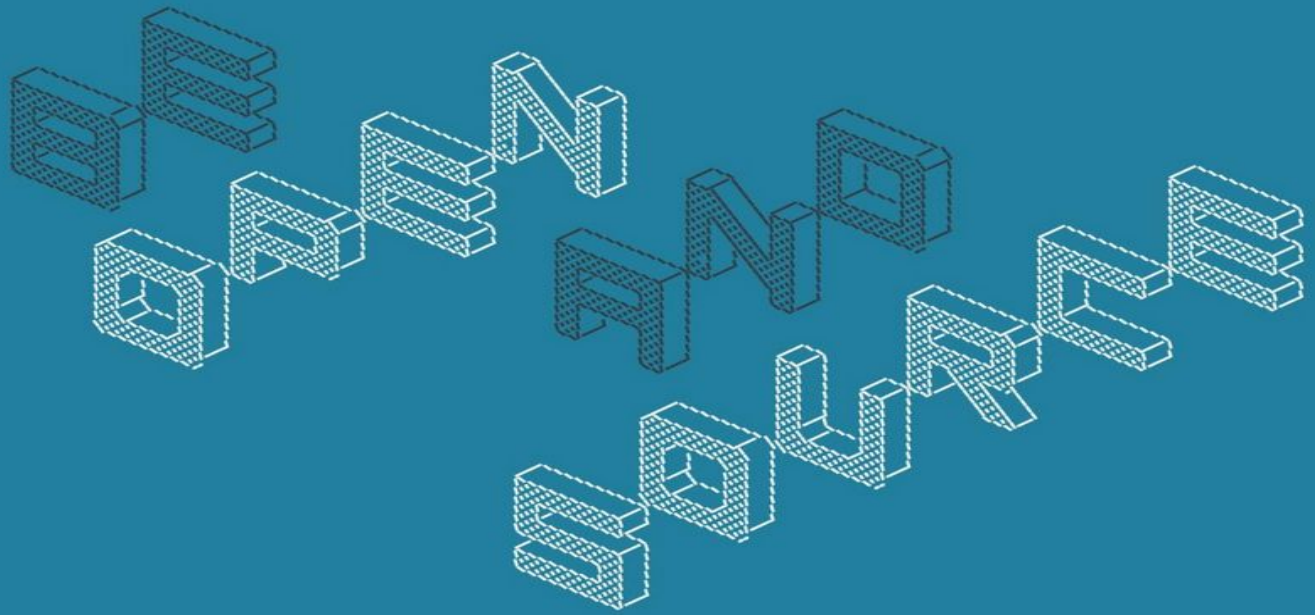
SLAC 2025

Secure Linux Administration Conference
vom 2.–4. Juni 2025 in Berlin



SLAC 2025

APIs für Administrierende?

