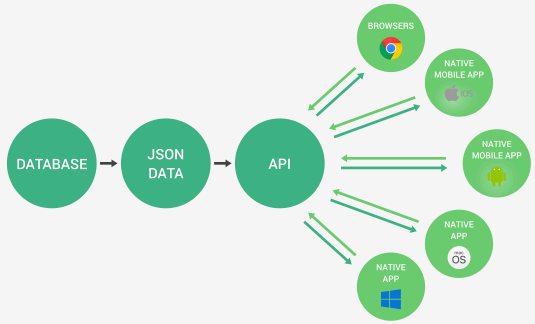
WHAT IS AN API ANYWAY?

API

Application **P**rogramming **I**nterface: a piece of software that can be used by another piece of software, in order to allow applications to talk to each other.

👉 Web APIs



👉 But, "Application" can be other things:

- 👉 Node.js' `fs` or `http` APIs ("node APIs");
- 👉 Browser's DOM JavaScript API;
- 👉 With object-oriented programming, when exposing methods to the public, we're creating an API;
- 👉 ...

THE REST ARCHITECTURE

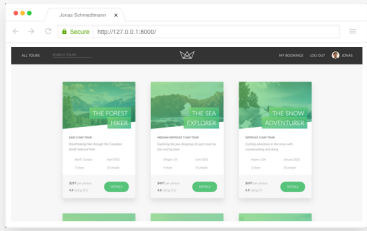
1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

👉 **Resource:** Object or representation of something, which has data associated to it. Any information that can be **named** can be a resource.

`tours` `users` `reviews`



URL

`https://www.natours.com/addNewTour`

ENDPOINT

`/getTour`

`/updateTour`

BAD 👎

`/getToursByUser`

`/deleteToursByUser`

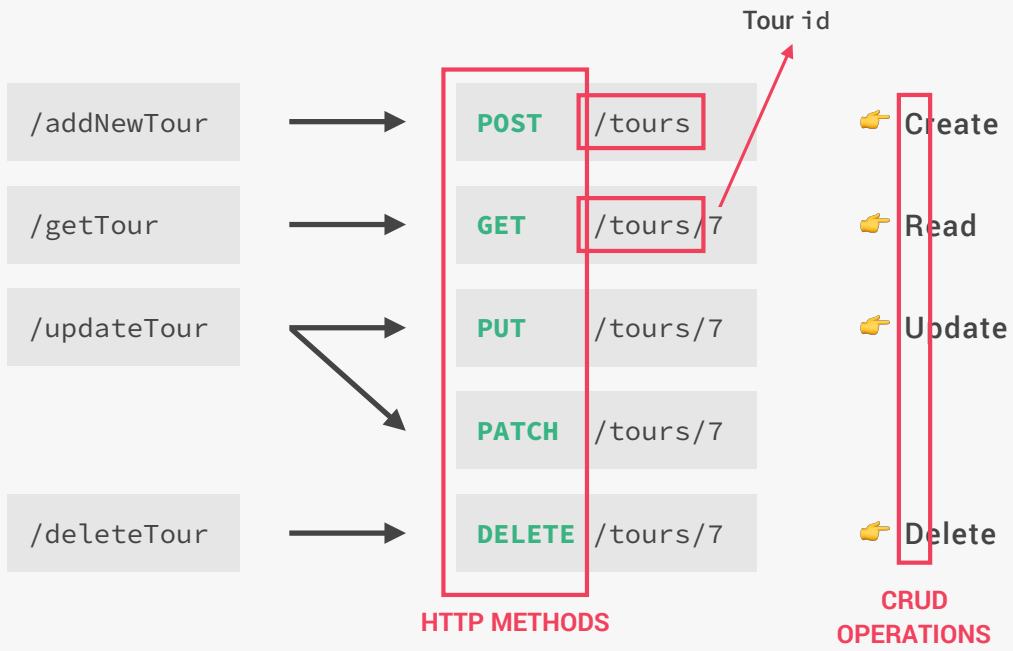
Never use actions in endpoints!

👉 Endpoints should contain **only resources** (nouns), and use **HTTP methods** for actions!

THE REST ARCHITECTURE

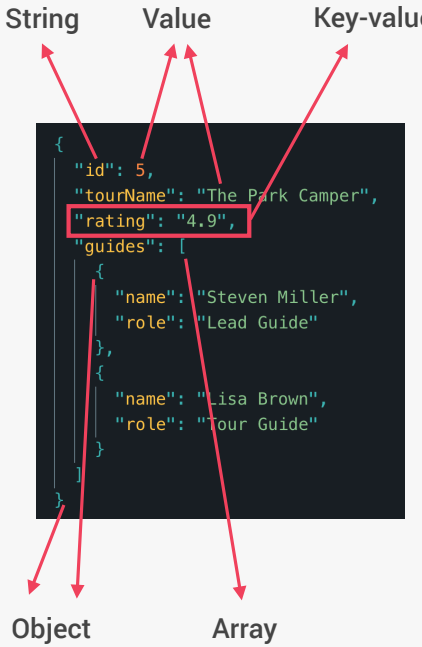
It's a convention to use resource name in the plural (eg: tours)

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**



THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**



JSend - create a new object then add a status message in it to inform the client about the status and the we put our data in the object called data.

RESPONSE FORMATTING

👉 **JSend**

👉 **JSON:API**

👉 **OData JSON Protocol**

👉 **...**

https://www.natours.com/tours/5

Enveloping - Wrapping the data in additional obj is a common practice to mitigate security issues

THE REST ARCHITECTURE

1. Separate API into logical **resources**
2. Expose structured, **resource-based URLs**
3. Use **HTTP methods** (verbs)
4. Send data as **JSON** (usually)
5. Be **stateless**

👉 **Stateless RESTful API:** All state is handled **on the client**. This means that each request must contain **all** the information necessary to process a certain request. The server should **not** have to remember previous requests.

👉 **Examples of state:** `loggedIn` `currentPage`