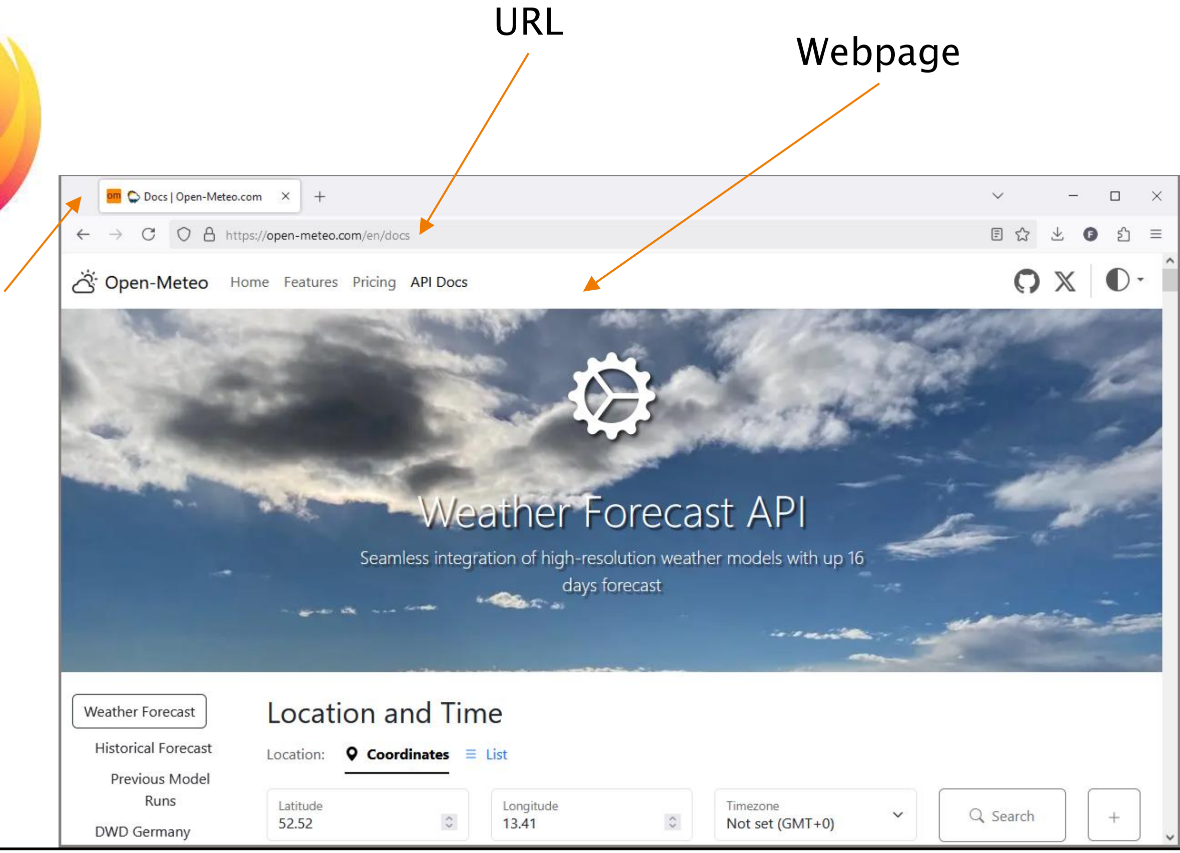
01 – Hypertext Transfer Protocol and Representational State Transfer

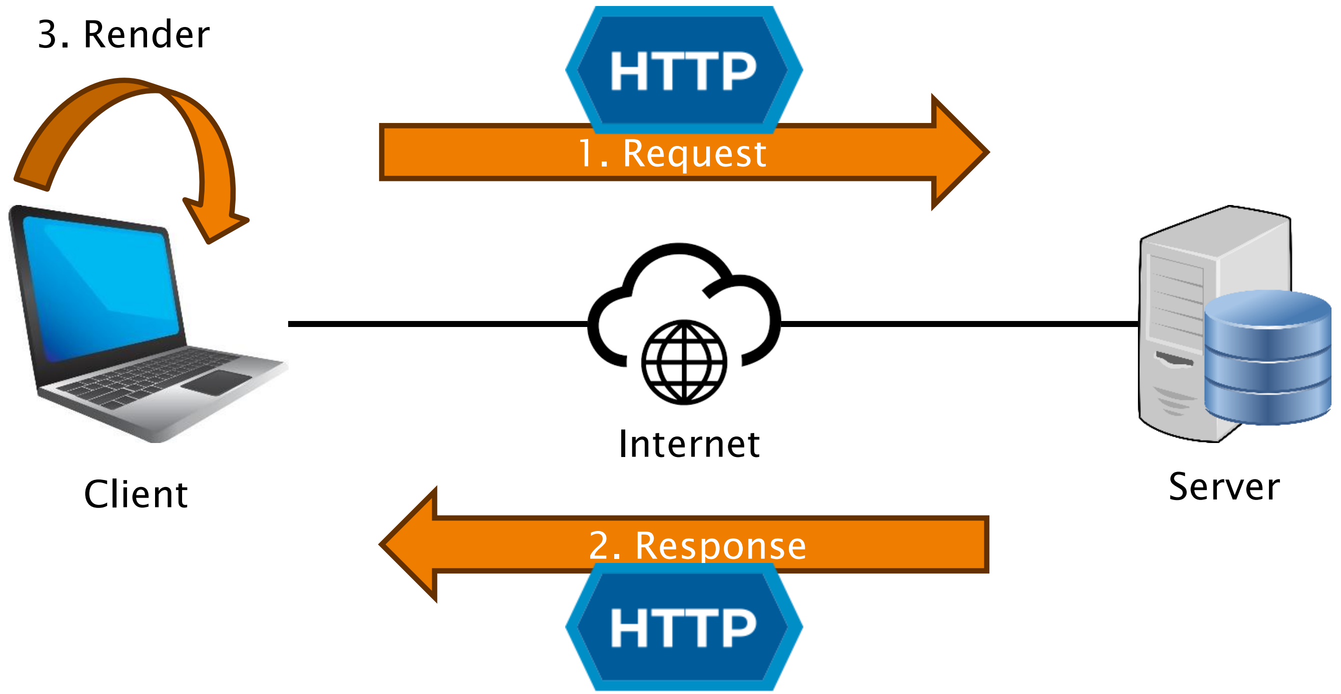
Web Technology Project (International Computer Science)
Summer semester 2025
Prof. Dr. Felix Schwägerl

OTH REGENSBURG

What will the weather be like this week?



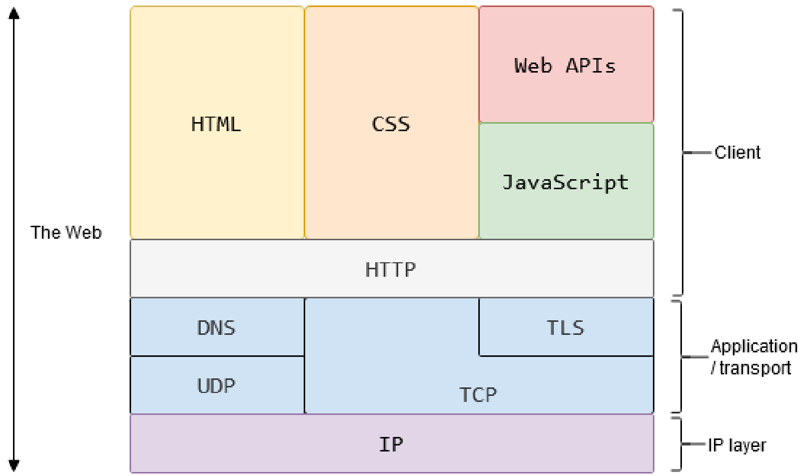
How does a web browser work?



HTTP is the basis of all communication over the internet.

- Introduced 1991 by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C)
- Current version: HTTP/3 (published as RFC 9114 in June 2022)
- Traditionally used for transferring *HTML* documents (→ chapter 03)
- *HTTPS* = *HTTP* + *SSL* (secure communication)
- Built on top of existing protocols of the transport layer (*TCP/IP*) (→ **Computer Networks**)
- Official specification [IETF 2025] is dedicated to browser developers; a useful reference for application developers is [Mozilla 2025]

[Mozilla 2025]



Structure of a HTTP request

Verb (Method) *Path* *Protocol version*

```
GET https://open-meteo.com/en/docs HTTP/2
Host: open-meteo.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml
Accept-Language: en,de-DE;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Alt-Used: open-meteo.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Priority: u=0, i
```

Request headers

- The browser generates verb, protocol version, and request headers automatically (based on user preferences) when we enter the path into the address bar.
- After the headers, a *request body* may follow after a blank line. (e.g., when using verbs **POST** or **PUT** to create/update data)

Structure of a HTTP response

Protocol version *Status code* *Status message*

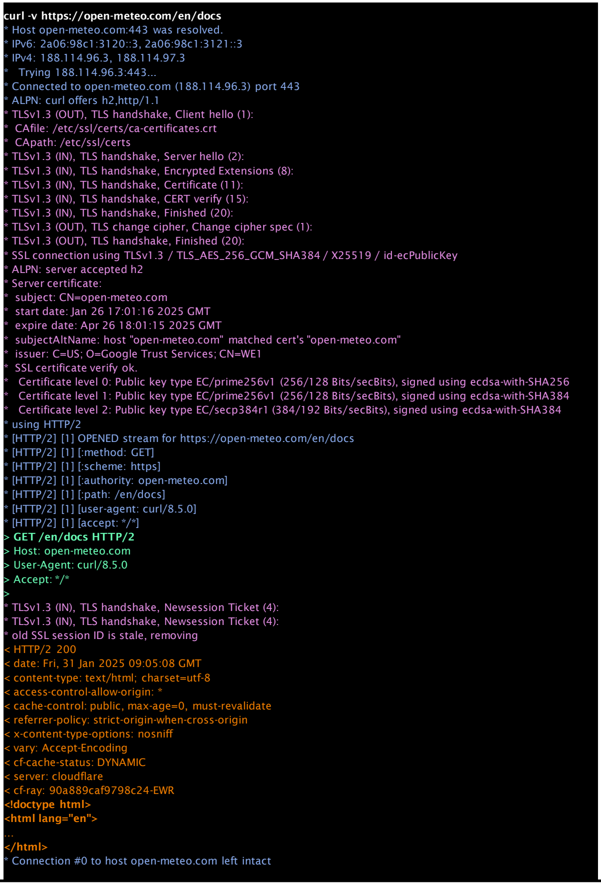
```
HTTP/2 200 OK
date: Fri, 31 Jan 2025 09:25:12 GMT
content-type: text/html; charset=utf-8
access-control-allow-origin: *
cache-control: public, max-age=0, must-revalidate
referrer-policy: strict-origin-when-cross-origin
x-content-type-options: nosniff
vary: Accept-Encoding
cf-cache-status: DYNAMIC
server: cloudflare
cf-ray: 90a8a732b976c356-EWR
content-encoding: br
alt-svc: h3=":443"; ma=86400

<!doctype html>
<html lang="en">
…
</html>
```

