ERHVERVSAKADEMI AARHUS
BUSINESS ACADEMY AARHUS

API & REST

Client Server, HTTP, Request & Response,
JSON, CRUD & REST API

JSON

Client Server

Data Exhange

HTTP Request & Response

Communication

HTTP Verbs

REST API

SaaS

PaaS

Cloud Services

CRUD

Dataforsyningen - REST API

One of many REST APIs

https://dataforsyningen.dk/

Dataforsyningen REST API

https://cederdorff.com/city-zip-code/

ERHVERVSAKADEMI AARHUS
BUSINESS ACADEMY AARHUS

How do systems communicate?

1. How do systems exchange information?
2. How does data travel between locations?
3. How can the same data be reused across platforms?
4. How do we integrate external data into our app?
5. What enables two systems to communicate?

Erhvervsakademi Aarhus - E
eaaa.dk
ERHVERVSAKADEMI AARHUS
Uddanne
Skal du vælge videregå
karriere? Tal med

Calendar
eaaa.instructure.com/calendar#view_name=month&view_start=2024-02-11
Account
Admin
Dashboard
Courses
Calendar
Inbox
History
Commons
Student links
Today
February 2024
Week
Month
Agenda
MON
TUE
WED
THU
FRI
SAT
SUN
CALENDARS
Rasmus Cederdorff
AU-VDB E23
DICON-E23 - 2. semester
DIGIMEDIE undervisere
DIKON-E23 - 2. semester
Internship MDC Spring 2024
MDC-E22INT - 4. semester
MDC-E23y - 2. semester
MDU Interaktionsundervisere
MDU praktik forår 2024
MDU-E23c - 1. semester -
UNDATED
Calendar feed

google.com/search?sca_esv=3165b152171cb470&sxsrf=ACQVn0-UN32_PfoMxNEKtBZfCLxW88mkGQ:1707669145453&q=house...
Google
house of vincent ring
Billeder
Videoer
Shopping
Nyheder
Bøger
Finans
Alle filtre
Værktøjer
Beskyttet søgning
Ca. 28.200.000 resultater (0,37 sekunder)
Resultater for Tilst · Vælg område
Sponsoreret
TILBUD
House of Vincent Archaic Shado...
636,65 kr.
Avilé.dk
Gratis forsend...
Fra TopPriser.dk
House of Vincent - Orb Chronicle...
649,00 kr.
Jwlry.dk
Gratis forsend...
Fra WakeupData
House of Vincent Moonlit Secret...
594,15 kr.
Avilé.dk
Gratis forsend...
Fra TopPriser.dk
House of Vincent Candy Rock Ri...
1.499,00 kr.
Me & Eliza
Gratis forsend...
Fra Google
House of Vincent Stolen...
636,65 kr.
Avilé.dk
Gratis forsend...
Fra Google
House of Vincent Candy Rock Sk...
1.274,15 kr.
Avilé.dk
Gratis forsend...
Fra Google
House of Vincent - Stolen...
749,00 kr.
Jwlry.dk
Gratis forsend...
Fra WakeupData
House of Vincent - Eternal Stair...
649,00 kr.
Jwlry
Gratis forsend...
Fra Google
Se billeder
Nordsøen
Danmark
Holland
Berlin
Polen
HOUSE OF VINCENT
Website
Gem
Åbningstider: Åben 24 timer
Foreslå en ændring
Tilføj manglende oplysninger

Payment Gateways

Social Media Integration

Healthcare Integration

Maps and Location Services

Weather Data

IoT (Internet of Things)

API

Cloud Services

News and Content Aggregation

Financial Data

E-commerce Platforms

Twitter API

Stripe API

Microsoft HealthVault API

Google Maps API

API

OpenWeatherMap API

Fitbit API

Amazon S3 API

The New York Times API

Alpha Vantage API

Shopify API

Communication between two systems

Find examples

Client Server

The Basic Architecture of the Web

Client Server Architecture

APP

REST API

DATA SOURCE

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

Web Development

WEB APP

REST API

DATA SOURCE

cederdorff.com/dat-js-crud-intro

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

Webudvikling

FRONTEND

REST API

BACKEND

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

Express

Client-Server Model

Communication between web **clients** and web **servers**.

CLIENT

SERVER

Client-Server Model

Communication between web **clients** and web **servers**.

Browsers
Or any type of
program or device

Cloud Computers
Often computers
in the cloud

CLIENT

SERVER

Client-Server Model

Communication between web **clients** and web **servers**.

Hyper Text Transfer Protocol

- A protocol and standard for fetching data, HTML and other resources (text, images, videos, scripts, JSON).
- The foundation of the web.

What is HTTP

Not Secure | w3schools.com/whatis/whatis_http.asp

HTML CSS JAVASCRIPT SQL PYTHON

HTTP Request / Response

Communication between clients and servers is done by **requests** and **responses**:

1. A client (a browser) sends an **HTTP request** to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an **HTTP response** (output) to the browser
5. The client (the browser) receives the response

The HTTP Request Circle

A typical HTTP request / response circle:

1. The browser requests an HTML page. The server returns an HTML file.
2. The browser requests a style sheet. The server returns a CSS file.
3. The browser requests an JPG image. The server returns a JPG file.
4. The browser requests JavaScript code. The server returns a JS file
5. The browser requests data. The server returns data (in XML or JSON).

https://www.w3schools.com/whatis/whatis_http.asp

How HTTP Works

https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol

Network Tab

Erhvervsakademi Aarhus - EA
eaaa.dk
ERHVERVSAKADEMI AARHUS
Uddannelser til erhvervslivet
Skal du vælge videregående uddannelse? Eller finde det kursus, der løfter din karriere? Tal med os. Vi er et af landets største erhvervsakademier.
Videregående uddannelser
Professionsbachelor- og
Efteruddannelse og kurser
Få ny viden, du kan bruge direkte i
Elements
Console
Sources
Network
3
51
50
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
1000 ms
2000 ms
3000 ms
4000 ms
5000 ms
6000 ms
Name
Method
Status
Type
Initiator
Size
T..
Waterfall
1.gif?dgi=70fb8fdd-4... GET 200 gif uc.js?cbid... (me... 0...
heatmaps.js GET 200 script monsido-s... (disk... 2...
page-correct.js GET 200 script monsido-s... (disk... 2...
?a=h0r3JOS3pvXaDa... GET 200 gif monsido-s... 57 B 1...
favicon.ico GET 200 x-icon Other (disk... 1...
h0r3JOS3pvXaDabSjt... GET 200 xhr heatmaps.... (disk... 0...
h0r3JOS3pvXaDabSjt... GET 200 xhr page-corr... (disk... 0...
cast_sender.js?loadC... GET 200 script vendor.mo... (disk... 0...
1765804027-75aafef... GET 200 avif Other (disk... 0...
master.json?base64_i... GET 200 xhr vendor.mo... 3.0 kB 1...
1765804027-75aafef... GET 200 avif vendor.mo... (me... 0...
cast_framework.js GET 200 script cast_send... 14 B 1...
cast_sender.js GET 200 script cast_send... (disk... 1...
settings.json GET 200 xhr uc.js?cbid... (disk... 0...
widgetIcon.min.js GET 200 script uc.js?cbid... (disk... 0...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.4 kB 2...
d509129e.mp4?r=dX... GET 200 xhr vendor.mo... 264 kB 3...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 4.3 ... 2...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.4 kB 1...
2b51a972.mp4?r=dX... GET 200 xhr vendor.mo... 5.1 kB 2...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 4.6 ... 2...
cf6acf66.mp4?r=dXM... GET 200 xhr vendor.mo... 3.5 ... 3...
collect?v=2&tid=G-D... POST 204 ping js?id=G-D... 17 B 6...
91 requests
12.8 MB transferred
16.2 MB resources
Finish: 5.47 s
DOMContentLoaded: 290 ms

