



empenoso 24 августа 2020 в 04:25

# Поиски фундаментальных данных для акций через API Financial Modeling Prep

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. \(AAPL\)](#):

Реклама

**ТЕРАБАЙТ  
ИНТЕРНЕТА  
В ПОДАРОК**

iPhone 12

TELE2

### ЧИТАЮТ СЕЙЧАС

Как увидеть  $\pi$ ? Нужно швырнуть  $\pi$  в стену

3,3k 18

9 тяжелых уроков, которые я усвоил за 18 лет разработки

21,3k 43

Что в стоматологии поменялось за 30 лет, и почему заниматься наукой в России так больно

3,2k 16

Косплеи и как это работает

2,5k 4

Библиотека от AMD стала причиной плохой производительности процессоров AMD в Cyberpunk 2077

59,4k 63

Как мы переносили современные игры на процессор Эльбрус-8С

financialmodelingprep.com/financial-ratios/AAPL

Financial Ratios

Apple Inc. (AAPL)

\$459.63

-0.41 (-0.09%)

BUY

SELL

Home > Company Profile > Financial Ratios

All | Liquidity Measurement Ratios | Profitability Indicator Ratios | Debt Ratios | Operating Performance Ratios | Cash Flow Indicator Ratios | Investment Valuation Ratios

< Financial StatementsDupont >

Liquidity Measurement Ratios

Current Ratio	$\frac{Current\ Assets}{Current\ Liabilities}$	1.54	A current ratio of 1.0 or greater is an indication that the company is well-positioned to cover its current or short-term liabilities.
Quick Ratio	$\frac{Cash\ and\ Cash\ Equivalents + Short\ Term\ Investments + Account\ Receivables}{Current\ Liabilities}$	1.17	The quick ratio is more conservative than the current ratio because it excludes inventory and other current assets, which generally are more difficult to turn into cash. A higher quick ratio means a more liquid current position.
Cash Ratio	$\frac{Cash\ and\ Cash\ Equivalents}{Current\ Liabilities}$	0.46	The cash ratio is almost like an indicator of a firm's value under the worst-case scenario where the company is about to go out of business.
Days of Sales Outstanding	$\frac{(Account\ Receivable(start) + Account\ Receivable(end))/2}{Revenue/365}$	32.16	DSO tells you how many days after the sale it takes people to pay you on average.
Days of Inventory Outstanding	$\frac{(Inventories(start) + Inventories(end))/2}{COGS/365}$	9.26	DIO tells you how many days inventory sits on the shelf on average.
Operating Cycle	$DSO + DIO$	41.43	(DSO + DIO) Basically the Operating Cycle tells you how many days it takes for something to go from first being in inventory to receiving the cash after the sale.

5,6k46

R2D2 курит в сторонке: что могут инструменты для AI на смартфонах HUAWEI

Мераност

Редаторский дайджест

Присылаем лучшие статьи раз в месяц

Электрпочта

Эти же данные по годам можно получать и через API:

https://financialmodelingprep.co

financialmodelingprep.com/api/v3/ratios/AAPL?apikey=demo

JSON :

