



Все важнейшие
IT-события здесь

5

6

7

8

9

10

11

12

13

14

15

16

17

18

**empenoso**

24 авг 2020 в 04:25

Поиски фундаментальных данных для акций через API Financial Modeling Prep

🕒 127 мин 👁 10K

API*, Node.JS*, Алгоритмы*, 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. \(AAPL\)](#):

financialmodelingprep.com/financial-ratios/AAPL

Financial Ratios

Apple Inc. (AAPL) \$459.63 -0.41 (-0.09%) BUY SELL

Home > Company Profile > Financial Ratios

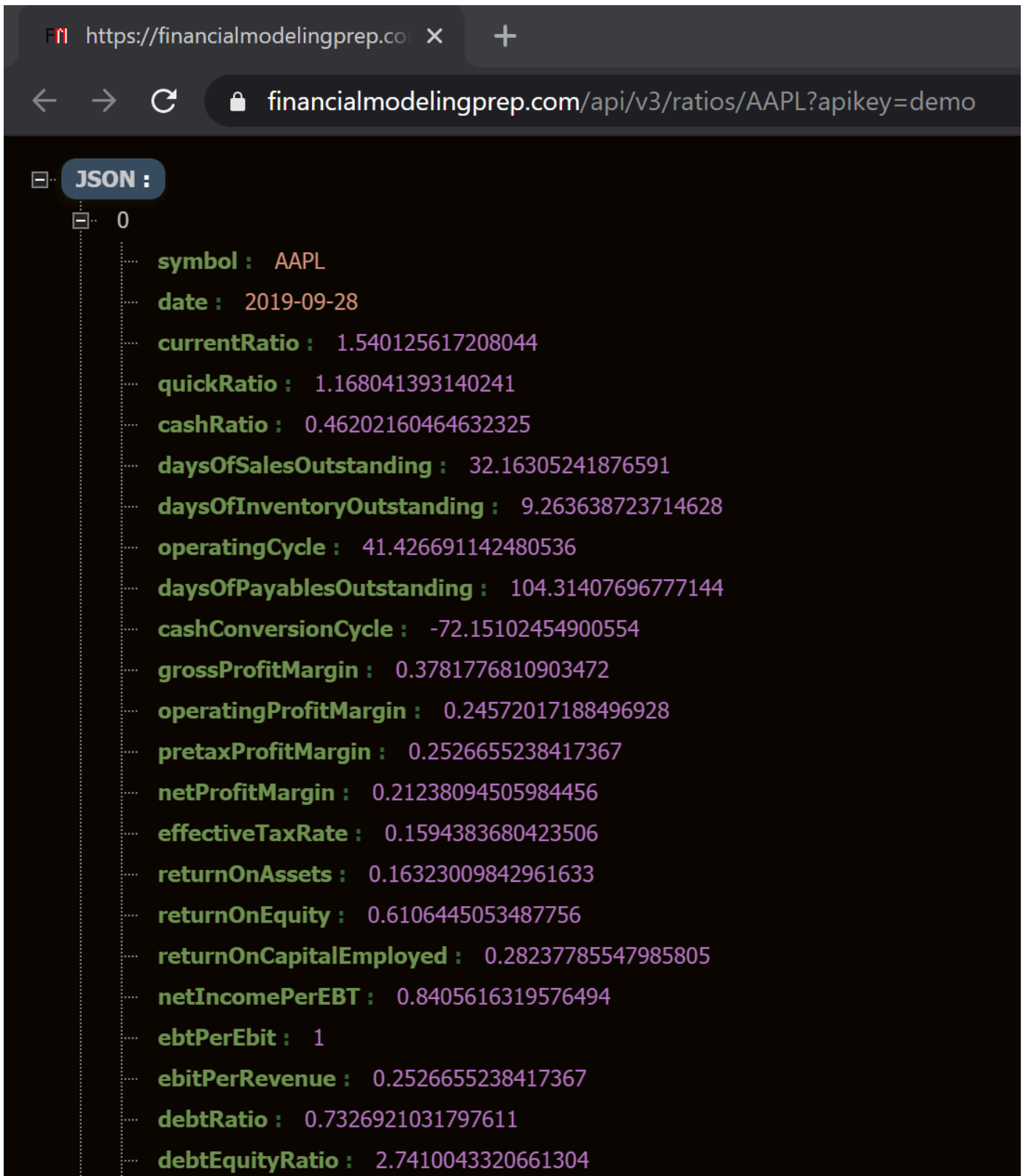
All | Liquidity Measurement Ratios | Profitability Indicator Ratios | Debt Ratios | Operating Performance Ratios | Cash Flow Indicator Ratios | Investment Valuation Ratios

