

API

Программный интерфейс означает совокупность средств, методов и правил взаимодействия с приложением или сервисом. Происходит это понятие от перевода значения английской аббревиатуры **API** — *application programming interface* — программный интерфейс приложений.

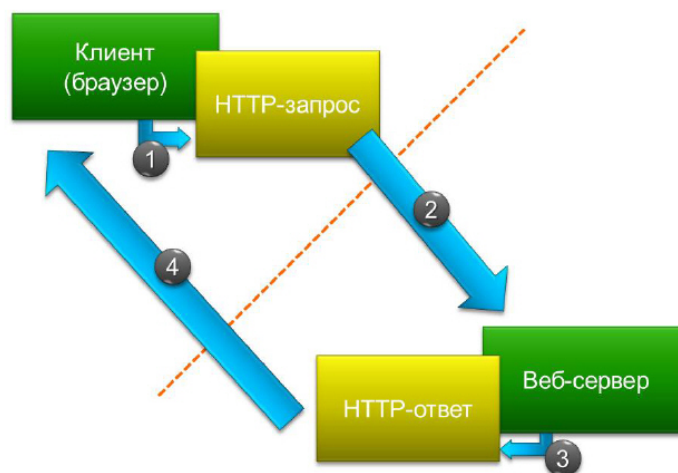
Возможно, вы уже встречались и с понятием «тестирование API». Если с пользовательским или даже консольным интерфейсом все более-менее понятно, то у новичка может возникнуть вопрос, а где я могу увидеть эти самые API? И как я смогу их протестировать?

Прежде чем мы начнем работать непосредственно с API, давайте вспомним, что известно об HTTP-протоколе

Знание его структуры будет необходимо, так как нам предстоит научиться понимать HTTP-запросы и ответы на них, а также формировать запросы с использованием специального инструмента — программного обеспечения *Postman*.

Обмен сообщениями по протоколу HTTP идёт от клиента к серверу по схеме запрос-ответ: клиент (1) отправляет запрос серверу (2), сервер обрабатывает запрос (3) и отправляет ответ клиенту (4). Основной объект, к которому происходит обращение по HTTP-протоколу, — это **ресурс**.

Ресурсом может быть что угодно: файл, логический объект или нечто абстрактное. Ресурс, к которому производится обращение по HTTP-протоколу, идентифицируется по *URI*, передаваемом в стартовой строке запроса. Операция, которую сервер выполняет по запросу клиента, идентифицируется методом.



Структура запроса HTTP-протокола

Запрос состоит из:

1. Стартовой строки.
2. Заголовков запросов.
3. Тела сообщения, передаваемого вместе с запросом.

[Глава 4 стандарта RFC7231](#) определяет следующие **основные методы HTTP-запроса**:

1. *GET*
2. *POST*
3. *PUT*
4. *DELETE*
5. *HEAD*
6. *OPTIONS*
7. *CONNECT*
8. *TRACE*

Также в этом стандарте определяются форматы *URI*, передаваемые в стартовой строке запроса.

Заголовки запроса характеризуют тело сообщения, параметры передачи и содержат прочие сведения. Заголовки представлены парой поле-значение, разделенных двоеточием.

Например: `Host: 130.193.37.179`

Заголовок *Host* является обязательным и должен присутствовать в каждом запросе, так как сервер изначально не обладает никакой информацией о том, какой именно адрес необходимо использовать для соединения.

Тело сообщения, передаваемое в запросе, — объект, с которым ассоциирован данный запрос. Если стартовая строка и заголовки всегда присутствуют в запросе, то тело сообщения может отсутствовать. Например, оно, как правило, отсутствует в *GET*-запросах.

Структура ответа HTTP-протокола

Ответ *HTTP*-протокола также состоит из стартовой строки, заголовков и тела сообщения, передаваемого вместе с ответом.

Стартовая строка ответа состоит из:

- версии протокола;
- кода состояния;
- его расшифровки.

