торговой программы

PHP



**binance\_api.php**

Получение активных ордеров происходит при обращении к странице и передаче необходимых параметров. В результате страница выдает ответ в формате JSON.

## Запрос

```
GET /api/binance_api.php
```

## Параметры

| <u>Параметр</u> | <u>Тип</u> | <u>Обязательный</u> | <u>Описание</u>                 |
|-----------------|------------|---------------------|---------------------------------|
| mode            | Строка     | Да                  | getOrders                       |
| api_key         | Строка     | Да                  | API ключ для торговли           |
| api_secret      | Строка     | Да                  | API секретный ключ для торговли |

## Ответ

```
{
  "status":0
}
```

## или

```
{
  "status":1,
  "data": [
    {
      "id":12345,
      "symbol":"BTCUSDT",
      "side":"BUY",
      "qty":0.01,
      "price": 7700.0,
      "filled":0.0,
      "open_time": 1546008440
    },
    { ... }
  ]
}
```

**8.2 Получение данных от API торговой программы и расшифровка****RНР****main.php**

метод **get\_orders**

Функция основного кода передает запрос к странице **binance\_api.php**, получает данные в формате JSON и расшифровывает их и сохраняет на время цикла в памяти.

## 9 Получение котировок

9.1 Получение котировок через API биржи и вывод в API торговой программы

PHP

**binance\_api.php**

Запрос

```
GET /api/binance_api.php
```

Параметры

| <u>Параметр</u> | <u>Тип</u> | <u>Обязательный</u> | <u>Описание</u> |
|-----------------|------------|---------------------|-----------------|
| mode            | Строка     | Да                  | getPrices       |

Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1,
  "data": {
    "BTCUSDT": 7509.95,
    "ETHUSDT": 130.54,
    "LTCUSDT": 75.0,
    ...
  }
}
```

9.2 Получение данных от API торговой программы и расшифровка

PHP

**main.php**

метод **get\_prices**

Функция основного кода передает запрос к странице **binance\_api.php**, получает данные в формате JSON и расшифровывает их и сохраняет на время цикла в памяти.

## 10 Отправка ордера

### 10.1 Работа API торговой программы

PHP

**binance\_api.php**

Запрос

```
GET /api/binance_api.php
```

Параметры

| <u>Параметр</u> | <u>Тип</u>     | <u>Обязательный</u> | <u>Описание</u>                 |
|-----------------|----------------|---------------------|---------------------------------|
| mode            | Строка         | Да                  | sendOrder                       |
| api_key         | Строка         | Да                  | API ключ для торговли           |
| api_secret      | Строка         | Да                  | API секретный ключ для торговли |
| symbol          | Строка         | Да                  | Обозначение торговой пары       |
| price           | Число с точкой | Да                  | Цена покупки или продажи        |
| qty             | Число с точкой | Да                  | Объем покупки или продажи       |
| side            | Строка         | Да                  | Направление сделки (BUY, SELL)  |

Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1,
  "order_id":1
}
```

### 10.2 Работа основного кода программы

PHP

**main.php**

После отправки ордера и получении ответа [**status**] = 1 записываем в хранилище orders данные об отправленном ордере.

## 11 Отмена ордера

### 11.1 Работа API торговой программы

PHP

**binance\_api.php**

Запрос

```
GET /api/binance_api.php
```

Параметры

| <u>Параметр</u> | <u>Тип</u> | <u>Обязательный</u> | <u>Описание</u>                 |
|-----------------|------------|---------------------|---------------------------------|
| mode            | Строка     | Да                  | cancelOrder                     |
| api_key         | Строка     | Да                  | API ключ для торговли           |
| api_secret      | Строка     | Да                  | API секретный ключ для торговли |
| order_id        | Строка     | Да                  | Номер ордера на бирже           |

Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1
}
```

### 11.2 Работа основного кода программы

PHP

**main.php**

После получении ответа [**status**] = 1 удаляем из хранилища orders данные об отправленном ордере.

## 12 Получение параметров биржи

### 12.1 Получение параметров биржи через API биржи и вывод в API торговой программы

PHP

**binance\_api.php**

## Запрос

GET /api/binance\_api.php

## Параметры

| <u>Параметр</u> | <u>Тип</u> | <u>Обязательный</u> | <u>Описание</u> |
|-----------------|------------|---------------------|-----------------|
| mode            | Строка     | Да                  | getRules        |

## Ответ

```
{
  "status":0
}
```

или

```
{
  "status":1,
  "data": [
    {
      "symbol": "ETHBTC",
      "base": "ETH",
      "quote": "BTC",
      "min_price": 0.00000100,
      "max_price": 100000.00000000,
      "step_price": 0.00000100,
      "min_qty": 0.00100000,
      "max_qty": 100000.00000000,
      "step_qty": 0.00100000
    },
    { ... }
  ]
}
```

## 12.2 Получение данных от API торговой программы, расшифровка и хранение PHP

**main.php**метод **get\_rules**

Функция основного кода передает запрос к странице **binance\_api.php**, получает данные в формате JSON, расшифровывает их и сохраняет в хранилище rules.

### 13 Обработка сигналов

PHP

main.php

Функция основного кода.

- Расшифровка данных **signals\_api**
- Получение из хранилища **positions** записей совпадающих по полю **symbol**
- Если запись существует, то проверяем стоит ли она на паузе.
  - Если пауза еще действует, то докупка позиции запрещается. Конец.
  - Если время паузы истекло, то позиция докупается. Отправляется ордер на покупку. Конец.
- Если запись не существует, то проверяем, отправлен ли у нас ордер на покупку. Получение из хранилища **orders** записей совпадающих по полю **symbol**
  - Если ордер отправлен, проверяем соответствует ли он критериям сигналов
    - Если цена ордера вышла за рамки **delta** в %, то отменяем данные ордер и отправляем новый, по данным из сигналов; удаляем из хранилища **orders** старую запись об неисполненном ордере и создаем новую запись.
    - Если цена ордера не вышла за рамки **delta** в %, конец.
  - Если ордер отсутствует в хранилище, отправляем новый ордер на покупку.

### 14 Обработка параметров покупки и продажи

PHP

main.php

Функция основного кода

- Расшифровка данных метода **get\_balances**
- Проверка достаточности средств для совершения сделки
- Проверка размера цены посылаемого ордера критериям **min\_price**, **max\_price** и **step\_price** биржи
- Проверка размера объема посылаемого ордера критериям **min\_qty**, **max\_qty** и **step\_qty** биржи

### 15 Обработка отправленных ордеров

PHP

main.php

Функция основного кода

- Расшифровка данных об ордерах на бирже из **get\_orders**
- Получение данных по отправленным ордерам в хранилище **orders**
- Проверка открывает или закрывает ордер позицию.
  - Если открывает, проверяем исполнился ордер на бирже или нет

- Если ордер исполнился на бирже, удаляем ордер из хранилища **orders** и создаем запись об открытии позиции в **positions**
- Если ордер не исполнился, конец.
- Если ордер закрывает позицию, проверяем исполнился он или нет
  - Если ордер исполнился на бирже, удаляем ордер из хранилища **orders**, удаляем запись в хранилище **positions** и создаем запись об завершенной позиции в **history**
  - Если ордер не исполнился, конец.

## 16 Обработка открытых позиций

PHP

main.php

метод **check\_positions**

Функция основного кода

- Расшифровка данных о котировках из **get\_prices**
- Получение данных по открытым позициям в хранилище **positions**.
- Проверка стоит ли позиция на паузе
  - Если стоит на паузе, пропускаем проверки закрытия по условиям и проверяем критерий падения цены ниже **close\_exit**.
  - Если время паузы истекло, проверяем условия выхода
    - Если предусмотрено закрытие позиции по ссылке, то получаем сигналы на закрытие по ссылке
      - Если сигнал на закрытие совпадает с открытой позицией, то позиция закрывается, иначе пропускаем закрытие по ссылке
    - Если предусмотрено закрытие по времени, проверяем сколько времени прошло
      - Если прошло времени больше, чем требовалось для выдерживания позиции, то позиция закрывается, иначе пропускаем закрытие по времени
    - Если предусмотрено закрытие по падению профита, то вычисляем отношение текущего профита к максимальному профиту за все время
      - Если профит упал ниже предусмотренного критерия, то позиция закрывается, иначе пропускаем закрытие по падению профита
- Проверяем упала ли цена ниже критической цены **close\_exit** в %.
  - Если цена упала ниже **close\_exit**, отправляем ордер на закрытие позиции.
  - Если цена выше **close\_exit**, конец.

# Создание необходимых таблиц

## Подготовка MySQL

Файл **install.sql**

Загружается через phpmyadmin

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

CREATE TABLE `settings` (
  `id` int(10) UNSIGNED NOT NULL,
  `api_key` text NOT NULL,
  `api_secret` text NOT NULL,
  `open_url` text NOT NULL,
  `amount` float UNSIGNED NOT NULL,
  `fee` float UNSIGNED NOT NULL,
  `delta` float UNSIGNED NOT NULL,
  `pause` int(10) UNSIGNED NOT NULL,
  `close_url` text NOT NULL,
  `close_time` int(10) UNSIGNED NOT NULL,
  `close_lose` float UNSIGNED NOT NULL,
  `close_exit` float UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `settings`
  ADD PRIMARY KEY (`id`);
COMMIT;

CREATE TABLE `orders` (
  `id` int(10) UNSIGNED NOT NULL,
  `symbol` text NOT NULL,
  `order_id` int(10) UNSIGNED NOT NULL,
  `price` float UNSIGNED NOT NULL,
  `qty` float UNSIGNED NOT NULL,
  `side` text NOT NULL,
  `open_time` int(10) UNSIGNED NOT NULL,
  `position` int(10) UNSIGNED NOT NULL,
  `close_res` text NOT NULL,
  `set_id` int(10) UNSIGNED NOT NULL
```



```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `orders`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `orders`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;

CREATE TABLE `positions` (
  `id` int(10) UNSIGNED NOT NULL,
  `symbol` text NOT NULL,
  `price` float UNSIGNED NOT NULL,
  `qty` float UNSIGNED NOT NULL,
  `open_time` int(10) UNSIGNED NOT NULL,
  `max_price` float UNSIGNED NOT NULL,
  `set_id` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `positions`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `positions`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;

CREATE TABLE `history` (
  `id` int(10) UNSIGNED NOT NULL,
  `symbol` text NOT NULL,
  `open_price` float UNSIGNED NOT NULL,
  `close_price` float UNSIGNED NOT NULL,
  `qty` float UNSIGNED NOT NULL,
  `open_time` int(10) UNSIGNED NOT NULL,
  `close_time` int(10) UNSIGNED NOT NULL,
  `close_res` text NOT NULL,
  `set_id` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `history`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `history`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;
```

```

CREATE TABLE `pairs` (
  `id` int(10) UNSIGNED NOT NULL,
  `symbol` text NOT NULL,
  `base` text NOT NULL,
  `quote` text NOT NULL,
  `min_price` float UNSIGNED NOT NULL,
  `max_price` float UNSIGNED NOT NULL,
  `step_price` float UNSIGNED NOT NULL,
  `min_qty` float UNSIGNED NOT NULL,
  `max_qty` float UNSIGNED NOT NULL,
  `step_qty` float UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `pairs`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `pairs`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT;
COMMIT;