SLAC 2025

Andreas Fritzsche

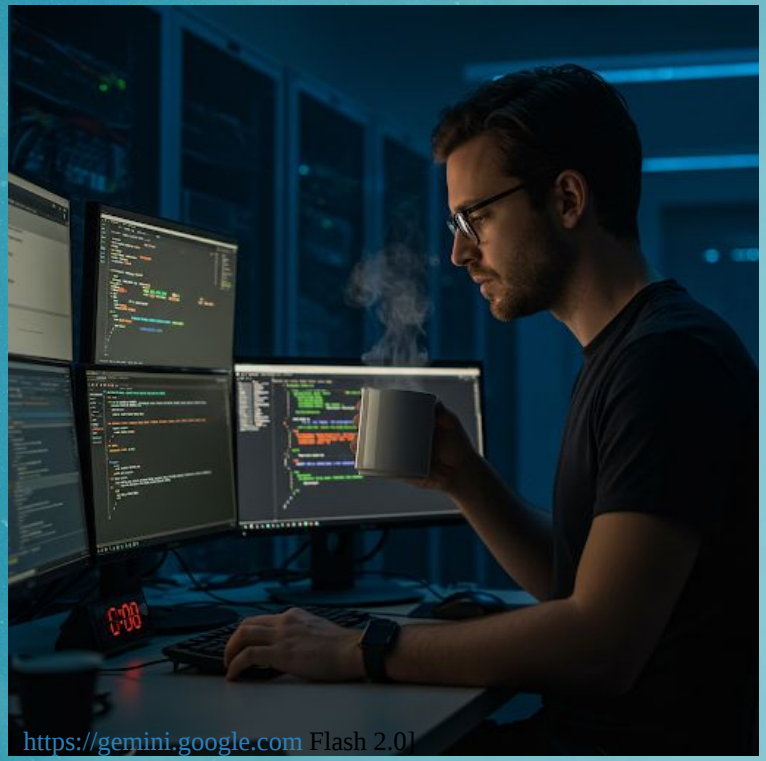
Systemadministrator

Danke an meine Freundin und Familie, die mir bei dem Vortrag sehr geholfen und mich unterstützt haben.
Die mir viele Aufgaben abgenommen haben.
Ohne sie gäbe es heute keinen Vortrag.

SLAC 2025

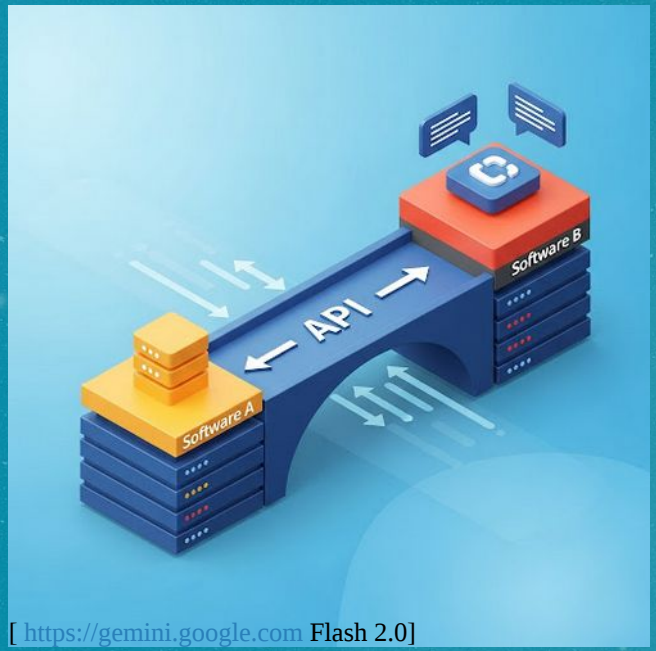
API

[https://gemini.google.com Flash 2.0]



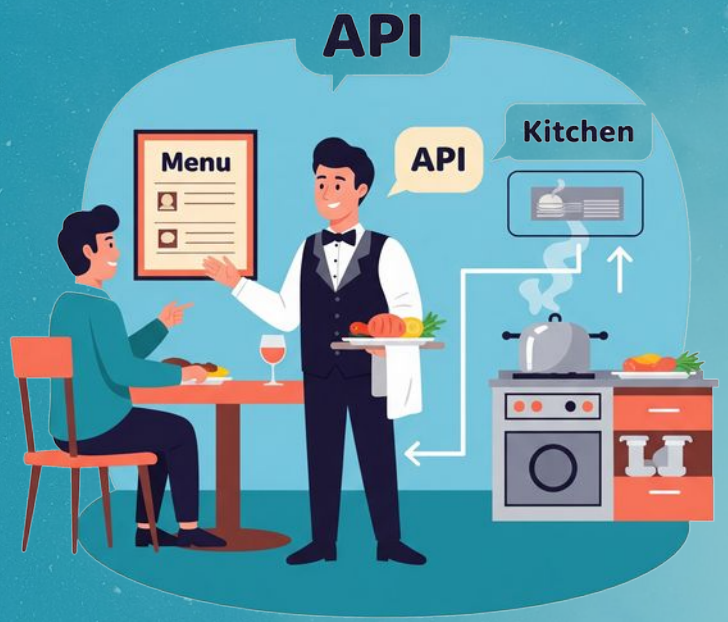
https://gemini.google.com Flash 2.0]

SLAC 2025



[https://gemini.google.com Flash 2.0]

[https://gemini.google.com Flash 2.0]



Was ist ein API?

