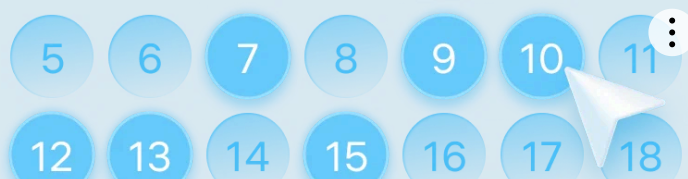




Все важнейшие  
IT-события здесь

**empenoso**

27 ноя в 03:23

## Отслеживание позиций торгового робота Московской биржи через CSV файл

👉 Простой ⌚ 7 мин 👁 1.9K

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](#)

РЕКЛАМА

**Ночь в дата-центре**

Ваша самая необычная смена

```

src > services > JS csvHandler.js > loadPositions > <function> > on('data') callback
15 const path = require('path'); // модуль для работы с путями, файлов и директорий
16 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:



**Ночь в дата-центре**

Ваша самая необычная смена

## 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.



**Ночь в дата-центре**

Ваша самая необычная смена

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 функ

// 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 ф
          purchaseDate: row.purchaseDate,
          maxPrice: parseFloat(row.maxPrice), // Преобразование максимальной ц
```



**Ночь в дата-центре**

Ваша самая необычная смена

```
        profitLoss: parseFloat(row.profitLoss) // Преобразование прибыли/убы
    });
  })
  .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', 'u
  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);
  });
}
```

**Ночь в дата-центре**

Ваша самая необычная смена

## 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 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



**Ночь в дата-центре**

Ваша самая необычная смена

- 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.

### 3. Безопасность робота:

- Любые обнаруженные расхождения вызывают ошибку, останавливающую торговлю.



**Ночь в дата-центре**

Ваша самая необычная смена

## 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 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', accountId);

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

    // Возвращаем только позиции с ценными бумагами (securities)
    return response.positions.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
    const csvPositions = csvHandler.loadCSVPositions();
```



**Ночь в дата-центре**

Ваша самая необычная смена



```

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);

  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 робота
      logger.error(`Ошибка: Несоответствие по FIGI ${csvPosition.figi}. CSV баланс: ${csvTotal}, серверный баланс: ${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



### Ночь в дата-центре

Ваша самая необычная смена

Автор: [Михаил Шардин](#)

27 ноября 2024 г.

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

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

## Редакторский дайджест

Присылаем лучшие статьи раз в месяц



**176** **135.2**

Карма Рейтинг

**Михаил Шардин** [@empenoso](#)

Разработчик

Подписаться



[Сайт](#) [Сайт](#) [Github](#) [Telegram](#)

РЕКЛАМА



### Вы развиваете, мы защищаем

Инструменты кибербезопасности для вашего бизнеса

Реклама. АО «Лаборатория Касперского». ОГРН 1027739867473  
125212, Россия, г. Москва, Ленинградское шоссе, д. 39А, стр. 2. [www.kaspersky.ru](http://www.kaspersky.ru)

