Чит-лист для тестирования API	
Author	**Alex Meshkov**
YouTube	https://www.youtube.com/@AlexMeshkovQA
Telegram	https://t.me/QAtestgrow
Инструменты тестирования API	
Katalon Studio	Postman
SoapUI	Rest-Assured
CITRUS	Karate
ReadyAPI	Airborne
Jmeter	apigee
API Терминология	
API	**Application Programming Interface (API)** описание способов взаимодействия одной компьютерной программы с другими.
HTTP	**Hypertext Transfer Protocol** это набор правил для передачи данных в глобальной сети Интернет, например, для передачи текста, картинок, звука, видео и других мультимедийных файлов.
HTTPS	**«S» в HTTPS** означает «Защищенность (secure)». Является расширением протокола HTTP для поддержки шифрования в целях повышения безопасности передачи данных.
URI	**Uniform Resource Identifier** - унифицированный (единообразный) идентификатор ресурса. Это символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д.
URL	**Uniform Resource Locator** - Унифицированный указатель ресурса. Представляет собой систему унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса (файла). Используется как стандарт записи ссылок на объекты Гипертекстовые в Интернете (ссылки во «всемирной паутине» www).
Уровни организации приложения	
Три уровня	Уровень презентации (или пользовательский интерфейс), уровень логики, и уровень баз данных для хранения и работы с данными приложения.
Веб сервисы	
SOAP	**Simple Object Access Protocol** - это протокол, определенный стандартами W3C, для отправки и получения запросов и ответов между веб сервисами. Протокол использует формат XML для обмена произвольными сообщениями.
REST	**Representational State Transfer** – это архитектурный стиль (набор правил) для взаимодействия приложения в сети. Другими словами, REST – это набор правил, которые использует программист для корректного написания кода серверной части приложения, чтобы все системы могли легко обмениваться данными. Может использовать любой формат передачи данных.
Формат передачи данных	
JSON	**JSON (JavaScript Object Notation)** – это формат передачи данных. Синтаксис JSON происходит от синтаксиса записи объектов JavaScript: - Данные записываются в виде пар «имя/значение»; - Данные разделяются запятыми; - В фигурных скобках {} записываются объекты; - В квадратных скобках [] записываются массивы.
XML	XML, в переводе с англ **eXtensible Markup Language** — расширяемый язык разметки. Используется для хранения и передачи данных. У XML-файлов древовидная структура. Это значит, что в них используется набор тегов, внутри которых могут находиться другие теги со своими значениями. Самый верхнеуровневый узел называется корнем, а все, что находится внизу, — листьями. Теги заключены в скобки <>.
YAML	YAML — это язык для сериализации данных, который отличается простым синтаксисом и позволяет хранить сложноорганизованные данные в компактном и читаемом формате.

Клиент, сервер, хост	
Client	Клиент - это компьютер, запрашивающий некоторую функцию или данные у сервера. **Сервер чаще всего (но не всегда) расположен на отдельных физических мощностях.**
Server	Сервер - это компьютер, на котором хранятся данные, или который выполняет определенные служебные функции для других компьютеров сети.
Host	Хост — любое устройство, предоставляющее сервисы формата «клиент-сервер» в режиме сервера по каким-либо интерфейсам и уникально определённое на этих интерфейсах. В более широком смысле под хостом могут понимать любой компьютер, подключённый к локальной или глобальной сети.
Основные коды ответов для HTTP	
Код	*Описание*
1xx	**Информационные ответы, означает, что сервер получил и понял запрос и что браузеру следует немного подождать, пока сервер обработает информацию.**
100	Продолжить (Continue): пока все в порядке, клиент должен продолжить выполнение запроса или проигнорировать его, если он уже завершен.
101	Протоколы переключения (Switching Protocols): протокол, на который сервер переключается по запросу клиента, который отправил сообщение, включая заголовок запроса на обновление.
102	Обработка (Processing): сервер принял полный запрос, но все еще обрабатывает его.
103	Ранние подсказки (Early Hints): разрешение пользовательскому агенту начать предварительную загрузку ресурсов, пока сервер все еще готовит ответ
2xx	**Успешные запросы. Означает, что запрос на доступ к ресурсу был успешным.**
200	OK: Успешный запрос.
201	Создан (Created): сервер подтвердил создание ресурса.
202	Принят (Accepted): запрос клиента получен, но сервер все еще обрабатывает его.