```

В результате получаем следующие таблицы

Таблица **settings**


| #  | Имя  | Тип     | Сравнение       | Атрибуты | Null | По умолчанию | Дополнительно |
|----|--|---------|-----------------|----------|------|--------------|---------------|
| 1  | id  | int(10) |                 | UNSIGNED | Нет  | Нет          |               |
| 2  | api_key  | text    | utf8_general_ci |          | Нет  | Нет          |               |
| 3  | api_secret   | text    | utf8_general_ci |          | Нет  | Нет          |               |
| 4  | open_url   | text    | utf8_general_ci |          | Нет  | Нет          |               |
| 5  | amount   | float   |                 | UNSIGNED | Нет  | Нет          |               |
| 6  | fee  | float   |                 | UNSIGNED | Нет  | Нет          |               |
| 7  | delta  | float   |                 | UNSIGNED | Нет  | Нет          |               |
| 8  | pause  | int(10) |                 | UNSIGNED | Нет  | Нет          |               |
| 9  | close_url  | text    | utf8_general_ci |          | Нет  | Нет          |               |
| 10 | close_time   | int(10) |                 | UNSIGNED | Нет  | Нет          |               |
| 11 | close_lose   | float   |                 | UNSIGNED | Нет  | Нет          |               |
| 12 | close_exit   | float   |                 | UNSIGNED | Нет  | Нет          |               |

Таблица **orders**

| #  | Имя       | Тип     | Сравнение       | Атрибуты | Null | По умолчанию | Дополнительно  |
|----|-----------|---------|-----------------|----------|------|--------------|----------------|
| 1  | id 🔑      | int(10) |                 | UNSIGNED | Нет  | Нет          | AUTO_INCREMENT |
| 2  | symbol    | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 3  | order_id  | int(10) |                 | UNSIGNED | Нет  | Нет          |                |
| 4  | price     | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 5  | qty       | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 6  | side      | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 7  | open_time | int(10) |                 | UNSIGNED | Нет  | Нет          |                |
| 8  | position  | int(10) |                 | UNSIGNED | Нет  | Нет          |                |
| 9  | close_res | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 10 | set_id    | int(10) |                 | UNSIGNED | Нет  | Нет          |                |

Таблица **positions**

| # | Имя       | Тип     | Сравнение       | Атрибуты | Null | По умолчанию | Дополнительно  |
|---|-----------|---------|-----------------|----------|------|--------------|----------------|
| 1 | id 🔑      | int(10) |                 | UNSIGNED | Нет  | Нет          | AUTO_INCREMENT |
| 2 | symbol    | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 3 | price     | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 4 | qty       | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 5 | open_time | int(10) |                 | UNSIGNED | Нет  | Нет          |                |
| 6 | max_price | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 7 | set_id    | int(10) |                 | UNSIGNED | Нет  | Нет          |                |

Таблица **history**

| # | Имя         | Тип     | Сравнение       | Атрибуты | Null | По умолчанию | Дополнительно  |
|---|-------------|---------|-----------------|----------|------|--------------|----------------|
| 1 | id 🔑        | int(10) |                 | UNSIGNED | Нет  | Нет          | AUTO_INCREMENT |
| 2 | symbol      | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 3 | open_price  | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 4 | close_price | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 5 | qty         | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 6 | side        | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 7 | open_time   | int(10) |                 | UNSIGNED | Нет  | Нет          |                |
| 8 | close_res   | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 9 | set_id      | int(10) |                 | UNSIGNED | Нет  | Нет          |                |

Таблица **pairs**

| #  | Имя  | Тип     | Сравнение       | Атрибуты | Null | По умолчанию | Дополнительно  |
|----|--|---------|-----------------|----------|------|--------------|----------------|
| 1  | id  | int(10) |                 | UNSIGNED | Нет  | Нет          | AUTO_INCREMENT |
| 2  | symbol   | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 3  | base   | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 4  | quote  | text    | utf8_general_ci |          | Нет  | Нет          |                |
| 5  | min_price  | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 6  | max_price  | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 7  | step_price   | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 8  | min_qty  | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 9  | max_qty  | float   |                 | UNSIGNED | Нет  | Нет          |                |
| 10 | step_qty   | float   |                 | UNSIGNED | Нет  | Нет          |                |

## Основной код программы

### Описание работы всех функций

*/\*с комментариями\*/*

#### Файл **mysql5.php**

Класс работы с базой данных MySQL для php версии 5.5 и ранее

```
<?php
class db
{
    var $db_id = false;
    var $mysql_error = '';
    var $mysql_error_num = 0;

    function connect($db_user='DBUSER', $db_pass='DBPASS', $db_name='DBNAME', $db_location =
'DBHOST', $show_error=1)
    {
        if(!$this->db_id = @mysql_connect($db_location, $db_user, $db_pass)) {
            if($show_error == 1) {
                $this->display_error(mysql_error(), mysql_errno());
            } else {
                return false;
            }
        }
        if(!@mysql_select_db($db_name, $this->db_id)) {
            if($show_error == 1) {
                $this->display_error(mysql_error(), mysql_errno());
            } else {
                return false;
            }
        }
    }
}
```

```

    }
}
return true;
}

function query($query, $show_error=false)
{
    if(!$this->db_id) $this->connect(DBUSER, DBPASS, DBNAME, DBHOST);
    if(!($result = mysql_query($query, $this->db_id) )) {
        $this->mysql_error = mysql_error();
        $this->mysql_error_num = mysql_errno();
        if($show_error) {
            $this->display_error($this->mysql_error, $this->mysql_error_num, $query);
        }
    }
    return $result;
}

function get_row($query_result)
{
    return mysql_fetch_assoc($query_result);
}

function get_array($query_result)
{
    return mysql_fetch_array($query_result);
}

function get_object($query_result)
{
    return mysql_fetch_object($query_result);
}

function super_query($query, $multi = false)
{
    if(!$this->db_id) $this->connect(DBUSER, DBPASS, DBNAME, DBHOST);
    if(!$multi) {
        return $this->get_row($this->query($query));
    } else {
        $query_result = $this->query($query);
        $rows = array();
        while($row = $this->get_row($query_result)) {
            $rows[] = $row;
        }
        return $rows;
    }
}

function num_rows($query_result)
{
    return mysql_num_rows($query_result);
}

function insert_id()
{
    return mysql_insert_id($this->db_id);
}

function get_result_fields($result) {
    while ($field = mysql_fetch_field($result))
    {
        $fields[] = $field;
    }
    return $fields;
}

function close()

```

```

{
    @mysql_close($this->db_id);
}

function display_error($error, $error_num, $query = '')
{
    if($query) {
        $query = preg_replace("/([0-9a-f]){32}/", "*****",
$query);
        $query_str = "$query";
    }

    echo '<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>MySQL Fatal Error</title>

<style type="text/css">
<!--
body {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    font-style: normal;
    color: #000000;
}
-->
</style>
</head>
<body>
    <font size="4">MySQL Error!</font>
    <br />-----<br />
    <br />

    <u>The Error returned was:</u>
    <br />
        <strong>' . $error . '</strong>

    <br /><br />
    </strong><u>Error Number:</u>
    <br />
        <strong>' . $error_num . '</strong>
    <br />
        <br />

    <textarea name="" rows="10" cols="52" wrap="virtual">' . $query_str . '</textarea><br />

</body>
</html>';

    exit();
}
}
?>

