empenoso
24 авг 2020 в 06:25

Поиски фундаментальных данных для акций через API Financial Modeling Prep

127 мин 8.8K

Алгоритмы*, Node.JS*, API*, Финансы в IT

Недавно мне понадобилось обработать экономические показатели для нескольких тысяч американских акций.

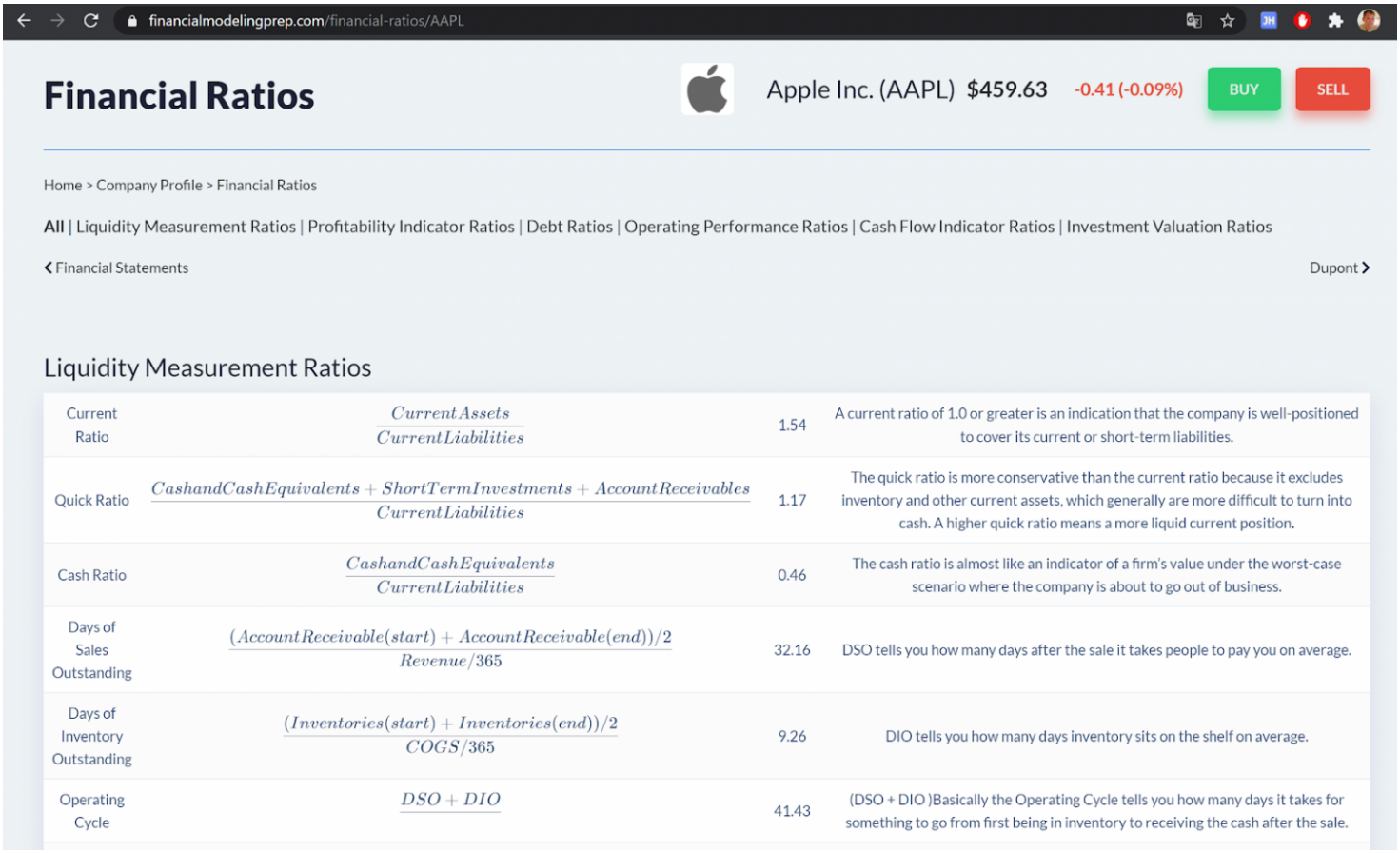
Их невозможно было получить через привычный скринер бумаг вроде яху финанс, потому что методика расчёта нестандартная.