Application Programming Interface

SLAC 2025

API Typen

Funktionsorientierte

Funktion mit oder ohne Rückgabe Handle

Dateiorientierte

Open, read, write, close

Objektorientierte

flexibler als funktionsorientierte
häufig Typbibliothek

Protokollorientierte

Protokoll
Kapselung Interface und funktionsorientiert
allgemein(SOAP) anwendungsspezifisch(SMTP)

SLAC 2025

API Typen

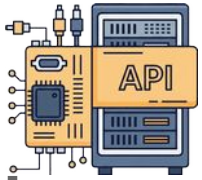
Web APIs

Per Netzwerk

(REST, SOAP)

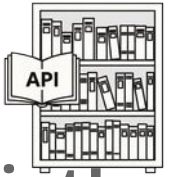
OS-APIs

Anwendung interagiert mit OS



Hardware-APIS

Zugriff auf Hardware(Kamera)

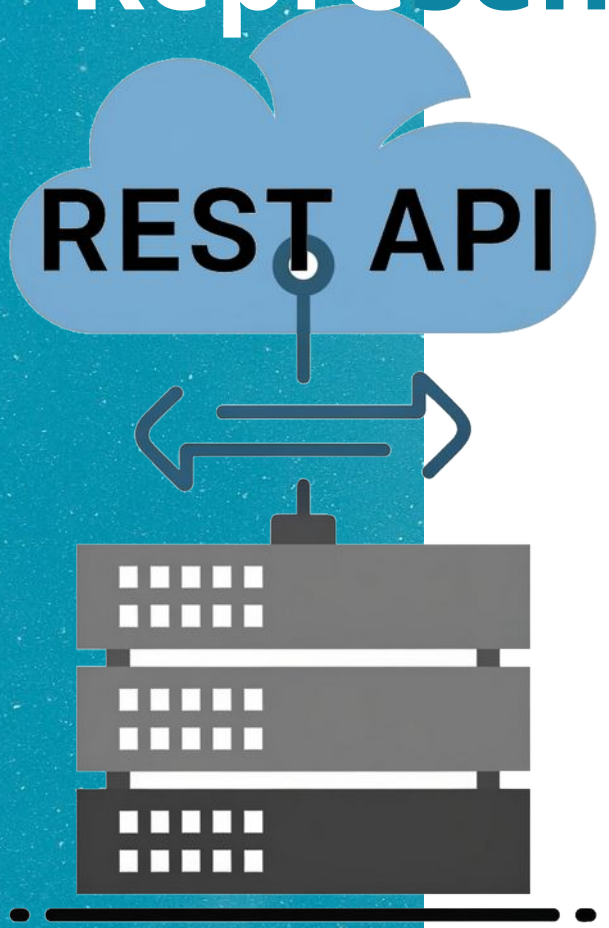


Bibliothek-APIs

innerhalb einer Programmiersprache

SLAC 2025

Representational State Transfer



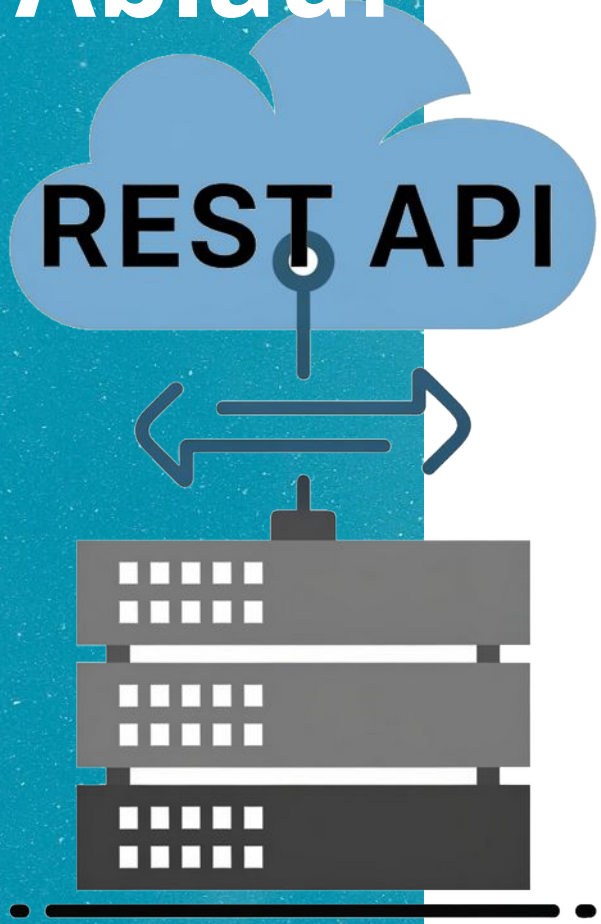
[https://pixnio.com/ Flash 2.0]

