What is an API

Software as a Service - Back-End Development
Session 01
Developed by Adrian Gould

Embracing the irrefutable correlation between novelty and quality
Essential
Shiny New Things
O RLY?

Contents

Development Environment

Please make sure you check the required development environment details.

Application Programming Interface

A	Application
P	Programming
I	Interface

A method of communicating between two different systems or parts of a system.

A 'translator' of data to allow communication between two points

---

Examples where APIs are used

- Hardware (Drivers provide an API for the developer)
- Operating Systems (APIs to interact with the OS)
- Applications (APIs to interact and extend)
- Web Services (Google, Facebook, GitHub, …)
- …

API Types

- REST
- GraphQL
- SOAP
- gRPC

> Question: Are there others?

API Types: REST

	{REST}
Re	Representational
S	State
T	Transfer

Concept was first presented in 2000 by Roy Fielding

Data usually transferred as JSON

- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Ics.uci.edu. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

---

API Types: SOAP

	SOAP
S	Simple
O	Open
A	Access
P	Protocol

Data transferred as XML

API Types: GraphQL

	GraphQL
Graph	Graph
Q	Query
L	Language

Data transferred as ???

API Types: gRPC

	gRPC
g	general
R	Remote
P	Procedure
C	Call

Data transferred as ???

API Type Comparison

{ REST }	GraphQL	gRPC	SOAP
Positives ✅	✅	✅	✅
Stateless	Avoids over-fetching	Built on HTTP/2	…
Standard HTTP methods	Avoids under-fetching	Multiplexing	…
JSON data exchange		Server push	
		Uses Protocol Buffers	
		Efficient	
Negatives 🟥	🟥	🟥	🟥
Often over-fetched	Queries impact server performance	Less human readable	…
Often under-fetches	POST requests only	Requires HTTP/2	…
	Response HTTP 200		

Simonyan, H. (2023). API Design 101: From Basics to Best Practices [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=7QfswaV0re4

Exercise: What positives and negatives could be added to the SOAP column.

API Design 101: Basics to Best Practices

Video content to watch:

- [API Design 101: From Basics to Best Practices (youtube.com)]

REST: Six Guiding Principles

Constraints/principles to promote simplicity, scalability, and statelessness in the design.

1. Uniform Interface
2. Client-Server
3. Stateless
4. Cacheable
5. Layered System
6. Code on Demand (Optional)

- Gupta, L. (2019, June 5). *REST Principles and Architectural Constraints – REST API Tutorial*. Restfulapi.net. https://restfulapi.net/rest-architectural-constraints/

Stateless

- Requests are not bound to a single server
- No knowledge of the previous request(s)
- State not stored in application
- Client takes care of tracking state

An API where each client request contains all the necessary information needed to process the request and the the application does not maintain any session state of context information between requests.

Making stateless 'stateful'

e.g. shopping cart

- identify state of application
- store state in database/cache
- include session id / cookie for correct cart access

The API is designed to be stateless

REST API

> A REST API consists of an assembly of interlinked resources.
>
> This set of resources is known as the REST API’s ***resource model***.

REST Resources

Any information that we can name can be a **resource**.

We will know the state of the resource:

- at any particular time
- **resource representation** made of:
  - the **data**,
  - the **metadata**, &
  - the **hypermedia links**

Resource Identifiers

- Identify each resource involved in interaction between the client and the server components.

Hypermedia

- The data format of a representation is known as a media type
- The media type identifies a specification that defines how a representation is to be processed.

Self-Descriptive

- **Resource representations shall be self-descriptive**

- The client does not need to know what the resource is.
- Client should act based on the resource's media type.