Response headers

Blank line

Response body (here: a HTML document)

Detailed protocol trace of a HTTP request



```
curl -v https://open-meteo.com/en/docs
* Host open-meteo.com:443 was resolved.
* IPv6: 2a06:98c1:3120::3, 2a06:98c1:3121::3
* IPv4: 188.114.96.3, 188.114.97.3
*   Trying 188.114.96.3:443...
* Connected to open-meteo.com (188.114.96.3) port 443
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: /etc/ssl/certs
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519 / id-ecPublicKey
* ALPN: server accepted h2
* Server certificate:
*  subject: CN=open-meteo.com
*  start date: Jan 26 17:01:16 2025 GMT
*  expire date: Apr 26 18:01:15 2025 GMT
*  subjectAltName: host "open-meteo.com" matched cert's "open-meteo.com"
*  issuer: C=US; O=Google Trust Services; CN=WE1
*  SSL certificate verify ok.
*   Certificate level 0: Public key type EC/prime256v1 (256/128 Bits/secBits), signed using ecdsa-with-SHA256
*   Certificate level 1: Public key type EC/prime256v1 (256/128 Bits/secBits), signed using ecdsa-with-SHA384
*   Certificate level 2: Public key type EC/secp384r1 (384/192 Bits/secBits), signed using ecdsa-with-SHA384
* using HTTP/2
* [HTTP/2] [1] OPENED stream for https://open-meteo.com/en/docs
* [HTTP/2] [1] [:method: GET]
* [HTTP/2] [1] [:scheme: https]
* [HTTP/2] [1] [:authority: open-meteo.com]
* [HTTP/2] [1] [:path: /en/docs]
* [HTTP/2] [1] [user-agent: curl/8.5.0]
* [HTTP/2] [1] [accept: */*]
> GET /en/docs HTTP/2
> Host: open-meteo.com
> User-Agent: curl/8.5.0
> Accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
< HTTP/2 200
< date: Fri, 31 Jan 2025 09:05:08 GMT
< content-type: text/html; charset=utf-8
< access-control-allow-origin: *
< cache-control: public, max-age=0, must-revalidate
< referrer-policy: strict-origin-when-cross-origin
< x-content-type-options: nosniff
< vary: Accept-Encoding
< cf-cache-status: DYNAMIC
< server: cloudflare
< cf-ray: 90a889caf9798c24-EWR
<!doctype html>
<html lang="en">
...
</html>
* Connection #0 to host open-meteo.com left intact
```

Calling the **curl** Linux app from the command line

Initiating TCP/IP connection

Establishing secure connection
(HTTP with SSL over TLS)

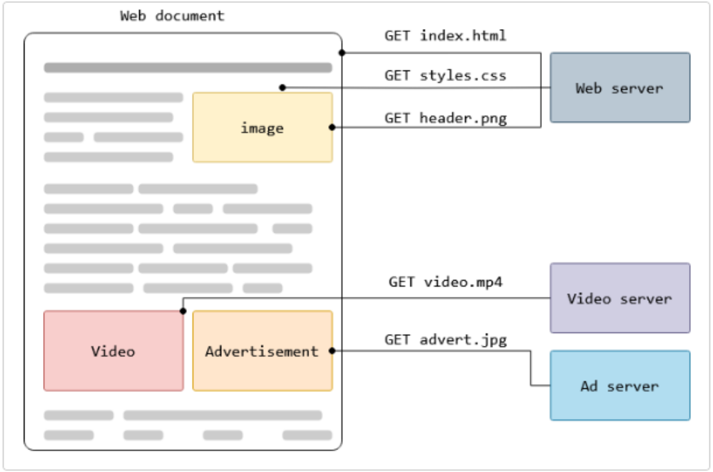
Initiating HTTP session

Sending HTTP request

Receiving HTTP response

Static background HTTP requests

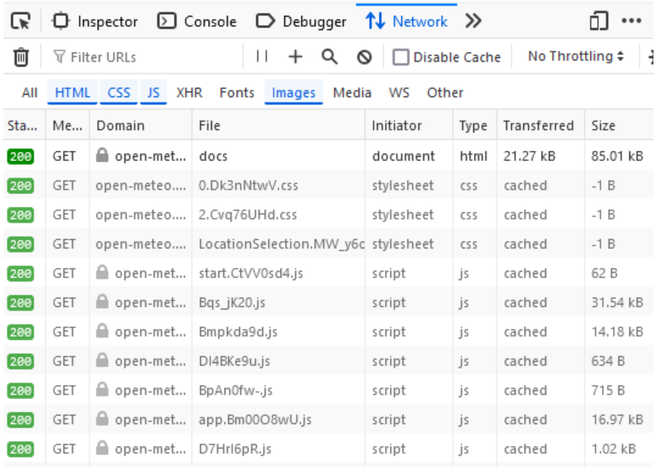
A HTML document contains links to additional resources.



[Mozilla 2025]

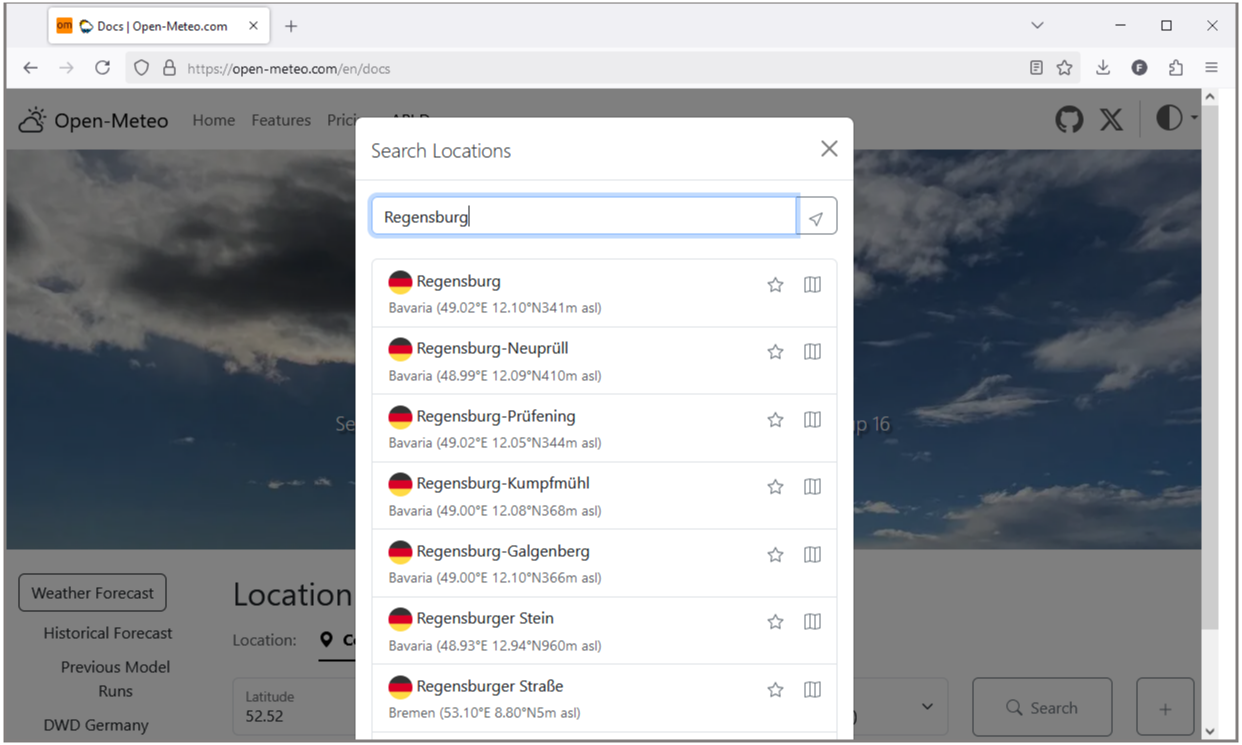
- Media content: images, videos, etc.
- Additional HTML documents (→ chapter 03)
- CSS (→ chapter 04)
- JavaScript (→ chapter 05)

- Browser reveals all background requests
  - E.g. in Firefox: Ctrl+Shift+I
- Go to Network tab and filter by resource type (HTML, CSS, JS, Images



Sta...	Me...	Domain	File	Initiator	Type	Transferred	Size
200	GET	open-met...	docs	document	html	21.27 kB	85.01 kB
200	GET	open-meteo....	0.Dk3nNtwV.css	stylesheet	css	cached	-1 B
200	GET	open-meteo....	2.Cvq76UHd.css	stylesheet	css	cached	-1 B
200	GET	open-meteo....	LocationSelection.MW_y6c	stylesheet	css	cached	-1 B
200	GET	open-met...	start.CtVV0sd4.js	script	js	cached	62 B
200	GET	open-met...	Bqs_jK20.js	script	js	cached	31.54 kB
200	GET	open-met...	Bmpkda9d.js	script	js	cached	14.18 kB
200	GET	open-met...	DI4BKe9u.js	script	js	cached	634 B
200	GET	open-met...	BpAn0fw-.js	script	js	cached	715 B
200	GET	open-met...	app.Bm00O8wU.js	script	js	cached	16.97 kB
200	GET	open-met...	D7Hrl6pR.js	script	js	cached	1.02 kB