```

## Файл mysql7.php

Класс работы с базой данных MySQL для php версии 6.0 и позднее

```

<?php
class db
{
    var $db_id = false;
    var $mysqli_error = '';
    var $mysqli_error_num = 0;

```

```

function connect($db_user='DBUSER', $db_pass='DBPASS', $db_name='DBNAME', $db_location =
'DBHOST', $show_error=1)
{
    if(!$this->db_id = @mysqli_connect($db_location, $db_user, $db_pass, $db_name)) {
        if($show_error == 1) {
            $this->display_error(mysqli_error(), mysqli_errno());
        } else {
            return false;
        }
    }
    return true;
}

function query($query, $show_error=false)
{
    if(!$this->db_id) $this->connect(DBUSER, DBPASS, DBNAME, DBHOST);
    if(!($result = mysqli_query($this->db_id, $query) )) {
        $this->mysqli_error = mysqli_error($this->db_id);
        $this->mysqli_error_num = mysqli_errno($this->db_id);
        if($show_error) {
            $this->display_error($this->mysqli_error, $this->mysqli_error_num, $query);
        }
    }
    return $result;
}

function get_row($query_result)
{
    return mysqli_fetch_assoc($query_result);
}

function get_array($query_result)
{
    return mysqli_fetch_array($query_result);
}

function get_object($query_result)
{
    return mysqli_fetch_object($query_result);
}

function super_query($query, $multi = false)
{
    if(!$this->db_id) $this->connect(DBUSER, DBPASS, DBNAME, DBHOST);
    if(!$multi) {
        return $this->get_row($this->query($query));
    } else {
        $query_result = $this->query($query);
        $rows = array();
        while($row = $this->get_row($query_result)) {
            $rows[] = $row;
        }
        return $rows;
    }
}

function num_rows($query_result)
{
    return mysqli_num_rows($query_result);
}

function insert_id()
{
    return mysqli_insert_id($this->db_id);
}

function get_result_fields($result) {
    while ($field = mysqli_fetch_field($result))

```

```

    {
        $fields[] = $field;
    }
    return $fields;
}

function close()
{
    @mysqli_close($this->db_id);
}

function display_error($error, $error_num, $query = '')
{
    if($query) {
        $query = preg_replace("/([0-9a-f]){32}/", "*****",
$query);
        $query_str = "$query";
    }

    echo '<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>MySQL Fatal Error</title>

<style type="text/css">
<!--
body {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    font-style: normal;
    color: #000000;
}
-->
</style>
</head>
<body>
    <font size="4">MySQL Error!</font>
    <br />-----<br />
    <br />

    <u>The Error returned was:</u>
    <br />
    <strong>'. $error. '</strong>

    <br /><br />
    </strong><u>Error Number:</u>
    <br />
    <strong>'. $error_num. '</strong>
    <br />
    <br />

    <textarea name="" rows="10" cols="52" wrap="virtual">'. $query_str. '</textarea><br />

</body>
</html>';

    exit();
}
?>

```

### Файл dbconfig.php

Файл с настройками базы данных MySQL



```

<?php
/* Нахождение баз данных MySQL */
define ("DBHOST", "localhost");
/* Название базы данных */
define ("DBNAME", "database");
/* Имя пользователя */
define ("DBUSER", "username");
/* Пароль пользователя */
define ("DBPASS", "password");

$db = new db;
$db->query("SET NAMES utf8");
?>

```

## Файл api\_binance.php

Интерфейс обмена запросов и ответов между торговой программой и биржей.

```

<?php

/* Приводим все к одному часовому поясу */
date_default_timezone_set(DateTimeZone::listIdentifiers(DateTimeZone::UTC)[0]);

/* Получение параметра типа запроса */
if(isset($_POST['mode'])) { $mode = $_POST['mode']; }
elseif(isset($_GET['mode'])) { $mode = $_GET['mode']; }
else { $mode = ''; }
$mode = htmlspecialchars(strip_tags(trim($mode)));

/* Функция передачи запроса на биржу */
function binance_query($path, $method, array $req = array()) {

    if(isset($_POST['api_key'])) { $api_key = $_POST['api_key']; }
    elseif(isset($_GET['api_key'])) { $api_key = $_GET['api_key']; }
    else { $api_key = 0; }
    if(isset($_POST['api_secret'])) { $api_secret = $_POST['api_secret']; }
    elseif(isset($_GET['api_secret'])) { $api_secret = $_GET['api_secret']; }
    else { $api_secret = 0; }
    if(isset($_POST['arg'])) { $arg = $_POST['arg']; }
    elseif(isset($_GET['arg'])) { $arg = $_GET['arg']; }
    else { $arg = 0; }

    $api_key = htmlspecialchars(strip_tags(trim($api_key)));
    $api_secret = htmlspecialchars(strip_tags(trim($api_secret)));
    $arg = htmlspecialchars(strip_tags(trim($arg)));

    $req['recvWindow'] = 5000;

    /* Корректировка времени сервера и биржи */
    $correctTime = time()*1000 - $arg;

    $req['timestamp'] = $correctTime;

    $post_data = http_build_query($req, '', '&');
    $sign = hash_hmac("sha256", $post_data, $api_secret);
    $req['signature'] = $sign;

    $post_data = http_build_query($req, '', '&');

    $ch = null;
    if (is_null($ch)) {
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/4.0 (compatible; binance PHP client;
'.php_uname('s').'; PHP/'.phpversion().')');
    }
}

```

```

if ($method == 'GET') {
    $headers = array(
        'X-MBX-APIKEY: '.$api_key,
    );
    $url = 'https://api.binance.com'.$path."?".$post_data;
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HTTPGET, 1);
}
if ($method == 'POST') {
    $headers = array(
        'X-MBX-APIKEY: '.$api_key,
        'Content-Type: application/x-www-form-urlencoded',
    );
    $url = 'https://api.binance.com'.$path;
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);
}
if ($method == 'DELETE') {
    $headers = array(
        'X-MBX-APIKEY: '.$api_key,
        'Content-Type: application/x-www-form-urlencoded',
    );
    $url = 'https://api.binance.com'.$path;
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");
}
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

$res = curl_exec($ch);

if ($res === false) {
    print "{\"status\":\"0\"}";
    exit;
}

$dec = json_decode($res, true);
if (!$dec) {
    print "{\"status\":\"0\"}";
    exit;
}
return $dec;
}

/* Получение балансов пользователя */

if ($mode == 'getBalances'){

    $result = binance_query("/api/v3/account","GET");

    if ($result['updateTime'] != null) {
        $balances = "{";
        $balances .= "\"status\":1,";
        $balances .= "\"data\":{\"";
        $count = count($result['balances']);

        for ($i = 0; $i < $count; $i++) {
            $balances .= "\"".$strtoupper($result['balances'][$i]['asset'])."\": ";
            $balances .= $result['balances'][$i]['free'].",";
        }
        $balances = substr($balances, 0, -1) . "}}";
    } else {
        $balances = "{\"status\":\"0\"}";
    }

    echo $balances;
}

```

```

    exit;
}

/* Получение ордеров */

if ($mode == 'getOrders') {

    $result = binance_query("/api/v3/openOrders","GET", array());
    $count = count($result);

    if ($count) {
        $return = "{";
        $return .= "\"status\":1,";
        $return .= "\"data\":[";
        for ($i = 0; $i < $count; $i++) {
            $return .= "{\"id\":\".$result[$i]['orderId'].\",\";
            $return .= "\"symbol\":\".$result[$i]['symbol'].\",\";
            $return .= "\"side\":\".$result[$i]['side'].\",\";
            $return .= "\"qty\":\".$result[$i]['origQty'].\",\";
            $return .= "\"price\":\".$result[$i]['price'].\",\";
            $return .= "\"time\":\".$result[$i]['time']/1000,0).\",\";
        }
        $return = substr($return, 0, -1) . "]}";
    } else {
        $return = "{\"status\":0}";
    }
    echo $return;
    exit;
}

/* Отмена ордера */

if ($mode == 'cancelOrder') {

    if(isset($_POST['order_id'])) { $order_id = $_POST['order_id']; }
    elseif(isset($_GET['order_id'])) { $order_id = $_GET['order_id']; }
    else { $order_id = 0; }
    $order_id = htmlspecialchars(strip_tags(trim($order_id)));

    if(isset($_POST['symbol'])) { $symbol = $_POST['symbol']; }
    elseif(isset($_GET['symbol'])) { $symbol = $_GET['symbol']; }
    else { $symbol = 0; }
    $symbol = htmlspecialchars(strip_tags(trim($symbol)));

    $result = binance_query("/api/v3/order","DELETE", array("symbol" => "$symbol", "orderId"
=> "$order_id", ));

    if ($result['orderId'] != null) {
        $return = "{\"status\":1}";
    } else {
        $return = "{\"status\":0}";
    }

    echo $return;
    exit;
}

/* Отправка ордера */

if ($mode == 'sendOrder') {

    if(isset($_POST['symbol'])) { $symbol = $_POST['symbol']; }
    elseif(isset($_GET['symbol'])) { $symbol = $_GET['symbol']; }
    else { $symbol = 0; }
    $symbol = htmlspecialchars(strip_tags(trim($symbol)));

