

empenoso

27 ноя в 03:23

Отслеживание позиций торгового робота Московской биржи через CSV файл

Простой 7 мин 1.8K

Open source*, Финансы в IT, JavaScript*, Node.JS*

Кейс

Нахожусь в процессе написания механизма торгового робота, работающего на Московской бирже через API одного из брокеров. Брокеров имеющих своё АПИ для МосБиржи катастрофически мало — мне известно только о трёх. При этом, когда я стал публиковать модули робота (и полностью [выложу готовый механизм робота на GitHub](#)), то стал получать непонимание — например, мне писали в комментариях — зачем придумывать велосипед, когда уже есть QUIK — популярная российская платформа для биржевых торгов. В Квике уже есть готовый функционал «импорт транзакций из файла» или таблица «карман транзакций». В тех же комментариях предлагали даже рассмотреть использование платформы 1C для робота, но оказалось, что торговля все равно будет осуществляться через импорт `.tri-файла` в Квик.

Лично мне Квик не очень нравится тем, что это программа для Windows. Хочется иметь механизм торгового робота, который был бы кроссплатформенным и легким — это позволит использовать его даже на «слабом» сервере. К тому же, [много лет назад](#), когда Квик был единственной альтернативой для частного лица, невозможно было внутри одной Windows без использования виртуальной машины запустить несколько копий программы технического анализа с разными системами - для того, чтобы каждая из этих копий отправляла свои сигналы на покупку и продажу в соответствующий Квик. Это было нужно для разных торговых стратегий.

По субъективным причинам я стал писать [торгового робота в среде исполнения JavaScript Node.js](#), но для тестирования на истории пришлось [использовать Python](#) и его библиотеки.

РЕКЛАМА

**Реакции уже на Хабре**

Разблокируй синих слизней

```

src > services > JS csvHandler.js > loadPositions > on('data') callback
19 const path = require('path'); // Модуль для работы с путями, файлов и директорий
20 const filePath = path.join(__dirname, '../data/positions.csv'); // Путь к файлу CSV
21 const logger = require('./logService'); // Подключаем модуль для логирования
22 const logFunctionName = require('./logFunctionName'); // Модуль для получения имени функции (для логирования)
23
24
25 // Загружаем все позиции из CSV файла
26 function loadPositions() {
27   return new Promise((resolve, reject) => {
28     const positions = [];
29     fs.createReadStream(filePath)
30       .pipe(csv())
31       .on('data', (row) => {
32         positions.push({
33           ticker: row.ticker,
34           figi: row.figi,
35           quantity: parseFloat(row.quantity), // Преобразование количества в float
36           purchaseDate: row.purchaseDate,
37           purchasePrice: parseFloat(row.purchasePrice), // Преобразование цены покупки в float
38           updateDate: row.updateDate,
39           maxPrice: parseFloat(row.maxPrice), // Преобразование максимальной цены в float
40           profitLoss: parseFloat(row.profitLoss) // Преобразование прибыли/убытков в float
41         });
42       })
43       .on('end', () => resolve(positions))
44       .on('error', reject);
45     });
46 }
47
48 // Сохраняем актуальные данные о позициях в CSV файл
49 function savePositions(positions) {
50   const csvFields = ['ticker', 'figi', 'quantity', 'purchaseDate', 'purchasePrice', 'updateDate', 'maxPrice', 'profitLoss'];
51   const csvData = parse(positions, { fields: csvFields });
52
53   fs.writeFileSync(filePath, csvData);

```

Модуль, считывающий позиции из файла

Проблемы с записью позиций в Node.js

Вообще именно этот модуль пришлось пару раз переписывать, потому что не смог сразу отладить его. Проблема была в том, что вызов модуля записи и обновления позиций осуществлялся сразу из нескольких мест и одни результаты перезаписывали другие. Но удалось разобраться и теперь всё протестировано и работает.

Дополнительно использую библиотеки `csv-parser` и `json2csv` — это популярные инструменты Node.js для обработки данных CSV, каждая из которых служит различным целям:

- **csv-parser**, это легкая и быстрая библиотека для анализа файлов CSV. Она основана на потоках, что делает ее очень эффективной для обработки больших наборов данных.
- **json2csv**, это утилита для преобразования данных JSON в формат CSV. Идеально подходит для экспорта данных из приложений в структуру, удобную для CSV, может работать как синхронно, так и асинхронно.

Установка этих библиотек:

```
npm install csv-parser json2csv
```



Реакции уже на Хабре
Разблокируй синих слизней

Этот код определяет модуль для взаимодействия с CSV-файлом для управления финансовыми торговыми позициями. Служит для загрузки, сохранения, обновления и удаления финансовых позиций, хранящихся в CSV-файле.