Network Tab

Erhvervsakademi Aarhus - EA
eaaa.dk
ERHVERVSAKADEMI AARHUS
Uddannelser til erhvervslivet
Skal du vælge videregående uddannelse? Eller finde det kursus, der løfter din karriere? Tal med os. Vi er et af landets største erhvervsakademier.
Videregående uddannelser
Efteruddannelse og kurser
Samarbejd og skoler
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
Name
Headers
Payload
Preview
Response
Initiator
Timing
items
wa/
247144?d=eaaa.dk
oembed.json?url...
h0r3JOS3pvXaD...
h0r3JOS3pvXaD...
master.json?base...
2b51a972.mp4?r...
settings.json
d509129e.mp4?r...
cf6acf66.mp4?r=...
2b51a972.mp4?r...
2b51a972.mp4?r...
cf6acf66.mp4?r=...
cf6acf66.mp4?r=...
15 / 111 requests
{type: "video", version: "1.0", provider_name: "Vimeo", provider_u
account_type: "starter"
author_name: "Erhvervsakademi Aarhus"
author_url: "https://vimeo.com/user209157220"
description: ""
duration: 17
height: 240
html: "<div style=\"padding:56.25% 0 0 0;position:relative;\"><if
is_plus: "0"
provider_name: "Vimeo"
provider_url: "https://vimeo.com/"
thumbnail_height: 166
thumbnail_url: "https://i.vimeocdn.com/video/1765804027-75aafef51
thumbnail_url_with_play_button: "https://i.vimeocdn.com/filter/ov
thumbnail_width: 295
title: "Silent video D4"
type: "video"
upload_date: "2023-12-08 06:54:18"
uri: "/videos/892590144"
version: "1.0"
video_id: 892590144
width: 426

Network Tab

Investigating Network Activity in the Browser

Fetch, HTTP Request & Response

WEB APP

REST API

DATA SOURCE

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

Fetch, HTTP Request & Response

WEB APP

CLIENT

FETCH

REQUEST (GET)

JSON DATA

RESPONSE

DATA SOURCE

potters

pokemons

southpark

SERVER

Frontend (client)

JSON Data Source (Server)

https://cederdorff.com/potter-app/

https://raw.githubusercontent.com/cederdorff/dat-js/main/data/potter.json

Web Dev Architecture

Web Dev Architecture

API

Application Programming Interface

API

Application Programming Interface

A way for two systems to communicate - get and post data.

API

Enables communication between different software applications, specifying how they can request and exchange information.

API

- A set of rules and tools that allows different software applications to communicate and interact with each other.
- It defines the methods and data formats that applications can use to request and exchange information.
- APIs are crucial for enabling the integration of different services and systems.

API

API

WEB APP
DATA SOURCES
API
WIKIPEDIA
API
GOOGLE MAPS
API
CVR API
API
SoMe
API
WEATHER API
API
HEADLESS CMS
API
Firebase REST

https://www.youtube.com/watch?v=s7wmiS2mSXY

CLIENT
(WEB APP)
API
DATA SOURCE
RECEIVER
WAITER
(MESSENGER)
KITCHEN
(SOURCE)

API
End User with Browser
Request
Response
API
Server Back-end System
Customer
Make the Order
Delivery of order
Waiter
Take the Order
Bringing from Kitchen
Chef

API

APPLICATION
PROGRAMMING **I**NTERFACE

API'er - definition og funktion.

chat.openai.com

API'er - definition og funktion.

Hvad er en API?

API står for "Application Programming Interface", på dansk "Programmeringsgrænseflade". En API er en samling af software-komponenter, protokoller og værktøjer, der gør det muligt for forskellige programmer eller systemer at kommunikere med hinanden og udveksle data på en standardiseret måde.

API'er kan fungere som en bro mellem forskellige applikationer, platforme eller tjenester, hvilket gør det muligt for udviklere at udnytte funktionalitet eller data fra andre systemer uden at skulle skrive alt koden fra bunden. API'er gør det også muligt at integrere forskellige tjenester og applikationer for at skabe mere komplekse og avancerede systemer.

Der findes mange forskellige typer af API'er, inklusiv web-API'er, biblioteks-API'er og operativsystem-API'er. Web-API'er er en af de mest almindelige typer, og disse gør det muligt for applikationer at kommunikere via internettet ved hjælp af standardiserede protokoller såsom HTTP eller REST.

Send a message...

ChatGPT Mar 23 Version. Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts.

Application Programming Interface

- **API Definition**: Application Programming Interface
- **Purpose**: Facilitates communication between software applications
- **Functionality**: Defines methods and data formats for information exchange
- **Integration**: Enables seamless interaction between different systems
- **Essential for**: Connecting and integrating diverse software services
- **Examples**: RESTful APIs, GraphQL, SOAP
- **Key Actions**: Requesting, sending, and exchanging data
- **Crucial for**: System interoperability and building modular applications

Application Programming Interface

- **Communication Standard**: API sets a standard for how different software components should interact. It’s an interface that allows two systems to communicate and exchange data in a standardised way.
- **Platform Agnostic / Platform independent**: APIs allow applications on different platforms to connect and work together. It can be used by different clients, devices, servers and users: websites, web apps, mobile apps, webshops and other clients
- **Abstraction Layer**: Serves as an abstraction layer, shielding internal complexities and exposing only necessary functionalities.
- **Scalability**: Facilitates scalability by allowing the integration of new features or services.
- **Developer Collaboration**: Enables collaboration between developers, allowing them to leverage each other's work.
- **Documentation Importance**: Well-documented APIs are crucial for developers to understand and use them effectively.
- **Security Measures**: APIs often include security measures like authentication and authorization for secure data exchange.
- **Real-time Interaction**: Supports real-time data transfer, enhancing dynamic and interactive applications.