0

symbol : AAPL

date : 2019-09-28

currentRatio : 1.540125617208044

quickRatio : 1.168041393140241

cashRatio : 0.46202160464632325

daysOfSalesOutstanding : 32.16305241876591

daysOfInventoryOutstanding : 9.263638723714628

operatingCycle : 41.426691142480536

daysOfPayablesOutstanding : 104.31407696777144

cashConversionCycle : -72.15102454900554

grossProfitMargin : 0.3781776810903472

operatingProfitMargin : 0.24572017188496928

pretaxProfitMargin : 0.2526655238417367

netProfitMargin : 0.21238094505984456

effectiveTaxRate : 0.1594383680423506

returnOnAssets : 0.16323009842961633

returnOnEquity : 0.6106445053487756

returnOnCapitalEmployed : 0.28237785547985805

netIncomePerEBT : 0.8405616319576494

ebtPerEbit : 1

ebitPerRevenue : 0.2526655238417367

debtRatio : 0.7326921031797611

debtEquityRatio : 2.7410043320661304

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,

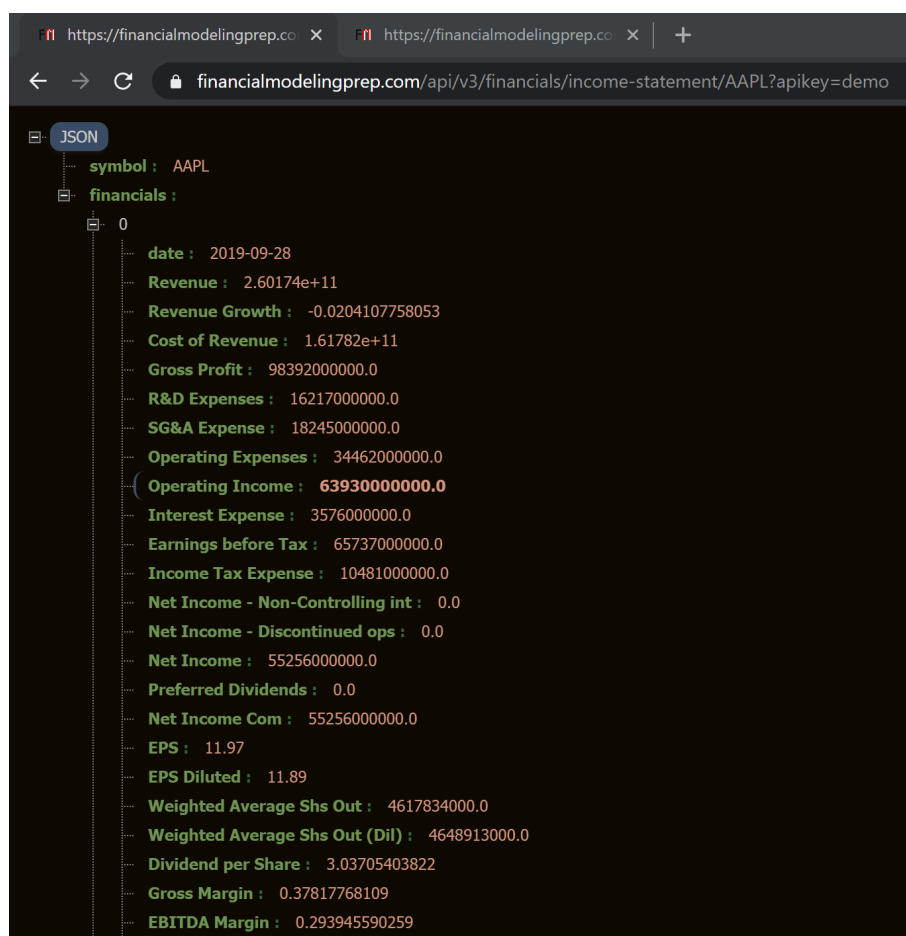
которые доступны за тот же период времени, что и финансовые отчеты компании:

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>

```

JSON :
{
  "date": "2019-09-28",
  "symbol": "AAPL",
  "fillingDate": "2019-10-31 00:00:00",
  "acceptedDate": "2019-10-30 18:12:36",
  "period": "FY",
  "revenue": 260174000000,
  "costOfRevenue": 161782000000,
  "grossProfit": 98392000000,
  "grossProfitRatio": 0.378178000000000014,
  "researchAndDevelopmentExpenses": 16217000000,
  "generalAndAdministrativeExpenses": 18245000000,
  "sellingAndMarketingExpenses": 0,
  "otherExpenses": 1807000000,
  "operatingExpenses": 34462000000,
  "costAndExpenses": 196244000000,
  "interestExpense": 3576000000,
  "depreciationAndAmortization": 12547000000,
  "ebitda": 81860000000,
  "ebitdaratio": 0.3146360000000000027,
  "operatingIncome": 63930000000,
  "operatingIncomeRatio": 0.245719999999999994,
  "totalOtherIncomeExpensesNet": 422000000,
  "incomeBeforeTax": 65737000000,
  "incomeBeforeTaxRatio": 0.25266600000000000002,
  "incomeTaxExpense": 10481000000
}

```

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 FinancialModelingPrepScreener() { // Stock Screener
    var sectorArray = ["Consumer Cyclical", "Energy", "Technology", "Indus
trials", "Financial Services", "Basic Materials", "Communication Services"
, "Consumer Defensive", "Healthcare", "Real Estate", "Utilities", "Industr
ial Goods", "Financial", "Services", "Conglomerates"]
    var symbolArray = []
    var log = ''
    for (var e = 0; e < sectorArray.length; e++) {
        const url = `https://financialmodelingprep.com/api/v3/stock-screen
er?sector=${sectorArray[e]}&apikey=${secrets.financialmodelingprep}`
        // console.log(`\n${getFunctionName()}. Ссылка на скринер в сектор
e ${sectorArray[e]}: ${url}.`)
        const response = await fetch(url)
        const json = await response.json()
        for (var j = 0; j < json.length; j++) {
            symbol = json[j].symbol
            companyName = json[j].companyName
            sector = json[j].sector
            exchange = json[j].exchange
            if (sector) {
                // console.log(`${getFunctionName()}. Компания № ${j+1} из
${json.length}: ${companyName} (${symbol}) из отрасли ${sector}, торгуется
на ${exchange}.`)
                // symbolArray.push([symbol, companyName, sector, exchang
e])
                symbolArray.push(symbol)
            }
        }
        console.log(`${getFunctionName()}. В секторе ${sectorArray[e]} (
${e+1} из ${sectorArray.length}) найдено ${json.length} компаний.`)
        log += `<li>В секторе ${sectorArray[e]} (${e+1} из ${sectorArray.l
ength}) найдено ${json.length} компаний.</li>`
    }
    return {
        symbolArray: symbolArray,
        log: log
    }
}