Naming API Endpoints

- Use Nouns (Fruit, Cart, User)
- Lower Case! (fruit, cart, user)
- Pluralise! (fruit, carts, users…)
- No Verbs (get, put, save…)
- Prepend with `api`
- Prepend with version, eg `v1`
- Use **correct/appropriate** HTTP Method
- Kebab style (use hyphens between words is needed)

BAD	Good	HTTP
getUsers	/api/v1/users	GET
getUserById	/api/v1users/{id}	GET
getUsersV1	/api/v1/users	GET
store-inventory	/api/v2/inventories	POST
updateUserByID	/api/v1/users/{id}	PUT/PATCH
removeProductFromCart	/api/v2/cart/{cart_id}/product/	DELETE

CoderOne. (2023). REST API Mistakes Every Junior Developer should Avoid | clean-code [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=JxeTegu4dD8&t=8s

REST & Resource Methods

Resource methods are used to perform the desired transition between two resource states.

No specification as to what we use for methods, could be:

- GET, POST, PATCH,…
- GET, SAVE, CHANGE,…

Resource methods **do not directly relate to** the HTTP methods (i.e., GET/PUT/POST/DELETE)

REST and HTTP are not the Same

Many people prefer to compare HTTP with REST.

REST and HTTP are not the same.

> **REST != HTTP**

Aside: Often the HTTP methods are used due to familiarity.

HTTP Request & Structure

> **All** the lines end with a carriage return (CR) and line feed (LF)

- A request line to get a required resource (e.g. GET /content/page1.html)
- Headers (e.g. Accept-Language: EN)
- An empty line (must contain ONLY CR & LF)
- A message body (optional)

What is HTTP, Structure of HTTP Request and Response?» WebNots. (2013, June 28). WebNots. https://www.webnots.com/what-is-http/

What Is an HTTP Request. (n.d.). Sematext. https://sematext.com/glossary/http-requests/

HTTP Verbs

- GET
- POST
- PUT
- PATCH
- DELETE
- OPTIONS

HTTP Request Response

Basic response from the server contains:

- HTTP Status Code (e.g. HTTP/1.1 301 Moved Permanently)
- Headers (e.g. Content-Type: html)
- An empty line
- A message body (optional)

> **All** the lines end with a carriage return (CR) and line feed (LF)

HTTP Status/Response Codes

- **1xx: Informational** – Communicates transfer protocol-level information.
- **2xx: Success** – Indicates that the client's request was accepted successfully.
- **3xx: Redirection** – Indicates that the client must take some additional action in order to complete their request.
- **4xx: Client Error** – This category of error status codes points the finger at clients.
- **5xx: Server Error** – The server takes responsibility for these error status codes.

Gupta, L. (2018, May 30). *HTTP Status Codes*. REST API Tutorial. https://restfulapi.net/http-status-codes/

HTTP Common Response Codes

Useful / most used:

- 200 - OK
- 201 - Created
- 301 - Moved Permanently
- 302 - Found
- 400 - Bad Request
- 401 - Unauthorized
- 403 - Forbidden

HTTP Common Response Codes

Useful / most used - Continued:

- 404 - Not Found
- 405 - Method not allowed
- 429 - Too many requests
- 500 - Internal Server Error
- 502 - Bad Gateway
- 503 - Service Unavailable
- 504 - Gateway Timeout

Resources used

- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Ics.uci.edu. https://ics.uci.edu/~fielding/pubs/dissertation/top.htm
- Gupta, L. (2019, June 5). *REST Principles and Architectural Constraints – REST API Tutorial*. Restfulapi.net. https://restfulapi.net/rest-architectural-constraints/
- *What is HTTP, Structure of HTTP Request and Response?» WebNots*. (2013, June 28). WebNots. https://www.webnots.com/what-is-http/
- *What Is an HTTP Request*. (n.d.). Sematext. https://sematext.com/glossary/http-requests/
- Gupta, L. (2018, May 30). *HTTP Status Codes*. REST API Tutorial. https://restfulapi.net/http-status-codes/
- Santos, L. (2020, March 4). The Complete Guide to Status Codes for Meaningful ReST APIs - Let's start! DEV Community. https://dev.to/_staticvoid/the-complete-guide-to-status-codes-for-meaningful-rest-apis-1-5c5
- ...

End

Next up: Laravel and APIs