Für Webservice

- Zustandslos
- Client-Server-basiert
- HTTP-Methoden
- Einfachheit
- Skalierbarkeit
- Ressourcen durch URL identifiziert

SLAC 2025

Ablauf

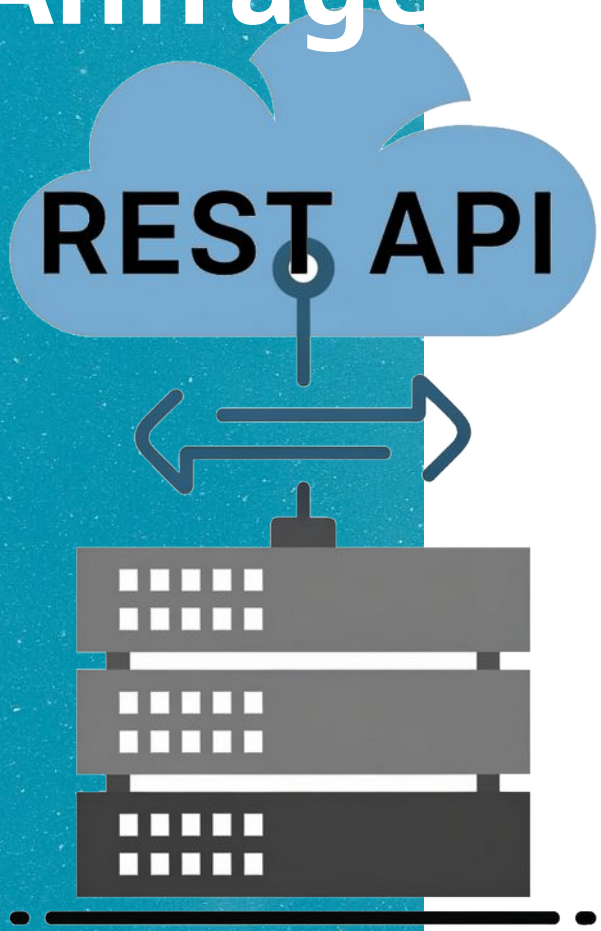


1 Client sendet eine Anfrage

Der Client (z.B. eine Webanwendung, eine mobile App) sendet eine Anfrage an den Server über das HTTP-Protokoll.

SLAC 2025

Anfrage



[https://gemini.google.com Flash 2.0]

HTTP-Methode

* *GET* Abrufen
* *POST* Erstellen
* *PUT* Aktualisieren
* *PATCH* Teilweises Aktualisieren
* *DELETE* Löschen einer Ressource.

URL (Endpoint)