async function FinancialModelingAvailableYears() { // поиск статистики дос
тупных лет
    console.log(`Получаем список компаний через скринер financialmodelingp
rep.`)
    symbolArray = await FinancialModelingPrepScreener()
    symbolArray = symbolArray.symbolArray
    symbolArrayUnique = symbolArray.filter((v, i, a) => a.indexOf(v) ===
i)
    console.log(`\nИтого: найдено ${symbolArray.length} компаний. Без повто
ров: ${symbolArrayUnique.length} шт.`)

    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/${tic
```

```

ker}?apikey=${secrets.financialmodelingprep}`
// 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 ${new Date().toLocaleTimeString()}: ${Year} (${a+1}-й) год.
            })
            }
            console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) глубина ${json.length} лет.`)
        } else {
            notIncluded += 1
            console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) нет истории.`)
        }
    } catch (e) {
        console.log(`${ticker} (${s+1} из ${symbolArrayUnique.length}) пропускаем в ${new Date().toLocaleTimeString()} с ошибкой: "${e}".`)
    }
}
let avg = averageYears.reduce((a, v, i) => (a * i + v) / (i + 1))
console.log(`\nДля ${averageYears.length} акций средняя глубина отчетов: ${avg.toFixed(2)} лет. Однако в базе нет отчетов у ${notIncluded} бумаг.`)

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.

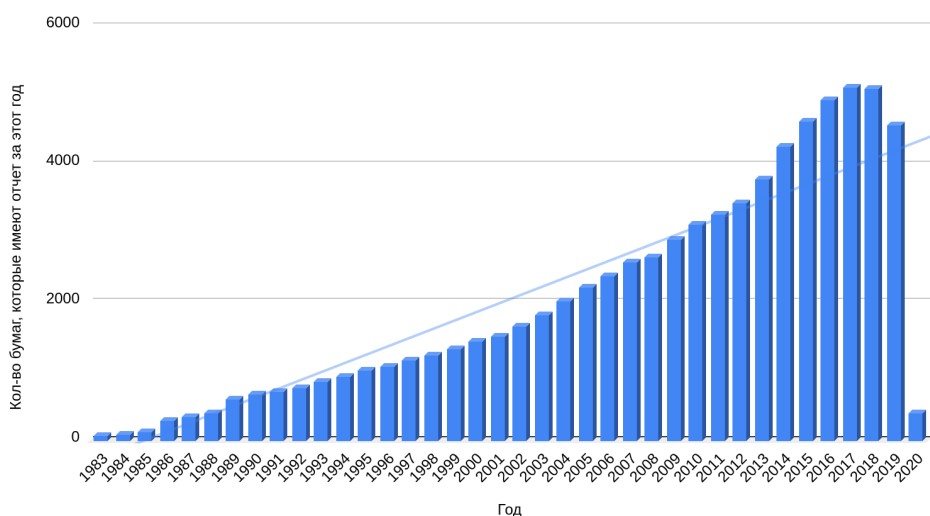
Итог: найдено 12627 компаний. Без повторов: 8422 шт.

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

► [Свёл итоги работы скрипта в разбивке отчетов по годам в таблицу, а ниже приведён график.](#)

Получившийся график:

Кол-во бумаг, которые имеют отчет за этот год относительно параметра "Год"



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 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.