Interacting with a webpage (1)



Search Locations

Regensburg

Regensburg
Bavaria (49.02°E 12.10°N341m asl)

Regensburg-Neuprüll
Bavaria (48.99°E 12.09°N410m asl)

Regensburg-Prüfening
Bavaria (49.02°E 12.05°N344m asl)

Regensburg-Kumpfmühl
Bavaria (49.00°E 12.08°N368m asl)

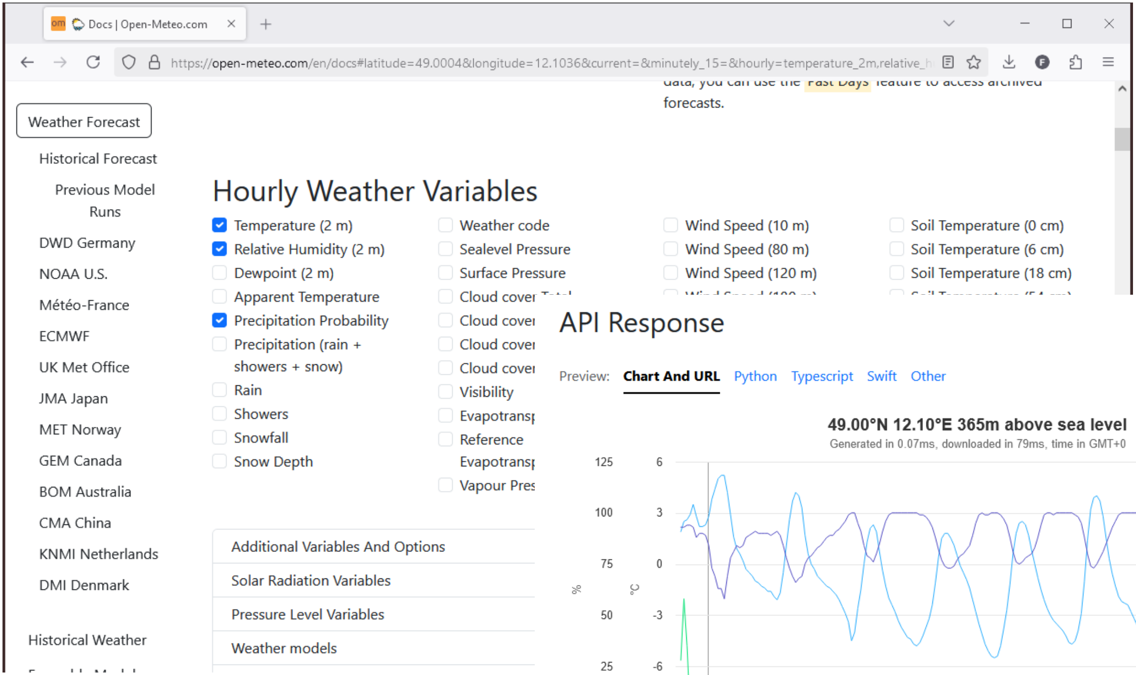
Regensburg-Galgenberg
Bavaria (49.00°E 12.10°N366m asl)

Regensburger Stein
Bavaria (48.93°E 12.94°N960m asl)

Regensburger Straße
Bremen (53.10°E 8.80°N5m asl)

→ Where do these suggestions come from? (We didn't request new data via the address bar.)

Interacting with a webpage (2)



Docs | Open-Meteo.com

https://open-meteo.com/en/docs#latitude=49.0004&longitude=12.1036&current=&minutely_15=&hourly=temperature_2m,relative_h

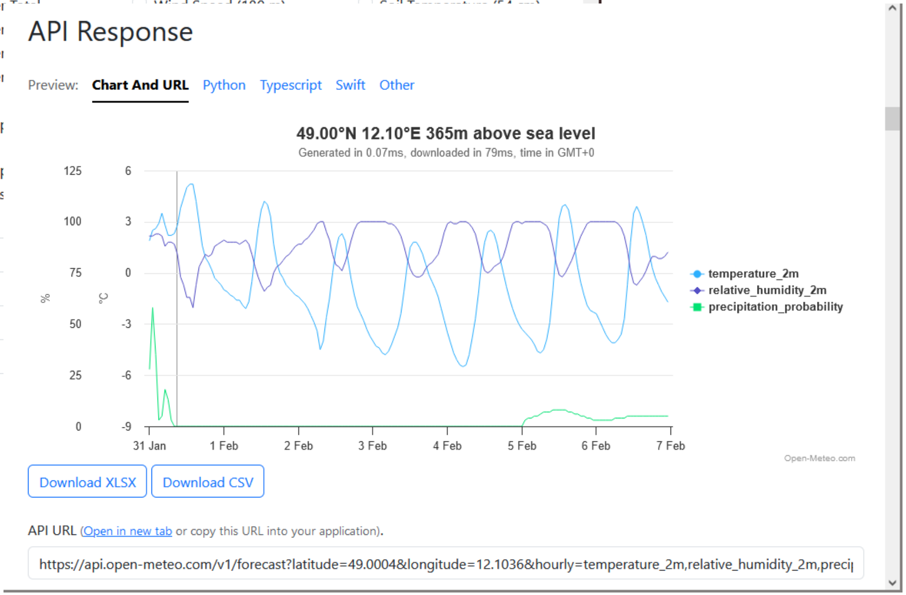
forecasts.

Weather Forecast
Historical Forecast
Previous Model Runs
DWD Germany
NOAA U.S.
Météo-France
ECMWF
UK Met Office
JMA Japan
MET Norway
GEM Canada
BOM Australia
CMA China
KNMI Netherlands
DMI Denmark
Historical Weather

Hourly Weather Variables

- [✓] Temperature (2 m)
- [✓] Relative Humidity (2 m)
- [ ] Dewpoint (2 m)
- [ ] Apparent Temperature
- [✓] Precipitation Probability
- [ ] Precipitation (rain + showers + snow)
- [ ] Rain
- [ ] Showers
- [ ] Snowfall
- [ ] Snow Depth
- [ ] Weather code
- [ ] Sealevel Pressure
- [ ] Surface Pressure
- [ ] Cloud cover
- [ ] Cloud cover
- [ ] Cloud cover
- [ ] Cloud cover
- [ ] Visibility
- [ ] Evapotransp
- [ ] Reference Evapotransp
- [ ] Vapour Pres
- [ ] Wind Speed (10 m)
- [ ] Wind Speed (80 m)
- [ ] Wind Speed (120 m)
- [ ] Soil Temperature (0 cm)
- [ ] Soil Temperature (6 cm)
- [ ] Soil Temperature (18 cm)

Additional Variables And Options
Solar Radiation Variables
Pressure Level Variables
Weather models



API Response

Preview: **Chart And URL** Python Typescript Swift Other

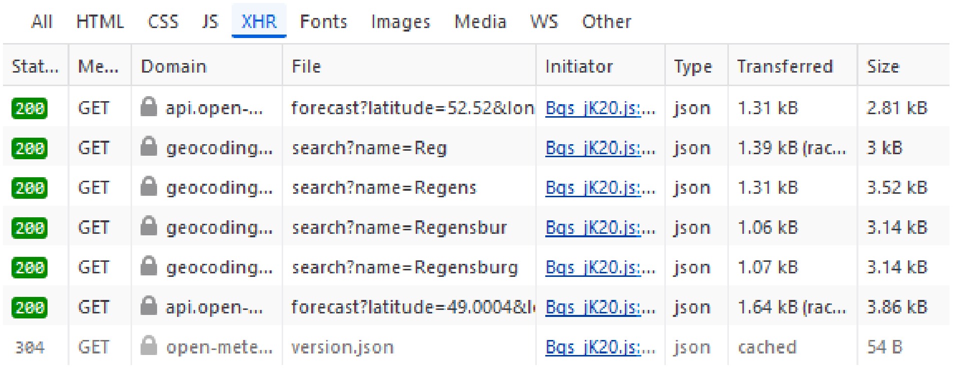
Download XLSX Download CSV

API URL (Open in new tab or copy this URL into your application).

https://api.open-meteo.com/v1/forecast?latitude=49.0004&longitude=12.1036&hourly=temperature_2m,relative_humidity_2m,preci

→ How are the graphs updated when the variables are changed?

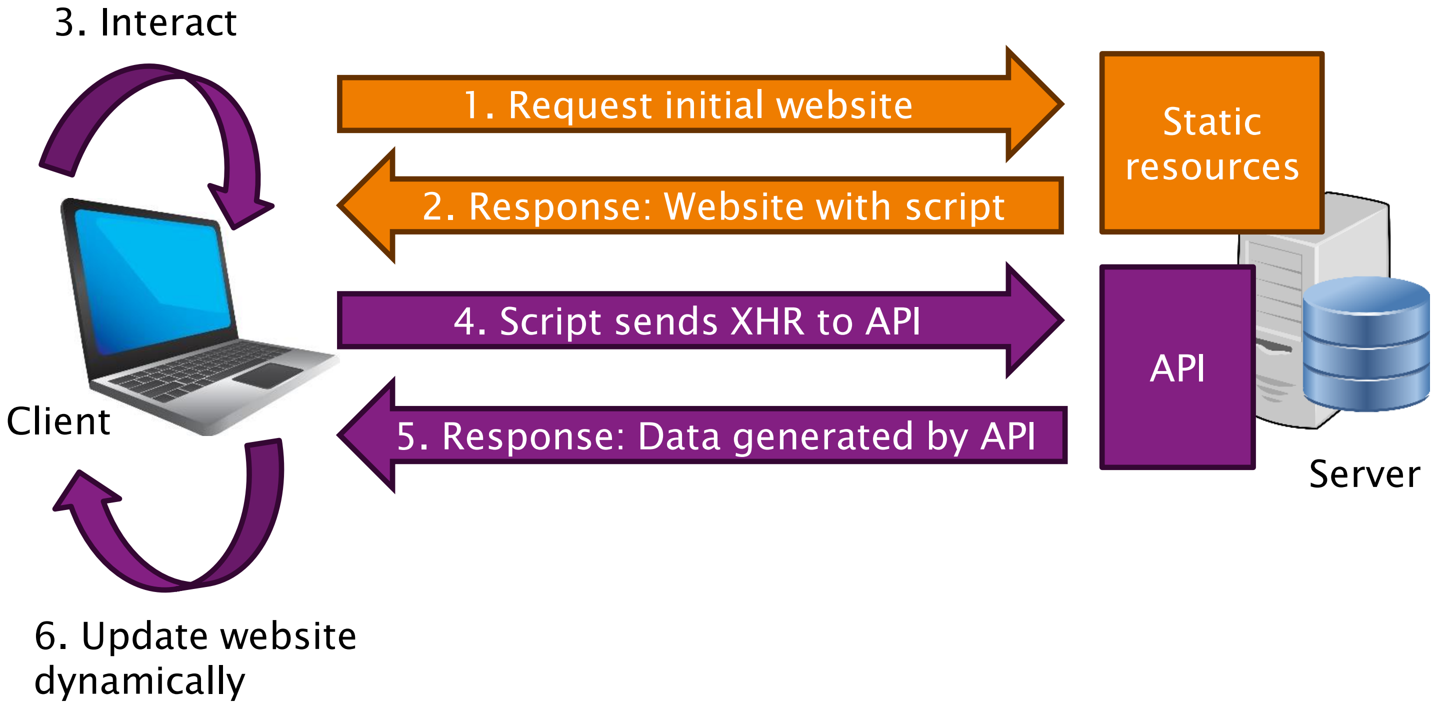
What about the XHR category in the Network tab?