Application Programming Interface

- **Data Formats**: APIs often use common data formats like JSON or XML for easy comprehension and parsing.
- **HTTP Methods**: Commonly employs HTTP methods such as GET, POST, PUT, and DELETE to perform different actions.
- **Versioning**: APIs may have versioning to manage changes and updates without disrupting existing integrations.
- **Rate Limiting**: Implementing rate limiting helps control the number of requests a client can make within a specified time frame.
- **Feedback Mechanism**: APIs often provide feedback in the form of status codes to indicate the success or failure of a request.
- **Testing and Debugging**: Developers can test and debug applications independently since APIs provide a clear interface.
- **Third-party Integration**: Enables third-party developers to build on top of existing services, fostering a broader ecosystem.
- **RESTful Principles**: RESTful APIs adhere to principles like statelessness and a uniform interface for consistency.

Application Programming Interface

Course roster: WU-E22a - 1. se

https://eaaa.instructure.com/courses/15482/users

Account
Admin
Dashboard
Courses
Calendar
Inbox
History
Commons
Student links

WU-E22a › People

Student view

Home
Announcements
Modules
Assignments
Discussions
People
BigBlueButton
Grades
Pages
Files
Syllabus
Outcomes
Rubrics
Quizzes
Collaborations
Settings

Everyone | Groups

+ Group set

Search people | All roles

+ People

	Name	Login ID	SIS ID	Section	Role	Last Activity	Total Activity
	Clara Juul Birk	eaaclbi@students.eaaa.dk		WU-E22a - 1. semester	Student	24 Aug at 13:16	01:04:21
	Martin Rieper Boesen	eaamrbo@students.eaaa.dk		WU-E22a - 1. semester	Student	24 Aug at 7:54	01:07:06
	Dan Okkels Brendstrup	dob@eaaa.dk		WU-E22a - 1. semester	Teacher	3 Aug at 8:55	
	Rasmus Cederdorff	race@eaaa.dk		WU-E22a - 1. semester	Teacher	25 Aug at 9:28	01:19:23
	Jeffrey David Serio	jds@eaaa.dk		WU-E22a - 1. semester	Teacher	17 Aug at 16:39	
	Chalotte Meng Emanuel Dyrholm	eaacmed@students.eaaa.dk		WU-E22a - 1. semester	Student	23 Aug at 16:59	22:24

Elements | Components | Network | 22

Preserve log | Disable cache | No throttling

Invert | Hide data URLs

All | Fetch/XHR | JS | CSS | Img | Media | Font | Doc | WS | Wasm | Manifest | Other

Has blocked cookies | Blocked Requests | 3rd-party requests

5000 ms | 10000 ms | 15000 ms | 20000 ms | 25000 ms | 30000 ms | 35000 ms

Name | Key | Headers | Payload | Preview | value | Initiator

?sentry_ke...
en-GB-84e...
unread_co...
group_cate...
users?inclu...
unread_co...

6 / 157 requests

[{id: "23974", name: "Clara Juul Birk", created_at: "2020-08-
0: {id: "23974", name: "Clara Juul Birk", created_at: "2020
1: {id: "36267", name: "Martin Rieper Boesen", created_at:
2: {id: "29923", name: "Dan Okkels Brendstrup", created_at:
avatar_url: "https://eaaa.instructure.com/images/thumbnai
created_at: "2021-07-30T00:46:05+02:00"
custom_links: []
email: "dob@eaaa.dk"
enrollments: [{id: "391064", user_id: "29923", course_id:
id: "29923"
integration_id: null
login_id: "dob@eaaa.dk"
name: "Dan Okkels Brendstrup"
short_name: "Dan Okkels Brendstrup (adjunkt – dob@eaaa.dk
sis_user_id: null
sortable_name: "Brendstrup, Dan Okkels"
3: {id: "14427", name: "Rasmus Cederdorff", created_at: "20
4: {id: "41", name: "Jeffrey David Serio", created_at: "201
5: {id: "24043", name: "Chalotte Meng Emanuel Dyrholm", cre
6: {id: "23978", name: "Jeppe Frik", created_at: "2020-08-0
7: {id: "23963", name: "Daniel Tjerrild Gamborg", created_a
8: {id: "23992", name: "Casper Hedegaard Hansen", created_a
9: {id: "36266", name: "Morten Gedsted Hansen", created_at:
10: {id: "23980", name: "Anders Husted", created_at: "2020-
11: {id: "23531", name: "Søren Bo Jørgensen", created_at: "

Canvas

Find API uses to integrate systems.

Application Programming Interface

Communication between two systems

Application Programming Interface

Communication between two systems

Web App

Camera API

Camera

Rasmus Cederdorff - Nybygg

https://www.instagram.com/cederdorff/

cederdorff | Edit profile | View archive

196 posts | 5,374 followers | 808 following

Rasmus Cederdorff - Nybyggerne ham/him

cederdorff

Nybyggerne 2025 – Rødt Hus
Lektor & Web App Udvikler
Aarhus med @matias.vase
Far til Alicia & Ida
Født og... more
cederdorff.com/links

Note...

Nybyggerne | Fuerteventura | Hjem | Vincent | KOS 2024

POSTS | REELS | SAVED | TAGGED

Elements | Console | Sources | Network | Performance | Memory

Preserve log | Disable cache | No throttling

Filter | Invert | More filters

All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | WS | Wasm | Other

10,000 ms | 20,000 ms | 30,000 ms | 40,000 ms | 50,000 ms | 60,000 ms | 70,000 ms | 80,000 ms | 90,000 ms | 100,000 ms | 110,000 ms

Name

- bz?__a=1&_...
- bz?__a=1&_...
- graphql
- query
- query
- query
- query
- query
- bz?__a=1&_...
- bulk-route-d...
- ig_sso_users/
- aIF9XMAVSa...
- v6glBYNK95...
- -l4caGgp-w...
- fH54y0OP2a...
- B1AsC9xG-h...
- main
- query
- query
- query
- graphql/
- graphql/
- graphql/
- graphql