Например, стартовая строка ответа может быть:

```
HTTP/1.1 200 OK
```

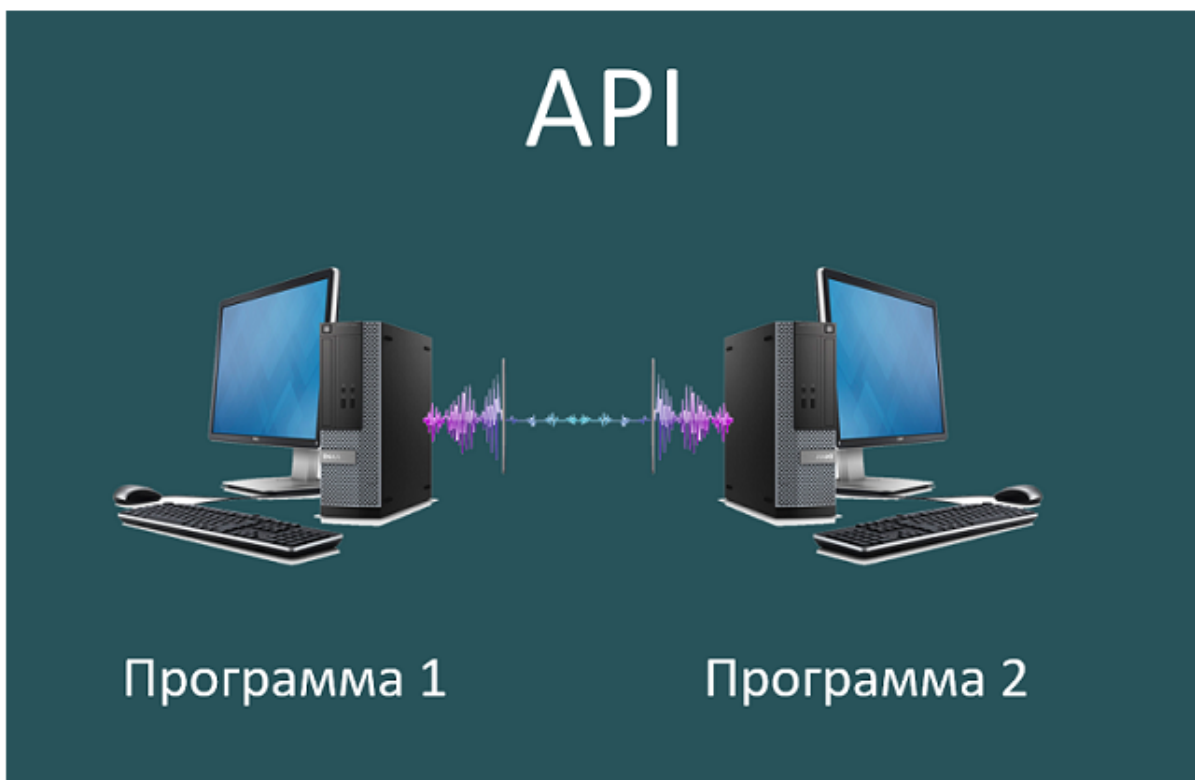
Согласно стандарту *RFC7231*, **код состояния** — трехзначный цифровой код, представляющий результат выполнения запроса. Коды состояния могут попадать в одну из пяти категорий определенного формата:

- **1xx** — информационные;
- **2xx** — коды, относящиеся к категории успешных, означающие успешное выполнение запроса;
- **3xx** — коды, относящиеся к категории перенаправления, означающие, что клиент должен сделать повторный запрос по другому URI, для успешного выполнения операции;
- **4xx** — коды, обозначающие, что была ошибка на стороне клиента;
- **5xx** — коды, обозначающие, что была ошибка на стороне сервера.

Заголовки ответа характеризуют тело полученного сообщения, параметры передачи и содержат прочие сведения. По формату они аналогичны заголовкам запроса.

Тело ответа содержит данные, связанные с запрошенным ресурсом, которые сервер отправляет на запрос клиенту. Например, ответ на пример запроса, приведенного выше, содержит сообщение в формате *HTML*.

Прочитать про заголовки *HTTP*-протокола можно в [главах 5 и 7 RFC7231](#), про коды ответа в главе 6 этого же стандарта. Тем, кто не любит сухой язык стандартов, можно прочитать про *HTTP*-протокол в документации, приведенной на [сайте разработчиков](#) браузера *Mozilla Firefox*.



Что значит «Тестирование API»

В первую очередь, мы подразумеваем тестирование ЧЕРЕЗ API. «Тестирование API» — общеупотребимый термин, так действительно говорят, но технически термин некорректен. Мы не тестируем API, мы не тестируем GUI (графический интерфейс). Мы тестируем какую-то функциональность через графический или программный интерфейс.

Но это устоявшееся выражение. Можно использовать его и говорить “тестирование API”. И когда мы про это говорим, мы имеем в виду:

- автотесты на уровне API
- или интеграцию между двумя разными системами.