Ключевые библиотеки:

- `fs` : для операций файловой системы, таких как чтение и запись файлов.
- `csv-parser` : для анализа CSV-файлов в объекты JavaScript.
- `json2csv` : для преобразования объектов JavaScript в формат CSV для сохранения.
- `path` : для управления путями к файлам.
- **Интеграция**: включает пользовательские модули для ведения журнала (`logService`) и получения имен функций для лучшей отладки.

Функциональность

1. Обработка пути к файлу: использует модуль `path` для поиска CSV-файла, хранящего данные о позиции: `../../data/+positions.csv` .

2. Функции управления позицией:

`loadPositions()`:

1. Считывает CSV-файл и анализирует его в массив объектов позиции.
2. Преобразует числовые поля (`quantity`, `purchasePrice`, `maxPrice`, `profitLoss`) в числа с плавающей точкой для вычислений.
3. Возвращает обещание, которое разрешается с проанализированными данными или отклоняется в случае ошибки.

`savePositions(positions)`:

1. Преобразует массив объектов позиции обратно в формат CSV с помощью `json2csv`.
2. Перезаписывает CSV-файл обновленными данными.

`removePosition(figi)`:



Реакции уже на Хабре

Разблокируй синих слизней

```
updatePosition(newPosition):
```

1. Добавляет новую позицию или обновляет существующую в CSV-файле:
 2. Если figi существует, обновляет соответствующую позицию.
 3. В противном случае добавляет новую позицию.
 4. Сохраняет обновленный список обратно в CSV-файл.
3. Экспортированные модули: функции `loadPositions`, `updatePosition` и `removePosition` для использования в других частях робота.

Полный код csvHandler.js:

```
const fs = require('fs');
const csv = require('csv-parser');
const { parse } = require('json2csv');
const path = require('path'); // Модуль для работы с путями файлов и директорий
const filePath = path.join(__dirname, '../..data/positions.csv'); // Путь к файлу CSV
const logger = require('./logService'); // Подключаем модуль для логирования
const logFunctionName = require('./logFunctionName'); // Модуль для получения имени функции

// Загружаем все позиции из CSV файла
function loadPositions() {
  return new Promise((resolve, reject) => {
    const positions = [];
    fs.createReadStream(filePath)
      .pipe(csv())
      .on('data', (row) => {
        positions.push({
          ticker: row.ticker,
          figi: row.figi,
          quantity: parseFloat(row.quantity), // Преобразование количества в float
          purchaseDate: row.purchaseDate,
          purchasePrice: parseFloat(row.purchasePrice), // Преобразование цены по
          updateDate: row.updateDate,
          maxPrice: parseFloat(row.maxPrice), // Преобразование максимальной цены
          profitLoss: parseFloat(row.profitLoss) // Преобразование прибыли/убыток
        });
      })
      .on('end', () => {
        resolve(positions);
      })
      .on('error', reject);
  });
}
```



Реакции уже на Хабре

Разблокируй синих слизней

```
.on( 'error', reject);
```

```

    });
}

// Сохраняем актуальные данные о позициях в CSV файл
function savePositions(positions) {
    const csvFields = ['ticker', 'figi', 'quantity', 'purchaseDate', 'purchasePrice', 'updateDate'];
    const csvData = parse(positions, { fields: csvFields });

    fs.writeFileSync(filePath, csvData);
}

// Удаляем позицию из CSV файла (после продажи)
function removePosition(figi) {
    loadPositions().then(positions => {
        const updatedPositions = positions.filter(position => position.figi !== figi);
        savePositions(updatedPositions);
    });
}

// Добавляем новую позицию или обновляем существующую в CSV файле
function updatePosition(newPosition) {
    loadPositions().then(positions => {
        const index = positions.findIndex(pos => pos.figi === newPosition.figi);

        if (index === -1) {
            // Добавляем, если не нашли существующую позицию
            positions.push(newPosition);
        } else {
            // Обновляем, если позиция уже существует
            positions[index] = newPosition;
        }

        savePositions(positions);
    });
}

module.exports = { loadPositions, updatePosition, removePosition };

```

Мой модуль checkCSVpositions.js

Этот модуль важен для обеспечения согласованности данных между локальным CSV-файлом и



Реакции уже на Хабре

Разблокируй синих слизней

Основные функции

1. Интеграция с внешними системами

- T-Bank Invest API: взаимодействует с API для извлечения торговых позиций в реальном времени.
- CSV File Management: использует локальный CSV-файл для хранения и управления представлением бота о торговых позициях.

2. Проверка согласованности

- Сравнивает позиции из CSV-файла с позициями с сервера T-Bank Invest API.
- Проверяет как количество, так и наличие позиций для обнаружения несоответствий.