54 / 192 requests

Headers | Payload | Preview | Response | Initiator | Timing | Cookies


```
{,…}
  data: {user: {friendship_status: null, gating: null, is_memorialized: false, is_private: fals
    user: {friendship_status: null, gating: null, is_memorialized: false, is_private: false,…}
      account_badges: []
      account_type: 3
      address_street: ""
      ai_agent_type: null
      bio_links: [{,…}]
      biography: "Nybyggerne 2025 – Rødt Hus\n👷 Lektor & Web App Udvikler\n📍 Aarhus med @matias.
      biography_with_entities: {entities: [{hashtag: null, user: {username: "matias.vase", id:
      category: "Personal blog"
      city_name: ""
      external_lynx_url: "https://l.instagram.com/?u=https%3A%2F%2Fcederdorff.com%2Flinks%3Ffbc
      external_url: "https://cederdorff.com/links"
      fb_profile_bio_link_web: null
      fbid_v2: "17841405578189001"
      follower_count: 5374
      following_count: 808
      friendship_status: null
      full_name: "Rasmus Cederdorff - Nybyggerne"
      gating: null
      has_chaining: true
      has_profile_pic: true
      has_story_archive: true
      hd_profile_pic_url_info: {,…}
      id: "5672009939"
      is_business: false
      is_embeds_disabled: false
      is_memorialized: false
      is_private: false
      is_professional_account: true
      is_regulated_c18: false
      is_unpublished: false
```

Rasmus Cederdorff - Nybygg

https://www.instagram.com/p/DGBr7EbOyh9/?img_index=2

cederdorff | Edit profile | View archive

196 posts | 5,374 followers | 808 following

Rasmus Cederdorff - Nybyggerne ham/him

cederdorff

Nybyggerne 2025 – Rødt Hus
Lektor & Web App Udvikler
Aarhus med @matias.vase
Far til Alicia & Ida
Født og... more
cederdorff.com/links

matias.vase and cederdorff

matias.vase Vi malede alt mintgrønt og syede et par matchende gardiner – den røde fuge blev sgu da også meget god! Første ud af ti
5d See translation

For you

af.jost Så fucking flot!
5d 2 likes Reply See translation

View replies (3)

Liked by aniacederdorff and others
5 days ago

Add a comment... Post

Elements | Console | Sources | Network | Performance | Memory

1 | 18 | 19

Preserve log | Disable cache | No throttling

Filter | Invert | More filters

All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | WS | Wasm | Other

500 ms | 1,000 ms | 1,500 ms | 2,000 ms | 2,500 ms | 3,000 ms | 3,500 ms

Name

- comments/?...
- logging_clie...
- query
- query
- query

5 / 61 requests

Headers | Payload | Preview | Response | Initiator | Timing | Cookies


```
{can_view_more_preview_comments: false,…}
  can_view_more_preview_comments: false
  caption: {pk: "18089478400497019", user_id: "603311262", type: 1, did_report_as_spam: false,…
    bit_flags: 0
    content_type: "comment"
    created_at: 1739478368
    created_at_for_fb_app: 1739478368
    created_at_utc: 1739478368
    did_report_as_spam: false
    has_translation: true
    is_covered: false
    is_created_by_media_owner: true
    is_ranked_comment: true
    media_id: "3567325555055077501"
    pk: "18089478400497019"
    private_reply_status: 0
    share_enabled: true
    status: "Active"
    strong_id__: "18089478400497019"
    text: "Vi malede alt mintgrønt og syede et par matchende gardiner – den røde fuge blev sgu
    type: 1
    user: {pk: "603311262", pk_id: "603311262", id: "603311262", username: "matias.vase",…}
    user_id: "603311262"
  caption_is_edited: false
  comment_count: 69
  comment_cover_pos: "bottom"
  comment_filter_param: "no_filter"
  comment_likes_enabled: true
  comments: [{pk: "18085219642597053", user_id: "603763003", type: 0, did_report_as_spam: false
    0: {pk: "18085219642597053", user_id: "603763003", type: 0, did_report_as_spam: false,…}
    1: {pk: "18025521470358245", user_id: "5633114583", type: 0, did_report_as_spam: false,…}
    2: {pk: "18039454121593713", user_id: "7232374367", type: 0, did_report_as_spam: false,…}
    3: {pk: "18042334928366268", user_id: "1638175743", type: 0, did_report_as_spam: false,…}
```

Rasmus Cederdorff - Nybygg

https://www.instagram.com/p/DGBr7EbOyh9/?img_index=2

cederdorff | Edit profile | View archive

196 posts | 5,374 followers | 808 following

Rasmus Cederdorff - Nybyggerne ham/him

cederdorff

Nybyggerne 2025 – Rødt Hus
Lektor & Web App Udvikler
Aarhus med @matias.vase
Far til Alicia & Ida
Født og... more
cederdorff.com/links

matias.vase and cederdorff

matias.vase Vi malede alt mintgrønt og syede et par matchende gardiner - den røde fuge blev sgu da også meget god! Første ud af ti
5d See translation

For you

af.jost Så fucking flot!
5d 2 likes Reply See translation

View replies (3)

Liked by aniacederdorff and others
5 days ago

Add a comment... Post

Elements | Console | Sources | Network | Performance | Memory | Application | Security

Preserve log | Disable cache | No throttling

Filter | Invert | More filters | All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | WS | Wasm | Other

Name
comments/?can_...
logging_client_e...
query
query
query

Headers | Payload | Preview | Response | Initiator | Timing | Cookies


```
comment_likes_enabled: true
comments: [{pk: "18085219642597053", user_id: "603763003", type: 0, did_report_as_spam: false,…},…]
0: {pk: "18085219642597053", user_id: "603763003", type: 0, did_report_as_spam: false,…}
bit_flags: 0
child_comment_count: 3
comment_index: 0
comment_like_count: 2
content_type: "comment"
created_at: 1739479790
created_at_for_fb_app: 1739479790
created_at_utc: 1739479790
did_report_as_spam: false
has_disliked_comment: false
has_liked_comment: true
has_more_head_child_comments: false
has_more_tail_child_comments: true
has_translation: true
inline_composer_display_condition: "never"
is_covered: false
is_liked_by_media_owner: true
is_ranked_comment: true
media_id: "3567325555055077501"
next_max_child_cursor: "QVFCcHB2WjBjOUtobklvMHFScGJrZkNEYWJ6M1l6am5uLUpJbVZkSDBCVmNhc3k0RHBUUXNjU2EwakFrUTQwa
num_tail_child_comments: 2
other_preview_users: []
pk: "18085219642597053"
preview_child_comments: [{pk: "18134926063385737", user_id: "5672009939", type: 2, did_report_as_spam: false,
private_reply_status: 0
share_enabled: true
status: "Active"
strong_id__: "18085219642597053"
text: "Så fucking flot! "
type: 0
user: {pk: "603763003", pk_id: "603763003", id: "603763003", username: "af.jost",…}
user_id: "603763003"
1: {pk: "18025521470358245", user_id: "5633114583", type: 0, did_report_as_spam: false,…}
2: {pk: "18039454121593713", user_id: "7232374367", type: 0, did_report_as_spam: false,…}
3: {pk: "18042334928366268", user_id: "1638175743", type: 0, did_report_as_spam: false,…}
4: {pk: "18058454794955163", user_id: "3028542997", type: 0, did_report_as_spam: false,…}
5: {pk: "18059270978058735", user_id: "39137901", type: 0, did_report_as_spam: false,…}
6: {pk: "18080350462623617", user_id: "455050897", type: 0, did_report_as_spam: false,…}
7: {pk: "18055910128869143", user_id: "60029610162", type: 0, did_report_as_spam: false,…}
8: {pk: "18052344175933631", user_id: "7182529838", type: 0, did_report_as_spam: false,…}
9: {pk: "18060586327980616", user_id: "201958804", type: 0, did_report_as_spam: false,…}
```