В качестве поставщика данных использовался сервис FinancialModelingPrep, который в 2019 году был бесплатен, но в 2020 году уже нет.

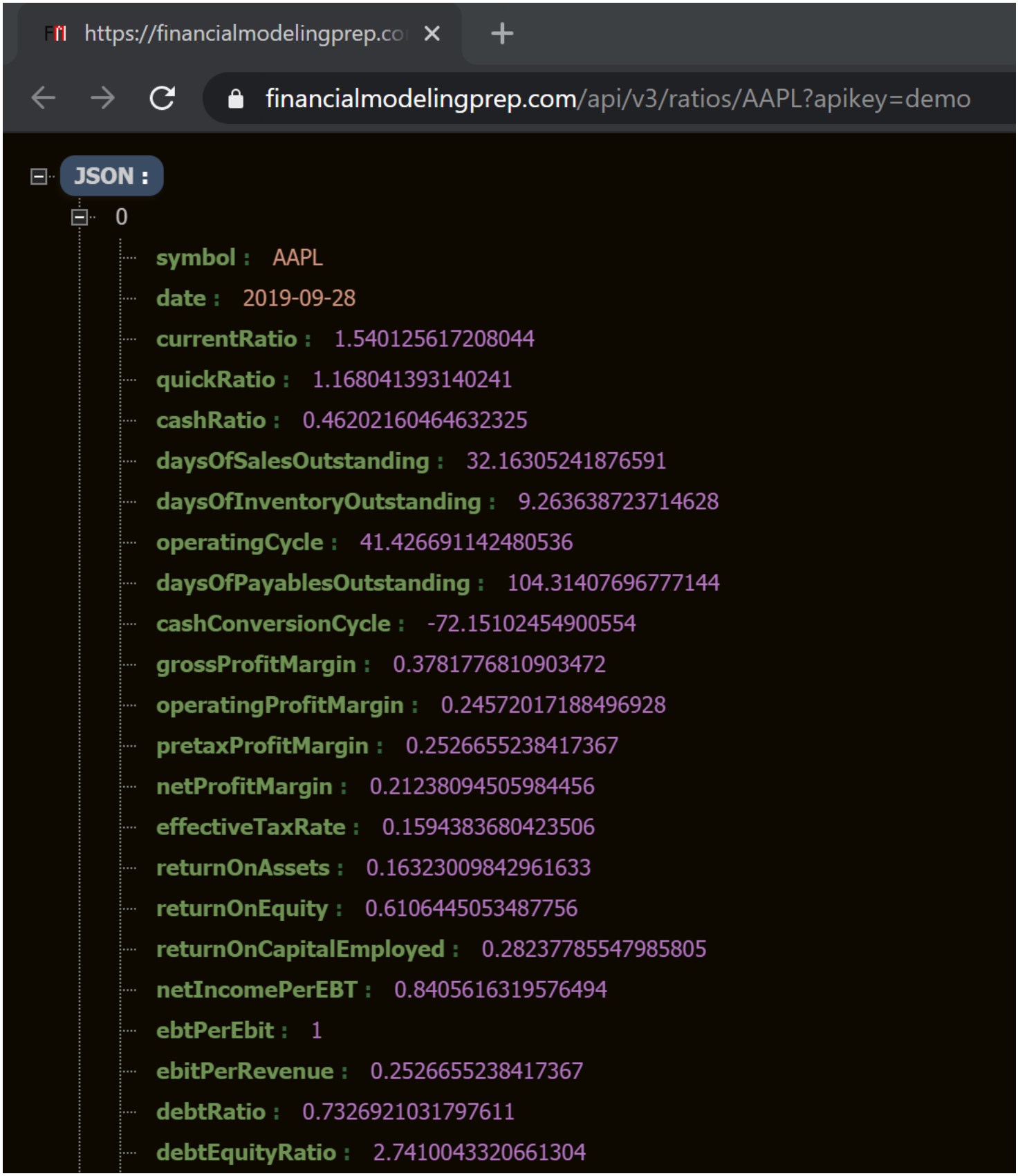
В статье разбираюсь в нюансах формирования запросов к базе данных сервиса. А ещё исследую глубину доступных финансовых отчетов компаний за прошлые годы.

