

**Михаил Шардин**

личный блог



05 ноября 2024, 07:18

[+ Подписаться](#)

Работа в «песочнице» с торговым роботом на Московской бирже

Перед тем как использовать торгового робота на живых деньгах хочется всё протестировать на демо-счете (или «песочнице»). Это когда программные ошибки не имеют особой стоимости.

Я планирую использовать робота на Московской бирже, через АПИ одного из брокеров. Чтобы частному инвестору начать торговать на бирже нужен брокерский счет. Однако минимальное число российских брокеров имеют свои API (на текущий момент я знаю только [ФИНАМ](#), [Алор](#), [Тинькофф Инвестиции](#)). По субъективным причинам я выбрал работать с T-Bank Invest API (это бывший Тинькофф) через среду выполнения JavaScript Node.JS.

В статье разбираюсь как использую песочницу:

1. Открыть счёт.
2. Пополнить баланс счёта рублями через специальный запрос.
3. Посмотреть все свои открытые счета в песочнице.
4. Купить 1 акцию.
5. Продать 1 акцию.
6. Получить все открытые позиции указанного счёта.
7. Закрывать счёт.

Введите текст комментария

```

38 async function sandboxAccount() {
58   const GetSandboxAccounts = await tinkoffClient.callApi('Sandbox/GetAccounts', {
59     'accessToken': secrets.accessToken,
60   });
61   logger.info('Список счетов в песочнице:\n ${JSON.stringify(GetSandboxAccounts, null, '\t')}\n\n');
62
63   // // Получить все открытые позиции указанного счёта
64   // // const accountId = {
65   //   "accountId": secrets.AccountID
66   // };
67   // const GetSandboxPositions = await tinkoffClient.callApi('OperationsService/GetPositions', accountId);
68   // logger.info('Все открытые позиции счёта ${secrets.AccountID}:\n ${JSON.stringify(GetSandboxPositions, null, '\t')}\n\n');
69
70   // // Функция для отправки рыночного ордера
71   // tinkoffClient.placeMarketOrder('BBG004730N88', 1, 'ORDER_DIRECTION_BUY'); // Купить 1 акцию
72   tinkoffClient.placeMarketOrder('BBG004730N88', 1, 'ORDER_DIRECTION_SELL'); // Продать 1 акцию
73 }
74 // ===== Запуск функций =====
75 // =====
76
77 sandboxAccount().catch(logger.error);

```

```

2024-11-01 14:11:58 [INFO]: Детали операции:
{
  "responseMetadata": {
    "serverTime": "2024-11-01T09:11:57.919185435Z"
  }
}
2024-11-01 14:11:58 [INFO]: Идентификатор продажи: 27a35903-3b38bc38c5e5.
2024-11-01 14:11:58 [INFO]: Общая стоимость сделки: 2358.1 руб.
2024-11-01 14:11:58 [INFO]: Цена за 1 шт. Сбер Банк (SBER): 235.81 руб.
2024-11-01 14:11:58 [INFO]: Комиссия за сделку: 1.17905 руб.
[Done] exited with code=0 in 1.146 seconds

```

Операция продажи через OrdersService/PostOrder

SilverFir-TradingBot\src\sandbox.js

Этот код [Node.js](#) взаимодействует с [API Tinkoff Invest](#), позволяя имитировать торговые операции на виртуальном счете, что позволяет протестировать некоторые функции API в ручном режиме. Вот что делает этот код:

1. Импорт модулей

- `secrets`: импортирует ключи доступа и идентификаторы из внешнего файла конфигурации (`secrets`), что помогает защитить конфиденциальную информацию.
- `logger`: импортирует модуль ведения журнала, который записывает журналы в файл или консоль. Это важно для отслеживания активности бота и отладки.
- `logFunctionName`: импортирует утилиту для получения имен функций, что упрощает ведение журнала текущего контекста функции.
- `TinkoffClient`: импортирует клиентский модуль для взаимодействия с API Tinkoff Invest. Этот клиент обрабатывает запросы к API.

2. Настройка клиента

- `API_TOKEN`: получает токен API (в режиме песочницы) из внешнего файла конфигурации (`secrets`) для аутентификации.

• `TinkoffClient` создает экземпляр `TinkoffClient` с токеном песочницы, настраивая

- `sandboxAccount()`: это основная функция, демонстрирующая различные операции с учетной записью песочницы, с несколькими действиями, которые в настоящее время закомментированы.
- `logFunctionName()`: регистрирует имя функции в консоли, что полезно для отслеживания в сложных приложениях.
- `GetSandboxAccounts`: получает все открытые позиции указанного счёта.

