**Rest API**

КАК ПРАВИЛЬНО РАБОТАТЬ С REST API <https://itvdn.com/ru/blog/article/rest-api-18>

REST API Best Practices <https://habr.com/ru/post/351890/>

API расшифровывается как Application Programming Interface. Это набор правил, позволяющий программам «общаться» друг с другом. Разработчик создает API на сервере и позволяет клиентам обращаться к нему.

REST – это архитектурный подход, определяющий, как API должны выглядеть. Читается как «Representational State Transfer». Этому набору правил и следует разработчик при создании своего приложения. Одно из этих правил гласит, что при обращении к определенному адресу, вы должны получать определенный набор данных (ресурс).

Каждый адрес маршрутом, пакет данных - запросом, в то время как результирующий ресурс – ответом.

**Структура запроса**

Важно понимать структуру запроса:

* Маршрут отправки
* Тип метода
* Заголовки
* Тело (или данные)
* ***Маршрут отправки***

Маршрут - это адрес, по которому отправляется ваш запрос. Root-endpoint - это точка приема запроса на стороне сервера (API).

Путь определяет запрашиваемый ресурс. Это что-то вроде автоответчика, который просит нажать 1 для одного сервиса, 2 для другого и так далее. Для понимания того, какие именно пути доступны, следует просмотреть документацию.

Последняя часть маршрута – это параметры запроса. Технически запросы не являются частью REST-архитектуры, но на практике сейчас всё строится на них. Параметры запроса позволяют использовать в запросе наборы пар «ключ-значение». Они всегда начинаются знаком вопроса. Каждая пара параметров после чего разделяется амперсантом.

JSON – JavaScript Object Notation – общий формат для отправки и приема данных посредством REST API.

* ***Тип метода***

Тип метода - обозначает тип производимого запроса, де-факто он является спецификацией операции, которую должен произвести сервер. Всего существует пять типов запросов:

* GET – используется для получения со стороны севера определенного ресурса. Если вы производите этот запрос, сервер ищет информацию и отправляет ее вам назад. По сути, он производит операцию чтения на сервере. Дефолтный тип запросов.
* POST – нужен для создания определенного ресурса на сервере. Сервер создает в базе данных новую сущность и оповещает вас, был ли процесс создания успешным. По сути, это операция создания.
* PUT и PATCH – используются для обновления определенной информации на сервере. В таком случае сервер просто изменяет информацию существующих сущностей в базе данных и оповещает об успехе выполнения операции.
* DELETE – как и следует из названия, удаляет указанную сущность из базы или сигнализирует об ошибке, если такой сущности в базе не было.
* ***Заголовки***

Заголовки используются, чтобы предоставить информацию как клиенту, так и серверу. Вообще, их можно использовать для много чего – пример – та же самая аутентификация и авторизация. Найти список доступных заголовком можно на официальной странице MDN. Заголовки представляют из себя пары ключей-значений.

* **Статус-коды и возможные ошибки**

HTTP определяет различные коды ответов для указания клиенту различной информации об операциях. Ваше REST приложение могло бы эффективно использовать все доступные HTTP-коды, чтобы помочь клиенту правильно настроить ответ. Далее представлен список кодов ответов HTTP:

* 200+: запрос успешен
* 300+: запрос перенаправлен на другой маршрут
* 400+: ошибка на стороне клиента
* 500+: ошибка на стороне сервера
* **Версии API**

Время от времени разработчики обновляют свои API. Порой обновления могут быть такими сильными, что разработчик желает выпустить релиз новой версии. В таком случае, если ваше приложение ломается, это происходит по причине, что вы писали код с учетом старого компонента, тогда как новый несколько отличается в плане реализации.

Запросить текущую версию API можно двумя путями.

* Через маршрут
* Через заголовок