empenoso
27 ноя 2024 в 03:23

Отслеживание позиций торгового робота Московской биржи через CSV файл

Простой 7 мин 2.5K

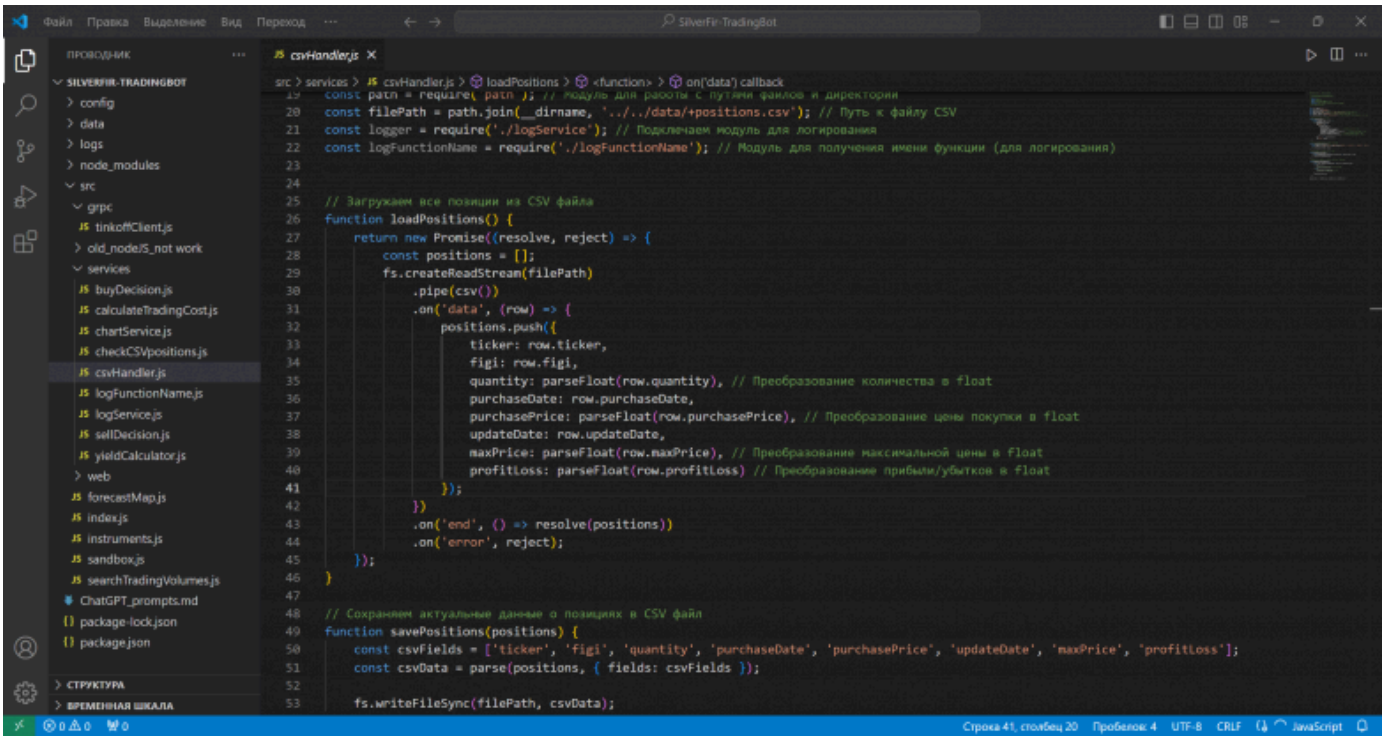
Open source*, Финансы в IT, JavaScript*, Node.JS*

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 1С 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.



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:

Мой модуль csvHandler.js

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

1. Удаляет позицию из CSV-файла на основе ее figi (уникального идентификатора).
2. Загружает все позиции, отфильтровывает указанную и перезаписывает файл.

`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 фун


// 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
```

```
                    maxPrice: parseFloat(row.maxPrice), // Преобразование максимальной
                    profitLoss: parseFloat(row.profitLoss) // Преобразование прибыли/уб
                });
            })
            .on('end', () => resolve(positions))
            .on('error', reject);
    });
}

// Сохраняем актуальные данные о позициях в CSV файл
function savePositions(positions) {
    const csvFields = ['ticker', 'figi', 'quantity', 'purchaseDate', 'purchasePrice', '
    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 текущими позициями, полученными из T-Bank Invest API. Он проверяет наличие несоответствий, которые могут привести к ошибкам в торговых операциях, и останавливает робота, если обнаруживаются несоответствия.