spezifische Ressource identifiziert

Header

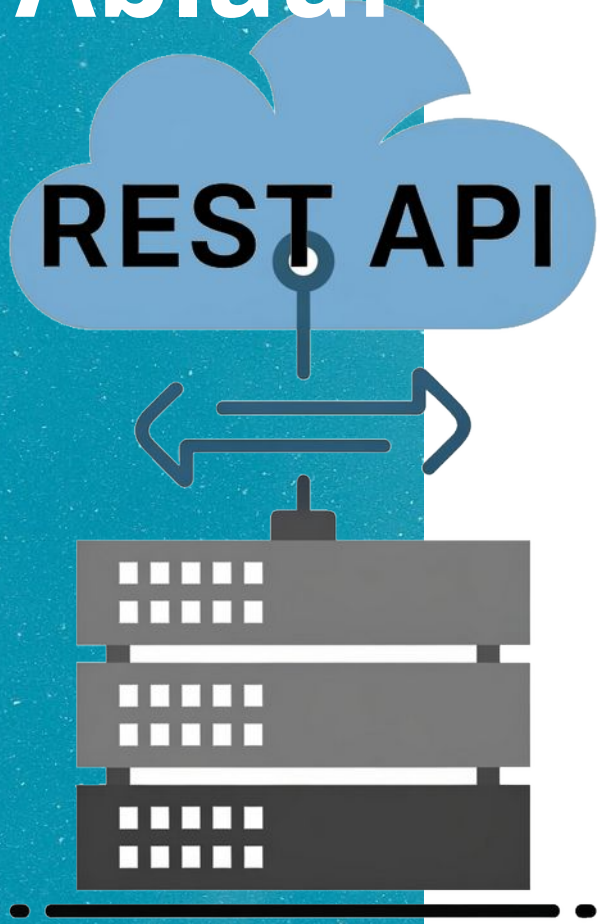
zusätzliche Informationen enthalten
(z.B. Content-Type, Authentifizierungsinformationen).

Body (optional)

POST, *PUT* und *PATCH* enthält der Body die Daten
(oft im JSON- oder XML-Format)

SLAC 2025

Ablauf



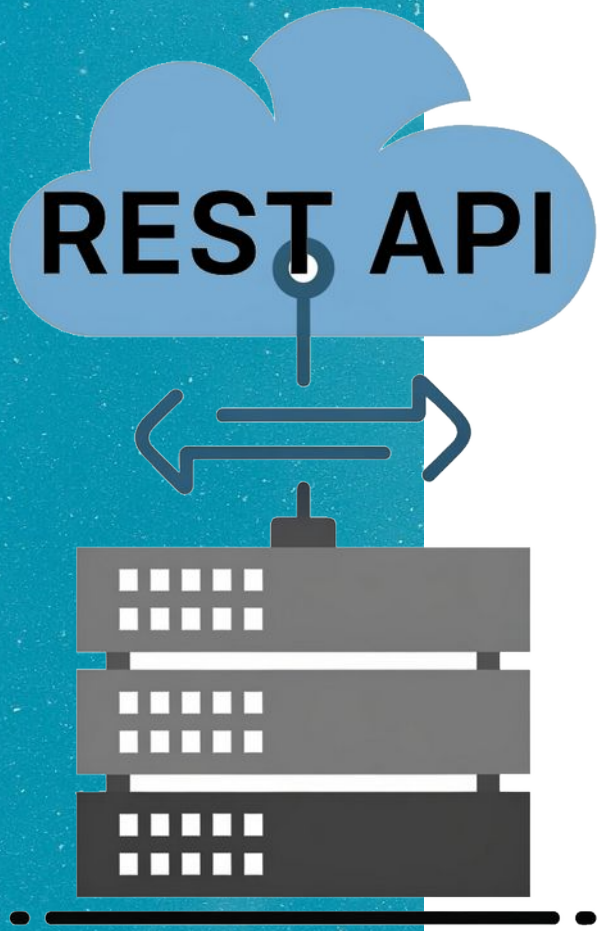
[https://pixabay.com/ Flash 2.0]

1. Client sendet eine Anfrage

Der Client (z.B. eine Webanwendung, eine mobile App) sendet eine Anfrage an den Server über das HTTP-Protokoll.

2. Server verarbeitet die Anfrage

Der Server empfängt die Anfrage,
verarbeitet sie und greift auf die angeforderte Ressource zu oder führt die angeforderte Aktion aus.