All HTML CSS JS XHR Fonts Images Media WS Other

Stat...	Me...	Domain	File	Initiator	Type	Transferred	Size
200	GET	api.open-...	forecast?latitude=52.52&lon	Bqs_jK20.js:...	json	1.31 kB	2.81 kB
200	GET	geocoding...	search?name=Reg	Bqs_jK20.js:...	json	1.39 kB (rac...	3 kB
200	GET	geocoding...	search?name=Regens	Bqs_jK20.js:...	json	1.31 kB	3.52 kB
200	GET	geocoding...	search?name=Regensbur	Bqs_jK20.js:...	json	1.06 kB	3.14 kB
200	GET	geocoding...	search?name=Regensburg	Bqs_jK20.js:...	json	1.07 kB	3.14 kB
200	GET	api.open-...	forecast?latitude=49.0004&l	Bqs_jK20.js:...	json	1.64 kB (rac...	3.86 kB
304	GET	open-mete...	version.json	Bqs_jK20.js:...	json	cached	54 B

- XHR is for “XML HTTP request”
  - The name is misleading since it does not require XML at all.
  - Formerly called AJAX (“asynchronous JavaScript and XML”)
  - Here called *dynamic HTTP requests*, or *API calls*.
- Idea: Websites may execute *scripts* (→ chapter 05), which use XHRs to retrieve data from or send data to servers as the user interacts with the website.
  - Technically, XHRs are normal HTTP requests.
  - The requests/responses are not directly shown to the user.

How do interactive web applications work?


3. Interact
1. Request initial website
2. Response: Website with script
4. Script sends XHR to API
5. Response: Data generated by API
Static resources
API
Client
Server
6. Update website dynamically

HTTP Uniform Resource Locators (URLs)

Schema
(here: network protocol)

Host
(public name of the server on the internet)

Path
(hierarchical, with segments separated by '/')

Fragment
(not sent to server, but evaluated by client for addressing relevant part of response)

```
https://user:pw@www.example.com:8080/documentation/index.html?p1=A&p2=B#resource
```

Identification
(should not be done in this way; clear-text password)

Port
(default: 80 for HTTP, 443 for HTTPS)

Query
(additional parameters sent to the server in key=value form)

Important request headers

Name	Meaning	Example
`User-Agent`	Browser/OS that originated the request	`Mozilla/5.0 (Windows)`
`Origin`	Origin server (if request is server-to-server)	`http://my-api.de:8080`
`Referer`	Server from where a link was followed	`http://www.google.com`
`Host`	Domain the request will be sent to	`open-meteo.com`
`Accept`	Accepted/preferred response content type	`text/html`
`Accept-Language`	Accepted/preferred response language	`en,de-DE`
`Accept-Encoding`	Compression formats understood by client	`gzip, deflate`
`Authorization`	Credentials for protected access (→)	`Basic dXNlcjpwdw==`
`Cookie`	Session data stored by the client	`sessionId=12345678`
`If-Modified-Since`	Request only if modified since	`19 Jan 2025 10:24:26`
`If-None-Match`	Request if version (e.g., hash) changed	`753af3c3e0`
`Connection`	Whether HTTP connection should be kept	`keep-alive`

More examples: https://developer.mozilla.org/en-US/docs/Glossary/Request_header charset=utf-8

Important response headers

Name	Meaning	Example
`Server`	Server software answering the request	`Apache`
`Access-Control-Allow-Origin`	Server whitelist	`https://www.my-api.de`
`Date`	When the server sent the response	`19 Jan 2025 10:24:26`
`Expires`	Response is considered stale after	`20 Jan 2025 10:24:26`
`Allow`	HTTP methods allowed for requested URL	`GET, POST, HEAD`
`Location`	URL supposed to be visited by requester	`http://new-domain.com`
`Vary`	Request headers that influenced response	`Cookie, Accept`
`Content-Type`	Actual type of the response body	`text/html; charset=utf-8`
`Content-Length`	Length of response body in bytes	`181`
`Content-Language`	Language used in returned content	`de`
`Content-Encoding`	Compression format applied to response	`gzip`
`Content-Disposition`	Download information	`attachment; filename="a.txt"`
`WWW-Authenticate`	Scheme of requested authentication	`Basic`
`Set-Cookie`	Requests client to update session data	`sessionId=12345678`
`Last-Modified`	Modification stamp of returned data	`19 Jan 2025 08:21:19`
`Etag`	Identifier of resource version (e.g. hash)	`W/"753af3c3e0"`
`Connection`	Whether HTTP connection should be kept	`keep-alive`
`Keep-Alive`	Parameters of connection keep-alive	`timeout=5 max=997`

More examples: https://developer.mozilla.org/en-US/docs/Glossary/Response_header

JavaScript Object Notation (JSON)

```
{
  "results": [
    {
      "id": 2849483,
      "name": "Regensburg",
      "latitude": 49.01513,
      "longitude": 12.10161,
      "elevation": 341.0,
      "country_code": "DE",
      "timezone": "Europe/Berlin",
      "population": 129151,
      "postcodes": [
        "93057"
      ],
      "country_id": 2921044,
      "country": "Germany",
      "admin1": "Bavaria",
      "admin2": "Upper Palatinate",
      "admin3": "Regensburg"
    },
    {
      "id": 8378695,
      "name": "Regensburg-Neuprüll",   …
    }, …
  ],
  "generationtime_ms": 0.87690353
}
```

- JSON is the preferred data serialization format on the web.
  - Compared to XML (eXtensible Markup Language), JSON is more lightweight and easier to read and write.
  - JSON documents may be validated with a JSON *Schema* (less often used than XML schema)
- *Objects* are structured data enclosed in `{ }`
- The data of objects is represented as key-value pairs, where the key is a string.
- *Values* may be *strings*, *numbers*, `true`, `false`, or `null`.
- Objects also count as values, such that they may be *nested*.
- *Arrays* are another type of value, enclosing sequences of values in `[ ]`
  - Array values are typically *homogeneous*, although not strictly required by JSON.

REGENSBURG

HTTP Verbs

Name	Purpose
GET	Request a representation of the resource specified by the URL.
POST	Submit an entity to the specified resource. (Often: create a new resource under the specified resource and return the address of the created resource.)
PUT	Replace representation of the specified resource with request body.
DELETE	Permanently delete the specified resource.
CONNECT	Establish a communication tunnel to the server owning the target resource.
TRACE	Perform a message loop-back test along the path to the target resource.
OPTIONS	Communication options for specified resource (returns allowed methods).
HEAD	GET without response body (just headers)
PATCH	Apply partial modifications to the resource. (partial PUT)

<u>Example CRUD (Create, Read, Update, Delete)</u>

Verb	Path	Request Body		Response Body
POST	`/students/`	`{“name”: “Alice”}`	→	`{“id”: 1, “name”: “Alice”}`
GET	`/students/1`		→	`{“id”: 1, “name”: “Alice”}`
PUT	`/students/1`	`{“name”: “Ada”}`	→	`{“id”: 1, “name”: “Ada”}`
DELETE	`/students/1`		→	

Response status codes

Code	Meaning	Category
100 Continue	Part of request was received; waiting for rest	Information
101 Switching Protocols	Upgrading, e.g., from HTTP to Websockets	
200 OK	Request was successful; result is in body	
201 Created	New resource was created; link is part of body	Success
204 No Content	Update/delete was successful; nothing to return	
301 Moved Permanently	Resource was moved; see Location header	
303 See Other	Recommends browser to redirect to Location	Redirects
307/308 Redirect	Automatic temporary/permanent redirect	
400 Bad Request	Generic client error (invalid request)	
401 Unauthorized	Missing or invalid credentials	
403 Forbidden	Credentials ok, but insufficient permissions	
404 Not Found	Server could not find the requested resource	
405 Method Not Allowed	Resource exists, but verb is not allowed	Client errors
406 Not Acceptable	Cannot return content suiting Accept-* headers	
409 Conflict	Concurrent modifications	
412 Precondition Failed	Conditions required by, e.g., If-Match do not hold	
500 Internal Server Error	Generic unexpected exception on server	
501 Not Implemented	Requested verb/resource is not implemented (yet)	
502 Bad Gateway	Error in the routing of the request	Server errors
503 Service Unavailable	Server is alive, but too busy to produce a response	
504 Gateway Timeout	Timeout while routing the request	

Examples of content types

`text/plain`
`text/html`
`text/css`
`text/javascript`
`image/png`
`image/svg+xml`
`video/h264`
`font/woff`
`application/xml`
`application/json`
`application/pdf`
`application/zip`