Financial Statements Dupont >

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.

Эти же данные по годам можно получать и [через API](#):



```
https://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>

```

{
  "symbol": "AAPL",
  "financials": [
    {
      "date": "2019-09-28",
      "Revenue": 2.60174e+11,
      "Revenue Growth": -0.0204107758053,
      "Cost of Revenue": 1.61782e+11,
      "Gross Profit": 98392000000.0,
      "R&D Expenses": 16217000000.0,
      "SG&A Expense": 18245000000.0,
      "Operating Expenses": 34462000000.0,
      "Operating Income": 63930000000.0,
      "Interest Expense": 3576000000.0,
      "Earnings before Tax": 65737000000.0,
      "Income Tax Expense": 10481000000.0,
      "Net Income - Non-Controlling int": 0.0,
      "Net Income - Discontinued ops": 0.0,
      "Net Income": 55256000000.0,
      "Preferred Dividends": 0.0,
      "Net Income Com": 55256000000.0,
      "EPS": 11.97,
      "EPS Diluted": 11.89,
      "Weighted Average Shs Out": 4617834000.0,
      "Weighted Average Shs Out (Dil)": 4648913000.0,
      "Dividend per Share": 3.03705403822,
      "Gross Margin": 0.37817768109,
      "EBITDA Margin": 0.293945590259
    }
  ]
}

```

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": 0.2526660000000000002,
  "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", "Industrials", "Fir
  var symbolArray = []
  var log = ''
  for (var e = 0; e < sectorArray.length; e++) {
    const url = `https://financialmodelingprep.com/api/v3/stock-screener?sector=${s
    // console.log(`\n${getFunctionName()}. Ссылка на скринер в секторе ${sectorArr
    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 ${sectorArray.length}) найдено ${log} компаний.

```

```

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

    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/${ticker}?apikey=${API_KEY}`
        // 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 ${Year} году`)
                }
                console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) глубина отчета: ${averageYears[s]}`)
            } else {
                notIncluded++
            }
        } catch (error) {
            console.log(`Ошибка при получении данных для ${ticker}: ${error}`)
        }
    }
}

