

Горячее

Лучшее

Свежее **Подписки** 







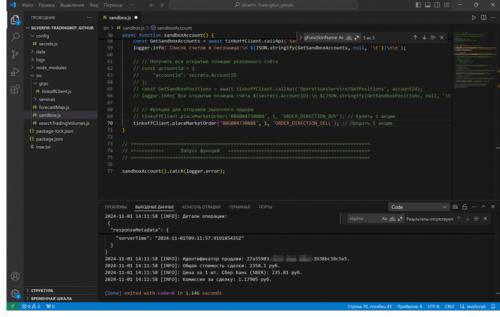




еmpenoso 2 часа назад
Лига биржевой торговли

# Тестирование торгового робота на Московской бирже в режиме «песочницы»

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



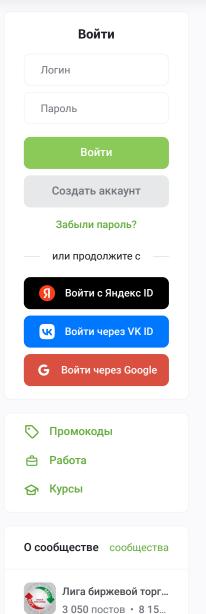
Операция продажи через OrdersService/PostOrder

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 АРІ (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 используя 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. Закрыть счёт.

SilverFir-TradingBot\src\sandbox.js



Сообщество посвящено обсуждению биржевой

торговли.

Популярные авторы

RomanPaluch

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 связь API для среды песочницы.

#### 3. Функции песочницы

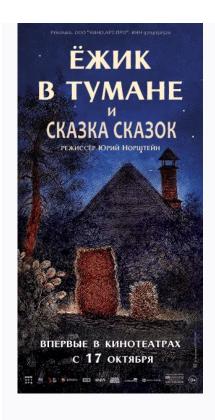
- sandboxAccount(): это основная функция, демонстрирующая различные операции с учетной записью песочницы, с несколькими действиями, которые в настоящее время закомментированы.
- logFunctionName(): регистрирует имя функции в консоли, что полезно для отслеживания в сложных приложениях.
- GetSandboxAccounts: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

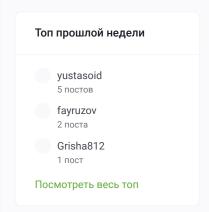
#### Закомментированные операции:

- 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 accountld, 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.

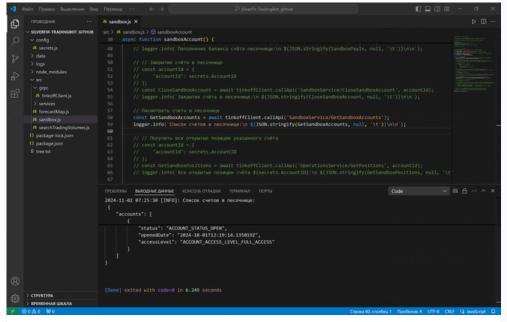
### Ошибки

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





Эта структура кода демонстрирует, как взаимодействовать с виртуальным торговым счетом в АРІ 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.



Запрос SandboxService/GetSandboxAccounts

// Импорт необходимых модулей const secrets = require('../config/secrets'); // Ключи доступа и идентификаторы

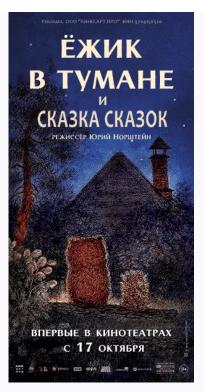
const logger = require('./services/logService'); // Логирование в файл и консоль

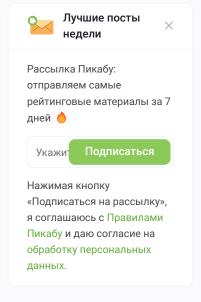
const logFunctionName = require('./services/logFunctionName'); // Получение имени функции

const TinkoffClient = require('./grpc/tinkoffClient'); // модуль для взаимодействия с API Tinkoff Invest const API\_TOKEN = secrets.TbankSandboxMode; const tinkoffClient = new TinkoffClient(API\_TOKEN);

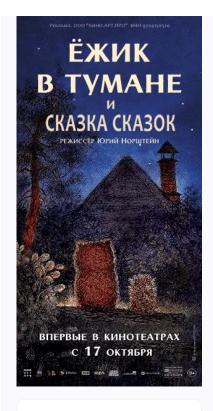
async function sandboxAccount() {
// https://tinkoff.github.io/investAPI/swaggerui/#/SandboxServ...
logger.info(`Запуск функции
\${JSON.stringify(logFunctionName())}\n`);

// // 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`);
// // Пополнение баланса счёта песочницы
// const RUB = {
// "accountId": secrets.AccountID,
// "amount": {
// "nano": 0, // Дробная часть отсутствует
// "currency": "RUB",
// "units": 30000, // Сумма в рублях
// }
// };
// const SandboxPayIn = await
tinkoffClient.callApi('SandboxService/SandboxPayIn', RUB);
// logger.info(`Пополнение баланса счёта песочницы:\n
${JSON.stringify(SandboxPayIn, null, '\t')}\n\n`);
// // Закрытие счёта в песочнице
// const accountId = {
// "accountId": secrets.AccountID
// }:
// const CloseSandboxAccount = await
tinkoffClient.callApi('SandboxService/CloseSandboxAccount',
accountId);
// logger.info(`Закрытие счёта в песочнице:\n
$\{JSON.stringify(CloseSandboxAccount, null, '\t')\\n\n');
// Посмотреть счета в песочнице
const GetSandboxAccounts = await
tinkoffClient.callApi('SandboxService/GetSandboxAccounts');
logger.info(`Список счетов в песочнице:\n
${JSON.stringify(GetSandboxAccounts, null, '\t')}\n\n');
// // Получить все открытые позиции указанного счёта
// const accountId = {
// "accountId": secrets.AccountID
// };
// const GetSandboxPositions = await
tinkoffClient.callApi('OperationsService/GetPositions',
accountId);
// logger.info(`Все открытые позиции счёта
```



Новости Пикабуерификации Помощь Награды Кодекс ПикабуКонтакты Реклама О проекте О компании Зал славы

Промокоды Купоны Мегамаркет
Скидки Купоны Спортмастер
Работа Купоны М.Видео
Курсы Купоны Aliexpress
Блоги Купоны Lamoda

Android







# Быстройдействие

Я не ждал какого-то особо быстродействия. Для человека это очень быстро, но вот для робота это медленно. Это придётся учесть при разработке торговой стратегии.

```
[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 (SBER) (BBG004730N88).
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": 1000000000
},
```

```
"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": "BBG004730N88",
"direction": "ORDER_DIRECTION_SELL",
"initialSecurityPrice": {
"currency": "rub",
"units": "235",
"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-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

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 свингтрейдинг, 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 течение нескольких дней или нескольких недель. Цель — извлечь прибыль из «колебаний» цены, используя рыночный импульс, когда цены колеблются в рамках тренда или между уровнями поддержки и сопротивления.

## 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.