203	Неавторизованная информация (Non-Authoritative Information): ответ, который сервер отправил клиенту, отличается от того, который был, когда сервер отправил его.
204	Нет содержимого (No Content): сервер обработал запрос, но не предоставил никакого содержимого.
205	Сбросить содержимое (Reset Content): клиент должен обновить образец документа.
206	Частичное содержимое (Partial Content): сервер отправляет только часть ресурса.
207	Мульти-статус (Multi-Status): тело следующего сообщения по умолчанию является XML-сообщением и может содержать несколько отдельных кодов ответа.
3xx	**Перенаправление, для завершения запроса необходимо перенаправление на URL, отличный от исходного**
300	Множественный выбор (Multiple Choices): запрос, сделанный клиентом, имеет несколько возможных ответов.
301	Перемещено навсегда (Moved Permanently): сервер сообщает клиенту, что ресурс, который он ищет, был навсегда перемещен на другой URL-адрес.
302	Временно перемещен (Moved Temporarily): веб-сайт или страница были временно перемещены по другому URL-адресу.
303	См. Другое (See Other): этот код сообщает клиенту, что сервер перенаправляет их не на запрошенный ресурс, а на другую страницу.
305	Использовать прокси (Use Proxy): клиент может получить доступ к запрошенному ресурсу только через прокси, указанный в ответе.
307	Временное перенаправление (Temporary Redirect): сервер сообщает клиенту, что ресурс, который он ищет, был временно перенаправлен на другой URL-адрес.
308	Постоянное перенаправление (Permanent Redirect): сервер сообщает клиенту, что

	искомый ресурс был временно перенаправлен на другой URL-адрес.
4xx	**Ошибка клиента, запрос содержит неправильный синтаксис или не может быть выполнен.**
400	Плохой запрос(Bad Request): клиент отправляет запрос с неполными данными, плохо построенными данными или недопустимыми данными.
401	Неавторизованный (Unauthorized): для доступа клиента к запрошенному ресурсу требуется авторизация.
403	Запрещено (Forbidden): ресурс, к которому клиент пытается получить доступ, запрещен.
404	Не найден (Not Found): сервер доступен, но конкретная страница, которую ищет клиент, нет.
405	Метод запрещен (Method Not Allowed): сервер получил и распознал запрос, но отклонил конкретный метод запроса.
407	Требуется проверка подлинности прокси (Proxy Authentication Required): этот код состояния аналогичен 401 Неавторизован. Единственная разница в том, что авторизация должна выполняться через прокси.
408	Время ожидания запроса (Request Timeout): срок запроса, отправленного клиентом серверу веб-сайта, истек.
409	Конфликт (Conflict): запрос на его отправку конфликтует с внутренними операциями сервера.
411	Требуемая длина (Length Required): для указанного ресурса клиент должен указать `Content-Length` в заголовке запроса
415	Тип медиа не поддерживается (Unsupported Media Type): по каким-то причинам сервер отказывается работать с указанным типом данных при данном методе.
417	Ожидание не удалось (Expectation Failed): ожидание, указанное в поле заголовка запроса «Expect», не может быть выполнено сервером.
421	Misdirected Request: запрос был перенаправлен на сервер, не способный дать ответ.
423	Заблокировано (Locked): целевой ресурс из запроса заблокирован от применения к нему указанного метода.
429	Слишком много запросов (Too Many Requests): клиент попытался отправить слишком много запросов за короткое время, что может указывать, например, на попытку DDoS-атаки.
5xx	**Ошибка сервера. Запрос, сделанный клиентом, верен, но сервер не может сгенерировать запрошенный ресурс.**
500	Внутренняя ошибка сервера (Internal Server Error): сервер столкнулся с ситуацией, которую он не может обработать при обработке запроса клиента.
501	Не реализовано (Not Implemented): сервер не знает или не может разрешить метод запроса, отправленный клиентом.
502	Плохой шлюз (Bad Gateway): сервер действовал как шлюз или прокси и получил недопустимое сообщение от входящего сервера.
503	Служба недоступна (Service Unavailable): сервер недоступен и не может обработать запрос клиента.
504	Время ожидания шлюза (Gateway Timeout): сервер в роли шлюза или прокси-сервера не дождался ответа от вышестоящего сервера для завершения текущего запроса.
505	HTTP версия не поддерживается (HTTP Version Not Supported): сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP.
507	Переполнение хранилища (Insufficient Storage): не хватает места для выполнения текущего запроса.