Подробнее

**kaspersky**

Комментарии 2



### Ночь в дата-центре

Ваша самая необычная смена

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ



Devastator82

14 часов назад

## Заметки поададца о Руанде 2000-х



13 мин



5.5K

Из песочницы



+98



23



14



ru\_vds

21 час назад

## Как долго можно извлечь данные из RAM после отключения компьютера?



Средний



11 мин



11K

Обзор

Перевод



+53



50



14



Loxmatiyamamont

23 часа назад

## Будущее PostgreSQL: как 64-битный счетчик транзакций решает проблему масштабирования



Средний



7 мин



5K



+37



34



14



Finam\_Broker

22 часа назад

## \$20 000 000 для хикикомори: история успеха Такаши Котегавы, подловившего банк Mizuho на ошибке



Простой



5 мин



5.7K

Ретроспектива



Ночь в дата-центре

Ваша самая необычная смена

**techno\_mot**

22 часа назад

## Самые дерзкие фишинговые схемы 2024 года, которые сработали

**Простой**

10 мин



8.7K

Обзор

**+29**

19



6

**gserge**

4 часа назад

## Менеджмент менеджмента: как во «Фланте» внедрили принятие решений эфемерными рабочими группами

**Простой**

7 мин



540

**+26**

5



0

**kuksepa**

23 часа назад

## AWX: упрощаем работу с Ansible

**Простой**

12 мин



3K

**+24**

36



1

**OldFashionedEngineer**

2 часа назад

## Защита схемы от переполюсовки, что может быть проще?

**Средний**

15 мин



1.3K

Тutorial

**+22**

19



9

**GordienkoAnd**

2 часа назад

## Как настроить сеть выделенный и облачный серверы

**Ночь в дата-центре**

Ваша самая необычная смена

Обзор

+16

9

0



spacesteel

23 часа назад

## «Как не надо проектировать ракеты» или путь разработки ракеты на ЖРД. Часть 4 — Как не сделать парашют



Простой



9 мин



2.1K

Тutorial

+16

13

1

## Рабочая схема: делимся рейтингом работодателей в IT за 2024 год

Турбо

Показать еще

### ИСТОРИИ



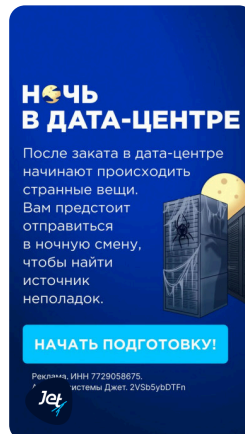
Новый хаб для разработчиков облака



Если Elixir — твоя партия



Спасибо, КЭП



Неполадки в ночном дата-центре



Космическая акселерация



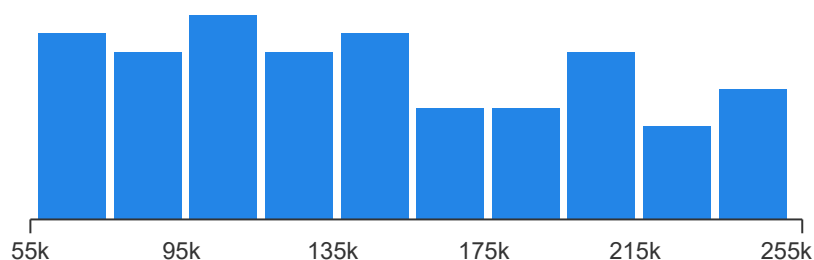
Маги: тести зовет

### СРЕДНЯЯ ЗАРПЛАТА В IT

**Ночь в дата-центре**

Ваша самая необычная смена

— средняя зарплата во всех IT-специализациях по данным из 63 370 анкет, за 2-ое пол. 2024 года. Проверьте «в рынке» ли ваша зарплата или нет!

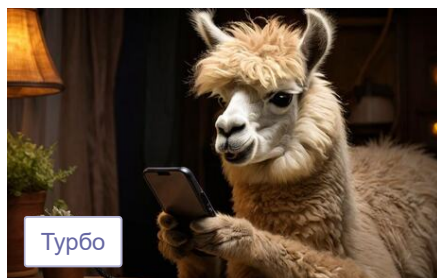


[Проверить свою зарплату](#)

## МИНУТОЧКУ ВНИМАНИЯ



**Конференции, хакатоны, митапы: что посетить в декабре**



**Карманная Llama: запускаем языковую модель в браузере**



**Боли стартапов и акселерация: выводим MVP на рынок**

## РАБОТА

[React разработчик](#)

41 вакансия

[JavaScript разработчик](#)

146 вакансий

[Node.js разработчик](#)

55 вакансий

[Все вакансии](#)



**Ночь в дата-центре**

Ваша самая необычная смена



25 – 26 апреля

## IT-конференция Merge Tatarstan 2025

Казань

Разработка

Маркетинг

Другое

[Больше событий в календаре](#)

### Хабр



 [Настройка языка](#)



#### Ночь в дата-центре

Ваша самая необычная смена

© 2006–2024, Habr



### Ночь в дата-центре

Ваша самая необычная смена