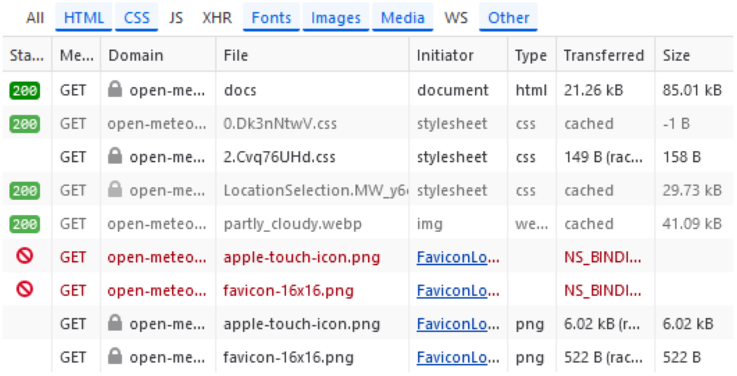
Individual file formats (text or binary)

`application/octet-stream`

Generic binary data

`application/x-www-form-urlencoded`
`multipart/form-data`

Container types for heterogeneous request data (e.g., text + file upload)



All | HTML | CSS | JS | XHR | Fonts | Images | Media | WS | Other

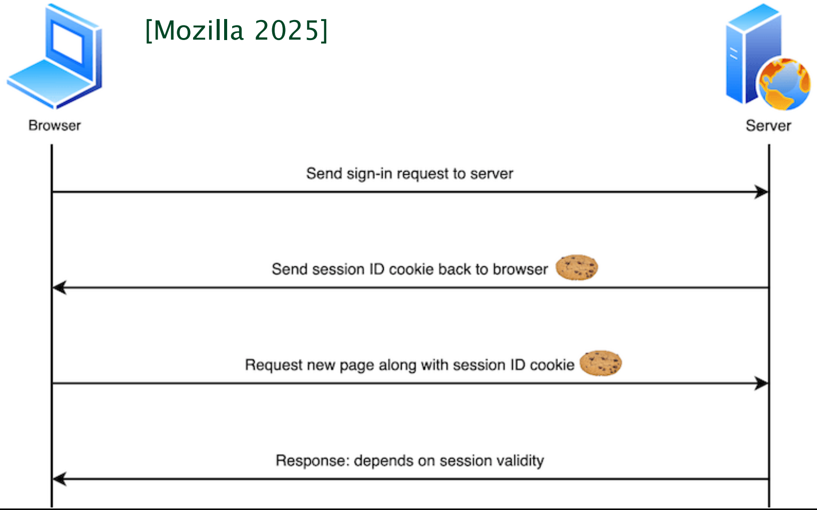
Sta...	Me...	Domain	File	Initiator	Type	Transferred	Size
200	GET	open-me...	docs	document	html	21.26 kB	85.01 kB
200	GET	open-meteo...	0.Dk3nNtwV.css	stylesheet	css	cached	-1 B
	GET	open-me...	2.Cvq76UHd.css	stylesheet	css	149 B (rac...	158 B
200	GET	open-me...	LocationSelection.MW_y6	stylesheet	css	cached	29.73 kB
200	GET	open-meteo...	partly_cloudy.webp	img	we...	cached	41.09 kB
⊘	GET	open-meteo...	apple-touch-icon.png	FaviconLo...		NS_BINDI...	
⊘	GET	open-meteo...	favicon-16x16.png	FaviconLo...		NS_BINDI...	
	GET	open-me...	apple-touch-icon.png	FaviconLo...	png	6.02 kB (r...	6.02 kB
	GET	open-me...	favicon-16x16.png	FaviconLo...	png	522 B (rac...	522 B

Full list see https://www.iana.org/assignments/media-types/media-types.xhtml

HTTP is stateless, but not strictly sessionless.

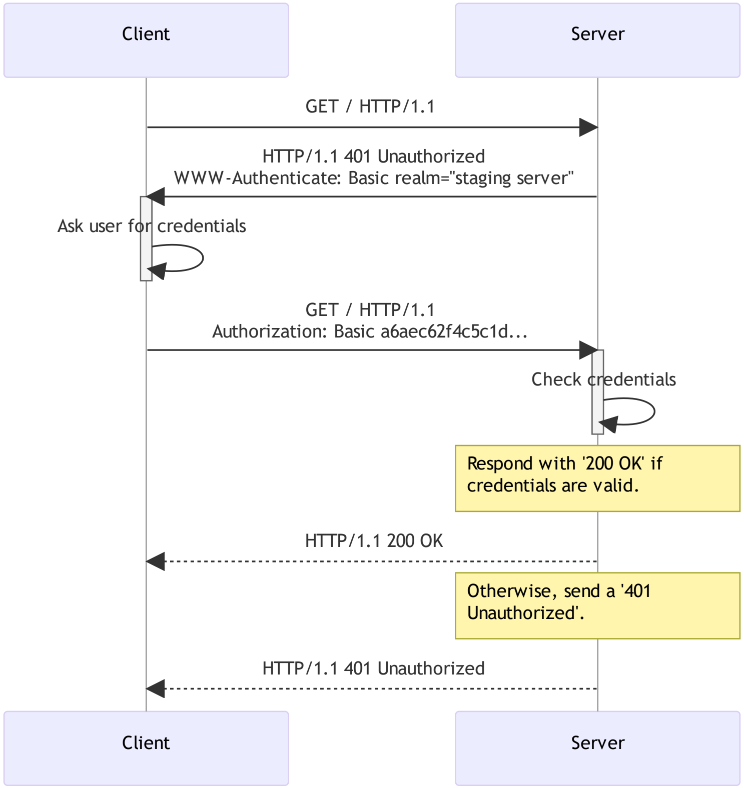
- Cookies store session data relevant for long-running applications.
- Mechanism: Client (=browser) stores a cookies string for every host.
  - Client sends current cookies with every request in `Cookies` header.
  - Server may modify the current cookie string and set it back in `Set-Cookies` header.
- Cookies should be kept minimal.
  - Often used for session ID or short-lived credentials (→)

[Mozilla 2025]


Browser
Server
Send sign-in request to server
Send session ID cookie back to browser
Request new page along with session ID cookie
Response: depends on session validity

HTTP requests may try to access *protected resources.* [Mozilla 2025]

- **Authentication**: User must log-in with valid *credentials* (e.g., username and password)
- **Authorization**: Depending on *roles* and *permissions*, different authenticated users may or may not have access to different resources.
- Based on an `Authorization` request header, the server may decide about both authentication and authorization.
- Different authentication schemes exist. The simplest (and least secure) one is `Basic`.
  - Base64-encoded <username>:<password>
  - E.g., `hello:world` → `Basic aGVsbG86d29ybGQ=`


Client
Server
GET / HTTP/1.1
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="staging server"
Ask user for credentials
GET / HTTP/1.1
Authorization: Basic a6aec62f4c5c1d...
Check credentials
Respond with '200 OK' if credentials are valid.
HTTP/1.1 200 OK
Otherwise, send a '401 Unauthorized'.
HTTP/1.1 401 Unauthorized
Client
Server

OTH REGENSBURG

Representational State Transfer (REST)

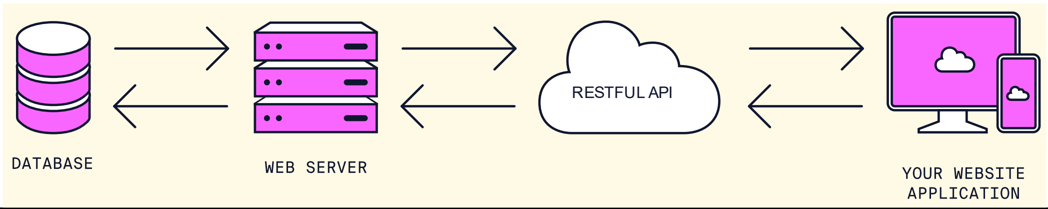
REST is an architectural style for creating web services.

Defined as a set of *constraints* over HTTP: [Fielding 2000]

- ***Stateless***: The server shouldn't hold any information about the client state. (e.g., which view is currently active)
- ***Client-server independence***: Client and server should act independently. Server should not send any information without request from client.
- ***Cacheable***: Resources should be cacheable to improve performance.
- ***Uniform interface***: Same requests from different clients (e.g., browsers, mobile apps)
- ***Layered system***: Components can be added or modified without affecting the service.
- *Code on demand* (optional): Server may send executable code if needed (e.g., JavaScript).

→ REST describes how HTTP *should be* used. REST was a reaction to inefficient usage of HTTP for web services (e.g., verbs were not used in a semantically correct way, causing hidden dependencies, caching and scalability problems).

→ Popular variant of REST architectural style: RESTful API

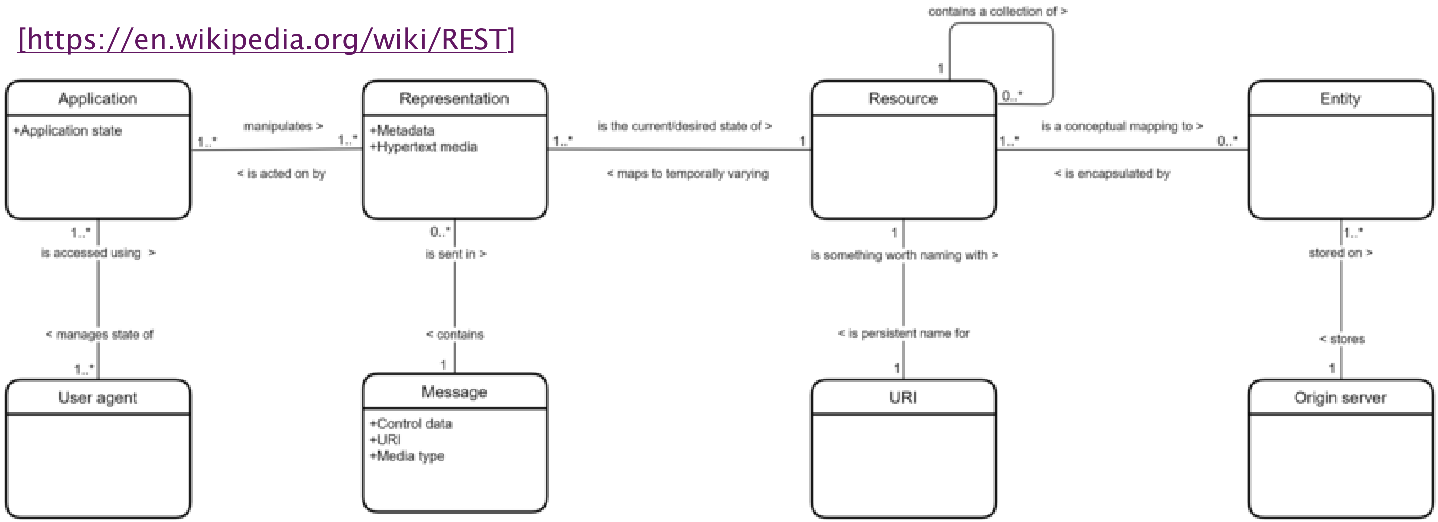


[https://www.codecademy.com/article/what-is-rest]