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 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:

- Для каждой позиции CSV ищет соответствующую позицию на сервере с помощью FIGI (уникальный идентификатор).
- 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

- 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

const secrets = require('../../config/secrets'); // Ключи доступа и идентификаторы
const config = require('../../config/config'); // Параметры
const csvHandler = require('./csvHandler'); // Работа с CSV файлами

const TinkoffClient = require('../grpc/tinkoffClient'); // Модуль для взаимодействия с
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',

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

        // 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;
    }
}
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 === csvPositior

            if (serverPosition) {
                const lotSize = await tinkoffClient.getLot(csvPosition.figi);
                logger.info(`Количество бумаг в лоте ${csvPosition.figi}: ${lotSize} шт
                const csvTotal = csvPosition.quantity * lotSize;

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

Итоги

Проект полностью представлен на Гитхабе. Новые модули будут загружаться по мере написания и тестирования.

Автор: Михаил Шардин
Моя онлайн-визитка
Telegram «Умный Дом Инвестора»

27 ноября 2024 г.

Теги: московская биржа, мосбиржа, moex, moexalgo, tbank, t-bank invest api, алгоритмическая торговля

Хабы: Open source, Финансы в IT, JavaScript, Node.JS

Комментарии 2

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

Exosphere
15 часов назад

Ещё 10 ошибок авторов Хабра

11 мин 3.2K

+93 35 65

vital_pavlenko
22 часа назад

Больше нет входа в IT. Только выход

+49 78 327

duran-duran
21 час назад

Трамплин в интернет: как мы ускорили запуск Яндекс Браузера

6 мин 3.3K

+42 9 29

Nickmob
16 часов назад

Введение в Angie: краткая история и отличия от Nginx

Простой 7 мин 2.8K

Ретроспектива

+37 29 8

Myskat_90
21 час назад

Распределённый инференс и шардирование LLM. Часть 1: настройка GPU, проброс в Proxmox и настройка Kubernetes

Сложный 14 мин 1.8K

Туториал

+33 46 0

slava_rumin
15 часов назад

Мое производство электрощитов приносит 40 млн в год. Спасибо нейросетям и СССР за конструкторскую школу

Простой 14 мин 22K

Интервью

AlexeyNadezhin
23 часа назад

Важное обновление BatteryTest 2

Простой 3 мин 2.5K

+27 37 9

ru_vds
16 часов назад

Как создавались вокальные эффекты Daft Punk

Средний 13 мин 842

Обзор Перевод

+23 5 1

GordienkoAnd
19 часов назад

Как настраивать сети: готовые решения Selectel для максимальной отказоустойчивости

Средний 20 мин 2.5K

Обзор

+21 28 0

alizar
20 часов назад

Почему из технологий делают культы

Простой 8 мин 1.3K

Обзор

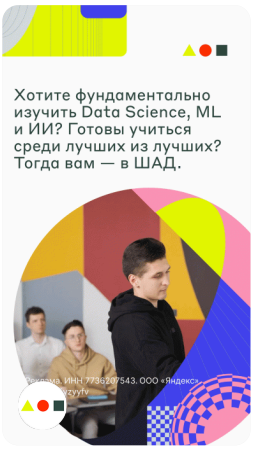
+21 7 6

Показать еще

ИСТОРИИ

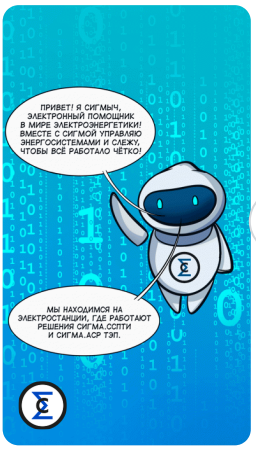
Старт гонки разработчиков

Торопись в сезон Open source



Открыт приём в Школу анализа данных

С Днём радио!



HELLO, WORLD — теперь на 110 киловольтах

Буду лучш

ВАКАНСИИ

Senior Frontend (JavaScript) разработчик
от 350 000 до 400 000 ₽ · Vital Partners · Можно удаленно

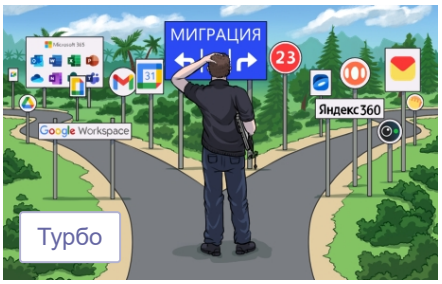
Tech Lead/ Team Lead (JavaScript)
до 200 000 ₽ · SteadyControl · Воронеж

Fullstack JavaScript разработчик
от 30 000 до 80 000 ₽ · MakeDifference · Можно удаленно

Backend developer (nodejs)
от 200 000 до 300 000 ₽ · SwiftDrive · Можно удаленно

Backend Engineer (Rust, NodeJS)
до 125 000 ₽ · REES46 · Можно удаленно

Больше вакансий на Хабр Карьере



Как избежать ошибок при бизнес-миграции и сохранить данные?