511	Требуется сетевая аутентификация (Network Authentication Required): ответ посылается не сервером, которому был предназначен запрос, а сервером-посредником — например, сервером провайдера — в случае, если клиент должен сначала авторизоваться в сети, например, ввести пароль для платной

	точки доступа к Интернету.
Основные задачи при тестировании API	
Проверьте правильность отправки запроса	Проверить валидацию параметров (если она есть), отправка пустого JSON, отправка не полностью заполненного JSON и т.д.
Проверить полученный ответ	Проверить правильность заполнения тела JSON при ответе, например, корректность имен, типов и значений в полях ответа, в том числе в ответах на ошибочные запросы.
Проверить безопасность запроса	Проверить, что запрос невозможно отправить с HTTP протоколом (при наличии HTTPS), также убедится, что запрос невозможно выполнить с невалидным или некорректным токеном.
Проверить корректность получения правильного статус кода	Например, создание ресурса должно возвращать 201 Создано, а неразрешенные запросы должны возвращать 403 Запрещено и т.д.
Проверьте базовую производительность метода	Если операция была завершена успешно, но заняла неоправданно много времени, тест не пройден.
HTTP Request Methods	
GET	Метод получает информацию с сервера. Более того, это наиболее часто используемый метод, который не имеет тела запроса. Каждый раз, когда вы открываете веб-сайт, срабатывает запрос Get для получения содержимого веб-сайта. Кроме того, это эквивалентно операции чтения.
POST	Используется для отправки информации на ресурс. Запрос приводит к изменениям на сервере, многократное применение запроса POST может привести к дублировании записей в БД. Содержит тело запроса, которое может быть в виде xml / html / json или в в простом текстовом формате.
PUT	Предназначен для изменения данных на ресурсе, отправка запроса с методом PUT полностью обновить запись на ресурсе. У запроса всегда присутствует тело, где передается информация для ресурса.
PATCH	Благодаря отправке запроса с методом PATCH можно частично изменять указанный ресурс. У запроса обязательно должно быть тело, где передаются параметры для корректировки данных на стороне сервера (форматы возможны, как в теле POST).
HEAD	Метод похож на облегченный метод GET, так как он также запрашивает информацию с ресурса, но в ответ от сервера не приходит тело ответа, хотя заголовки ответа возвращаются. Пригодится, чтобы проверить, что ресурс доступен, с него можно получить информацию, если нужно, а также увидеть дополнительные сведения, как дату последнего изменения или тип возвращаемого контента, например.
DELETE	Метод позволяет отправить запрос на удаление указанной сущности. У запроса может быть тело, но оно необязательно. В нем можно указать параметры для поиска записи в базе данных и ее последующего удаления.
OPTIONS	Данный HTTP-метод позволяет получить сведения о деталях соединения с ресурсом. Благодаря данному методу можно узнать какие другие методы допустимы при обращении к ресурсу (передается в ответе в хэдере Allow)
Типы API	
Private APIs	API-интерфейсы, созданные исключительно для использования внутри организации, классифицируются как внутренние приложения для сотрудников для автоматизации бизнес-процессов компании.
Public/Partner APIs	Открыто продвигаемые API-интерфейсы, но доступные для определенных разработчиков или деловых партнеров, обычно представляют собой интеграцию программного обеспечения между организациями.
External APIs	Полностью внешние API, которые доступны любому стороннему разработчику и в основном предназначены для предоставления возможности любому клиенту пользоваться данными сервисами.
Категории тестов для API	

1	Базовые позитивные тесты (happy paths)
2	Расширенные позитивные тесты с дополнительными параметрами
3	Деструктивное тестирование - тип тестирования ПО для поиска точек отказа в ПО, который проверяет систему на обработку исключительных ситуаций (срабатывание валидаторов на некорректные данные), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.
4	Тест на проверку безопасности, авторизации, прав доступа
5	Негативные тесты с валидными данными
6	Негативные тесты с невалидными данными
Проектирование HTTP методов	
Безопасность	Безопасность — это просто. Если операция может изменить ресурс — она называется небезопасной в рамках REST архитектуры. - GET, OPTIONS, HEAD — работают только на чтение, поэтому они безопасны для ресурса. - POST, PUT, DELETE — могут модифицировать данные, поэтому они небезопасные.
Идемпотентность	Идемпотентная операция — действие, многократное повторение которого эквивалентно однократному. - PUT, GET, OPTIONS, DELETE, HEAD — идемпотентные. - POST, PATCH — неидемпотентные.