Front-end

JavaScript

JS. Api, rest, crud, endpoint, swagger

- Api
- Rest API
- CRUD
- Endpoint
- Swagger

JS. Api, rest, crud, endpoint, swagger

Разные программы могут быть написаны на разных языках.

Это очевидно, и на первый взгляд кажется, что не вызывает никаких проблем. На деле же, если программы написаны на разных языках, их может быть трудно «подружить» и сделать так, чтобы они могли друг с другом «общаться».

Именно для того, чтобы подружить разные модули, системы, языки, программы — и существуют **API**.

JS. Api, rest, crud, endpoint, swagger

Когда нам нужны какие-то данные, мы запрашиваем их у сервера. Однако сервер написан, скорее всего, не на JavaScript, а на каком-то другом языке: Python, C#, Java. Чтобы сервер понял, что мы от него хотим, нам нужно как-то объяснить наш запрос.

Именно для этого нужно API — оно позволяет разным системам общаться, понимая друг друга.

API (Application Programming Interface) — это набор фич, которые одна программа представляет всем остальным. Она как бы говорит: «Смотрите, со мной можно говорить вот так и вот так, можете меня спросить о том-то через эту часть, а попросить что-то сделать — через эту».

В случае с клиент-серверным общением API может выступать как набор ссылок, по которым клиент обращается на сервер:

JS. Api, rest, crud, endpoint, swagger

Пример: Twitter API. Если мы хотим создать нового пользователя, нам нужно передать на сервер данные об этом пользователе. Мы не можем сделать это с помощью JS-объекта, потому что сервер использует другой язык. Значит, нам надо «перевести» данные на какой-то промежуточный язык (чаще всего это JSON).

Та же история с получением данных с сервера. Серверу надо получить данные из базы данных, перевести их в какой-то язык, который понятен клиенту, и отправить.

JS. Api, rest, crud, endpoint, swagger

REST

REST (Representational State Transfer) — стиль общения компонентов, при котором все необходимые данные указываются в параметрах запроса.

REST сейчас — один из самых распространённых стилей API в интернете.

Отличительная особенность этого стиля — это стиль построения адресов и выбор метода. Всё взаимодействие между клиентом и сервером сводится к 4 операциям (CRUD):

- созданию чего-либо, например, объекта пользователя (create, C);
- чтению (read, R);
- обновлению (update, U);
- удалению (delete, D).

JS. Api, rest, crud, endpoint, swagger

Для каждой из операций есть собственный HTTP-метод:

- POST для создания;
- GET для чтения;
- PUT, PATCH для обновления;
- DELETE для удаления.

Разница между PUT и PATCH в том, что PUT обновляет объект целиком, а PATCH — только указанное поле.