```

```

if(isset($_POST['side'])) { $side = $_POST['side']; }
elseif(isset($_GET['side'])) { $side = $_GET['side']; }
else { $side = ""; }
$side = htmlspecialchars(strip_tags(trim($side)));

if(isset($_POST['qty'])) { $qty = $_POST['qty']; }
elseif(isset($_GET['qty'])) { $qty = $_GET['qty']; }
else { $qty = 0; }
$qty = htmlspecialchars(strip_tags(trim($qty)));

if(isset($_POST['price'])) { $price = $_POST['price']; }
elseif(isset($_GET['price'])) { $price = $_GET['price']; }
else { $price = 0; }
$price = htmlspecialchars(strip_tags(trim($price)));

$result = binance_query("/api/v3/order","POST", array("symbol" => "$symbol", "type" =>
"LIMIT", "side" => "$side", "timeInForce" => "GTC", "quantity" => $qty, "price" =>
$price));

if ($result['orderId'] != null) {
    $time = time();
    $order = "{";
    $order .= "\"status\":1,";
    $order .= "\"order_id\":\"".$result['orderId']."";
    $order .= "}";
} else {
    $order = "{\"status\":0}";
}

echo $order;
exit;
}

/* Получение цен торговых пар */

if ($mode == 'getPrices') {

    $link = "https://api.binance.com/api/v3/ticker/price";
    $fcontents = implode ('', file ($link));
    $fcontents = json_decode($fcontents, true);

    $count = count($fcontents);
    if ($count > 0) {
        $price = "{";
        $price .= "\"status\":1,";
        $price .= "\"data\":{\"";
        for ($i = 0; $i < $count; $i++) {
            $price .= "\"".$fcontents[$i]['symbol']."\":".$fcontents[$i]['price'].",";
        }
        $price = substr($price, 0, -1) . " }";
    } else {
        $price = "{\"status\":0}";
    }

    echo $price;
    exit;
}

/* Получение правил для торговых пар */

if ($mode == 'getRules') {

    $link = "https://api.binance.com/api/v1/exchangeInfo";
    $fcontents = implode ('', file ($link));
    $fcontents = json_decode($fcontents, true);

    $rules = "{";

```

```

$rules .= "\"status\":1,";

$count = count($fcontents['symbols']);

$rules .= "\"data\":[ ";

for ($i = 0; $i < $count; $i++) {
    if ($fcontents['symbols'][$i]['baseAsset'] != 123) {
        $rules .= "{";
        $rules .= "\"symbol\":".$fcontents['symbols'][$i]['symbol']."\",";
        $rules .= "\"base\":".$fcontents['symbols'][$i]['baseAsset']."\",";
        $rules .= "\"quote\":".$fcontents['symbols'][$i]['quoteAsset']."\",";
        $rules .= "\"min_price\":";
        $rules .= $fcontents['symbols'][$i]['filters'][0]['minPrice']."\",";
        $rules .= "\"max_price\":";
        $rules .= $fcontents['symbols'][$i]['filters'][0]['maxPrice']."\",";
        $rules .= "\"step_price\":";
        $rules .= $fcontents['symbols'][$i]['filters'][0]['tickSize']."\",";
        $rules .= "\"min_qty\":";
        $rules .= $fcontents['symbols'][$i]['filters'][2]['minQty']."\",";
        $rules .= "\"max_qty\":";
        $rules .= $fcontents['symbols'][$i]['filters'][2]['maxQty']."\",";
        $rules .= "\"step_qty\":";
        $rules .= $fcontents['symbols'][$i]['filters'][2]['stepSize']."\",";
        $rules .= "}";
    }
}

$rules = substr($rules, 0, -1) . "]}";

echo $rules;
exit;
}

if ($mode == 'getTime') {
    /* Корректировка времени сервера и биржи */
    $link = "https://api.binance.com/api/v1/time";
    $fcontents = implode ('', file ($link));
    $fcontents = json_decode($fcontents, true);
    $serverTime = $fcontents["serverTime"];
    $deltaTime = time()*1000 - $serverTime;
    $delta = "{";
    $delta .= "\"server_time\": $serverTime,";
    $delta .= "\"delta_time\": $deltaTime";
    $delta .= "}";
    print $delta;
    exit;
}

print "{\"status\":0}";
exit;
?>

```

## Файл main.php

### Основной файл программы

```

<?php

if(isset($_GET['debug'])) { $debug = 1; } else { $debug = 0; }

/* Получение номера набора настроек */
if(isset($_GET['set_id'])) { $set_id = $_GET['set_id']; } else { $set_id = 0; }
$set_id = htmlspecialchars(strip_tags(trim($set_id)));

/* Если параметр равен 0, завершение программы */

```

```

if (!$set_id) {
    echo 'Необходим параметр set_id';
    exit();
}

/* Теперь все запросы в хранилище производим с параметром set_id */
/* Программа будет работать с определенным набором настроек, */
/* и мы сможем контролировать - как они работают */

if ($debug) {
    ini_set('error_reporting', E_ALL);
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
} else {
    ini_set('error_reporting', 0);
}

/* Подключаем файлы работы с базой данных */
require_once('mysql7.php');
require_once('dbconfig.php');

/* Настройки торговой программы */
/* Биржа, с которой будет работать программа через API */
$exchange = "binance";
/* Папка в которой лежит бот */
$dir = "http://localhost/bot/";

/* Настройки торговой программы. Менять запрещено. */
$ex_api = $dir . "api_" . $exchange . ".php";

/* Получение из хранилища настроек пользователя */
$result = $db->query("SELECT * FROM `settings` WHERE `id` = '$set_id'");
$row = $db->get_object($result);
$api_key = $row->api_key;
$api_secret = $row->api_secret;
$open_url = $row->open_url;
$amount = $row->amount;
$pause = $row->pause;
$delta = $row->delta;
$fee = $row->fee;
$close_url = $row->close_url;
$close_time = $row->close_time;
$close_loss = $row->close_loss;
$close_exit = $row->close_exit;
$current_time = time();

/* Получение параметра временной переменной для биржи */
$url = $ex_api . "?mode=getTime";
$fcontents = implode('', file($url));
$fcontents = json_decode($fcontents, true);
$arg = $fcontents['delta_time'];
if ($debug) {echo 'prepare: --Временная переменная = '.$arg.'

```

```

}

/* Получаем сигналы закрытия */
if ($close_url) {
    if ($debug) {echo 'prepare: --Получаем сигналы закрытия<br>';}
    $fcontents = implode('', file ($close_url));
    $fcontents = json_decode($fcontents, true);
    if ($fcontents['status'] == 1) {
        if ($debug) {echo 'prepare: --Сигналы закрытия получены<br>';}
        $close_signals = $fcontents['data'];
    } else {
        if ($debug) {echo 'prepare: --Сигналы закрытия не обнаружены<br>';}
        $close_signals = [];
    }
} else {
    $close_signals = [];
}

/* Получаем ордера из хранилища */
$result = $db->query("SELECT * FROM `orders` WHERE `set_id` = '$set_id'");
if ($debug) {echo 'prepare: --Получаем ордера из хранилища<br>';}
if ($db->num_rows($result) > 0) {
    if ($debug) {echo 'prepare: --Ордера из хранилища получены<br>';}
    $i = 0;
    while($row = $db->get_array($result)) {
        $db_orders[$i] = $row;
        $i++;
    }
} else {
    if ($debug) {echo 'prepare: --Ордера из хранилища не обнаружены<br>';}
    $db_orders = [];
}

/* Получаем позиции из хранилища */
$result = $db->query("SELECT * FROM `positions` WHERE `set_id` = '$set_id'");
if ($debug) {echo 'prepare: --Получаем позиции из хранилища<br>';}
if ($db->num_rows($result) > 0) {
    if ($debug) {echo 'prepare: --Позиции из хранилища получены<br>';}
    $i = 0;
    while($row = $db->get_array($result)) {
        $db_positions[$i] = $row;
        $i++;
    }
} else {
    if ($debug) {echo 'prepare: --Позиции из хранилища не обнаружены<br>';}
    $db_positions = [];
}

/* Получаем ордера из биржи */
$url = $ex_api . "?mode=getOrders&api_key=$api_key&api_secret=$api_secret&arg=$arg";
if ($debug) {echo 'prepare: --Получаем ордера из биржи<br>';}
$fcontents = implode('', file ($url));
$fcontents = json_decode($fcontents, true);
if ($fcontents['status'] == 1) {
    if ($debug) {echo 'prepare: --Ордера из биржи получены<br>';}
    $ex_orders = $fcontents['data'];
} else {
    if ($debug) {echo 'prepare: --Ордера из биржи не обнаружены<br>';}
    $ex_orders = [];
}

/* Получаем балансы пользователя */
$url = $ex_api . "?mode=getBalances&api_key=$api_key&api_secret=$api_secret&arg=$arg";
if ($debug) {echo 'prepare: --Получаем балансы из биржи<br>';}
$fcontents = implode('', file ($url));
$fcontents = json_decode($fcontents, true);
if ($fcontents['status'] == 1) {
    if ($debug) {echo 'prepare: --Балансы из биржи получены<br>';}