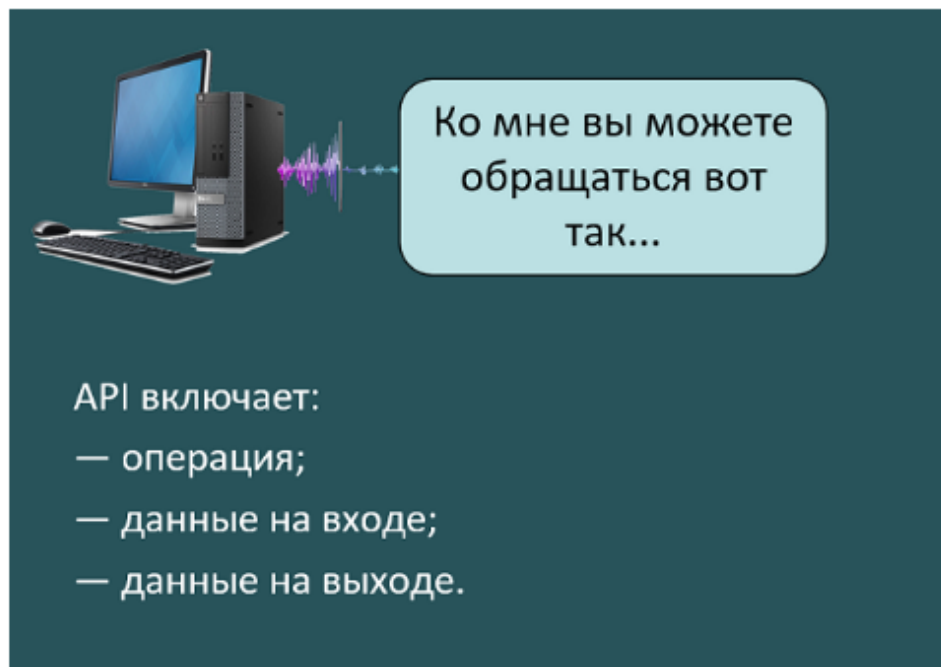
Интеграция — когда одна система общается с другой по какому-то протоколу передачи данных. Это называется Remote API, то есть общение по сети, по некоему протоколу (HTTP, JMS и т.д.). В противовес ему есть еще Local API (он же «Shared memory API») — это то API, по которому программа общается сама с собой или общается с другой программой внутри одной виртуальной памяти.

API — набор функций

Когда вы покупаете машину, вы составляете договор, в котором прописываете все важные для вас пункты. Точно также и между программами должны составляться договоры. Они указывают, как к той или иной программе можно обращаться.

Соответственно, API отвечает на вопрос "Как ко мне, к моей системе можно обратиться?", и включает в себя:

- саму операцию, которую мы можем выполнить,
- данные, которые поступают на вход,
- данные, которые оказываются на выходе (контент данных или сообщение об ошибке).



API (Application programming interface) — это контракт, который предоставляет программа. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».

Контракт включает в себя:

- саму операцию, которую мы можем выполнить,
- данные, которые поступают на вход,
- данные, которые оказываются на выходе (контент данных или сообщение об ошибке).

Вызвать API можно как напрямую, так и косвенно:

1. Система вызывает функции внутри себя
2. Система вызывает метод другой системы
3. Человек вызывает метод
4. Автотесты дергают методы
5. Пользователь работает с GUI

Когда говорят про API с тестировщиком, обсуждают два варианта:

- автотесты на уровне API (умение автоматизировать)
- интеграцию между двумя разными системами (обычно SOAP или REST, то есть работу в SOAP Ui или Postman).

10 лучших инструментов для тестирования API

Тестирование IT-систем*, API*






10 лучших инструментальных средств тестирования интерфейсов прикладного программирования 2018 года.

Интерес к тестированию неудержимо растёт на протяжении нескольких последних лет, согласно исследованиям [Google Trends](#). Опрос, проведенный компанией Smartbear в 2017 году среди 5000 профессионалов в области разработки программного обеспечения, показал, что более 50% опрошенных респондентов используют автоматические средства тестирования API, и ожидается рост их количества на 30% (с 59% до 77%) в течении следующих двух лет, причем 80% участников опроса указали, что отвечают за тестирование API.