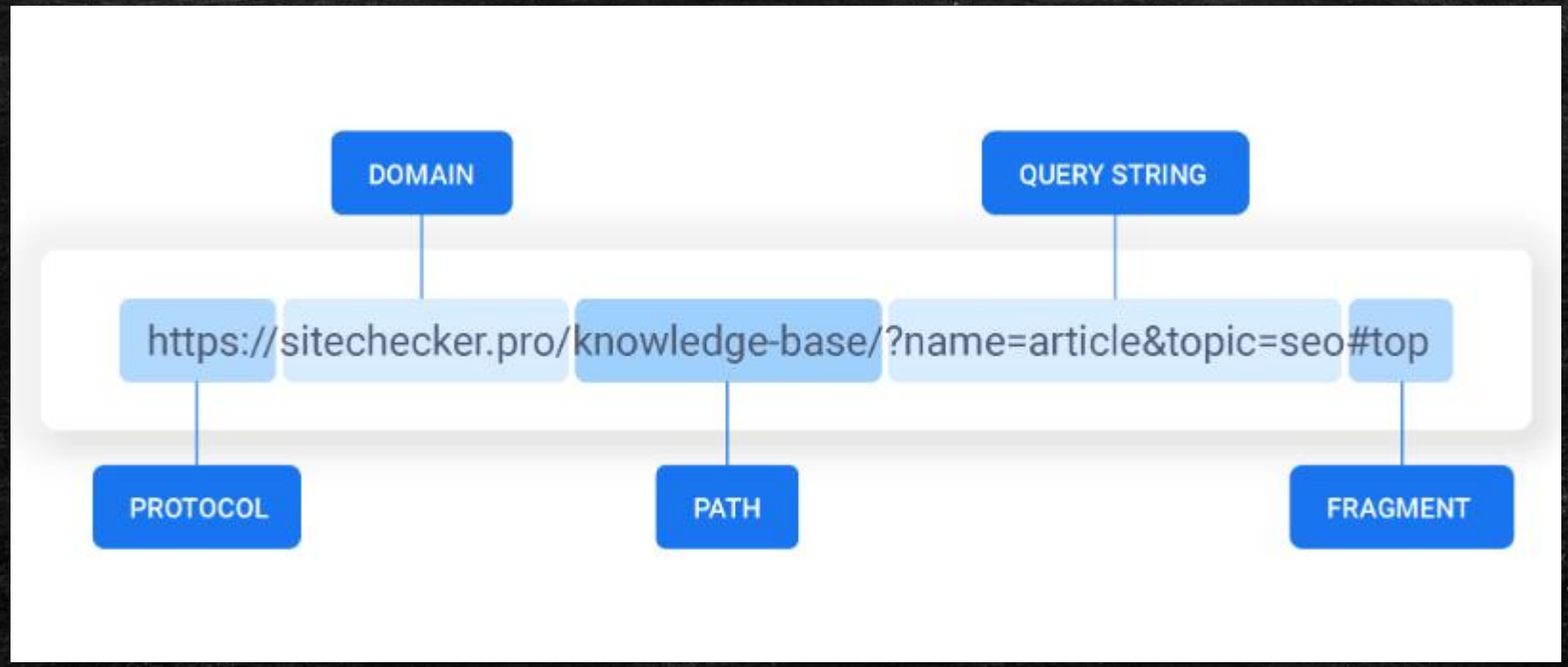
Адрес, как правило, остаётся почти одинаковым, а детали запроса указываются в HTTP-методе и параметрах или теле запроса.

JS. Api, rest, crud, endpoint, swagger

Каждый REST API запрос сообщает о результатах работы числовыми кодами — **HTTP-статусами**.

Например, редактирование записи на сервере может отработать успешно (код 200), может быть заблокировано по соображениям безопасности (код 401 или 403), а то и вообще сломаться в процессе из-за ошибки сервера (код 500). Цифровые статусы выполнения ошибок — аналог пользовательских сообщений с результатами работы программы.

JS. Api, rest, crud, endpoint, swagger



JS. Api, rest, crud, endpoint, swagger

Swagger — это набор инструментов, который позволяет автоматически описывать API на основе его кода. API — интерфейс для связи между разными программными продуктами, и у каждого проекта он свой. Документация, автоматически созданная через Swagger, облегчает понимание API для компьютеров и людей

JS. Api, rest, crud, endpoint, swagger

Коды состояния HTTP

Выделяют пять классов ответов. Идентифицировать класс можно по первой цифре.

5** – техническая ошибка на стороне сервера. Точная причина указывается сразу после кода. Иногда пятисотая говорит о внутренних сбоях, реже – о превышении статической нагрузки на сервер.

4** – сбой на стороне юзера.

3** – обнаружен редирект на другой адрес (не ошибка).

2** – запрос обработан успешно (не ошибка).

1** – служебный класс кодов, который чаще всего относится к информационным сообщениям (не ошибка).

JS. Api, rest, crud, endpoint, swagger

https://developer.mozilla.org/ru/docs/Web/HTTP/Status

https://restapitutorial.ru/httpstatuscodes.html

https://texterra.ru/blog/kody-sostoyaniya-http-proveryaem-otvety-servera-i-ubiraem-oshibki.html

JS. Api, rest, crud, endpoint, swagger

Заголовки запроса

Для установки заголовка запроса в fetch мы можем использовать опцию **headers**. Она содержит объект с исходящими заголовками

https://developer.mozilla.org/ru/docs/Glossary/MIME_type

https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Content-Type

https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Accept

JS. Api, rest, crud, endpoint, swagger

POST-запросы

Для отправки POST-запроса или запроса с другим методом, нам необходимо использовать fetch параметры:

- **method** – HTTP метод, например POST
- **body** – тело запроса