```

```

    $balances = $fcontents['data'];
} else {
    if ($debug) {echo 'prepare: --Балансы из биржи не обнаружены<br>';}
    $balances = [];
    exit();
}

/* Получаем котировки биржи */
$url = $ex_api . "?mode=getPrices";
if ($debug) {echo 'prepare: --Получаем котировки из биржи<br>';}
$fcontents = implode ('', file ($url));
$fcontents = json_decode($fcontents, true);
if ($fcontents['status'] == 1) {
    if ($debug) {echo 'prepare: --Котировки из биржи получены<br>';}
    $prices = $fcontents['data'];
} else {
    if ($debug) {echo 'prepare: --Котировки из биржи не обнаружены<br>';}
    $prices = [];
    exit();
}

/* Отправка ордеров на покупку */
function send_buy_order($symbol,$price,$qty) {
    global $db;
    global $arg;
    global $debug;
    global $ex_api;
    global $api_key;
    global $api_secret;
    global $balances;
    /* Получаем правила биржи для заданной торгуемой пары */
    $result = $db->query("SELECT * FROM `pairs` WHERE `symbol` = '$symbol'");
    $row = $db->get_object($result);
    /* Получаем округление по условиям биржи */
    $around_price = abs(log10($row->step_price));
    $around_qty = abs(log10($row->step_qty));
    /* Считаем необходимую сумму */
    $sum = $price * $qty;
    /* Проверяем, что баланса достаточно: */
    $funds = $balances[$row->quote];
    /* Если средств недостаточно, завершаем функцию */
    if ($funds > $sum) {
        /* Подготавливаем ссылку */
        $send_url = $ex_api .
        "?mode=sendOrder&api_key=$api_key&api_secret=$api_secret&symbol=$symbol&side=buy&qty=".number_
        er_format($qty,$around_qty,'.','')."&price=".number_format($price,$around_price,'.','')."&a
        rg=$arg";
        if ($debug) {echo 'send_buy: ссылка '.$send_url.<br>';}
        $fcontents = implode ('', file ($send_url));
        $fcontents = json_decode($fcontents, true);
        if ($fcontents['status'] == 1) {
            return $fcontents['order_id'];
        } else {
            return 0;
        }
    } else {
        if ($debug) {echo 'send_buy: баланса не хватает<br>';}
        return 0;
    }
}

function send_sell_order($symbol,$price,$qty) {
    global $db;
    global $arg;
    global $debug;
    global $ex_api;
    global $api_key;
    global $api_secret;

```



```

global $balances;
global $fee;
/* Получаем правила биржи для заданной торгуемой пары */
$result = $db->query("SELECT * FROM `pairs` WHERE `symbol` = '$symbol'");
$row = $db->get_object($result);
/* Проводим округление по условиям биржи */
$around_price = abs(log10($row->step_price));
$around_qty = abs(log10($row->step_qty));
$qty = $qty*(1-$fee/100);
/* Проверяем, что баланса достаточно: */
$funds = $balances[$row->base];
/* Если средств недостаточно, продаем все что есть */
if ($funds < $qty) {
    if ($debug) {echo 'send_sell: баланса недостаточно, продаем все что есть<br>';}
    $qty = $funds;
}
/* Подготавливаем ссылку */
$send_url = $ex_api .
"?mode=sendOrder&api_key=$api_key&api_secret=$api_secret&symbol=$symbol&side=sell&qty=".number_format($qty,$around_qty,'.','')."&price=".number_format($price,$around_price,'.','')."&arg=$arg";
if ($debug) {echo 'send_sell: ссылка '.$send_url.<br>';}
$fcontents = implode ('', file ($send_url));
$fcontents = json_decode($fcontents, true);
if ($fcontents['status'] == 1) {
    return $fcontents['order_id'];
} else {
    return 0;
}
}

////////////////////
/* Основной код расчета */
////////////////////

/* Обработка сигналов */
if ($debug) {
    echo 'signals: START<br>';
    echo 'signals: --Сравниваем сигналы и позиции<br>';
}
$num = 0;
if (count($open_signals) > 0) {
    $num = count($open_signals);
    /* Сравниваем сигналы и открытые позиции в базе */
    if (count($db_positions) > 0) {
        /* Перебираем все позиции */
        for ($i=0;$i<count($db_positions);$i++) {
            /* Перебираем все сигналы */
            for ($j=0;$j<count($open_signals);$j++) {
                /* Если в сигналах присутствует пара, по которой открыта позиция, проверяем стоит ли на ней пауза */
                if ($db_positions[$i]['symbol'] == $open_signals[$j]['symbol'] && ($current_time - $db_positions[$i]['open_time']) < $pause) {
                    /* Удаляем ее из списка сигналов */
                    array_splice($open_signals,$j,1);
                }
            }
        }
    }
}
if ($debug) {
    echo 'signals: --Поступило '.$num.' сигналов<br>';
    echo 'signals: --Перебрано '.count($db_positions).' позиций<br>';
    echo 'signals: --Вышло '.count($open_signals).' сигналов<br>';
    echo 'signals: --Сравниваем сигналы и ордера<br>';
}
$num = 0;
/* Если в массиве остались данные, проверяем есть ли у нас отправленные ордера */

```

```

if (count($open_signals) > 0) {
    $num = count($open_signals);
    /* Сравниваем сигналы и открытые позиции в базе */
    if (count($db_orders) > 0) {
        /* Перебираем все ордера */
        for ($i=0;$i<count($db_orders);$i++) {
            /* Перебираем все сигналы */
            for ($j=0;$j<count($open_signals);$j++) {
                /* Если в сигналах присутствует пара, по которой отправлен ордер */
                if ($db_orders[$i]['symbol'] == $open_signals[$j]['symbol']) {
                    /* Удаляем ее из списка сигналов */
                    array_splice($open_signals,$j,1);
                }
            }
        }
    }
}

if ($debug) {
    echo 'signals: --Поступило ' . $num . ' сигналов<br>';
    echo 'signals: --Перебрано ' . count($db_orders) . ' ордеров<br>';
    echo 'signals: --Вышло ' . count($open_signals) . ' сигналов<br>';
}

/* Если сигналы остались, берем первый по списку и отправляем */
/* Остальные сигналы отправятся при последующих прогонах программы */
/* Не будем посылать сразу несколько ордеров, чтобы не получить бан от биржи */
if (count($open_signals) > 0) {
    $order = array_shift($open_signals);
    if ($debug) {echo 'signals: Подготовка к отправке ордера на покупку ' . $order['symbol'] . '<br>';}
    /* Определяем размер позиции */
    $qty = $amount / $order['eq_usdt'];
    /* Цену повышаем на величину delta%, чтоб не упустить позицию */
    $price = $order['price']*(1 + $delta/100);
    $symbol = $order['symbol'];
    /* Отправляем ордер */
    $order_id = send_buy_order($symbol,$price,$qty);
    if ($order_id) {
        if ($debug) {echo 'signals: --Ордер успешно отправлен<br>';}
        /* Записываем в хранилище отправленный ордер */
        $db->query("INSERT INTO `orders`(`symbol`, `order_id`, `price`, `qty`, `side`, `open_time`, `position`, `close_res`, `set_id`) VALUES ('$symbol','$order_id','$price','$qty','BUY','$current_time','0','','$set_id')");
        /* И завершаем операцию */
    } else {
        if ($debug) {echo 'signals: --Ошибка! Сбой отправки ордера<br>';}
    }
}

if ($debug) {
    echo 'signals: END<br>';
    echo 'orders: START<br>';
    echo 'orders: --Сравниваем ордера хранилища и ордера биржи<br>';
}

/* Обработка ордеров */
$open_num = 0;
$close_num = 0;
$upd_num = 0;
if (count($db_orders) > 0) {
    /* Создаем очередь ордеров */
    $queue_orders = [];
    if (count($ex_orders) > 0) {
        /* Сравниваем ордера биржи и хранилища */
        for ($i=0;$i<count($db_orders);$i++) {
            /* Совпадение */
            $num = 0;
            for ($j=0;$j<count($ex_orders);$j++) {