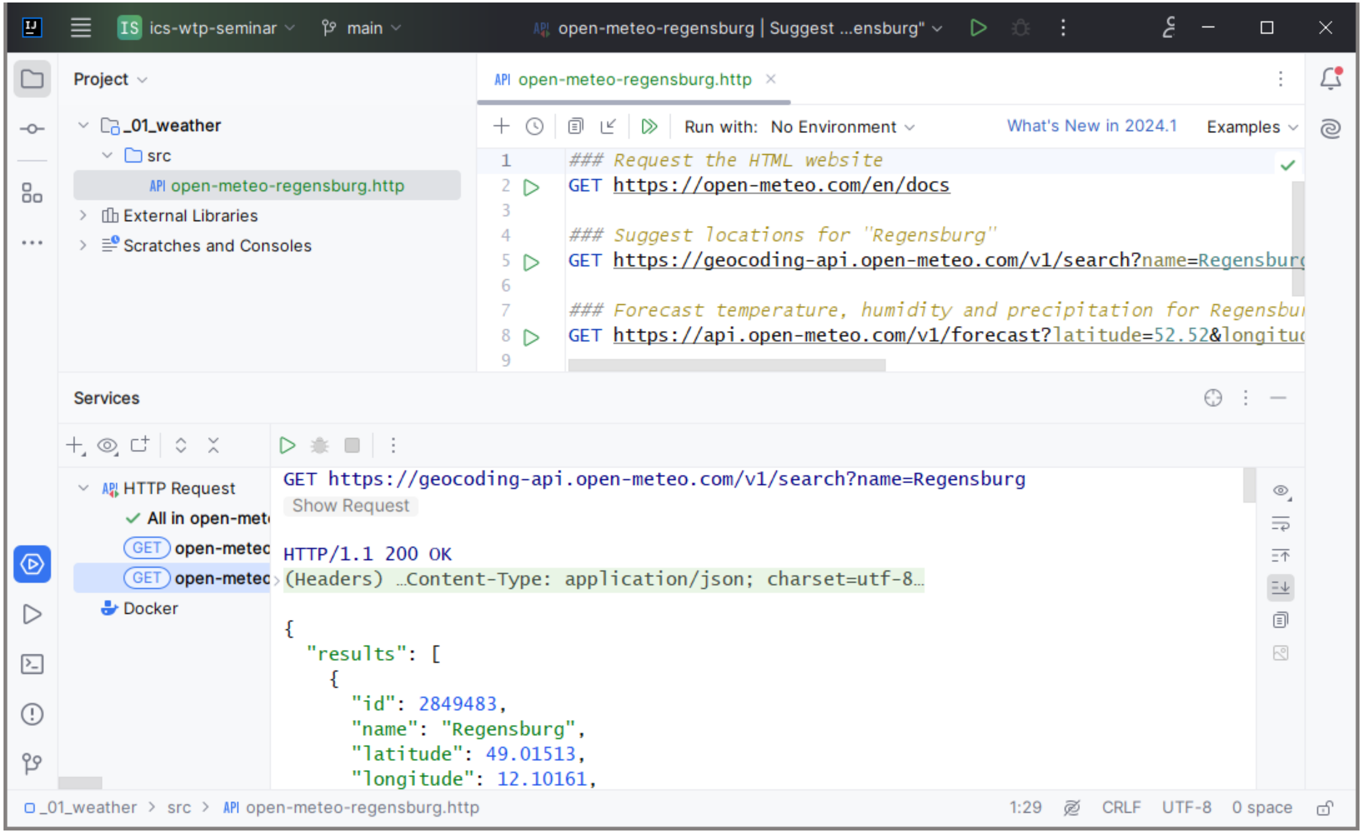
REST: Representations, Resources, and entities

- ***Resource <u>Identification</u>***: Resources should be identified by unique identifiers, e.g., URLs. REST resources should expose easily understood directory-structure-like URLs.
- ***Manipulation through <u>representation</u>***: When making a request to a resource, the server should respond with a representation of the resource. (Typically, JSON or XML)
- ***Self-descriptive <u>messages</u>***: Messages should contain enough information such that the server knowns how to process them. (e.g., using content headers)
- *Hypermedia as the Engine of Application State (HATEOAS, optional)*: Responses should contain links to other areas of the services.

[https://en.wikipedia.org/wiki/REST]



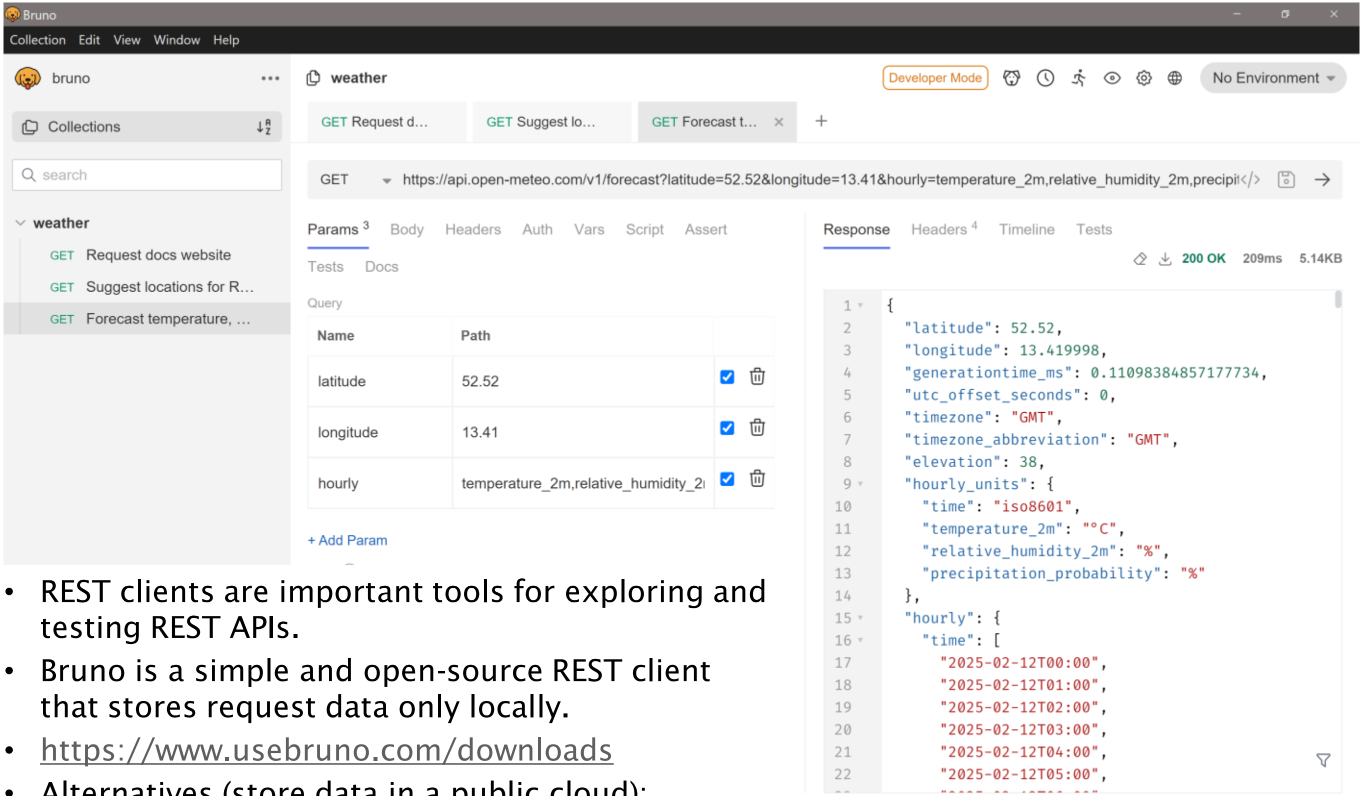
Managing HTTP requests with IntelliJ



```
### Request the HTML website
GET https://open-meteo.com/en/docs

### Suggest locations for "Regensburg"
GET https://geocoding-api.open-meteo.com/v1/search?name=Regensburg

### Forecast temperature, humidity and precipitation for Regensbur
GET https://api.open-meteo.com/v1/forecast?latitude=52.52&longitu
```

```
GET https://geocoding-api.open-meteo.com/v1/search?name=Regensburg

HTTP/1.1 200 OK
(Headers) …Content-Type: application/json; charset=utf-8…

{
  "results": [
    {
      "id": 2849483,
      "name": "Regensburg",
      "latitude": 49.01513,
      "longitude": 12.10161,
```

REST client Bruno



Bruno
Collection Edit View Window Help
bruno
Collections
search
weather
GET Request docs website
GET Suggest locations for R…
GET Forecast temperature, …
weather
Developer Mode
No Environment
GET Request d…
GET Suggest lo…
GET Forecast t…
Params 3 Body Headers Auth Vars Script Assert Tests Docs
Response Headers 4 Timeline Tests
200 OK 209ms 5.14KB
+ Add Param

GET https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&hourly=temperature_2m,relative_humidity_2m,precipit