**Теги:** [парсинг](#), [котировка](#), [биржа](#), [инвестиции](#), [статистика](#), [ценные бумаги](#), [облигации](#), [биржевая торговля](#), [JavaScript](#), [Node.JS](#)

**Хабы:** [Алгоритмы](#), [Node.JS](#), [API](#), [Финансы в IT](#)


↑ +4 ↓

38

2,8k

6

Поделиться



81,5

0,1

Карма

Рейтинг

Михаил Шардин

@empenoso

Разработчик

[Github](#)

ПОХОЖИЕ ПУБЛИКАЦИИ

- 27 ноября 2016 в 19:08

Настольный пульт управления на JavaScript/Node.js для работа на Ардуине

↑ +14

13k

94

14
- 24 февраля 2016 в 13:46

Тонкости Javascript/Node.js. Увеличиваем производительность в десятки раз

↑ +23

41,2k

310

38
- 14 декабря 2014 в 01:27

Выразительный JavaScript: Node.js


↑ +39

137k

499

8

МИНУТОЧКУ ВНИМАНИЯ



Опрос

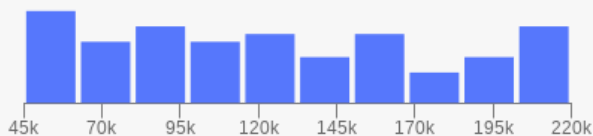
Опрос о модернизации российской банковской сферы



## СРЕДНЯЯ ЗАРПЛАТА В IT

**110 000** руб./мес.

Средняя зарплата по всем IT-специализациям на основании 9 345 анкет, за 2-ое пол. 2020 года

[Узнать свою зарплату](#)

Мегатест

Съешь ещё этих мягких булок, да тест на логику пройди

## Реклама



## Пульсоксиметр

Замеряет кислород в крови и пульс за 5 секунд. Заказывайте сегодня со скидкой 50%!

marketshoop.ru \*Есть противопоказания, проконсультируйтесь!



## Комментарии 6

fzn7 24 августа 2020 в 08:22 #

↑ +1 ↓

Quandl довольно удобен

empenoso 24 августа 2020 в 08:49 #

↑ +1 ↓

[docs.quandl.com](https://docs.quandl.com)

Спасибо!

Xazzzi 24 августа 2020 в 11:50 #

↑ +1 ↓

Или вот еще тоже неплохой сервис:

[www.alphavantage.co/documentation](https://www.alphavantage.co/documentation)

empenoso 24 августа 2020 в 13:15 #

↑ 0 ↓

Использовал его, он в прошлом полностью бесплатный был, если ничего не путаю.

artem\_larin 25 августа 2020 в 12:46 #

↑ +1 ↓

Скажите, какой финансовый API предоставляет данные, чтобы протестировать «магическую формулу» Гринблатта? Используется вот такой набор: ROA, P/E, тип компании (взаимные фонды, банки, страховые компании), признак «иностранная компания» (напр. суффикс ADR).

empenoso 25 августа 2020 в 13:28 #

↑ 0 ↓

Любой наверное. Можно этот платно или API Яху Финанс бесплатно, но с задержками сделать. Вот писал про него: [habr.com/ru/post/505674](https://habr.com/ru/post/505674)

От истории ещё зависит — сколько лет надо смотреть.

## ЧТО ОБСУЖДАЮТ

Сейчас

Вчера

Неделя

Абиогенез и алгоритмы естественного отбора

674

9

Мифический человеко-месяц 45 лет спустя

6,2k

66

PandaDoc просит освободить Виктора Кувшинова

11,2k

187

На Apple M1 заработали Firefox и Microsoft Office в нативном коде

5,1k

27

Где логика, где разум? Нескучный тест для mobile-разработчиков

Мегатест

Только [полноправные пользователи](#) могут оставлять комментарии. [Войдите](#), пожалуйста.

САМОЕ ЧИТАЕМОЕ

- Сутки
- Неделя
- Месяц

9 тяжелых уроков, которые я усвоил за 18 лет разработки

+36 21,3k 134 43

Библиотека от AMD стала причиной плохой производительности процессоров AMD в Cyberpunk 2077

+56 59,4k 28 63

До свидания, Google Fonts. Последний аргумент

+67 21,3k 144 47

Темно-серая зона экосистемы Telegram

+7 12,5k 23 56

Как мы пофиксили Apple security-приложением

Меропост

Ваш аккаунт

[Войти](#)  
[Регистрация](#)

Разделы

[Публикации](#)  
[Новости](#)  
[Хабы](#)  
[Компании](#)  
[Пользователи](#)  
[Песочница](#)

Информация

[Устройство сайта](#)  
[Для авторов](#)  
[Для компаний](#)  
[Документы](#)  
[Соглашение](#)  
[Конфиденциальность](#)

Услуги

[Реклама](#)  
[Тарифы](#)  
[Контент](#)  
[Семинары](#)  
[Мегапроекты](#)  
[Мерч](#)