Закомментированные операции:

- `OpenSandboxAccount`: регистрирует новый счет в песочнице, что позволит начать тестирование заново.
- `SandboxPayIn`: зачисляет средства на счет в песочнице в российских рублях (RUB). Здесь указанная сумма составляет 30 000 руб.
- `CloseSandboxAccount`: закрывает указанный счет в песочнице, используя его `accountId`, что позволяет выполнить сброс после тестирования.
- `GetSandboxPositions`: извлекает и регистрирует все открытые позиции для указанного идентификатора счета.
- `placeMarketOrder`: отправляет рыночные ордера на покупку и продажу указанного инструмента (здесь `BBG004730N88`). Это позволит протестировать функциональность размещения ордеров в песочнице.

Ошибки

- `sandboxAccount().catch(logger.error)`: запускает `sandboxAccount` асинхронно и регистрирует любые обнаруженные ошибки.

Эта структура кода демонстрирует, как взаимодействовать с виртуальным торговым счетом в API Тинькофф. Закомментированные блоки кода указывают на дополнительные функции, которые можно активировать при необходимости, такие как открытие, пополнение и закрытие счетов песочницы, а также размещение ордеров на покупку/продажу.

```

30 async function sandboxAccount() {
48 // logger.info( Пополнение баланса счёта песочницы:\n ${JSON.stringify(SandboxPayIn, null, '\t')}\n\n );
49
50 // // Закрытие счёта в песочнице
51 // const accountId = {
52 //   "accountId": secrets.AccountID
53 // };
54 // const CloseSandboxAccount = await tinkoffClient.callApi('SandboxService/CloseSandboxAccount', accountId);
55 // logger.info( Закрытие счёта в песочнице:\n ${JSON.stringify(CloseSandboxAccount, null, '\t')}\n\n );
56
57 // Посмотреть счёта в песочнице
58 const GetSandboxAccounts = await tinkoffClient.callApi('SandboxService/GetSandboxAccounts');
59 logger.info( Список счётов в песочнице:\n ${JSON.stringify(GetSandboxAccounts, null, '\t')}\n\n );
60
61 // // Получить все открытые позиции указанного счёта
62 // const accountId = {
63 //   "accountId": secrets.AccountID
64 // };
65 // const GetSandboxPositions = await tinkoffClient.callApi('OperationsService/GetPositions', accountId);
66 // logger.info( Все открытые позиции счёта ${secrets.AccountID}:\n ${JSON.stringify(GetSandboxPositions, null, '\t')}\n\n );
67

```

2024-11-02 07:25:30 [INFO]: Список счётов в песочнице:

```

{
  "accounts": [
    {
      "status": "ACCOUNT_STATUS_OPEN",
      "openedDate": "2024-10-01T12:19:14.135019Z",
      "accessLevel": "ACCOUNT_ACCESS_LEVEL_FULL_ACCESS"
    }
  ]
}

```

[Done] exited with code=0 in 6.249 seconds

```
<code>// Импорт необходимых модулей
```

```
const secrets = require('../config/secrets'); // Ключи доступа и идентификаторы
```

```
const logger = require('./services/logService'); // Логирование в файл и консоль
```

```
const logFunctionName = require('./services/logFunctionName'); // Получение имен
```

```
const TinkoffClient = require('./grpc/tinkoffClient'); // модуль для взаимодействия
```

```
const API_TOKEN = secrets.TbankSandboxMode;
```

```
const tinkoffClient = new TinkoffClient(API_TOKEN);
```

```
async function sandboxAccount() {
```

```
  // https://tinkoff.github.io/investAPI/swagger-ui/#/SandboxService/SandboxService
  logger.info(`Запуск функции ${JSON.stringify(logFunctionName())}\n`);
```

```
  // // Регистрации счёта в песочнице
```

```
  // const OpenSandboxAccount = await tinkoffClient.callApi('SandboxService/OpenSandboxAccount');
```

```
  // logger.info(`Регистрации счёта в песочнице:\n ${JSON.stringify(OpenSandboxAccount, null, '\t')}\n\n`);
```

```
//      "amount": {
//          "nano": 0, // Дробная часть отсутствует
//          "currency": "RUB",
//          "units": 30000, // Сумма в рублях
//      }
// };
// const SandboxPayIn = await tinkoffClient.callApi('SandboxService/SandboxP
// logger.info(`Пополнение баланса счёта песочницы:\n ${JSON.stringify(Sandb

// // Закрытие счёта в песочнице
// const accountId = {
//     "accountId": secrets.AccountID
// };
// const CloseSandboxAccount = await tinkoffClient.callApi('SandboxService/C
// logger.info(`Закрытие счёта в песочнице:\n ${JSON.stringify(CloseSandboxA

// Посмотреть счета в песочнице
const GetSandboxAccounts = await tinkoffClient.callApi('SandboxService/GetSa
logger.info(`Список счетов в песочнице:\n ${JSON.stringify(GetSandboxAccount

// // Получить все открытые позиции указанного счёта
// const accountId = {
//     "accountId": secrets.AccountID
// };
// const GetSandboxPositions = await tinkoffClient.callApi('OperationsServic
// logger.info(`Все открытые позиции счёта ${secrets.AccountID}:\n ${JSON.st

// // Функция для отправки рыночного ордера
// tinkoffClient.placeMarketOrder('BBG004730N88', 1, 'ORDER_DIRECTION_BUY');
// tinkoffClient.placeMarketOrder('BBG004730N88', 1, 'ORDER_DIRECTION_SELL')
}

// =====
// =====      Запуск функций      =====
// =====
```