5 / 63 requests | .0 k

API

More Examples: https://race.notion.site/API-Examples-48baf0ecb20f490d916edd27f70204ed

JSON

Java**S**cript **O**bject **N**otation

JSON

Java**S**cript **O**bject **N**otation

... a syntax for storing & exchanging data over the web

https://www.instagram.com/p/CVqbCzgsZUF/

JSON (& API) is the glue

APP

API

DATA SOURCE

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

JSON

... a syntax for storing and exchanging data over the web

```
{
  "name": "Alicia",
  "age": 6
}
```

JSON OBJECT

```
[{
  "name": "Alicia",
  "age": 6
}, {
  "name": "Peter",
  "age": 22
}]
```

LIST OF JSON OBJECTS

JSON

https://raw.githubusercontent.c

raw.githubusercontent.com/cederdorff/dat-js/m...

Raw | Parsed


```
{
    "name": "Harry Potter",
    "species": "human",
    "gender": "male",
    "house": "Gryffindor",
    "dateOfBirth": "31-07-1980",
    "yearOfBirth": 1980,
    "ancestry": "half-blood",
    "eyeColour": "green",
    "hairColour": "black",
    "wand": "holly,phoenix feather,11",
    "patronus": "stag",
    "hogwartsStudent": true,
    "hogwartsStaff": false,
    "actor": "Daniel Radcliffe",
    "alive": true,
    "image": "http://hp-api.herokuapp.com/images/harry.jpg"
}
```


JSON OBJECT

https://raw.githubusercontent.c

raw.githubuser...

Raw | Parsed


```
[
    {
        "name": "Harry Potter",
        "species": "human",
        "gender": "male",
        "house": "Gryffindor",
        "dateOfBirth": "31-07-1980",
        "yearOfBirth": 1980,
        "ancestry": "half-blood",
        "eyeColour": "green",
        "hairColour": "black",
        "wand": "holly,phoenix feather,11",
        "patronus": "stag",
        "hogwartsStudent": true,
        "hogwartsStaff": false,
        "actor": "Daniel Radcliffe",
        "alive": true,
        "image": "http://hp-api.herokuapp.com/images/harry.jpg"
    },
    {
        "name": "Hermione Granger",
        "species": "human",
        "gender": "female",
        "house": "Gryffindor",
        "dateOfBirth": "19-09-1979",
        "yearOfBirth": 1979,
        "ancestry": "muggleborn",
        "eyeColour": "brown",
        "hairColour": "brown",
        "wand": "vine,dragon heartstring",
        "patronus": "otter",
        "hogwartsStudent": true,
        "hogwartsStaff": false,
        "actor": "Emma Watson",
        "alive": true,
        "image": "http://hp-api.herokuapp.com/images/hermione.jpeg"
    },
    {
        "name": "Ron Weasley",
        "species": "human",
        "gender": "male",
        "house": "Gryffindor",
        "dateOfBirth": "01-03-1980",
        "yearOfBirth": 1980,
        "ancestry": "pure-blood",
        "eyeColour": "blue",
        "hairColour": "red",
        "wand": "willow,unicorn tail-hair,14",
```


LIST OF JSON OBJECTS

JSON Object

https://raw.githubusercontent.com/cederdorff/dat-js/main/data/harry.json

JSON Object

app.js — potter-app

app.js M

app.js > showCharacterModal

```
28
29  function showCharacter(character) {
30      document.querySelector("#characters").insertAdjacentHTML(
31          "beforeend",
32          /*html*/ `
33              <article class="grid-item">
34                  <img src="${character.image}">
35                  <h2>${character.name}</h2>
36                  <p>${character.house}</p>
37              </article>
38          `
39      );
40  }
```

grid* 0 0 Ln 48, Col 19 Spaces: 4 UTF-8 LF JavaScript Port : 5500 Prettier

https://raw.githubusercontent.c

raw.githubusercontent.com/cederdorff/dat-js/m...

Raw Parsed


```
{
    "name": "Harry Potter",
    "species": "human",
    "gender": "male",
    "house": "Gryffindor",
    "dateOfBirth": "31-07-1980",
    "yearOfBirth": 1980,
    "ancestry": "half-blood",
    "eyeColour": "green",
    "hairColour": "black",
    "wand": "holly,phoenix feather,11",
    "patronus": "stag",
    "hogwartsStudent": true,
    "hogwartsStaff": false,
    "actor": "Daniel Radcliffe",
    "alive": true,
    "image": "http://hp-api.herokuapp.com/images/harry.jpg"
}
```


https://raw.githubusercontent.com/cederdorff/dat-js/main/data/harry.json

JAVASCRIPT OBJECT NOTATION

- Collection of key-value pair: “`key”: “value”`
- List of values, collections or objects
- Lightweight data-interchange format
- Syntax / text format for storing and exchanging data over the web
- Human and machine readable **text**: small, fast and simple
- Language independent
- Can be parsed directly to JavaScript Object
- JavaScript Objects can be converted directly to JSON
- The glue between programs (interface between frontend and backend)


```
[
    {
        "id": "1",
        "firstname": "Kasper",
        "lastname": "Topp",
        "age": "34",
        "haircolor": "Dark Blonde",
        "countryName": "Denmark",
        "gender": "Male",
        "lookingFor": "Female"
    },
    {
        "id": "2",
        "firstname": "Nicklas",
        "lastname": "Andersen",
        "age": "22",
        "haircolor": "Brown",
        "countryName": "Denmark",
        "gender": "Male",
        "lookingFor": "Female"
    },
    {
        "id": "3",
        "firstname": " Sarah",
        "lastname": "Dybvad ",
        "age": "34",
        "haircolor": "Blonde",
        "countryName": "Denmark",
        "gender": "Female",
        "lookingFor": "Male"
    },
    {
        "id": "4",
        "firstname": "Alex",
```

JSON METHODS

```
const user = {
  name: "John",
  age: 30,
  gender: "male",
  lookingFor: "female"
};

// === JSON.stringify === //
const jsonUser = JSON.stringify(user);
console.log(jsonUser); // {"name":"John","age":30,"gender":"male","lookingFor":"female"}

// === JSON.parse === //
const jsonString = '{"name":"John","age":30,"gender":"male","lookingFor":"female"}';
const userObject = JSON.parse(jsonString);
console.log(userObject); // logging userObject
```

CRUD App

127.0.0.1:5500/index.html

EAAA | Canvas | KEA | BOOKS | Code & Templates | Summer School -...