Экономические показатели

Для каждой из акций Financial Modeling Prep приводит множество показателей на основании ежегодных отчетов компании, например для Apple Inc. (AAPL):



Эти же данные по годам можно получать и через API:



Для Apple Inc. (AAPL) глубина выборки в API 34 года: с 1985 по 2019 годы.

Эти показатели уже рассчитаны и доступна по годам — это готовый ряд данных, которые теоретически можно прогнозировать на несколько периодов вперед, поскольку есть история за 30 и более периодов назад.

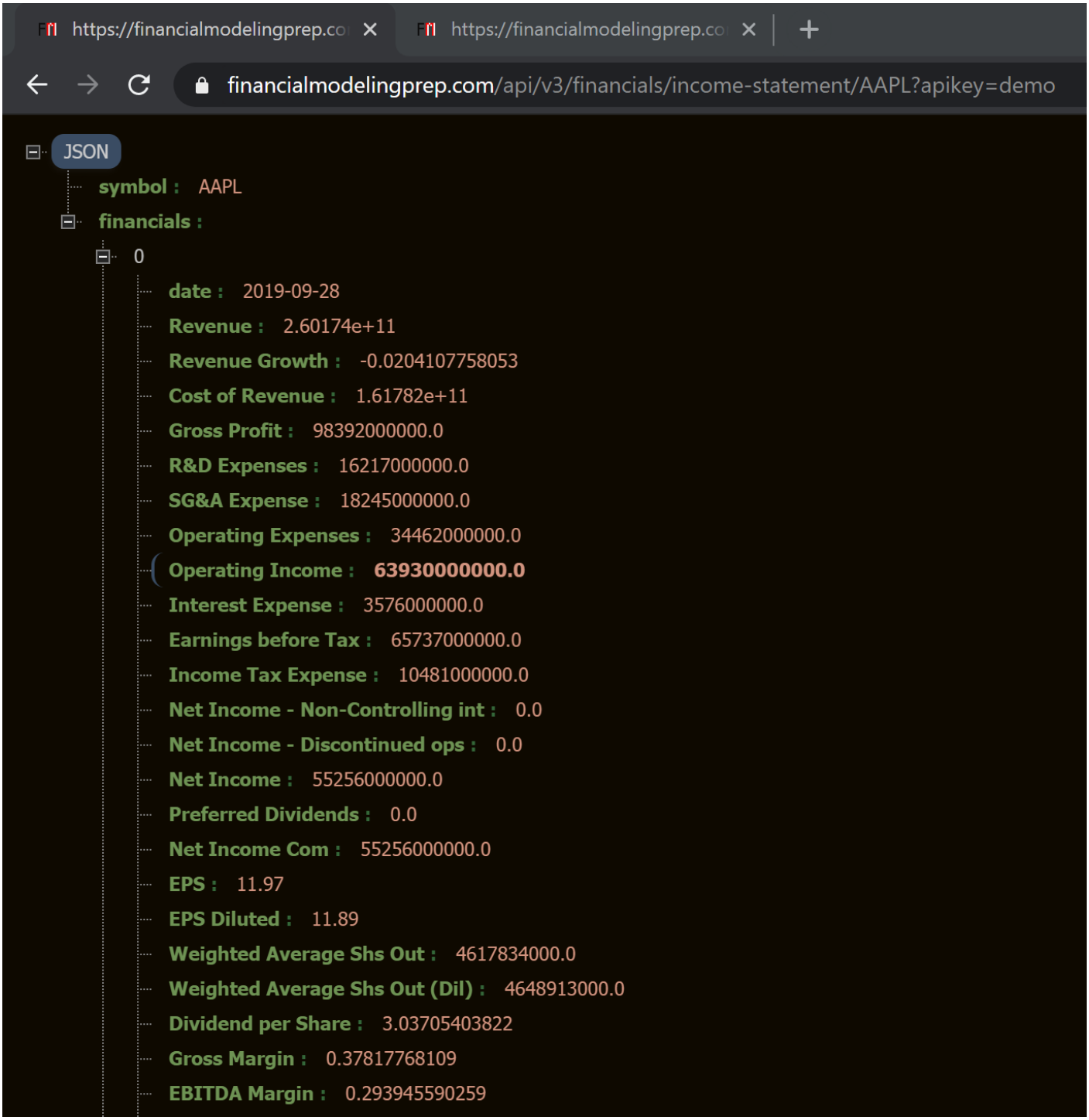
Эти показатели рассчитываются на основании данных из финансовых отчетов, которые доступны за тот же период времени, что и финансовые отчеты компании:

1. Income Statement
2. Balance Sheet Statement
3. Cash Flow Statement

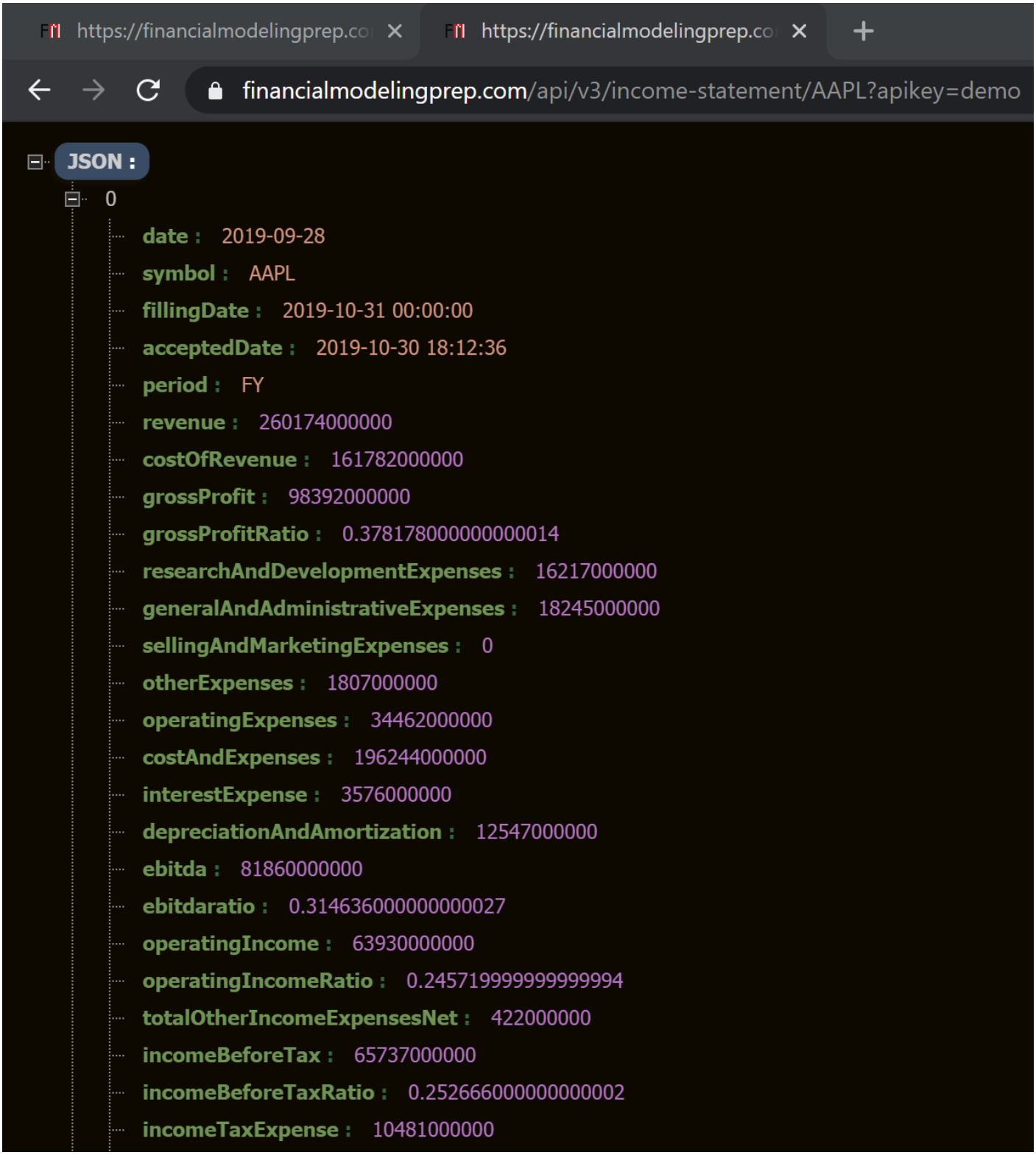
Тонкости, не описанные в документации

Однако есть нюанс — данные отчетов можно смотреть одновременно по двум адресам:

1. https://financialmodelingprep.com/api/v3/financials/income-statement/AAPL?apikey=demo



2. https://financialmodelingprep.com/api/v3/income-statement/AAPL?apikey=demo



Если в пути запроса добавлено `financials`, то данные приводятся только за 10 лет и названия параметров даны в человеческом формате, например разводненная прибыль на акцию (Diluted Earnings Per Share): « `EPS Diluted` » по адресу `JSON.financials[0][«EPS Diluted»]` в `financials`.

Против этого, без добавления « `financials» всё выглядит по другому:` « `epsdiluted` » по

адресу `JSON[0].epsdiluted`, однако в этом случае выводится полная история (все годы) по данному тикеру.

Ещё один нюанс — если искать например, Market cap — капитализацию компании (это стоимость одной акции, умноженную на их количество на бирже) на дату отчёта, то информация тоже есть в нескольких разделах API:

1. В разделе Company Key Metrics с понятными обозначениями «`Market Cap`», но только за 10 лет.
2. В разделе Company Enterprise Value с обозначением «`marketCapitalization`», зато за все доступные годы.

И последний нюанс — в разных разделах API financialmodelingprep.com одни и те же показатели могут называться совершенно по разному. Например связанное с обратным выкупом название «`Issuance (buybacks) of shares`» в других разделах трансформируется в «`commonStockRepurchased`».

Отчёты за последние годы

Ещё меня интересовала доступность отчетов за последние годы. Потому что столкнулся с тем, что на лето 2020 года по некоторым бумагам отчёты были только за 2018 год.

Написал скрипт на Node.js для того, чтобы провести исследование:

```
async function FinancialModelingPrepScrenner() { // Stock Screener
    var sectorArray = ["Consumer Cyclical", "Energy", "Technology", "Industrials", "Financ
    var symbolArray = []
    var log = ''
    for (var e = 0; e < sectorArray.length; e++) {
        const url = `https://financialmodelingprep.com/api/v3/stock-screener?sector=${sect
        // console.log(`\n${getFunctionName()}. Ссылка на скринер в секторе ${sectorArray[
        const response = await fetch(url)
        const json = await response.json()
        for (var j = 0; j < json.length; j++) {
            symbol = json[j].symbol
            companyName = json[j].companyName
            sector = json[j].sector
            exchange = json[j].exchange
            if (sector) {
                // console.log(`${getFunctionName()}. Компания № ${j+1} из ${json.length}:
                // symbolArray.push([symbol, companyName, sector, exchange])
                symbolArray.push(symbol)
```

```
            }
        }

        console.log(`${getFunctionName()}. В секторе ${sectorArray[e]} (${e+1} из ${sector
        log += `<li>В секторе ${sectorArray[e]} (${e+1} из ${sectorArray.length}) найдено
    }
    return {
        symbolArray: symbolArray,
        log: log
    }
}

async function FinancialModelingAvailableYears() { // поиск статистики доступных лет
    console.log(`Получаем список компаний через скринер financialmodelingprep.`)
    symbolArray = await FinancialModelingPrepScrenner()
    symbolArray = symbolArray.symbolArray
    symbolArrayUnique = symbolArray.filter((v, i, a) => a.indexOf(v) === i)
    console.log(`\nИтог: найдено ${symbolArray.length} компаний. Без повторов: ${symbolArr

    averageYears = []
    allYears = []
    notIncluded = 0
    for (var s = 0; s <= symbolArrayUnique.length - 1; s++) {
        // for (var s = 0; s <= 20; s++) { // для тестов
        ticker = symbolArrayUnique[s]

        // проверка есть ли данные в отчетах - начало
        const url = `https://financialmodelingprep.com/api/v3/ratios/${ticker}?apikey=${se
        // console.log(`Ссылка на Company financial ratios для ${ticker}: ${url}.`)
        try {
            const response = await fetch(url)
            const json = await response.json()
            if ((json.length - 1) > 0) {
                averageYears.push(json.length)
                for (var a = 0; a <= json.length - 1; a++) {
                    Year = new Date(json[a].date).getFullYear()
                    allYears.push(Year)
                    // console.log(`${ticker} (${s+1} из ${symbolArrayUnique.length}) в ${
                }
                console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) глубина
            } else {
                notIncluded += 1
                console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) нет ист
            }
        } catch (e) {
            console.log(`${ticker} (${s+1} из ${symbolArrayUnique.length}) пропускаем в ${
        }
    }
```

```
    let avg = averageYears.reduce((a, v, i) => (a * i + v) / (i + 1))
    console.log(`\nДля ${averageYears.length} акций средняя глубина отчетов: ${avg.toFixed

    let count = {};
    allYears.forEach(function (i) {
        count[i] = (count[i] || 0) + 1;
    })
    console.log(`Разбивка отчетов по годам:`)
    console.log(count)
}
```