3. Обработка ошибок

- Регистрирует подробные ошибки при обнаружении несоответствий.
- Останавливает торговые операции для предотвращения дальнейших действий на основе неверных данных.

Основные функции:

1. `getServerPositions()`

- Извлекает все открытые позиции из T-Bank Invest API.
- Извлекает позиции с ценными бумагами и преобразует баланс в float для сравнения.
- Регистрирует ответ сервера для отладки и аудита.

2. `checkForDiscrepancies()`

- Загружает данные CSV: считывает локальную запись позиций бота с помощью `csvHandler`.

Сравнивает позиции:



Реакции уже на Хабре

Разблокируй синих слизней

- Извлекает размер лота для точного сравнения количества.
- Если обнаружены расхождения в количестве или отсутствующие позиции, регистрирует ошибки и останавливает торговлю.
- Статус журнала: подтверждает, когда все позиции совпадают, и позволяет продолжить торговлю.

Рабочий процесс

1. Извлечение позиций:

- Локальные позиции загружаются из CSV-файла.
- Позиции сервера извлекаются через API Tinkoff.

2. Обнаружение расхождений:

Для каждой позиции в CSV-файле:

3. Код вычисляет общее количество в лотах (`csvPosition.quantity * lotSize`).
4. Сравнивает с балансом на сервере.

Ошибки регистрируются, если:

5. Количества не совпадают.
6. Позиция в CSV-файле отсутствует на сервере.

3. Безопасность робота:

- Любые обнаруженные расхождения вызывают ошибку, останавливающую торговые операции.
- Не позволяет роботу совершать сделки на основе устаревших или неверных данных.

Полный код `checkCSVpositions.js`:

```
const logger = require('./logService'); // Логирование в файл и консоль
const logFunctionName = require('./logFunctionName'); // Получение имени функции
```



Реакции уже на Хабре

Разблокируй синих слизней

```
const config = require('../config/config'), // параметры
```

```
const csvHandler = require('./csvHandler'); // Работа с CSV файлами

const TinkoffClient = require('../grpc/tinkoffClient'); // Модуль для взаимодействия с API
const API_TOKEN = secrets.TbankSandboxMode;
const tinkoffClient = new TinkoffClient(API_TOKEN);

// Функция для получения всех позиций с сервера
async function getServerPositions() {
  try {
    const accountId = {
      accountId: secrets.AccountID
    };
    const response = await tinkoffClient.callApi('OperationsService/GetPositions', account

    // Логируем полученные позиции с сервера
    logger.info(`Все открытые позиции счета ${secrets.AccountID}: \n ${JSON.stringify(response)}`);

    // Возвращаем только позиции с ценными бумагами (securities)
    return response.securities.map(sec => ({
      figi: sec.figi,
      balance: parseFloat(sec.balance) // Преобразуем баланс в float
    }));
  } catch (error) {
    logger.error(`Ошибка при получении позиций с сервера: ${error.message}`);
    throw error;
  }
}

// Функция для проверки расхождений
async function checkForDiscrepancies() {
  try {
    // Загружаем текущие позиции из CSV файла
    var csvPositions = await csvHandler.loadPositions();

    // Получаем позиции с сервера
    const serverPositions = await getServerPositions();

    // Проверяем каждую позицию из CSV
    for (const csvPosition of csvPositions) {
      // Находим соответствующую позицию с сервера
      const serverPosition = serverPositions.find(pos => pos.figi === csvPosition.figi);
```

**Реакции уже на Хабре**

Разблокируй синих слизней


```
const csvTotal = csvPosition.quantity * lotSize;

// Сравниваем количество позиций
if (csvTotal !== serverPosition.balance) {
    // Если есть расхождение, логируем ошибку и останавливаем торгового робота
    logger.error(`Ошибка: Несоответствие по FIGI ${csvPosition.figi}. CSV: ${csvTotal}, сервер: ${serverPosition.balance}`);
    throw new Error('Найдено несоответствие позиций. Остановка торговли.');
```

```
    }
} else {
    logger.error(`Ошибка: Позиция с FIGI ${csvPosition.figi} отсутствует на сервере`);
    throw new Error('Найдено несоответствие позиций. Остановка торговли.');
```

```
    }
}

logger.info('Все позиции совпадают. Торговля продолжается.');
```

```
} catch (error) {
    logger.error(`Ошибка при проверке позиций: ${error.message}`);
    // Останавливаем торгового робота (добавьте здесь вашу логику остановки)
}
```

```
}

// Экспортируем функции
module.exports = {
    checkForDiscrepancies
};

// checkForDiscrepancies().catch(logger.error);
```

Итоги

Проект полностью представлен на [Гитхабе](#). Новые модули будут загружаться по мере написания и тестирования.