Экономим деньги со скидками в Промокодусе

Работа в атомной энергетике: это вам не только про АЭС

РАБОТА

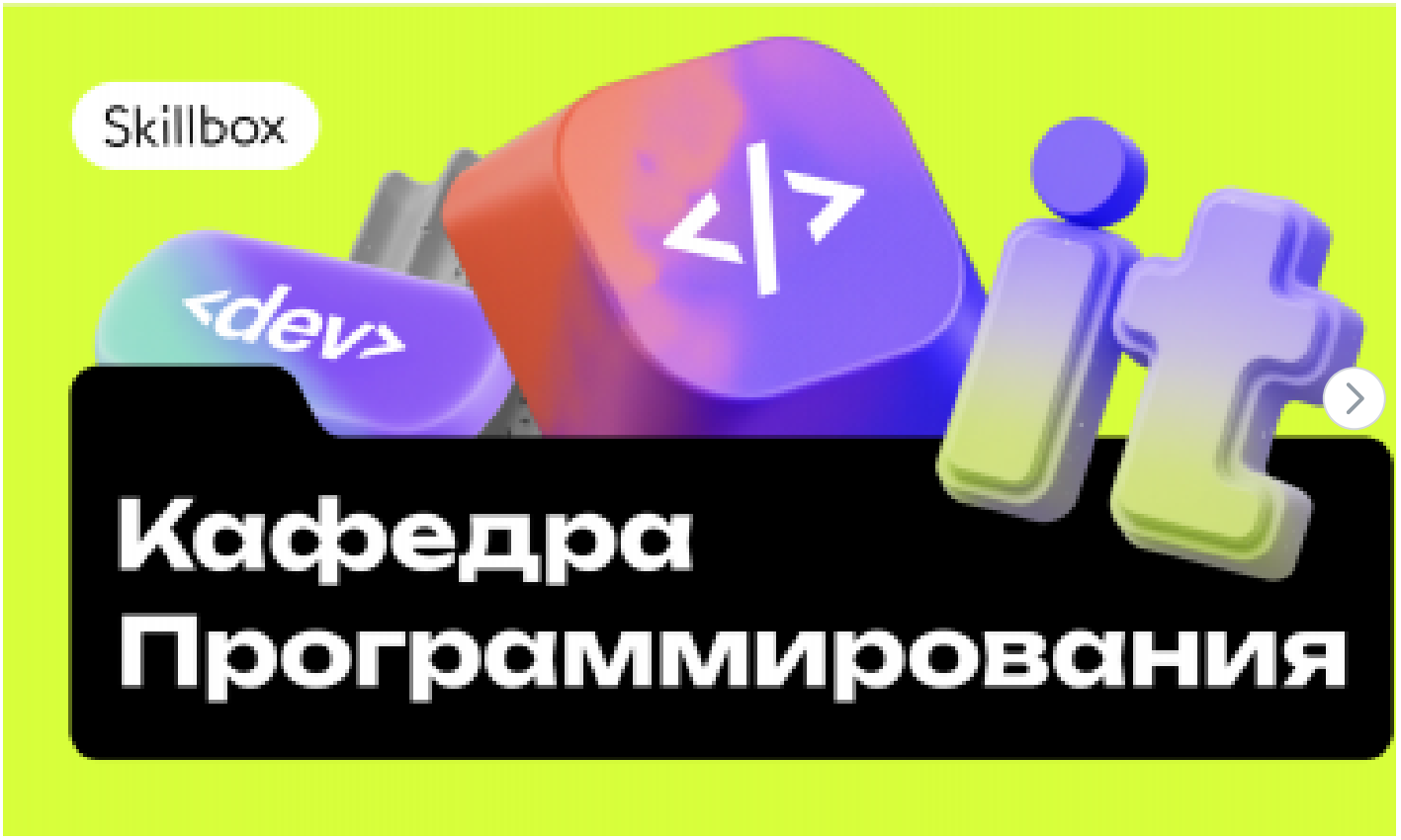
JavaScript разработчик
109 вакансий

React разработчик
37 вакансий

Node.js разработчик
41 вакансия

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ



17 апреля – 29 мая

Серия бесплатных офлайн-конференций «Кафедра Программировани от Skillbox

Москва

Разработка

Больше событий в календаре

Хабр