

**Михаил Шардин**

личный блог



11 ноября 2024, 04:51

[+ Подписаться](#)

Инструменты робота, торгующего на Московской бирже через API брокера

Поскольку хочу использовать для среднесрочной алгоритмической торговли на российском рынке скрипт — робота, то мне необходимо получать от брокера актуальную информацию о текущих ценах и сопутствующую информацию:

- Время работы биржи через `InstrumentsService/TradingSchedules`.
- Основную информацию об инструменте через `InstrumentsService/GetInstrumentBy`.
- Последнюю котировку по инструменту через `MarketDataService/GetLastPrices`.
- Торговые лоты — это определенное количество акций, которые можно купить или продать в рамках одной сделки.
- Свечи по инструменту для разных временных интервалов через `MarketDataService/GetCandles`.
- Технические индикаторы через `MarketDataService/GetTechAnalysis`.
- Понятное имя инструмента через `InstrumentsService/FindInstrument`.

В статье разбираюсь как проделать все эти операции при помощи программного кода.

Частному лицу для начала торговли на бирже частному инвестору необходим брокерский счёт. Но лишь у немногих российских брокеров есть собственные API (точно есть у [ФИНАМ](#), [Алор](#), [Тинькофф Инвестиции](#)). По личным предпочтениям я решил использовать API от Т-Банк (ранее известный как Тинькофф), работая в среде исполнения JavaScript Node.js.

Введите текст комментария

```

// config.js
// secrets.js
// data
// logs
// node_modules
// src
//   gpcc
//   tinkoffClient.js
//   services
//     logFunctionName.js
//     logService.js
//     forecastMaps.js
//   instruments.js
//   sandbox.js
//   searchTradingVolumes.js
// package-lock.json
// package.json
// test.js

// logger.error('Ошибка $stock', error.message);
// }
// }

// Получение полного имени инструмента
for (const stock of config.securitiesToMonitorFigiArray) {
  try {
    const name = await tinkoffClient.getName(stock);
    const nameId = name.uid;
    logger.info(`${name.nameCombination} это ${stock} или ${nameId}.`);
  } catch (error) {
    logger.error('Ошибка $stock:', error.message);
  }
}

// // Тест корректности размера автоов:
// const figi = "BBG004730NBB"; // Пример OTC
// const price = await tinkoffClient.getQuote(figi);

[Running] node "d:\Synology\SilverFir-TradingBot_github\src\instruments.js"
2024-11-08 10:11:27 [INFO]: Запуск функции "test"
2024-11-08 10:11:27 [INFO]: Запуск функции "Instruments"
2024-11-08 10:11:27 [INFO]: Банк ВТБ (VTBR) это BBG004730Z39 или 8e2b0325-0292-4654-8a18-4f63ed3b0e09.
2024-11-08 10:11:27 [INFO]: Мечел (MTLR) это BBG00456d598 или e64ba063-405f-4f80-bc29-f262793bae58.
2024-11-08 10:11:27 [INFO]: ОМК (OMK) это BBG000403V05 или a8dc1880-cae0-00c0-99e8-f561f4339751.
2024-11-08 10:11:28 [INFO]: Руссифиль (RMFT) это BBG000F9XX7M4 или c7485564-ed92-45fd-a724-1214aa202904.
2024-11-08 10:11:28 [INFO]: ЕвроТранс (EUTR) это TC500A1902V2 или 0262ea14-3c4b-47e8-9548-45a8bccc9f8a.
2024-11-08 10:11:28 [INFO]: Сургутнефтегаз - привилегированные акции (SMGSP) это BBG0045681M2 или a797f14a-b513-40b4-b15e-a3b98dc4cc00.
2024-11-08 10:11:28 [INFO]: Газпром (GAZP) это BBG0047200F0 или 9c2b2a95-02a9-4171-b0d7-a11f8bde4d3a.
2024-11-08 10:11:28 [INFO]: Роснефть (ROSN) это BBG004731354 или f0417230-19cf-4e7b-9623-f7c9c18ec0b.
2024-11-08 10:11:28 [INFO]: Сбер Банк (SBER) это BBG004730000 или a6123145-9665-43e0-8a13-cd18ba09011.
2024-11-08 10:11:29 [INFO]: Сетекс (SGTX) это BBG0100R9963 или 7bedd800-470d-4742-a20c-39d27f0db07d.
2024-11-08 10:11:29 [INFO]: Аэрофлот (AFLT) это BBG0041608M7 или 1c69e020-f3b1-415c-af6a-45f8b0049234.
2024-11-08 10:11:29 [INFO]: ВК (VKCO) это TC500A180V70 или b71bd174-c72c-41b0-a66f-5f90770d1f5f.
2024-11-08 10:11:29 [INFO]: РКСА (RUAL) это BBG0000P2T312 или f86e872b-8f68-4b6e-930f-749f09aa79c0.
2024-11-08 10:11:29 [INFO]: Татнефть (TATN) это BBG0048UUFFC0 или 88a62ffc-c67a-4fb4-a006-53ead03883c.

[Done] exited with code=0 in 2.793 seconds

```

SilverFir-TradingBot\src\instruments.js

Этот модуль служит для проверки части функций, которые будут использоваться потом в автоматическом режиме. Что он делает? Импортирует необходимые модули:

- secrets и config для конфиденциальной информации и настроек конфигурации.
- Службы для рисования диаграмм (chart), обработки CSV-файлов (csvHandler), решений о покупке/продаже (buyDecision и sellDecision) и расчета доходности (yieldCalculator).
- Служба ведения журнала (logger) для отслеживания действий и ошибок.
- TinkoffClient, модуль для взаимодействия с Tinkoff Invest API, и API_TOKEN для аутентификации.