```

```
        notIncluded += 1
        console.log(`Для ${ticker} (${s+1} из ${symbolArrayUnique.length}) нет
    }
} catch (e) {
    console.log(`${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(2)}

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.

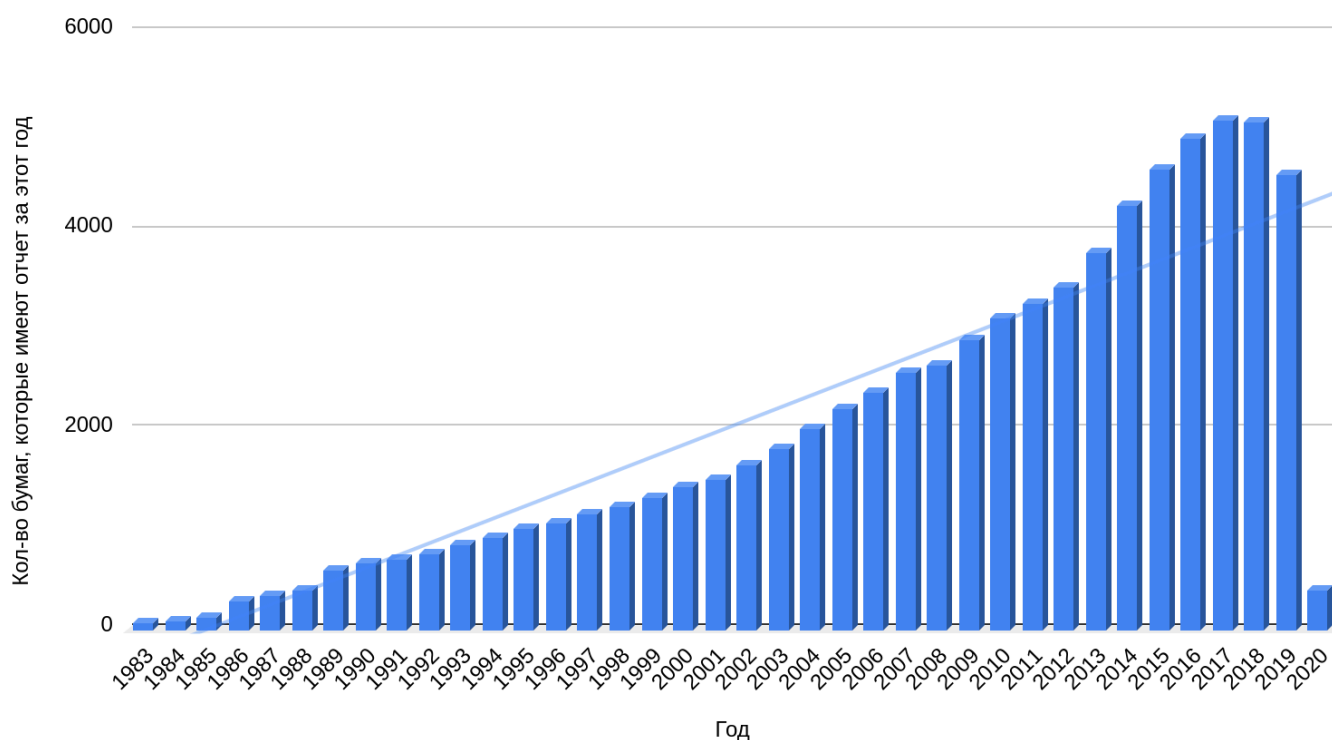
Итого: найдено 12627 компаний. Без повторов: 8422 шт.

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:

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 несоответствий: 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

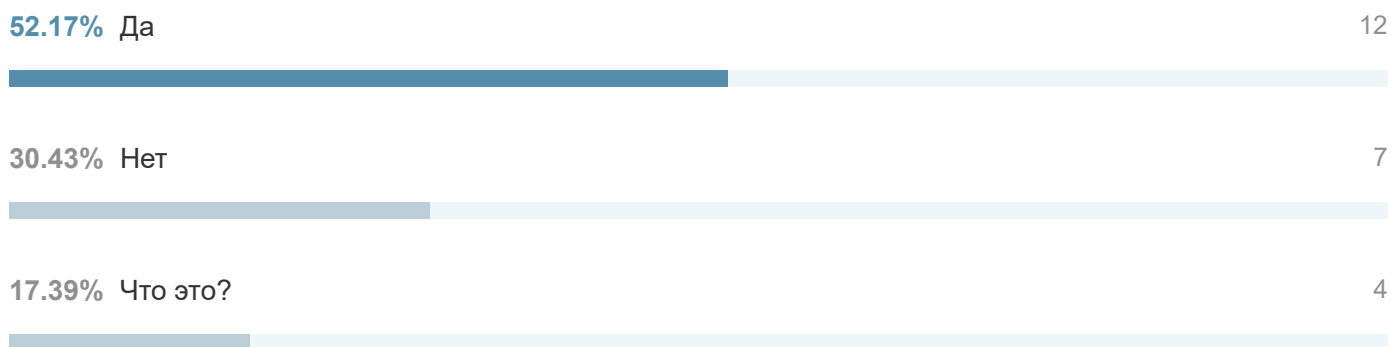
Если вы готовы платить, то Financial Modeling Prep предоставляет множество данных, через удобный, но слегка запутанный интерфейс.

Автор: [Михаил Шардин](#),

24 августа 2020 г.

Только зарегистрированные пользователи могут участвовать в опросе. [Войдите](#), пожалуйста.

Используете финансовые API?



Проголосовали 23 пользователя. Воздержались 8 пользователей.

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

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

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

**179**

Карма

30.4

Рейтинг

Михаил Шардин @empenoso

Разработчик

[Подписаться](#)

[Сайт](#) [Сайт](#) [Github](#)[Комментарии 6](#)

Публикации

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

5 часов назад