Автор: [Михаил Шардин](#)

27 ноября 2024 г.

Теги: [московская биржа](#), [мосбиржа](#), [моех](#), [моехalgo](#), [tbank](#), [t-bank invest api](#)

Хабы: [Open source](#), [Финансы в IT](#), [JavaScript](#), [Node.JS](#)



Реакции уже на Хабре

Разблокируй синих слизней

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

**168**

Карма

57

Рейтинг

Михаил Шардин @empenoso

Разработчик

[Подписаться](#)[Сайт](#) [Сайт](#) [Github](#) [Telegram](#) [Комментарии 2](#)

Публикации

[ЛУЧШИЕ ЗА СУТКИ](#)[ПОХОЖИЕ](#)**engine9**

20 часов назад

Экспортный измеритель влажности древесины IV1-1

**Простой**

4 мин



2.3K

[Обзор](#)**+47**

10



8

**myops**

23 часа назад

Йо-хо-хо и бутылка типографской краски

**Простой**

17 мин



2.2K

[Мнение](#)**Реакции уже на Хабре**

Разблокируй синих слизней

**zatih**

23 часа назад

Дешманский ремонт дешманского телевизора

**Простой**

8 мин



5.2K

Тutorial

+36

19

11

**AlexMih**

20 часов назад

Как я однажды P-CAD видел



10 мин



4.5K

Из песочницы

+33

18

19

**LeonidFirstov**

15 часов назад

Как я выучил английский



5 мин



8.5K

+30

58

12

**FanatPHP**

15 часов назад

В документации PHP теперь можно выполнять примеры кода прямо на странице

**Простой**

2 мин



3.3K

Обзор

+27

11

10

**youtube_com_chevichelov**

16 часов назад

Контролька автоэлектрика с голосовым управлением AI мобильным

**Реакции уже на Хабре**

Разблокируй синих слизней

Тutorial

+22

21

3



ptsecurity

44 минуты назад

DaMAgeCard: как индустрия карт памяти SD воскресила DMA-атаки

24 мин 403

Кейс

+21

4

2



DandyDan

34 минуты назад

Регулярные выражения простыми словами. Часть 2

Простой 10 мин 1.1K

Tutorial

+17

13

1



full_moon

21 час назад

Gemini вырывается вперед, Китай спамит моделями, в Minecraft запустили AI-агентов: главные события ноября в сфере ИИ

Средний 13 мин 1.2K

Обзор

+17

9

1

О чем поговорить с легендарным инженером на ИТ-конференции мечты?

Опрос

Показать еще



Реакции уже на Хабре

Разблокируй синих слизней

ИСТОРИИ



Запустили спутник «Кондор-ФКА» № 2



Симфония технологий



Путешествие в мир ИИ



Новости нейромира



Прокачать коммуникацию



**Переверни календарь
добавить**

КУРСЫ

SE Интеграция систем. Разработка требований и основы проектирования

11 января 2025 · Systems Education

SE Systems Analyst Bootcamp: Проектировщик корпоративных информационных систем

16 января 2025 · Systems Education

Java для начинающих

По факту набора · Университет «Синергия»

Frontend-разработчик

По факту набора · Университет «Синергия»

JS Intensive Course

По факту набора · IT INCUBATOR

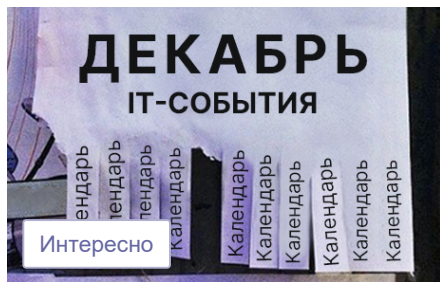
[Больше курсов на Хабр Карьере](#)

МИНУТОЧКУ ВНИМАНИЯ



Реакции уже на Хабре

Разблокируй синих слизней



Конференции, хакатоны, митапы:
что посетить в декабре



Кто завоевал сердца айтишников
в 2024 году



ИИ в DevOps: что изменилось за
этот год

РАБОТА

[Node.js разработчик](#)

55 вакансий

[JavaScript разработчик](#)

159 вакансий

[React разработчик](#)

43 вакансии

[Все вакансии](#)

БЛИЖАЙШИЕ СОБЫТИЯ



Реакции уже на Хабре

Разблокируй синих слизней



8 октября – 4 декабря

ТурбоХакатон «Решения для электроэнергетики на базе искусственного интеллекта»

Онлайн

Разработка

Другое

[Больше событий в календаре](#)

Хабр



🌐 [Настройка языка](#)



Реакции уже на Хабре

Разблокируй синих слизней

© 2006–2024, Habr



Реакции уже на Хабре

Разблокируй синих слизней