Основные функции

Функция test():

Цель: Тестирование функциональности API и регистрация данных для конкретных биржевых инструментов.

Примеры операций:

- Получить основную информацию об инструменте — вызывает InstrumentsService/GetInstrumentBy для получения информации о определенном инструменте с использованием его идентификатора.
- Получить список всех акций — вызывает InstrumentsService/Shares для составления списка акций и регистрации первых нескольких результатов.

Примеры операций:

- Получение времени работы биржи — получает и регистрирует часы торговли.
- Найти всю информацию об акциях в списке файла config — отображает всю информацию о каждом из тикеров в JSON формате.
- Последние цены и торговые лоты — извлекает последние цены акций и проверяет размеры лотов (это определенное количество акций, которые можно купить или продать в рамках одной сделки).
- Данные свечей — собирает данные свечей (ценовые точки с течением времени) в различные интервалы (5 минут, час, день).
- Технические индикаторы — извлекает индикаторы, такие как SMA (простая скользящая средняя), для анализа тенденций акций. По выходным данных нет, хотя свечи за это же время присутствуют.
- Разместить рыночный ордер — строки кода для прямого размещения ордеров на покупку/продажу.
- Позиции портфеля — перечисляет текущие активы и вычисляет годовую доходность.

В конце код запускает `test()` и `instruments()` с обработкой ошибок, регистрируя все возникшие проблемы.

Файл `instruments.js` это ещё одна часть бота, которая позволяет частному инвестору отслеживать и взаимодействовать с платформой Tinkoff, обрабатывая все: от анализа цен акций и тенденций до размещения сделок. Настройка этого бота подходит для среднесрочной торговли на основе данных, используя Node.js для быстрой обработки данных и взаимодействия с API.

```
<code>// Импорт необходимых модулей
const secrets = require('../config/secrets'); // Ключи доступа и идентификаторы
const config = require('../config/config'); // Параметры
const chart = require('./services/chartService'); // Отрисовка графиков
const csvHandler = require('./services/csvHandler'); // Работа с CSV файлами
const buyDecision = require('./services/buyDecision'); // Функции покупки
const sellDecision = require('./services/sellDecision'); // Функции продажи
const yieldCalculator = require('./services/yieldCalculator'); // Расчёт годовой

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

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

```
const API_TOKEN = secrets.ThankSandboxMode;
```

```
logger.info(`Запуск функции ${JSON.stringify(logFunctionName())}\n`);

// // Получить основную информацию об инструменте InstrumentsService/GetInst
// const testPayload = {
//   idType: "INSTRUMENT_ID_TYPE_FIGI", // Тип идентификатора INSTRUMENT_I
//   id: "BBG004730N88" // Идентификатор инструмента
// };
// const response = await tinkoffClient.callApi('InstrumentsService/GetInstr
// logger.info(`InstrumentsService/GetForecastBy: ${JSON.stringify(response,

// // // Получить список акций InstrumentsService/Shares
// const testPayload = {
//   "instrumentStatus": "INSTRUMENT_STATUS_BASE", // https://russianinves
//   "instrumentExchange": "INSTRUMENT_EXCHANGE_UNSPECIFIED"
// };
// const response = await tinkoffClient.callApi('InstrumentsService/Shares',
// // // Отображение ответа от API
// logger.info(`Ответ: ${JSON.stringify(response, null, 2)} `); // выводится
}

async function instruments() {
  logger.info(`Запуск функции ${JSON.stringify(logFunctionName())}\n`);

  // // Получение времени работы биржи
  // const response = await tinkoffClient.callApi('InstrumentsService/TradingS
  // logger.info(`Получение времени работы биржи: ${JSON.stringify(response, n
  // await tinkoffClient.getExchangeOpen());

  // // Найти всю информацию об акциях в списке файла config
  // for (const stock of config.securitiesToMonitorTikerArray) { // securities
  //   const securitiesToMonitorTikerArrayPayload = {
  //     "query": stock,
  //     "instrumentKind": "INSTRUMENT_TYPE_SHARE"
  //   };
  //   try {
```

```
//      }
// }

// // Получить последнюю цену для акций из списка в файле config
// for (const stock of config.securitiesToMonitorFigiArray) {
//     try {
//         const quote = await tinkoffClient.getQuote(stock);
//         const name = await tinkoffClient.getName(stock);
//         logger.info(`Цена акции ${name.nameCombination} [${stock}]: ${quo
//     } catch (error) {
//         logger.error(`Ошибка ${stock}:`, error.message);
//     }
// }