Кто поджёт Лос-Анджелес? Свежая конспирология о виноватых НЛО, Пи Дидди, урбанистах и корюшке

**Простой**

14 мин



3.7K

[Обзор](#)

+25

4

34

**DimDimDimDimDim**

6 часов назад

Rust 1.84: новый релиз отличного языка программирования. Еще лучше, еще эффективнее, как всегда



6 мин



2.5K

+17

8

4

**JBFW**

14 часов назад

Подключаем длинную линию 1-wire к Ардуино



3 мин



4.3K

+17




32

28




**arturdumchev**


1 час назад

Заговор разработчиков против корпораций




 **Средний**  15 мин  1.6K

[Мнение](#)




 +11  6  4


 **DENEVGSTAR**
5 часов назад

Распознавание образов в мозге с помощью микроплееров




 **Средний**  8 мин  1K

[Из песочницы](#)




 +11  16  6


 **chlorine**
7 часов назад

Кэш. Теория кэширования. Устройство и разновидности кэша



 **Простой**  7 мин  2K

[Из песочницы](#)




 +11  68  16


 **burenkov**
3 часа назад

Стереокамера машинного зрения с поддержкой ИИ на базе FPGA и Arduino Portenta H7




 10 мин  782

[Из песочницы](#)

 +10  14  0

 **mikhailmurzak**
21 час назад

Делаем Телеграм-бота в Cursor AI без знания кода

 **Простой**  5 мин  6.6K

[Тutorial](#)

 +10 76 15

DAN_SEA

1 час назад

«Профессор, конечно, лопух, но аппаратура при нём» — или немного о костной проводимости



Средний



9 мин



625

Обзор

 +7 1 3

andreybold

7 часов назад