```

```

        /* Если такой ордер был отправлен, плюсуем совпадения */
        if ($db_orders[$i]['symbol'] == $ex_orders[$j]['symbol']) {
            $num = 1;
            break;
        }
    }
    /* Если совпадение не нашлось, значит ордер исполнился */
    if (!$num) {
        if ($debug) {echo 'orders: --Обнаружен исполнившийся ордер<br>';}
        /* Запоминаем ордер */
        $queue_orders[] = $db_orders[$i];
    }
}

} else {
    /* Переводим все ордера из хранилища в позиции */
    for ($i=0;$i<count($db_orders);$i++) {
        if ($debug) {echo 'orders: --Обнаружен исполнившийся ордер<br>';}
        /* Запоминаем ордер */
        $queue_orders[] = $db_orders[$i];
    }
}

/* Проверяем, что очередь ордеров не пустая */
if (count($queue_orders) > 0) {
    if ($debug) {echo 'orders: --Перебираем исполнившиеся ордера<br>';}
    for ($i=0;$i<count($queue_orders);$i++) {
        /* Если ордер на покупку, то открываем позицию */
        if ($queue_orders[$i]['side'] == 'BUY') {
            if ($debug) {echo 'orders: --Обнаружен ордер на покупку<br>';}
            /* Удаляем ордер из хранилища ордеров и делаем запись в хранилище позиций */
            $id = $queue_orders[$i]['id'];
            $db->query("DELETE FROM `orders` WHERE `id` = '$id'");
            $symbol = $queue_orders[$i]['symbol'];
            $price = $queue_orders[$i]['price'];
            $qty = $queue_orders[$i]['qty'];
            /* Прежде чем сделать запись в хранилище position проверяем есть у нас позиция с
данной парой */
            $n=-1;
            /* Перебираем все позиции и ищем с нашей торговой парой */
            for ($j=0;$j<count($db_positions);$j++) {
                /* Если пара найдена, запоминаем ее индекс */
                if ($queue_orders[$i]['symbol'] == $db_positions[$j]['symbol']) {
                    /* Запоминаем индекс позиции */
                    $n = $j;
                }
            }
            /* Если индекс найден, обновляем позицию, если не найден, создаем позицию */
            if ($n != -1) {
                if ($debug) {echo 'orders: --Обновляем позицию<br>';}
                $id = $db_positions[$n]['id'];
                /* Вычисляем новый объем, новую цену */
                $new_qty = $qty + $db_positions[$n]['qty'];
                $new_price = ($price*$qty +
$db_positions[$n]['price']*$db_positions[$n]['qty'])/($new_qty);
                /* Обновляем позицию в хранилище */
                $db->query("UPDATE `positions` SET
`price`='$new_price',`qty`='$new_qty',`open_time`='$current_time',`max_price`='$new_price'
WHERE `id` = '$id'");
                /* Обновляем позицию в переменной */
                $db_positions[$n]['qty'] = $new_qty;
                $db_positions[$n]['price'] = $new_price;
                $db_positions[$n]['max_price'] = $new_price;
                $db_positions[$n]['open_time'] = $current_time;
                $upd_num++;
            } else {
                if ($debug) {echo 'orders: --Добавляем позицию<br>';}
                /* Если позиции с такой парой у нас нет, создаем новую позицию */

```

```

        $result = $db->query("SHOW TABLE STATUS LIKE 'positions'");
        $db->query("INSERT INTO `positions`(`symbol`, `price`, `qty`, `open_time`,
`max_price`, `set_id`) VALUES
('$symbol', '$price', '$qty', '$current_time', '$price', '$set_id')");
        /* Узнаем индекс под которым будет произведена запись */
        $row = $db->get_array($result);
        $new_id = $row['Auto_increment'];
        $n = count($db_positions);
        /* Добавляем позицию в переменную */
        $db_positions[$n]['symbol'] = $symbol;
        $db_positions[$n]['qty'] = $qty;
        $db_positions[$n]['price'] = $price;
        $db_positions[$n]['max_price'] = $price;
        $db_positions[$n]['open_time'] = $current_time;
        $db_positions[$n]['id'] = $new_id;
        $db_positions[$n]['set_id'] = $set_id;
        $open_num++;
    }
}
/* Если ордер на продажу, то закрываем позицию */
if ($queue_orders[$i]['side'] == 'SELL') {
    if ($debug) {echo "orders: --Обнаружен ордер на продажу<br>";}
    /* Удаляем ордер из хранилища ордеров и переводим позицию в историю */
    $id = $queue_orders[$i]['id'];
    $db->query("DELETE FROM `orders` WHERE `id` = '$id'");
    $symbol = $queue_orders[$i]['symbol'];
    $close_price = $queue_orders[$i]['price'];
    $position_id = $queue_orders[$i]['position'];
    $close_res = $queue_orders[$i]['close_res'];
    $n=-1;
    /* Перебираем все позиции и ищем с нашей торговой парой */
    for ($j=0; $j<count($db_positions); $j++) {
        /* Если пара найдена, запоминаем ее индекс */
        if ($queue_orders[$i]['symbol'] == $db_positions[$j]['symbol']) {
            /* Запоминаем индекс позиции */
            $n = $j;
        }
    }
    /* Если индекс найден, удаляем позицию и переводим ее в историю, если не найден
выводим ошибку */
    if ($n != -1) {
        if ($debug) {echo "orders: --Удаляем позицию<br>";}
        $db->query("DELETE FROM `positions` WHERE `id` = '$position_id'");
        $open_price = $db_positions[$n]['price'];
        $qty = $db_positions[$n]['qty'];
        $open_time = $db_positions[$n]['open_time'];
        $close_time = time();
        $qty = number_format($qty, 8, '.', '');
        $open_price = number_format($open_price, 8, '.', '');
        $close_price = number_format($close_price, 8, '.', '');
        if ($debug) {echo "orders: --Переводим позицию в историю<br>";}
        $db->query("INSERT INTO `history`(`symbol`, `open_price`, `close_price`, `qty`,
`open_time`, `close_time`, `close_res`, `set_id`) VALUES
('$symbol', '$open_price', '$close_price', '$qty', '$open_time', '$close_time', '$close_res', '$se
t_id')");
        /* Удаляем позицию из переменной */
        array_splice($db_positions, $n, 1);
    } else {
        if ($debug) {echo "orders: --Ошибка! Позиция не обнаружена<br>";}
    }
    $close_num++;
}
}
}
}

if ($debug) {
    echo "orders: --Позиций открыто '$open_num.'<br>";
}

```

```

echo 'orders: --Позиций обновлено ' . $upd_num . '<br>';
echo 'orders: --Позиций закрыто ' . $close_num . '<br>';
echo 'orders: END<br>';
echo 'positions: START<br>';
}

/* Обработка позиций */
if (count($db_positions) > 0) {
    /* Создаем очередь ордеров */
    $queue_orders = [];
    $temp_order = [];
    if ($debug) {echo 'position: --Позиции в базе получены<br>';}
    /* Перебираем все позиции и проверяем условия выхода */
    for ($i=0; $i<count($db_positions); $i++) {
        if ($debug) {echo 'position: --Обрабатываем данные ' . $i . ' позиции<br>';}
        $position_id = $db_positions[$i]['id'];
        $symbol = $db_positions[$i]['symbol'];
        $open_price = $db_positions[$i]['price'];
        $open_time = $db_positions[$i]['open_time'];
        $qty = $db_positions[$i]['qty'];
        /* Проверяем обновилась ли максимальная цена */
        if ($db_positions[$i]['max_price'] > $prices[$symbol]) {
            $max_price = $db_positions[$i]['max_price'];
        } else {
            if ($debug) {echo 'position: --Максимальная цена обновилась<br>';}
            $max_price = $prices[$symbol];
            $max_price = number_format($max_price, 8, '.', '');
            /* Обновляем максимальную цену в хранилище */
            $db->query("UPDATE `positions` SET `max_price` = '$max_price' WHERE `id` = '$position_id'");
        }
        /* Проверяем позиции на соответствие условиям выхода */
        /* Счетчик сигналов на продажу и причина продажи */
        $res = "";
        /* Проверка на аварийный выход, не зависящий от паузы */
        if ($close_exit) {
            /* Проверяем, что цена упала ниже цены открытия позиции на величину close_exit в % */
            if (($prices[$symbol]/$open_price) < (1 - $close_exit/100)) {
                if ($debug) {echo 'position: --Выход по close_exit<br>';}
                $res = "EXIT";
            }
        }
        /* Проверяем остальные критерии */
        if (!$res) {
            /* Проверяем, стоят ли позиции на паузе */
            if ($current_time > ($open_time + $pause)) {
                /* Выход по сигналам */
                if (count($close_signals) > 0) {
                    for ($j=0; $j<count($close_signals); $j++) {
                        /* Проверяем совпадение пары */
                        if ($symbol == $close_signals[$j]['symbol']) {
                            if ($debug) {echo 'position: --Выход по close_url<br>';}
                            /* Правим причину */
                            $res = "URL";
                            /* Прерываем цикл поиска по сигналам */
                            break;
                        }
                    }
                }
            }
            /* Выход по времени */
            if ($close_time && !$res) {
                /* Проверяем прошло ли время жизни позиции */
                if ($current_time > ($open_time + $close_time)) {
                    if ($debug) {echo 'position: --Выход по close_time<br>';}
                    /* Правим причину */
                    $res = "TIME";
                }
            }
        }
    }
}