Query

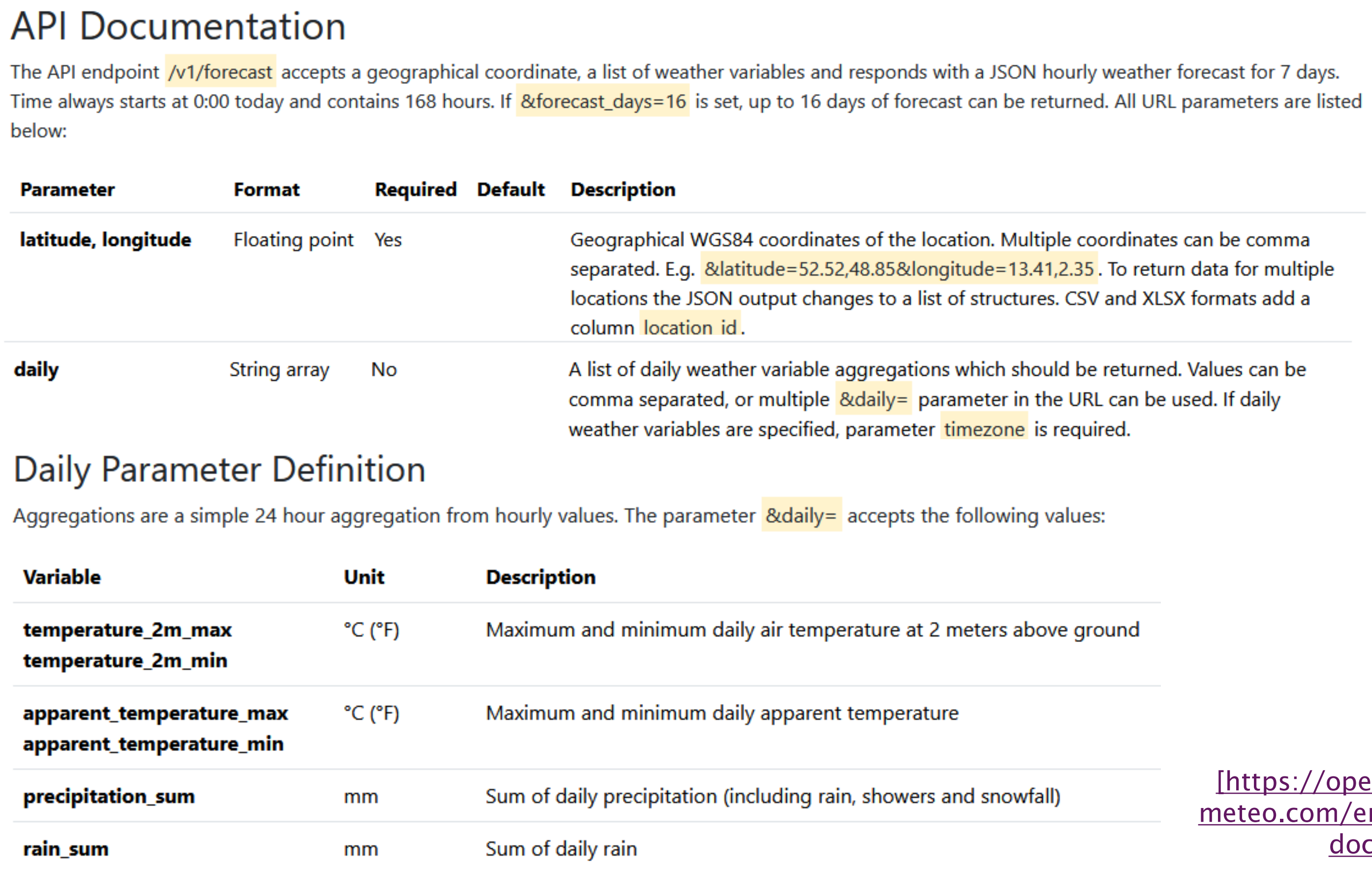
Name	Path
latitude	52.52
longitude	13.41
hourly	temperature_2m,relative_humidity_2

```
{
  "latitude": 52.52,
  "longitude": 13.419998,
  "generationtime_ms": 0.11098384857177734,
  "utc_offset_seconds": 0,
  "timezone": "GMT",
  "timezone_abbreviation": "GMT",
  "elevation": 38,
  "hourly_units": {
    "time": "iso8601",
    "temperature_2m": "°C",
    "relative_humidity_2m": "%",
    "precipitation_probability": "%"
  },
  "hourly": {
    "time": [
      "2025-02-12T00:00",
      "2025-02-12T01:00",
      "2025-02-12T02:00",
      "2025-02-12T03:00",
      "2025-02-12T04:00",
      "2025-02-12T05:00",
```

- REST clients are important tools for exploring and testing REST APIs.
- Bruno is a simple and open-source REST client that stores request data only locally.
- https://www.usebruno.com/downloads
- Alternatives (store data in a public cloud): Postman, Insomnia

OTH
REGENSBURG

APIs should be documented



API Documentation

The API endpoint /v1/forecast accepts a geographical coordinate, a list of weather variables and responds with a JSON hourly weather forecast for 7 days. Time always starts at 0:00 today and contains 168 hours. If &forecast_days=16 is set, up to 16 days of forecast can be returned. All URL parameters are listed below:

Parameter	Format	Required	Default	Description
latitude, longitude	Floating point	Yes		Geographical WGS84 coordinates of the location. Multiple coordinates can be comma separated. E.g. &latitude=52.52,48.85&longitude=13.41,2.35. To return data for multiple locations the JSON output changes to a list of structures. CSV and XLSX formats add a column location_id.
daily	String array	No		A list of daily weather variable aggregations which should be returned. Values can be comma separated, or multiple &daily= parameter in the URL can be used. If daily weather variables are specified, parameter timezone is required.

Daily Parameter Definition

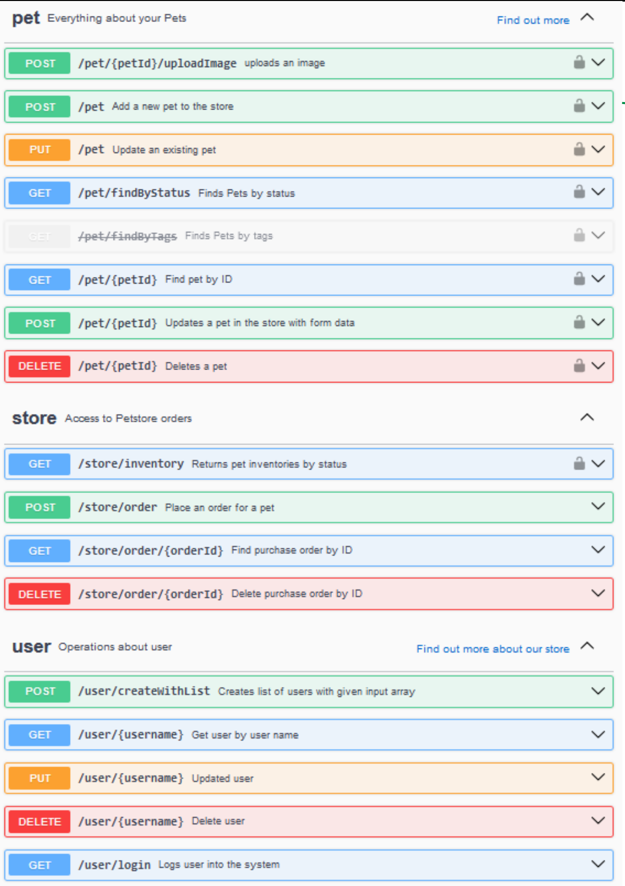
Aggregations are a simple 24 hour aggregation from hourly values. The parameter &daily= accepts the following values:

Variable	Unit	Description
temperature_2m_max **temperature_2m_min**	°C (°F)	Maximum and minimum daily air temperature at 2 meters above ground
apparent_temperature_max **apparent_temperature_min**	°C (°F)	Maximum and minimum daily apparent temperature
precipitation_sum	mm	Sum of daily precipitation (including rain, showers and snowfall)
rain_sum	mm	Sum of daily rain

[https://open-meteo.com/en/docs]

API documentation with Swagger

[https://petstore.swagger.io]
(Authorize with API Key “special-key”)



pet Everything about your Pets — Find out more

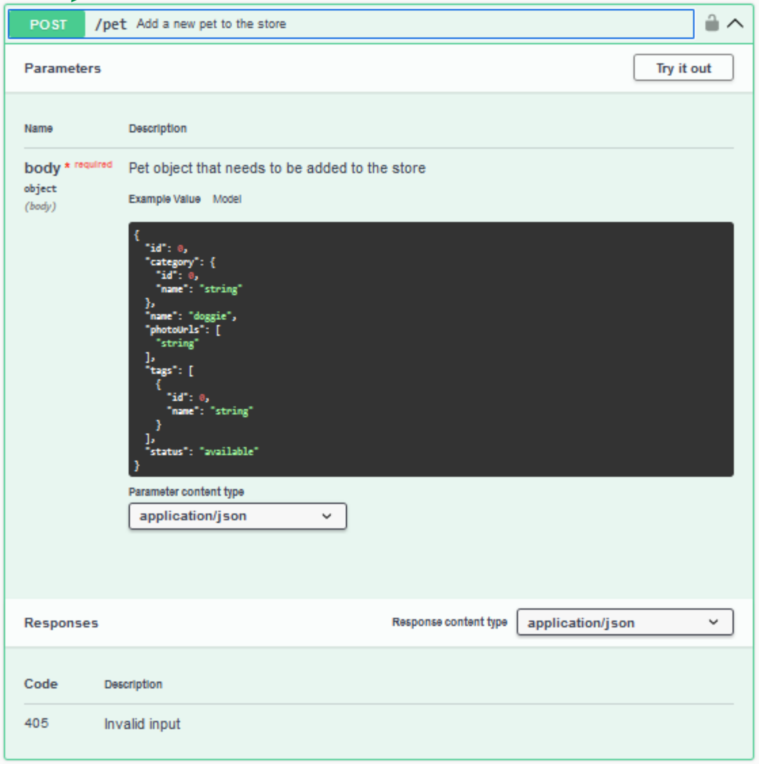
POST	/pet/{petId}/uploadImage	uploads an image
POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
~~GET~~	~~/pet/findByTags~~	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet

store Access to Petstore orders

GET	/store/inventory	Returns pet inventories by status
POST	/store/order	Place an order for a pet
GET	/store/order/{orderId}	Find purchase order by ID
DELETE	/store/order/{orderId}	Delete purchase order by ID

user Operations about user — Find out more about our store

POST	/user/createWithList	Creates list of users with given input array
GET	/user/{username}	Get user by user name
PUT	/user/{username}	Updated user
DELETE	/user/{username}	Delete user
GET	/user/login	Logs user into the system



POST /pet Add a new pet to the store

Parameters — Try it out

Name	Description
body * required object *(body)*	Pet object that needs to be added to the store Example Value Model

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
405	Invalid input

REGENSBURG

OpenAPI documentation format

OpenAPI is the standard for documenting REST APIs.