CRUD App

Rasmus Cederdorff

Senior Lecturer
race@dev.dk

UPDATE DELETE

Lars Bogetoft

Head of Education
larb@eaaa.dk

UPDATE DELETE

Edith Terte

Lecturer
edan@kea.dk

UPDATE DELETE

Frederikke Bender

Head of Education
fbe@kea.dk

UPDATE DELETE

Murat Kilic

Senior Lecturer
mki@eaaa.dk

UPDATE DELETE

Anne Andersen

Head of Education
anki@mail.dk

UPDATE DELETE

Elements | Console | Sources | Network | Performance | Memory | Application

Preserve log | Disable cache | No throttling

Filter | Invert | Hide data URLs | Hide extension URLs

All | Doc | JS | Fetch/XHR | CSS | Font | Img | Media | Manifest | WS | Wasm | Other | Blocked response cookies

Blocked requests | 3rd-party requests

10 ms | 20 ms | 30 ms | 40 ms | 50 ms | 60 ms | 70 ms | 80 ms | 90 ms | 100 ms | 110

Name | Headers | Preview | Response | Initiator | Timing

users


```
[
    {
        "id": 2,
        "name": "Rasmus Cederdorff",
        "mail": "race@dev.dk",
        "title": "Senior Lecturer",
        "image": "https://share.cederdorff.com/images/race.jpg"
    },
    {
        "id": 3,
        "name": "Lars Bogetoft",
        "mail": "larb@eaaa.dk",
        "title": "Head of Education",
        "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/larb.jpg"
    },
    {
        "id": 4,
        "name": "Edith Terte",
        "mail": "edan@kea.dk",
        "title": "Lecturer",
        "image": "https://kea.dk/slir/w180-c1x1/images/user-profile/synced/EDAN.jpg"
    },
    {
        "id": 5,
        "name": "Frederikke Bender",
        "mail": "fbe@kea.dk",
        "title": "Head of Education",
        "image": "https://kea.dk/slir/w200-c1x1/images/user-profile/chefer/fbe.jpg"
    },
    {
        "id": 6,
        "name": "Murat Kilic",
        "mail": "mki@eaaa.dk",
        "title": "Senior Lecturer",
        "image": "https://www.eaaa.dk/media/llyavasj/murat-kilic.jpg?width=800&heigh
    },
    {
        "id": 7,
        "name": "Anne Andersen",
        "mail": "anki@mail.dk",
        "title": "Head of Education",
        "image": "https://www.eaaa.dk/media/5buh1xeo/anne-kirketerp.jpg?width=800&he
    }
]
```


1 / 11 requests | 21

https://jsonplaceholder.typicode.com/

Test endpoints

- Install JSON Formatter
- Test the following endpoints:
  - https://jsonplaceholder.typicode.com/posts/
  - https://jsonplaceholder.typicode.com/posts/5
  - https://jsonplaceholder.typicode.com/users/
  - https://jsonplaceholder.typicode.com/users/1
  - https://jsonplaceholder.typicode.com/todos
  - https://jsonplaceholder.typicode.com/todos/5

```json
{
  "name": "Rasmus Cederdorff",
  "birthday": "1990-03-12",
  "title": "Experienced JavaScript Developer",
  "experienceYears": 10,
  "education": {
    "degree": "Master of Science in IT – Web Communication",
    "specialization": "Web Architecture"
  },
  "skills": ["JavaScript", "React", "Node.js", "UI/UX Design"],
  "currentPosition": "Senior Lecturer at EAAA",
  "teachingSubjects": ["Web Development", "JavaScript", "React", "BaaS & Node.js"],
  "lovesJavaScript": true
}
```

Define yourself with JSON

- https://race.notion.site/Define-yourself-with-JSON-391944b6b1a041bfa341a0d46f5c0e4a?pvs=4

CRUD

Create - **R**ead - **U**pdate - **D**elete

What's CRUD?

- CREATE objects like a post, user, movie, product, etc.
- READ objects like an array (or object) of objects (posts, users, movies, products, etc)
- UPDATE an object, often given by a unique id.
- DELETE an object, often given by a unique id.

codecademy.com/article/what-is-crud

CRUD

Create - **R**ead - **U**pdate - **D**elete

Find App Examples

REST API Design & CRUD

API Design 101: From Basics to Best Practices

REST

Representational **S**tate **T**ransfer

REST

REST (Representational State Transfer) is an architectural style for web services based on HTTP and stateless interactions.

REST

A set of architectural **principles** of implementing an API.

REST

A standard for systems (client & server) to communicate over HTTP.

REST

Key Principles

1. Stateless Communication
2. CRUD and HTTP Methods
3. Resource-Based Design
4. Using HTTP Status Codes
5. JSON as the Standard Format
6. Client-Server Architecture

codecademy.com/article/what-is-rest

1. Stateless Communication

Each client request to the server contains all necessary information – the server does not store client state between requests.

2. CRUD and HTTP Methods

- RESTful APIs follow the CRUD principle using HTTP methods

race — -zsh — 65×32

```
GET → Retrieve data

POST → Create data

PUT/PATCH → Update data

DELETE → Remove data
```

3. Resource-Based Design

- In REST, everything is considered a resource.
- REST APIs are structured around uniquely identifiable resources, accessed via specific URLs.
- Each resource is represented as a URI and can be manipulated using standard HTTP methods.GET, POST, PUT, DELETE).

race — -zsh — 65×32

```
GET /users → Retrieves a list of all users

GET /users/123 → Retrieves user with ID 123

POST /users → Creates a new user

PUT /users/123 → Updates user with ID 123

PATCH /users/123 → Partially updates user with ID 123

DELETE /users/123 → Deletes user with ID 123
```

3. Resource-Based Design

- **Resources and their Endpoints**
  - Products: `http://example.com/products`
  - Users: `http://example.com/users`
  - Orders: `http://domain.com/orders`
- **Resource Identifiers (IDs)**
  - Specific Product: `http://example.com/products/2/25`
  - Specific User: `http://example.com/users/7503`
  - Specific Order: `http://domain.com/orders/2014/06/4022`
- **Connected Data**
  - User Order Item: `http://example.com/users/7503/orders/3/item/1`

Clean snd Semantic URL

race — -zsh — 65×32

http://example.com/products/2/25

versus

http://example.com/products?category=2&id=25

4. Using HTTP Status Codes

- RESTful APIs use standard HTTP status codes to indicate the outcome of a request

race — -zsh — 65×32

```
200 OK → Success

201 Created → Resource created

400 Bad Request → Invalid request

404 Not Found → Resource not found

500 Internal Server Error → Server error
```

HTTP response status codes

RESTful API

- Base URL: http://some-url.com/products

domain ressource specific element

- http://some-url.com/products/product1

- Data type —> JSON

Ressource	GET	POST	PUT	DELETE
Collection: http://some-url.com/products	Returns a list with all products	Creates new product, added to the collections	Replaces a collection with a another	Deletes all products
Element: http://some-url.com/products/product1	Returns a specific product	÷	Replaces product with new (updated) data	Deletes product

5. JSON as the Standard Format

REST APIs typically use JSON as the data format for efficient and readable communication between client and server.

