REST – Representational State Transfer

Most commonly uses HTTP

Don't rely on HTTP

Summary HTTP services should follow this architectural style

REST is an architectural paradigm, which defines an architecture style. REST works with resources (for example books and authors in a bookstore). The server sends a representation of these resources in the form of their state to the client. This representation is usually sent in JSON format. REST defines several constraints. These constraints are guidelines, that restful services should follow.

Constraint 1: Uniform interface

Embrace standards, use the http request methods the right way (don't use POST to fetch data for example). Model your API around resources and not around functions. Send representational state transfer representational state of these resources. The client manipulates resources through json most of the time working on representations and doesn't work directly on the resources. don't directly manipulate the resources on the server

Constraint 2: Client Server

Client only knows interface of server, the endpoints we provide and how do they look like.

Client and Server should be able to evolve independent of each other. As long as the interface doesn't change, the client shouldn't have to care about changes on the server.

only changes in interface should affect our client

Constraint 3: Stateless

The server should treat each request seperately. as new request No request should depend on previous requests. There should not be any client data cached on the server. The client is responsible for handling the state of the application.

Constraint 4: Layered

REST allows you to build distributed systems. There could be load balancers, authentication servers, databases on different machines. These layers shouldn't affect your client. The client should only have to send a request to one single endpoint.

the client doesn't see layers

client only sees one interface

only one address to call these endpoints