- Basis: JSON-based API definition
- Swagger interprets this definition for rendering the UI.

```
{
  "host": "petstore.swagger.io",
  "basePath": "/v2",
  "schemes": [ "https", "http" ],
  "paths": {
    "/pet": { "post": {
        "summary": "Add a new pet to the store",
        "operationId": "addPet",
        "consumes": [ "application/json", "application/xml" ],
        "produces": [ "application/json", "application/xml" ],
        "parameters": [{
            "in": "body",
            "name": "body",
            "description": "Pet object added to the store",
            "required": true,
            "schema": { "$ref": "#/definitions/Pet" }
        }],
        "responses": {"405":{ "description":"Invalid input"}}
    }}
  },
  "definitions": {
    "Pet": {
      "type":"object",
      "required":["name","photoUrls"],
      "properties": {
        "id": {"type":"integer", "format":"int64"},
        "category": {"$ref":"#/definitions/Category"},
        "name": {"type":"string","example":"doggie"},
        "photoUrls": {"type":"array","items":{"type":"string"}},
        "tags": {"type":"array", "$ref":"#/definitions/Tag"}
      }
    }
  }
}
```

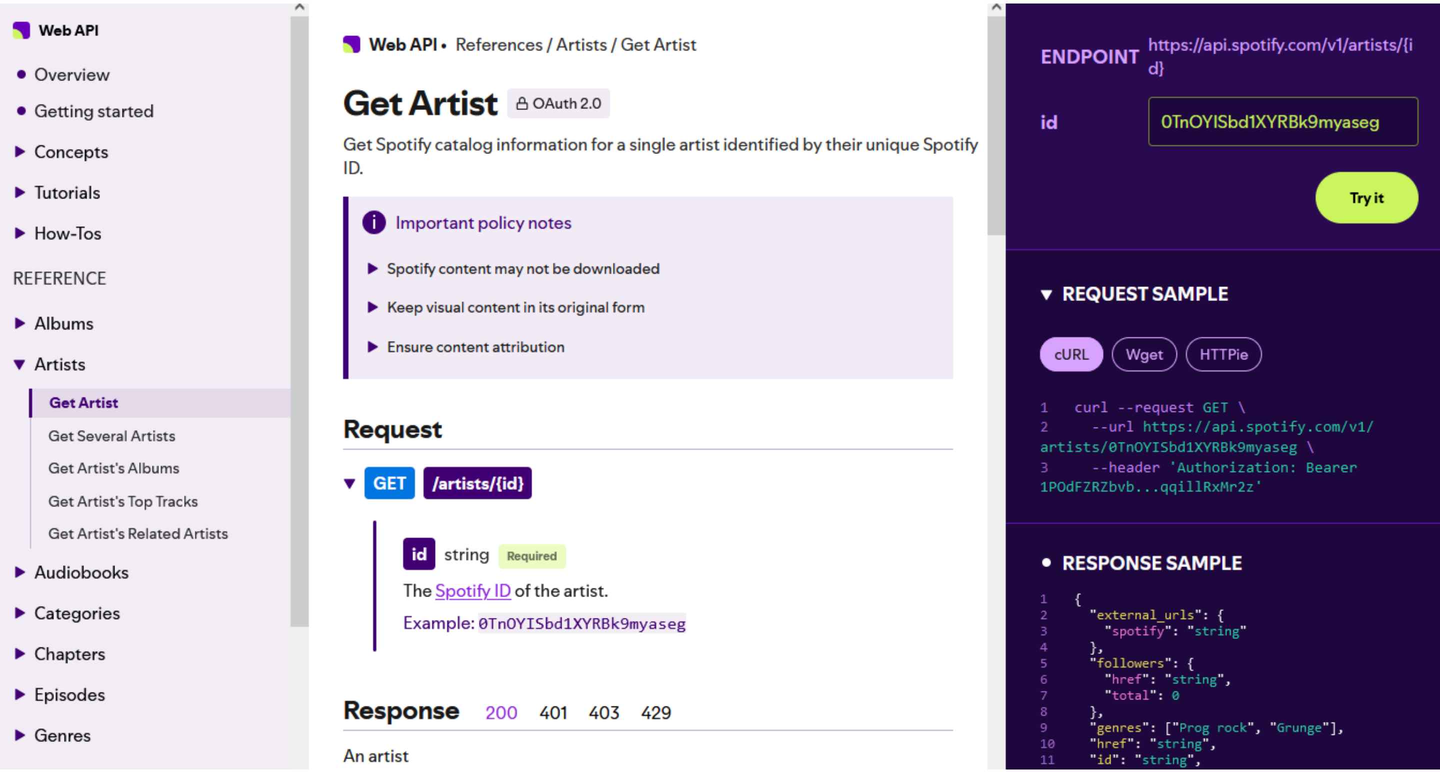
Metadata

Paths (with verbs, content types, possible responses

Schema definition (for representations used in request/response bodies above)

OTH REGENSBURG

A real-world interactive API documentation



Web API • References / Artists / Get Artist

Get Artist OAuth 2.0

Get Spotify catalog information for a single artist identified by their unique Spotify ID.

Important policy notes
- Spotify content may not be downloaded
- Keep visual content in its original form
- Ensure content attribution

Request

GET /artists/{id}

id string Required
The Spotify ID of the artist.
Example: `0TnOYISbd1XYRBk9myaseg`

Response 200 401 403 429

An artist

ENDPOINT https://api.spotify.com/v1/artists/{id}

id 0TnOYISbd1XYRBk9myaseg

Try it

REQUEST SAMPLE

cURL Wget HTTPie

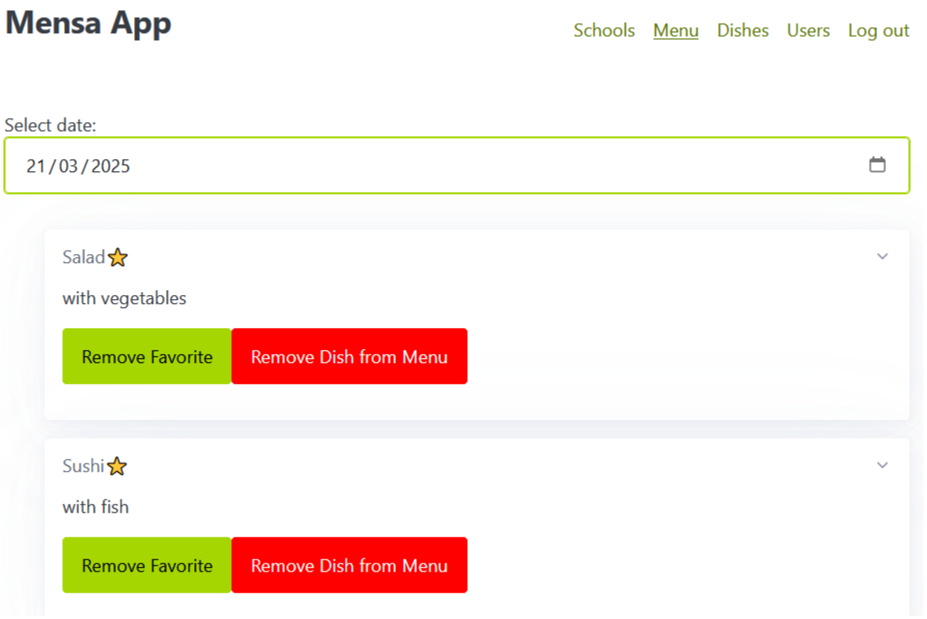
```
curl --request GET \
  --url https://api.spotify.com/v1/artists/0TnOYISbd1XYRBk9myaseg \
  --header 'Authorization: Bearer 1POdFZRZbvb...qqillRxMr2z'
```

RESPONSE SAMPLE

```
{
  "external_urls": {
    "spotify": "string"
  },
  "followers": {
    "href": "string",
    "total": 0
  },
  "genres": ["Prog rock", "Grunge"],
  "href": "string",
  "id": "string",
```

[https://developer.spotify.com/documentation/web-api/reference/get-an-artist]

The Mensa example app



Select date: 21/03/2025

Salad ⭐
with vegetables

Sushi ⭐
with fish

→ Chapter 02: Backend development (Spring Boot)
→ Chapter 06: Frontend development (React)

Live task: Explore the Mensa example app

OTH-LAN connection required (e.g., via VPN)!

- Frontend (based on React):
  http://im-vm-123.hs-regensburg.de
- Swagger API documentation:
  http://im-vm-123.hs-regensburg.de:8080/swagger-ui/index.html

Task: Explore the app! Which *REST calls* are made for UI actions? Check browser console.

- Create an admin and a school.
  - Select the school and create some dishes.
- Create a manager and create more dishes.
  - Create menus for the next few days in your school.
  - Add some dishes as favorites.
- Create a normal user.
  - Add some dishes as favorites and write some comments for existing menus.
- Re-login as admin.
  - Delete a comment.
  - Delete a dish.
  - Delete a user.
  - Delete a school.

References

- [Hinkula 2022] Juha Hinkula: Full Stack Development with Spring Boot 3 and React, Packt, 2022
- [Fielding 2000] Roy Fielding: Architectural Styles and the Design of Network-based Software Architectures, Dissertation, University of California, Irvine, 2000
- [Mozilla 2025] Mozilla Developer Network (MDN): HTTP web docs, https://developer.mozilla.org/en-US/docs/Web/HTTP
- [IETF 2025] Internet Engineering Task Force (IETF): HTTP Semantics (RFC 9110), https://www.rfc-editor.org/rfc/rfc9110.html