Наличие правильных процессов, инструментов и технических решений для автоматических тестирований API становится важным, как никогда ранее. И с помощью тенденции shift-left, тестирование API становится больше, чем просто решение по контролю за качеством, теперь это критически важный компонент успешной непрерывной интеграции и развёртывания программного обеспечения.

В данной статье предоставлен обзор лучших средств тестирования API, как с открытым доступом, так и коммерческих решений, из которых команды тестировщиков могут выбрать наиболее подходящие для себя.

подходящие для себя.

Product	 POSTMAN	 JMeter™	 Katalon	 TRICENTIS	 apigee
Available since	2012	1988	2015	2007	2004
Application Under Test	API	API	Web (UI & API) Mobile apps	Web (UI & API) Mobile apps	API
Pricing	Paid + Free	Open Source	Free	Paid + Free	Paid + Free
Supported Platform	Windows Linux MacOS	Windows Linux MacOS	Windows Linux MacOS	Windows	Windows Linux MacOS
Ease of installing and use	Easy to setup & use	Advanced programming skills needed to setup & use	Easy to setup & use	Easy to setup. Need training to properly use the tool	Require end-points management knowledge to use

1. SoapUI

SoapUI представляет собой консольный инструмент, предназначенный для тестирования API и позволяющий пользователям легко тестировать API REST и SOAP, а также Web-сервисы.

При помощи SoapUI, пользователи могут получить полный исходный документ и встроить предпочтительный набор функций, в дополнение к перечисленным ниже:

- Создать тест легко и быстро при помощи технологий перетаскивания объектов «мышкой» (Drag and drop) и метода «указания и щелчка» (Point-and-click)
- Быстро создать пользовательский код при помощи Groovy
- Мощное тестирование на основе данных: Данные загружаются из файлов, баз данных и Excel, поэтому они могут создать симуляцию взаимодействия пользователя и API.
- Создание комплексных сценариев и поддержка асинхронного тестирования
- Повторное использование скриптов: загрузка тестов и сканирование безопасности могут повторно использоваться в случае функционального тестирования всего в несколько шагов

2. Postman

Будучи изначально плагином браузера Chrome, теперь Postman расширяет свои технические решения вместе с оригинальными версиями как для Mac, так и для Windows.

Postman является отличным выбором API тестирования для тех, кто не желает иметь дела с кодировками в интегрированной среде разработки, используя тот же язык программирования, что и разработчик.

- Легкий в использовании клиент REST
- Богатый интерфейс делает этот инструмент простым и удобным в использовании
- Может использоваться как при автоматическом, так и при исследовательском тестировании
- Работает в приложениях для Mac, Windows, Linux и Chrome
- Обладает пакетом средств интеграции, таких как поддержка форматов Swagger и RAML
- Обладает функциями Run, Test, Document и Monitoring
- Не требует изучения нового языка программирования
- Позволяет пользователям легко делиться опытом и знаниями с другими членами команды, поскольку позволяет упаковывать все запросы и ожидаемые ответы и отправлять их коллегам.

3. Katalon Studio

Katalon Studio является бесплатным инструментом автоматического тестирования, предоставляющим общую среду для создания и выполнения UI функционала, служб API/Web и тестирования мобильных платформ.

Способность комбинировать уровни UI и Business (службы API/Web) для различных операционных сред (Windows, Mac OS, Linux) расценивается как значительное преимущество Katalon Studio перед аналогичными продуктами.

Katalon Studio поддерживает запросы SOAP и RESTful с различными типами команд (GET, POST, PUT, DELETE) с параметризованными возможностями.

Основные свойства:

- Поддержка теста комбинации между верификациями UI и API.
- Поддержка тестирования запросов как SOAP, так и RESTful.
- Сотни встроенных ключей для создания тестовых заданий.
- Поддержка одной из самых мощных библиотеки проверки утверждений AssertJ для создания динамических утверждений в BDD-стиле.
- Поддержка подхода, управляемого данными.
- Может использоваться как при автоматическом, так и при исследовательском тестировании.
- Отлично подходит как для профессионалов, так и для новичков

4. Tricentis Tosca

Tricentis Tosca представляет собой платформу непрерывного тестирования для Agile и DevOps. Среди преимуществ Tricentis Tosca следует отметить:

- Поддержку многих массивов протоколов: HTTP(s) JMS, AMQP, Rabbit MQ, TIBCO EMS, SOAP, REST, IBM MQ, NET TCP
- Интеграцию в циклы Agile и DevOps
- Максимизацию многократного использования и способность к сопровождению средств автоматизации тестирования на основе использования моделей
- Тесты API могут использоваться как на мобильных, так на браузерных и пакетных приложениях и т.д.
- Достигнута автоматизация, поддерживаемая новыми технологиями
- Снижено время, необходимое на проведение регрессивного тестирования

5. Apigee

Apigee является кросс-«облачным» средством тестирования API, позволяющим пользователям измерять и тестировать производительность API, обеспечивать техническую поддержку и разработку API при помощи других редакторов, таких как Swagger.