```

```

/* Выход по потере профита */
if ($close_lose && !$res) {
    /* Проверяем что профит положителен */
    $profit = $prices[$symbol] - $open_price;
    if ($profit > 0) {
        /* Проверяем падение профита на величину close_lose в % */
        if (($prices[$symbol]-$open_price)/($max_price-$open_price) < (1 -
$close_lose/100)) {
            if ($debug) {echo 'position: --Выход по close_lose<br>';}
            /* Если профит упал ниже доступной границы, правим причину */
            $res = "LOSE";
        }
    } else {
        if ($debug) {echo 'position: --Профит меньше ноля<br>';}
        /* Если профит отрицателен, сразу правим причину */
        $res = "LOSE";
    }
}
} else {
    if ($debug) {echo 'position: --Позиция еще на паузе<br>';}
}
}
/* Если мы нашли причину выхода, ставим ордер в очередь */
if ($res) {
    $temp_order['symbol'] = $symbol;
    $temp_order['position'] = $position_id;
    $temp_order['res'] = $res;
    /* Понижаем цену на величину delta */
    $temp_order['price'] = $prices[$symbol]*(1 - $delta/100);
    $temp_order['qty'] = $qty*(1-$fee/100);
    $queue_orders[] = $temp_order;
}
}
/* Если очередь не пуста отправляем ордера */
if (count($queue_orders) > 0) {
    /* Проверяем все ордера и ищем, были ли отправлен ордер ранее */
    /* Совпадение */
    $n = 0;
    for($i=0;$i<count($queue_orders);$i++) {
        for ($j=0;$j<count($db_orders);$j++) {
            if ($queue_orders[$i]['symbol'] == $db_orders[$j]['symbol'] &&
$db_orders[$j]['side'] == "SELL") {
                /* Если данная пара присутствует в сигналах, запоминаем ее */
                $n = $j;
                break;
            }
        }
    }
    /* Если ордер был отправлен ранее, удаляем его и отправляем новый */
    if ($n) {
        $id = $db_orders[$n]['id'];
        $order_id = $db_orders[$n]['order_id'];
        /* Если данная пара присутствует в сигналах, удаляем ордер */
        $url = $ex_api .
"?mode=cancelOrder&api_key=$api_key&api_secret=$api_secret&order_id=$order_id&arg=$arg";
        $fcontents = implode ('', file ($url));
        $fcontents = json_decode($fcontents, true);
        if ($fcontents['status'] == 1) {
            if ($debug) {echo 'position: --Ордер удален с биржи<br>';}
            $db->query("DELETE FROM `orders` WHERE `id` = '$id'");

        } else {
            if ($debug) {echo 'position: --Ошибка! Ордер не был удален<br>';}
        }
    }
}
}
$symbol = $queue_orders[$i]['symbol'];
$price = $queue_orders[$i]['price'];
$qty = $queue_orders[$i]['qty'];

```

```

$res = $queue_orders[$i]['res'];
$position_id = $queue_orders[$i]['position'];

/* Отправляем новый ордер */
$order_id = send_sell_order($symbol,$price,$qty);
if ($order_id) {
    if ($debug) {echo 'position: --Ордер отправлен по '.$res.'<br>';}
    $qty = number_format($qty,8,'.','');
    $price = number_format($price,8,'.','');
    /* Записываем в хранилище отправленный ордер */
    $db->query("INSERT INTO `orders` (`symbol`,`order_id`,`price`,`qty`,`side`,`open_time`,`position`,`close_res`,`set_id`) VALUES
    ('$symbol','$order_id','$price','$qty','SELL','$current_time','$position_id','$res','$set_id')");
}
}
}
}

if ($debug) {echo 'positions: END<br>';}

?>

```

## GUI

### Файл index.html

Навигационный файл программы

```

<html>
<head>
<title>
iBOT Index
</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<style type="text/css">
.c {
padding: auto;
margin: auto;
text-align: center;
}
</style>
</head>
<body>
<table cellpadding='0' cellspacing='0' border='0' width='100%' height='100%'>
<tr>
<td width='100%' height='100%'>
<div class="c" width='100%' height='100%'>
<a href="config.php">i-Bot Конфигурация</a><br>
</div>
</td>
</tr>
</table>
</body>
</html>

```

### Файл config.php

Страница для корректировки настроек бота (если нет желания устанавливать phpmyadmin)

```
<?php

/* Подключаем файлы работы с базой данных mysql5.php или mysql7.php */
require_once('mysql7.php');
require_once('dbconfig.php');

/* Получение необходимых параметров */
/* Номер настроек */
if(isset($_POST['set_id'])) { $set_id = $_POST['set_id']; }
elseif(isset($_GET['set_id'])) { $set_id = $_GET['set_id']; }
else { $set_id = 0; }
$set_id = htmlspecialchars(strip_tags(trim($set_id)));

/* API ключ */
if(isset($_POST['api_key'])) { $api_key = $_POST['api_key']; }
elseif(isset($_GET['api_key'])) { $api_key = $_GET['api_key']; }
else { $api_key = ""; }
$api_key = htmlspecialchars(strip_tags(trim($api_key)));

/* API секретный ключ */
if(isset($_POST['api_secret'])) { $api_secret = $_POST['api_secret']; }
elseif(isset($_GET['api_secret'])) { $api_secret = $_GET['api_secret']; }
else { $api_secret = ""; }
$api_secret = htmlspecialchars(strip_tags(trim($api_secret)));

/* ссылка открытия позиции */
if(isset($_POST['open_url'])) { $open_url = $_POST['open_url']; }
elseif(isset($_GET['open_url'])) { $open_url = $_GET['open_url']; }
else { $open_url = ""; }
$open_url = htmlspecialchars(strip_tags(trim($open_url)));

/* Дельта */
if(isset($_POST['delta'])) { $delta = $_POST['delta']; }
elseif(isset($_GET['delta'])) { $delta = $_GET['delta']; }
else { $delta = 0; }
$delta = htmlspecialchars(strip_tags(trim($delta)));

/* Комиссия биржи */
if(isset($_POST['fee'])) { $fee = $_POST['fee']; }
elseif(isset($_GET['fee'])) { $fee = $_GET['fee']; }
else { $fee = 0; }
$fee = htmlspecialchars(strip_tags(trim($fee)));

/* Сумма */
if(isset($_POST['amount'])) { $amount = $_POST['amount']; }
elseif(isset($_GET['amount'])) { $amount = $_GET['amount']; }
else { $amount = 0; }
$amount = htmlspecialchars(strip_tags(trim($amount)));

/* Пауза */
if(isset($_POST['pause'])) { $pause = $_POST['pause']; }
elseif(isset($_GET['pause'])) { $pause = $_GET['pause']; }
else { $pause = 0; }
$pause = htmlspecialchars(strip_tags(trim($pause)));

/* Ссылка закрытия позиции */
if(isset($_POST['close_url'])) { $close_url = $_POST['close_url']; }
elseif(isset($_GET['close_url'])) { $close_url = $_GET['close_url']; }
else { $close_url = ""; }
$close_url = htmlspecialchars(strip_tags(trim($close_url)));

/* Время закрытия */
if(isset($_POST['close_time'])) { $close_time = $_POST['close_time']; }
elseif(isset($_GET['close_time'])) { $close_time = $_GET['close_time']; }
```



```

else { $close_time = 0; }
$close_time = htmlspecialchars(strip_tags(trim($close_time)));

/* Потеря профита */
if(isset($_POST['close_loss'])) { $close_loss = $_POST['close_loss']; }
elseif(isset($_GET['close_loss'])) { $close_loss = $_GET['close_loss']; }
else { $close_loss = 0; }
$close_loss = htmlspecialchars(strip_tags(trim($close_loss)));

/* Снижение цены */
if(isset($_POST['close_exit'])) { $close_exit = $_POST['close_exit']; }
elseif(isset($_GET['close_exit'])) { $close_exit = $_GET['close_exit']; }
else { $close_exit = 0; }
$close_exit = htmlspecialchars(strip_tags(trim($close_exit)));

/* Дополнительный параметр */
if(isset($_POST['arg'])) { $arg = $_POST['arg']; }
elseif(isset($_GET['arg'])) { $arg = $_GET['arg']; }
else { $arg = ""; }
$arg = htmlspecialchars(strip_tags(trim($arg)));

if(isset($_POST['del'])) { $del = $_POST['del']; }
elseif(isset($_GET['del'])) { $del = $_GET['del']; }
else { $del = 0; }
$del = htmlspecialchars(strip_tags(trim($del)));

if(isset($_POST['send'])) { $send = $_POST['send']; }
elseif(isset($_GET['send'])) { $send = $_GET['send']; }
else { $send = 0; }
$send = htmlspecialchars(strip_tags(trim($send)));
$error = "";

if ($send) {
    if ($set_id) {
        $result = $db->query("SELECT * FROM `settings` WHERE `id` = '$set_id'");
        if ($db->num_rows($result)) {
            if (!$api_key) {$error .= "Вы не отправили API-ключ от биржи<br>";}
            if (!$api_secret) {$error .= "Вы не отправили API-ключ от биржи<br>";}
            if (!$open_url) {$error .= "Вы не отправили ссылку открытия позиции<br>";}
            if (!$amount) {$error .= "Вы не отправили размер открываемой позиции<br>";}
            if (!$close_exit) {$error .= "Вы не отправили значение экстренного выхода<br>";}
            if (!$error) {
                $db->query("UPDATE `settings` SET
`api_key`='$api_key',`api_secret`='$api_secret',`open_url`='$open_url',`amount`='$amount',`
fee`='$fee',`delta`='$delta',`pause`='$pause',`close_url`='$close_url',`close_time`='$close
_time',`close_loss`='$close_loss',`close_exit`='$close_exit',`arg`='$arg' WHERE `id` =
'$set_id'");
            }
        } else {
            if (!$api_key) {$error .= "Вы не отправили API-ключ от биржи<br>";}
            if (!$api_secret) {$error .= "Вы не отправили API-ключ от биржи<br>";}
            if (!$open_url) {$error .= "Вы не отправили ссылку открытия позиции<br>";}
            if (!$amount) {$error .= "Вы не отправили размер открываемой позиции<br>";}
            if (!$close_exit) {$error .= "Вы не отправили значение экстренного выхода<br>";}
            if (!$error) {
                $db->query("INSERT INTO `settings`(`id`, `api_key`, `api_secret`, `open_url`,
`amount`, `fee`, `delta`, `pause`, `close_url`, `close_time`, `close_loss`, `close_exit`,
`arg`) VALUES ('$set_id', '$api_key', '$api_secret', '$open_url', '$amount', '$fee',
'$delta', '$pause', '$close_url', '$close_time', '$close_loss', '$close_exit', '$arg')");
            }
        }
    } else {
        $error = "Вы не отправили номер сета<br>";
    }
}
if ($del) {
    if ($set_id) {