▸ Полный лог под этим спойлером (он огромный, не открывайте без необходимости, аналитика ниже по тексту).

Получившиеся сводные цифры при сканировании базы акций на 18 августа 2020 года:

В секторе Consumer Cyclical (1 из 15) найдено 770 компаний.

В секторе Energy (2 из 15) найдено 546 компаний.

В секторе Technology (3 из 15) найдено 937 компаний.

В секторе Industrials (4 из 15) найдено 1012 компаний.

В секторе Financial Services (5 из 15) найдено 1904 компаний.

В секторе Basic Materials (6 из 15) найдено 533 компаний.

В секторе Communication Services (7 из 15) найдено 397 компаний.

В секторе Consumer Defensive (8 из 15) найдено 351 компаний.

В секторе Healthcare (9 из 15) найдено 1284 компаний.

В секторе Real Estate (10 из 15) найдено 490 компаний.

В секторе Utilities (11 из 15) найдено 179 компаний.

В секторе Industrial Goods (12 из 15) найдено 3 компаний.

В секторе Financial (13 из 15) найдено 1916 компаний.

В секторе Services (14 из 15) найдено 2304 компаний.

В секторе Conglomerates (15 из 15) найдено 1 компаний.

Итог: найдено 12627 компаний. Без повторов: 8422 шт.

Для 5308 акций средняя глубина отчетов: 14.20 лет. Однако в базе нет отчётов у 3114 бумаг.

- Свёл итоги работы скрипта в разбивке отчетов по годам в таблицу, а ниже приведён график.

Получившийся график:

Как видно из графика больше всего отчетов за 2016-2019 года.

Financial Modeling Prep vs Morningstar

Если сравнивать данные отчетов, которые приведены в Financial Modeling Prep, с данными отчетов в Morningstar, то они немного отличаются. Часто это касается последнего года. Понятна примерная причина этих несовпадений: бухгалтера вносят изменения в уже представленную ранее информацию, находят свои ошибки или регуляторы требуют что-то изменить. А в базы данных эти запоздалые изменения или не вносятся или вносятся, но в разное время разными

данных эти запоздалые изменения или не вносятся или вносятся, но в разное время разными управляющими этих баз.

Итог

Если вы готовы платить, то Financial Modeling Prep предоставляет множество данных, через удобный, но слегка запутанный интерфейс.

Автор: Михаил Шардин,

24 августа 2020 г.

Только зарегистрированные пользователи могут участвовать в опросе. Войдите, пожалуйста.

Используете финансовые API?

52.38%	Да	11
28.57%	Нет	6
19.05%	Что это?	4

Проголосовал 21 пользователь. Воздержались 8 пользователей.