- Данный инструмент является многошаговым и находится под управлением Javascript
- Он позволяет разрабатывать, отслеживать, выполнять разворачивание и масштабирование API
- Идентифицирует проблемы путем отслеживания трафика API, уровня ошибок и времени ответа
- Легко создает прокси API из Open API Specification и выполняет их развертку в «облаке»
- Модели разворачивания в «облаке», локального разворачивания и гибридного разворачивания работают на основе одного кода
- PCI, HIPAA, SOC2, и PII для приложений и API
- Apigee разработан специально для цифрового бизнеса и задач с интенсивной обработкой данных на под управлением мобильных платформ API и приложений, которые управляют ним.

6. JMeter

JMeter (открытое программное обеспечение) широко используется для функционального тестирования API, однако изначально он создавался для нагрузочного тестирования.

- Поддерживает воспроизведение результатов тестирования
- Автоматически работает с файлами CSV, позволяя команде быстро создавать уникальные значения параметров для тестирования API.
- Благодаря интеграции между JMeter и Jenkins, пользователи могут включать тесты API в конвейерные обработки CI
- Данный инструмент может использоваться как для статического, так и динамического тестирования производительности ресурсов

7. Rest-Assured

Rest-Assured является общедоступным предметно-ориентированным Java-языком, который делает службу тестирования REST более простой и удобной.

- Обладает целым набором встроенных функционалов, наличие которых означает, что пользователю не придется кодировать заново.
- Надежно интегрирован со автоматизированной платформой Serenity, что позволяет пользователям комбинировать тесты UI и REST в рамках одной платформы и получать изумительные результаты.
- Поддерживает синтаксические конструкции BDD Given/When/Then
- Пользователям необязательно быть экспертами в области HTTP

8. Assertible

Assertible представляет собой инструмент тестирования API, который в первую очередь акцентируется на автоматизации процессов и надежности.

- Обеспечивает автоматическое тестирование API на каждом этапе процесса интеграции и поставки программного обеспечения.
- Осуществляет поддержку текущих тестов API после внедрения приложений и интегрирует их с привычными инструментами, такими как GitHub, Slack, и Zapier.
- Поддерживает проверку подлинности ответов HTTP при помощи готовых операторов подтверждения отсутствия ошибок, таких как проверка достоверности JSON Schema и проверка целостности данных JSON Path

9. Karate DSL

Karate DSL это новый инструмент для тестирования API, который помогает разрабатывать сценарии для BDD тестов на основе API простым способом, без написания характеристик этапов. Эти характеристики создаются самим KarateDSL, а поэтому пользователи могут запустить тестирование API легко и быстро.

- Встроен на вершине Cucumber-JVM
- Способен запускать тестирование и генерировать отчеты как любой другой стандартный проект Java
- Тест может быть разработан без обязательного владения знаниями языка Java
- Тесты могут легко создаваться даже непрограммистами
- Поддерживает конфигурацию продвижения /переключения, а также многопоточковое параллельное выполнение кода

10. Ни одно решение не может совместить все инструменты
Это больно осознавать, однако, это правда!

Мы верим, что приведенный выше список представляет наилучшие решения, доступные на текущий момент, если Вы решили использовать автоматическое тестирование API. Однако, как и в случае с большинством продуктов в данной сфере, найти один инструмент, который совмещает все указанные функции, практически невозможно.

Некоторые могут посчитать, что свойств коммерческих продуктов (Postman, Tricentis Tosca,...) будет достаточно, однако цена вопроса будет служить серьезным сдерживающим фактором. Бесплатные и общедоступные решения (Rest-Assured, Karate DSL,...) являются довольно-таки приемлемыми, но требуют квалифицированных умений и много усилий для имплементации правильной платформы. Инструменты, которые удерживают баланс между ценой и другими факторами (Katalon Studio, Postman), могут иметь недостатки для некоторых типов проектов, и эти недостатки требуют пристальной оценки.