расскажет медленно. Это пригодится только при разработке торговых стратегий.

```
<code>[Running] node "d:\Synology ...\SilverFir-TradingBot_github\src\sandbox.js
2024-11-01 14:11:57 [INFO]: Запуск функции "sandboxAccount"
```

```
2024-11-01 14:11:58 [WARN]: Операция продажи выполнена успешно для Сбер Банк (SB
```

```
2024-11-01 14:11:58 [INFO]: Детали операции:
```

```
{
  "orderId": "27a35903-2134-4aaf-XXXX-3b38bc38c5e5",
  "executionReportStatus": "EXECUTION_REPORT_STATUS_FILL",
  "lotsRequested": "1",
  "lotsExecuted": "1",
  "initialOrderPrice": {
    "currency": "rub",
    "units": "2358",
    "nano": 100000000
  },
  "executedOrderPrice": {
    "currency": "rub",
    "units": "235",
    "nano": 810000000
  },
  "totalOrderAmount": {
    "currency": "rub",
    "units": "2358",
    "nano": 100000000
  },
  "initialCommission": {
    "currency": "rub",
    "units": "1",
    "nano": 179050000
  },
  "executedCommission": {
    "currency": "rub",
    "units": "1",
    "nano": 179050000
  },
  "figi": "BPC004730N88",
```

```
  "initialSecurityPrice": {
```

```
"nano": 810000000
},
"orderType": "ORDER_TYPE_MARKET",
"message": "",
"initialOrderPricePt": {
  "units": "0",
  "nano": 0
},
"instrumentUid": "e6123145-9665-43e0-XXXX-cd61b8aa9b13",
"orderRequestId": "",
"responseMetadata": {
  "trackingId": "d059748a138038d3XXXXX93783d61a99",
  "serverTime": "2024-11-01T09:11:57.919185435Z"
}
}
}
2024-11-01 14:11:58 [INFO]: Идентификатор продажи: 27a35903-2134-4aaf-XXXX-3b38b
2024-11-01 14:11:58 [INFO]: Общая стоимость сделки: 2358.1 руб.
2024-11-01 14:11:58 [INFO]: Цена за 1 шт. Сбер Банк (SBER): 235.81 руб.
2024-11-01 14:11:58 [INFO]: Комиссия за сделку: 1.17905 руб.
```

[Done] exited with code=0 in 1.146 seconds</code>

Для торгового робота 1,146 секунды от отправки ордера до его исполнения можно считать довольно медленным временем.

В высокочастотной торговле (HFT), где компании конкурируют за время исполнения менее миллисекунды, время обработки ордера более одной секунды будет непозволительно долгим. Стратегии HFT основаны на выполнении тысяч сделок за доли секунды, поэтому 1,146 секунды сделают этого робота неконкурентоспособным.

Напротив, для долгосрочной стратегии, такой как дневной торговый бот или свинг-трейдинг, это время может быть приемлемым. Скорость исполнения остается важной, но не такой критической, как в HFT. В этих случаях компромисс часто склоняется в сторону надежности и экономической эффективности, а не чистой скорости. Задержка в 1 секунду, как правило, не подорвет прибыльность в стратегии, где сделки исполняются с интервалом в несколько минут или даже часов.

Я планирую использовать свинг-трейдинг — это торговая стратегия, ориентированная на захват краткосрочных и среднесрочных ценовых движений, обычно удерживая активы в

Итоги

Проект полностью представлен на Гитхабе: <https://github.com/empenoso/SilverFir-TradingBot>.
Новые модули будут загружаться по мере написания и тестирования.

Автор: [Михаил Шардин](#)

5 ноября 2024 г.

торговые роботы

339 

 2  0  4



Михаил Шар...

 Пермь

 34  381

 с 23 января 2019

 [empenoso](#)

[+ Подписаться](#)

0 КОММЕНТАРИЕВ

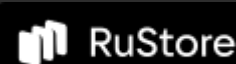
[Сначала старые](#) 

Напишите комментарий...



[ОТПРАВИТЬ](#)

Установите приложение Смартлаба:



The logo for MOEX (Moscow Exchange) is a small red square with the white text "MOEX" inside.

Московская Биржа является спонсором ресурса smart-lab.ru