Теги: парсинг, котировка, биржа, инвестиции, статистика, ценные бумаги, облигации, биржевая торговля, JavaScript, Node.JS

Хабы: Алгоритмы, Node.JS, API, Финансы в IT

124 3.3
КармаРейтинг

Михаил Шардин @empenoso

Разработчик

Сайт

Комментарии 6

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

zhzhitel
22 часа назад

Яндекс Маркет рассылает коды для получения чужих заказов и персональные данные

Простой 2 мин 21K

+66 13 149

averkij
2 часа назад

Это не чат, это GigaChat. Русскоязычная ChatGPT от Сбера

Средний 14 мин 8.5K

+58 45 33

voldemar_d
20 часов назад

MIDI chiptune + detune

Средний 15 мин 3.4K

Из песочницы

+35 23 22

Hawkeye889
21 час назад

Halo: Combat Evolved — отец современных шутеров

Простой 10 мин 2.2K

Ретроспектива

+31 15 15

AKlimenkov
21 час назад

Иллюзия упущенных возможностей. Вот если бы я тогда...

6 мин 15K

Мнение

+29 75 64

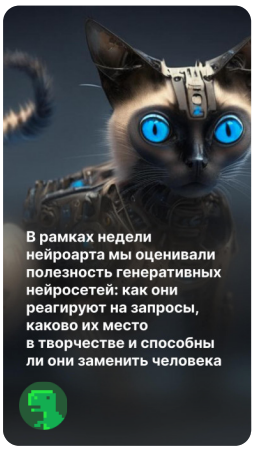
Нейросети, железо, ИИ: какой была неделя нейроарта на Хабре

Спецпроект

Показать еще

ИСТОРИИ

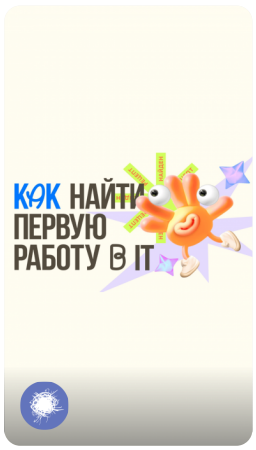
Позовите автора!



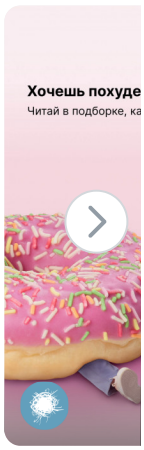
На Хабре завершилась неделя нейроарта

Достучаться до ИИ

Недельный топ годноты от компаний



Как найти первую работу в IT



Читай и ху

КУРСЫ

Офлайн-курс JavaScript-разработчик
29 мая 2023 · 29 900 ₽ · Бруноям

FullStack JavaScript программист в Москве
1 мая 2023 · 330 000 ₽ · Elbrus Coding Bootcamp

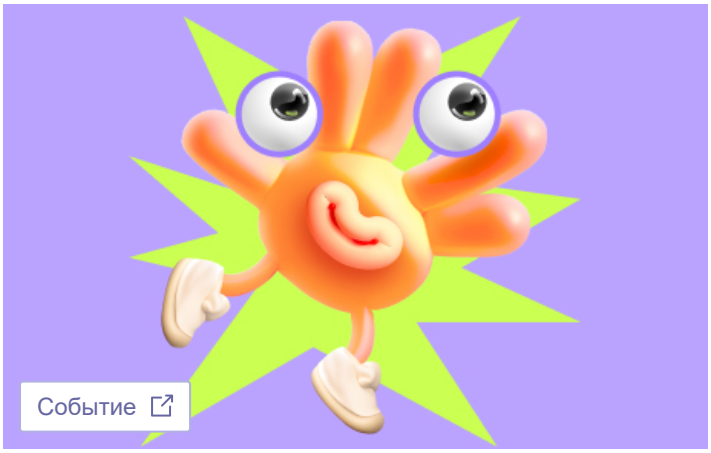
FullStack JavaScript программист в Санкт-Петербурге
1 мая 2023 · 290 000 ₽ · Elbrus Coding Bootcamp

FullStack JavaScript программист Онлайн
1 мая 2023 · 260 000 ₽ · Elbrus Coding Bootcamp

JavaScript-разработчик
16 мая 2023 · 159 000 ₽ · HTML Academy

Больше курсов на Хабр Карьере

МИНУТОЧКУ ВНИМАНИЯ



Как найти первую работу в IT и не облажаться

Как небольшой компании расцвести на Хабре за полгода

РАБОТА

Node.js разработчик
60 вакансий

JavaScript разработчик
275 вакансий

Все вакансии