Collections (JSON Array)

localhost:3000/users

localhost:3000/users

Raw | Parsed


```
[
  {
    "id": 1,
    "name": "Birgitte Kirk Iversen",
    "mail": "bki@mail.dk",
    "title": "Senior Lecturer",
    "image": "https://www.baaa.dk/media/u4gorzsd/birgitte-kirk-iversen2.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921541630000&format=webp"
  },
  {
    "id": 4,
    "name": "Rasmus Cederdorff",
    "mail": "race@mail.dk",
    "title": "Senior Lecturer",
    "image": "https://www.baaa.dk/media/devlvvgj/rasmus-cederdorff.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921695570000&format=webp"
  },
  {
    "id": 5,
    "name": "Dan Okkels Brendstrup",
    "mail": "dob@mail.dk",
    "title": "Lecturer",
    "image": "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921559630000&format=webp"
  },
  {
    "id": 6,
    "name": "Kasper Fischer Topp",
    "mail": "kato@mail.dk",
    "title": "Lecturer",
    "image": "https://www.eaaa.dk/media/lxzcybme/kasper-topp.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921618200000&format=webp"
  },
  {
    "id": 7,
    "name": "Line Skjødt",
    "mail": "lskj@mail.dk",
    "title": "Senior Lecturer & Internship Coordinator",
    "image": "https://www.eaaa.dk/media/14qpfeq4/line-skj%C3%B8dt.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921638700000&format=webp"
  },
  {
    "id": 8,
    "name": "Martin Aagaard Nøhr",
    "mail": "mnor@mail.dk",
    "title": "Lecturer",
    "image": "https://www.eaaa.dk/media/oayjq02h/martin-n%C3%B8hr.jpg?
    anchor=center&mode=crop&width=800&height=450&rnd=132792921658800000&format=webp"
```


Element (JSON Object)

localhost:3000/users/4

localhost:3000/users/4

Raw | Parsed


```
{
  "id": 4,
  "name": "Rasmus Cederdorff",
  "mail": "race@mail.dk",
  "title": "Senior Lecturer",
  "image": "https://www.baaa.dk/media/devlvvgj/rasmus-cederdorff.jpg?
  anchor=center&mode=crop&width=800&height=450&rnd=132792921695570000&format=webp"
}
```

6. Client-Server Architecture

- **Client-Server Architecture**: Separates the client and server responsibilities, enabling them to evolve independently. Clients are not concerned with data storage, while servers are not concerned with user interfaces.
- **Uniform Interface**: Defines a standardized way for clients to interact with resources. This includes resource identification through URIs, manipulation through representations, and self-descriptive messages.

https://www.youtube.com/watch?v=-MTSQjw5DrM&ab_channel=Fireship

GraphQL - REST Alternative

https://www.youtube.com/watch?v=eIQh02xuVw4&ab_channel=Fireship

What are REST APIs?

https://www.youtube.com/watch?v=SLwpqD8n3d0

Advantages of REST

- By following these REST principles, a REST API provides a simple and consistent way to interact with a database or other services. It allows developers to perform common operations using familiar HTTP methods, making it easier to integrate and interact with the database from different programming languages or platforms.

- It enables developers to perform CRUD (Create, Read, Update, Delete) operations on data resources using standard HTTP methods and benefit from the flexibility and interoperability that REST offers.

Advantages of REST

- Independent of platform and programming language
- Based on existing standards (on top of HTTP)
- Semantic URL —> Nice and clean URLs —> SEO
- Restful API
- Scalable
- Performance
- Exchange formats like JSON, XML, or both

Dataforsyningen - REST API

Why is it a REST API?

https://dataforsyningen.dk/

Dataforsyningen REST API

10 Best Practices for REST API Design

1. Data Format: Use JSON for Data Exchange
2. Endpoint Design: Use nouns, not verbs in endpoint paths
3. Resource Naming: Pluralize Collection Names
4. Error Handling: Utilize HTTP Status Codes & JSON-based error responses
5. Resource Relationships: Employ Nested Endpoints
6. Data Retrieval: Implement Filtering, Sorting, and Pagination
7. Security: Enforce SSL/TLS Encryption
8. Versioning: Clearly Define API Versions
9. Caching: Provide data caching to improve performance
10. Documentation: Provide Comprehensive API Docs

https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/#h-cache-data-to-improve-performance

https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/

API Design 101: From Basics to Best Practices

Relationships in API Design

API Design 101: From Basics to Best Practices

Queries, Limit, and Idempotence of GET Requests

Idempotent

NEVER mutate data

GET /api/products?limit=100&offset=0

GET /api/products?startDate={date}&endDate={date}

Backward Compatibility and Versioning

API Design 101: From Basics to Best Practices

DOs & DON’Ts

1. Let’s just use POST for everything!
2. While we’re at it…
3. Let’s use PUT… What do you mean “404”?!
4. Everything is “OK”
5. Server: “400 Bad Request. Body: Empty…” — Client: “Thanks…”
6. “400 Bad Request. Body: Some random error” — Client: “But… this is a GET request”
7. 404 (Not Found) vs 410 (Gone)
8. Get your priorities right
9. “We want our API to be public, completely open, and free to use!” — The Care Bears
10. “500 Internal Server Error” more like “500 Whoops, code broke”

https://medium.com/creative-black-pug-studio/restful-apis-5b0944900e6a

HTTP Requests

GET - POST - PUT - DELETE

HTTP REQUEST METHODS (verbs)

GET - POST - PUT - DELETE

HTTP (Hypertext Transfer Protocol) is the standard way to communicate between clients and servers (request-response protocol).

"HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource."

https://www.w3schools.com/tags/ref_httpmethods.asp
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

CRUD vs REST & HTTP Verbs

CREATE —> `POST`: create a new resource (object)

READ —> `GET`: retrieve a specific resource or a collection

UPDATE —> `PUT`: update a specific resource (by id)

DELETE —> `DELETE`: remove a specific resource by id

codecademy.com/article/what-is-rest

HTTP Request & Response

FRONTEND

REST API

BACKEND

FETCH(GET, POST, PUT, DELETE)

REQUEST

JSON DATA {...}

RESPONSE

CLIENT

SERVER

HTTP Request & Response

FRONTEND

REST API

BACKEND

```
fetch(url,
      {headers, method, body}
      );
```

REQUEST

JSON DATA {...}

RESPONSE

CLIENT

SERVER

HTTP Request & Response

```
const newUser = { name, title, mail, image };
const userAsJson = JSON.stringify(newUser);

const response = await fetch("http://localhost:3333/users", {
    headers: {
        "Content-Type": "application/json"
    },
    method: "POST",
    body: userAsJson
});
```

FR

BACKEND

REQUEST

JSON DATA {...}

RESPONSE

CLIENT

SERVER

URL → Specifies the web address of the resource you want to access.

Headers → Provide metadata and instructions for the request or response.

Body → Carries data, often used to send information to the server in requests.

REQUEST headers

- “[...] contain more information about the resource to be fetched, or about the client requesting the resource.”

- "A request header is an HTTP header that can be used in an HTTP request to provide information about the request context, so that the server can tailor the response. For example, the Accept-* headers indicate the allowed and preferred formats of the response. Other headers can be used to supply authentication credentials (e.g. Authorization), to control caching, or to get information about the user agent or referrer, etc.”

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers
https://developer.mozilla.org/en-US/docs/Glossary/Request_header

REQUEST headers

```
const response = await fetch("http://localhost:3333/users", {
    headers: {
        "Content-Type": "application/json"
    },
    method: "POST",
    body: userAsJson
});
```

```
const headers = {
    "Content-Type": "application/json"
};

// ============ CREATE ============ //
// Create (POST) user to backend using REST API
async function createUser(event) {
    event.preventDefault();
    const name = event.target.name.value;
    const title = event.target.title.value;
    const mail = event.target.mail.value;
    const image = event.target.image.value;
    // create a new user
    const newUser = { name, title, mail, image };
    const userAsJson = JSON.stringify(newUser);
    const response = await fetch(`${endpoint}/users`, {
        headers: headers,
        method: "POST",
        body: userAsJson
    });

    if (response.ok) {
        // if success, update the users grid
```

REQUEST body

In HTTP requests, there are occasions when data must be transmitted. This data is enclosed within what we call the request body.

Different types of request body:

- **JSON Request Body**: Data sent as a structured JSON-encoded string.
- **Form Data Request Body**: Data transmitted in a structured format, commonly used for web forms.
- **Binary Data Request Body**: Non-textual data, such as files or images, sent in its raw binary form.
- **URL Search Params Request Body**: Simple data, like key-value pairs, sent as URL search parameters.

https://javascript.info/fetch#post-requests

REQUEST body

```
const url = "http://localhost:3000/json-api/posts/";
await fetch(url, {
    method: "POST",
    body: JSON.stringify(newPost)
});
```

```
async function savePost(postToUpdate) {
    const response = await fetch(url, {
        method: "PUT",
        body: JSON.stringify(postToUpdate)
    });
    const data = await response.json();
    console.log(data);
    navigate("/");
}
```

Fetch and create post

Using `POST` to create a new post object in the resource.

```
// new post object
const newPost = {
    title: "My new post",
    body: "Body description of a new post",
    image: "image url or image data string"
};

const url = "https://race-rest-default-rtdb.firebaseio.com/posts.json";
const response = await fetch(url, {
    method: "POST", // fetch request using POST
    body: JSON.stringify(newPost) // newPost object to JSON
});
```

Fetch and update post

Using `PUT` to an existing post object by given id.

```
const postId = "5tl4jHHSRaKEB0UW9nQd"; // id of the object to update
const postToUpdate = { title: "...", body: "...", image: "..." };

const url = `https://race-rest-default-rtdb.firebaseio.com/posts/${postId}.json`;
const response = await fetch(url, {
    method: "PUT", // using HTTP method PUT
    body: JSON.stringify(postToUpdate)
});
```

Fetch and delete post

Using `DELETE` to an object by given id.

```
const postId = "5tl4jHHSRaKEB0UW9nQd"; // id of the object to update
const url = `https://race-rest-default-rtdb.firebaseio.com/posts/${postId}.json`;

const response = await fetch(url, {
    method: "DELETE"
});
```

JSON to Figma

Plugin: https://www.figma.com/community/plugin/789839703871161985
Video: https://www.youtube.com/watch?v=1qk3AbTQfYQ&ab_channel=Figma55

Integrate JSON data in Figma

https://race.notion.site/Integrate-JSON-data-in-Figma-eddb262bac8148c1a46761f20759b498

Final Perspective

Digital Integrations, APIs, Cloud Services & SaaS

The Role of APIs in Digital Integrations

- APIs act as digital bridges, enabling communication between systems.
- Example: Connecting a website to a payment service (Stripe) or CRM (HubSpot).
- Benefits: Automation, scalability, and streamlined workflows.

APIs and Cloud Services

- Cloud platforms (AWS, Google Cloud, Azure, Firebase) offer APIs to interact with their services.
- Example: Cloud storage APIs (Google Drive API) or AI services (OpenAI API).
- Impact: Reduces infrastructure costs and enables global access.

API
Web App
Firebase
Firestore
Hosting
Realtime Database
Genkit
Cloud Functions
Vertex AI in Firebase
Cloud Storage
Authentication

What is Firebase?

Platform, a suite of tools & Backend-as-a-Service
for Web & App Development

100 SECONDS OF

https://www.youtube.com/watch?v=vAoB4VbhRzM

SaaS & API-Driven Business Models

- SaaS platforms (Salesforce, Slack, Shopify) expose APIs for customization and integration.
- Example: Integrating a helpdesk tool (Zendesk API) with a customer database.
- Trend: "API-first" companies provide building blocks for digital products.

API Economy & Monetization

- Companies monetize APIs as products (e.g., Twilio for messaging, Google Maps for location services).
- Shift from standalone software to interconnected digital ecosystems.
- Key insight: APIs create new revenue streams and business opportunities.

Developers & Businesses

Mistral AI

Google Maps

SendGrid

stripe

Meta

PayPal

aws

The Future: AI, Automation & API-First Development

- AI-powered APIs (Mistral, OpenAI) enabling automation and smarter applications.
- Low-code/no-code platforms utilizing APIs for easy integrations.
- Emerging Trends: APIs in IoT, fintech, and digital transformation.

Claude

Mistral AI

Gemini

GitHub

perplexity