```

```

        $db->query("DELETE FROM `settings` WHERE `id` = $set_id");
    } else {
        $error = "Вы не отправили номер сета<br>";
    }
}

$url = "http://localhost/bot/api_binance.php?mode=getTime";
$fcontents = implode ('', file ($url));
$fcontents = json_decode($fcontents, true);
$delta_time = $fcontents['delta_time'];

print "
<html>
<head>
<title>iBot Config Page</title>
</head>
<body>
<center>
<h1>Конфигурация iBot</h1>
$error
<form action='config.php' method='post'>
    <input type='hidden' name='send' value='1'>

    <table width='50%'>
        <tr>
            <th>Параметр</th>
            <th>Значение</th>
            <th>Описание</th>
        </tr>
        <tr>
            <td>Номер набора</td>
            <td><input type='number' step='1' name='set_id' placeholder='1' value=''></td>
            <td>Под данным номером сохраняются настройки в хранилище</td>
        </tr>
        <tr>
            <td>API-ключ</td>
            <td><input type='text' name='api_key' placeholder='API key'></td>
            <td>Ключ, который выдают на бирже API-KEY</td>
        </tr>
        <tr>
            <td>API-секретный ключ</td>
            <td><input type='text' name='api_secret' placeholder='API key'></td>
            <td>Ключ, который выдают на бирже API-SECRET</td>
        </tr>
        <tr>
            <td colspan='3'><hr></td>
        </tr>
        <tr>
            <td>Ссылка открытия</td>
            <td><input type='text' name='open_url' placeholder='http://'></td>
            <td>Ссылка, по которой получаем сигналы на покупку</td>
        </tr>
        <tr>
            <td>Дельта</td>
            <td><input type='number' step='any' name='delta' placeholder='0.5'></td>
            <td>Корректировка цены в % в худшую сторону при отправке ордера. Необходима чтобы
не упустить позицию при волатильном рынке.</td>
        </tr>
        <tr>
            <td>Размер позиции</td>
            <td><input type='number' step='any' name='amount' placeholder='20'></td>
            <td>Размер позиции эквивалентная USDT</td>
        </tr>
        <tr>
            <td>Комиссия биржи</td>
            <td><input type='number' step='any' name='fee' placeholder='0.2'></td>
            <td>Комиссия в %, которую берет биржа за совершение операции</td>
        </tr>
    </table>
    </body>
</html>

```

```

        <tr>
            <td>Пауза</td>
            <td><input type='number' step='1' name='pause' placeholder='86400'></td>
            <td>Количество секунд, которое будет заблокирована позиция до начала проверок
выходов</td>
        </tr>
        <tr>
            <td>Дополнительный параметр</td>
            <td><input type='text' name='arg' placeholder='$delta_time'></td>
            <td>Дополнительный параметр</td>
        </tr>
        <tr>
            <td colspan='3'><hr></td>
        </tr>
        <tr>
            <td>Ссылка закрытия</td>
            <td><input type='text' name='close_url' placeholder='http://'></td>
            <td>Ссылка, по которой получаем сигналы на продажу</td>
        </tr>
        <tr>
            <td>Время закрытия</td>
            <td><input type='number' step='1' name='close_time' placeholder='604800'></td>
            <td>Количество секунд через которое позиция закрывается</td>
        </tr>
        <tr>
            <td>Потеря профита</td>
            <td><input type='number' step='any' name='close_lose' placeholder='20.0'></td>
            <td>Значение в %, которое мы готовы потерять от максимального профита, после чего
закрываем позицию</td>
        </tr>
        <tr>
            <td>Экстренный выход<br>по снижению цены</td>
            <td><input type='number' step='any' name='close_exit' placeholder='2.0'></td>
            <td>Значение в % на которое просядет цена и позиция закроется.</td>
        </tr>
        <tr>
            <td colspan='3'><hr></td>
        </tr>
        <tr>
            <td colspan='3'><input type='submit' value='Отправить'></td>
        </tr>
    </table>
</form>
Для binance дополнительный параметр равен $delta_time
<br><br>
";

$result = $db->query("SELECT * FROM `settings` WHERE 1");
print "    <table>
        <tr>
            <th>Номер</th>
            <th>API-ключ</th>
            <th>API-секрет</th>
            <th>Ссылка открытия</th>
            <th>Дельта</th>
            <th>Комиссия</th>
            <th>Пауза</th>
            <th>Параметр</th>
            <th>Ссылка закрытия</th>
            <th>Время закрытия</th>
            <th>Потери профита</th>
            <th>Экстремальные потери</th>
            <th></th>
        </tr>";
while ($row = $db->get_object($result)) {
print "
        <tr>
            <td>". $row->id. "</td>

```

```

        <td>" .substr($row->api_key,0,16)."...</td>
        <td>" .substr($row->api_secret,0,16)."...</td>
        <td>" . $row->open_url. "</td>
        <td>" . $row->delta. "</td>
        <td>" . $row->fee. "</td>
        <td>" . $row->pause. "</td>
        <td>" . $row->arg. "</td>
        <td>" . $row->close_url. "</td>
        <td>" . $row->close_time. "</td>
        <td>" . $row->close_lose. "</td>
        <td>" . $row->close_exit. "</td>
        <td><a href='config.php?set_id=".$row->id."&del=1'>Удалить</a></td>
    </tr>
";
}

print " </table>
</center>
</body>
</html>";

?>

```

## Файл monitor.php

Страница для текущего просмотра позиций

```

<?php

/* Подключаем файлы работы с базой данных */
require_once('mysql7.php');
require_once('dbconfig.php');

/* Настройки торговой программы */
/* Биржа, с которой будет работать программа через API */
$exchange = "binance";
/* Папка в которой лежит бот */
$dir = "http://localhost/bot/";

/* Надстройки торговой программы. Менять запрещено. */
$ex_api = $dir . "api_" . $exchange . ".php";

print "
<html>
<head>
<title>
Monitor
</title>
</head>
<body>
<center>
<h1>Мониторинг Funny Bot</h1>
<table>
<tr>
<th>№</th>
<th>Пара</th>
<th>Цена открытия</th>
<th>Количество</th>
<th>Сумма</th>
<th>Макс. цена</th>
<th>Посл. цена</th>
<th>Изменение</th>
<th>Макс. профит</th>
<th>Текущий профит</th>
<th>Настройки</th>
</tr>

```

```

";

/* Получаем цены биржи по ссылке и расшифровываем JSON */
$url = $ex_api . "?mode=getPrices";
$fcontents = implode('', file ($url));
$fcontents = json_decode($fcontents, true);
$prices = $fcontents['data'];

/* Получаем позиции из хранилища */
$result = $db->query("SELECT * FROM `positions` WHERE 1");

while ($row = $db->get_object($result)){
    print "
    <tr>
    <td>".$row->id." </td>
    <td>".$row->symbol." </td>
    <td>".number_format($row->price,8,'.','')."." </td>
    <td>".number_format($row->qty,8,'.','')."." </td>
    <td>".number_format(($row->price*$row->qty),8,'.','')."." </td>
    <td>".number_format($row->max_price,8,'.','')."." </td>
    <td>".number_format($prices[$row->symbol],8,'.','')."." </td>
    <td>".round((100*($prices[$row->symbol]-$row->price)/$row->price),2)." </td>
    <td>".number_format(((($row->max_price-$row->price)*$row->qty),8,'.','')."." </td>
    <td>".number_format(((($prices[$row->symbol]-$row->price)*$row->qty),8,'.','')."." </td>
    <td>".$row->set_id." </td>
    </tr>";
}

print "
</table>
</center>
</body>
</html>
";

?>

```

## Запуск программы

Запуск программы производится через планировщик заданий CRON. Большинство хостингов позволяют редактировать записи cron

Редактирование таблиц crontab на debian

В командной строке из-под root запускаем редактор

```
nano /etc/crontab
```

Вписываем строку

```
*/1 * * * * wget -q --spider http://localhost/bot/main.php?set_id=1
```

Жмем Ctrl+O и сохраняем файл

Жмем Ctrl+X и выходим

Теперь ваш скрипт выполняется с периодичностью 1 раз в минуту

Кирилл Савин