// // Получение торговых лотов - это определенное количество акций, которые
// for (const stock of config.securitiesToMonitorFigiArray) {
//     try {
//         const quote = await tinkoffClient.getLot(stock);
//         const name = await tinkoffClient.getName(stock);
//         logger.info(`Торговый лот акции ${name.nameCombination} [${stock}
//     } catch (error) {
//         logger.error(`Ошибка ${stock}:`, error.message);
//     }
// }

// Получение понятного имени инструмента
for (const stock of config.securitiesToMonitorFigiArray) {
    try {
        const name = await tinkoffClient.getName(stock);
        const nameUid = name.uid;
        logger.info(`${name.nameCombination} это ${stock} или ${nameUid}.`);
    } catch (error) {
        logger.error(`Ошибка ${stock}:`, error.message);
    }
}
```

```
// logger.info(`Тест количества лотов ${figi} для покупки: ${quantity}`);

// // Получение свечей по инструменту
// for (const stock of config.securitiesToMonitorFigiArray) {
//     try {
//         const name = await tinkoffClient.getName(stock);
//         const candles5Min = await tinkoffClient.getCandles(stock, "CANDLE_5_MIN");
//         logger.info(`5-минутные свечи для ${name.nameCombination}: ${JSON.stringify(candles5Min)}`);
//         const candlesHour = await tinkoffClient.getCandles(stock, "CANDLE_1_HOUR");
//         logger.info(`Часовые свечи для ${name.nameCombination}: ${JSON.stringify(candlesHour)}`);
//         const candlesDay = await tinkoffClient.getCandles(stock, "CANDLE_1_DAY");
//         logger.info(`Дневные свечи для ${name.nameCombination}: ${JSON.stringify(candlesDay)}`);
//     } catch (error) {
//         logger.error(`Ошибка ${stock}:`, error.message);
//     }
// }

// // Получение технических индикаторов по инструменту
// for (const stock of config.securitiesToMonitorFigiArray) {
//     try {
//         const instrument = await tinkoffClient.getName(stock);
//         const instrumentUid = instrument.uid;
//         const indicatorType = "INDICATOR_TYPE_SMA"; // Пример типа индикатора
//         const interval = "INDICATOR_INTERVAL_FIVE_MINUTES"; // Пример интервала
//         const typeOfPrice = "TYPE_OF_PRICE_CLOSE"; // Тип цены (например, закрытия)
//         const indicators = await tinkoffClient.getTechIndicators(instrumentUid, indicatorType, interval, typeOfPrice);
//         logger.info(`Индикатор ${indicatorType} для ${instrument.nameCombination}: ${JSON.stringify(indicators)}`);
//     } catch (error) {
//         logger.error(`Ошибка ${stock}: ${error.message}`);
//     }
// }

// // Создание графиков пересечения свечей и индикатора для акций из списка
// for (const stock of config.securitiesToMonitorFigiArray) {
//     try {
//         const charts = chart.generateCandlestickChart(stock);
//         // chart.saveCharts(stock, charts);
//     } catch (error) {
//         logger.error(`Ошибка ${stock}: ${error.message}`);
//     }
// }
```

```

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

// // Получить все открытые позиции счёта
// const GetSandboxPositions = await tinkoffClient.getPortfolio();
// logger.info(`Все открытые позиции счёта ${secrets.AccountID}:\n ${JSON.st

// // Расчёт годовой доходности от Торгового робота
// const SilverFirBotYield = await yieldCalculator.calculateAnnualYield();
// logger.info(`Годовая доходность от Торгового робота SilverFir Bot: ${Silv

// // Получить прогнозов инвестдомов по инструменту InstrumentsService/GetFo
// const ForecastPayload = {
//   "instrumentId": "1c69e020-f3b1-455c-affa-45f8b8049234" // У Аэрофлот
// };
// const response = await tinkoffClient.callApi('InstrumentsService/GetForec
// logger.info(`InstrumentsService/GetForecastBy: ${JSON.stringify(response,
}

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

test().catch(logger.error);
instruments().catch(err => logger.error(err));</code>

```

Многие строки закомментированы, но это не потому что они не рабочие, а потому что они используются для тестов той или иной функции.

Итоги

Проект полностью представлен на Гитхабе: <https://github.com/empenoso/SilverFir-TradingBot>.

Новые модули будут загружаться по мере написания и тестирования.

Автор: [Михаил Шардин](#)

2.4K



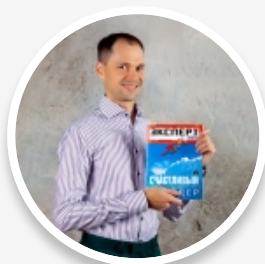
3



0



9

 **empenoso****Михаил Шар...**

Пермь

53 542

с 23 января 2019

+ Подписаться

0 КОММЕНТАРИЕВ

Сначала старые ▾

Напишите комментарий...

**ОТПРАВИТЬ**

Установите приложение Смартлаба:



Google Play



App Store



AppGallery



RuStore



О смартлабе

Реклама

Полная версия



Московская Биржа является спонсором ресурса smart-lab.ru