Как я развлекался с восходом солнца



Простой



6 мин



1.5K

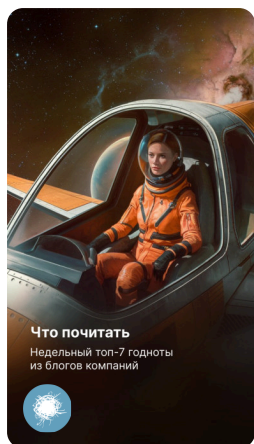
 +7 5 30

Как закрыть ноут в 18:00 и встать со стула — комикс о переработках

Промо

Показать еще

ИСТОРИИ



Годнота из блогов компаний



Выравнивания планет



Нейрозима 2025



Статьи с новогодним вайбом



Кто выступит на конференции мечты

ВАКАНСИИ

Бэкэнд-разработчик JavaScript

от 250 000 до 400 000 ₽ · Wanted. · Москва

JavaScript FullStack developer

до 220 000 ₽ · Wanted. · Санкт-Петербург

Middle+ NodeJS backend developer

от 2 000 до 4 500 \$ · DataLouna · Можно удаленно

Backend Node.JS разработчик (Nest.JS) One Day Offer

до 40 000 ₽ · Ревамп АйТи · Можно удаленно

Middle fullstack developer (NodeJS)

до 180 000 ₽ · Тетрика · Москва · Можно удаленно

[Больше вакансий на Хабр Карьере](#)

МИНУТОЧКУ ВНИМАНИЯ



Как хабравчане следят за здоровьем?



Испытание огнем и ловушками: спасти мир с помощью лассо



Автоматизируем процессы на базе event driven architecture

РАБОТА

[Node.js разработчик](#)

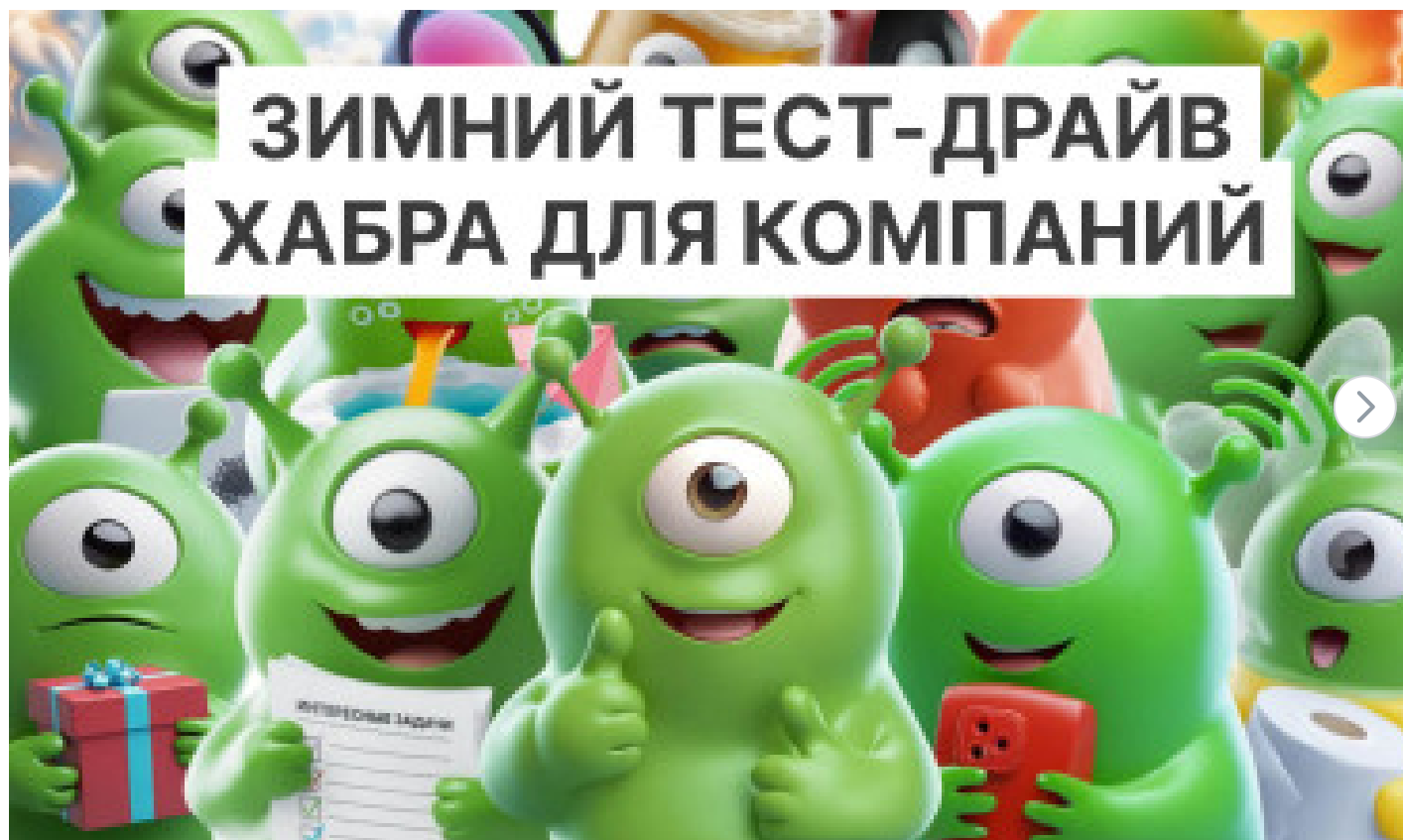
33 вакансии

[JavaScript разработчик](#)

98 вакансий

[Все вакансии](#)

БЛИЖАЙШИЕ СОБЫТИЯ



30 января

Зимний тест-драйв Хабра для компаний

Москва

Маркетинг

Другое

[Больше событий в календаре](#)

Хабр



🌐 [Настройка языка](#)

[Техническая поддержка](#)

© 2006–2025, Habr