[https://gemini.google.com Flash 2.0]

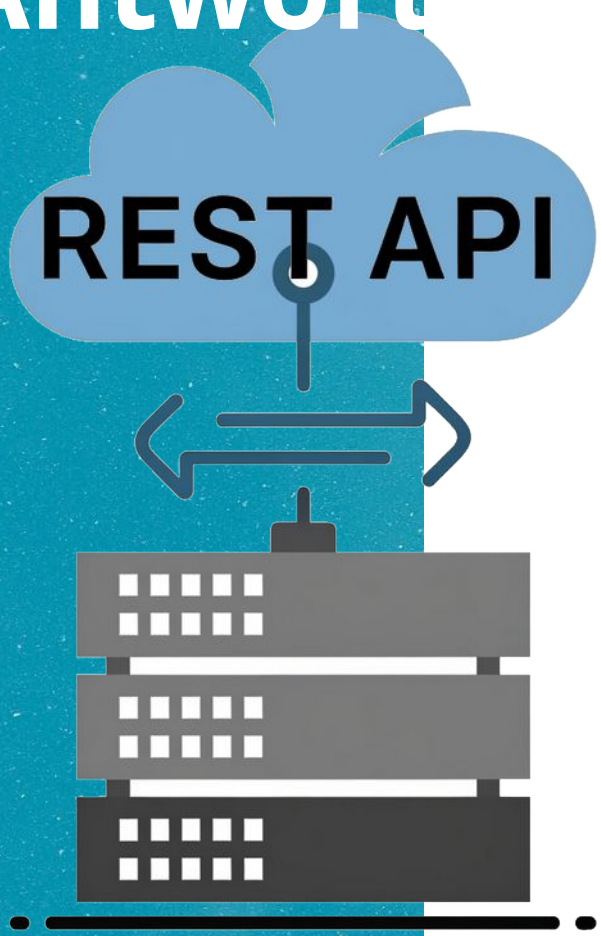
1. Client sendet eine Anfrage
Der Client (z.B. eine Webanwendung, eine mobile App) sendet eine Anfrage an den Server über das HTTP-Protokoll.

2. Server verarbeitet die Anfrage
Der Server empfängt die Anfrage,
verarbeitet sie und greift auf die angeforderte Ressource zu oder führt die angeforderte Aktion aus.

3. Server sendet eine Antwort
Der Server sendet eine Antwort
zurück an den Client über das HTTP-Protokoll.

SLAC 2025

Antwort



[https://gemini.google.com Flash 2.0]

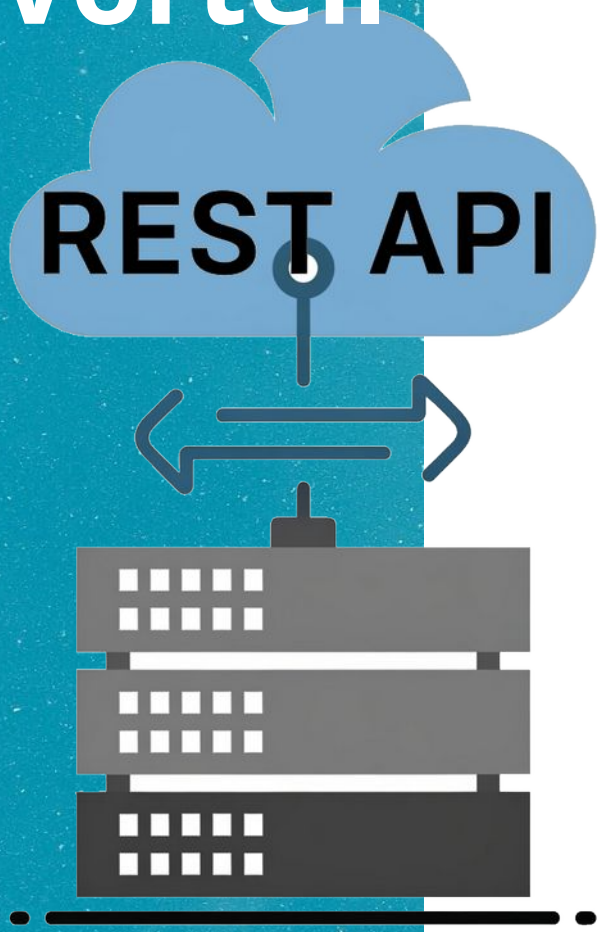
HTTP-Statuscode
HTTP-Statuscode
(z.B. 200 OK, 201 Created, 404 Not Found,
500 Internal Server Error).

Header
(z.B. Content-Type).

Body (optional)
Body mit den angeforderten Daten
oder Informationen über das Ergebnis
(oft im JSON- oder XML-Format).

SLAC 2025

Vorteil

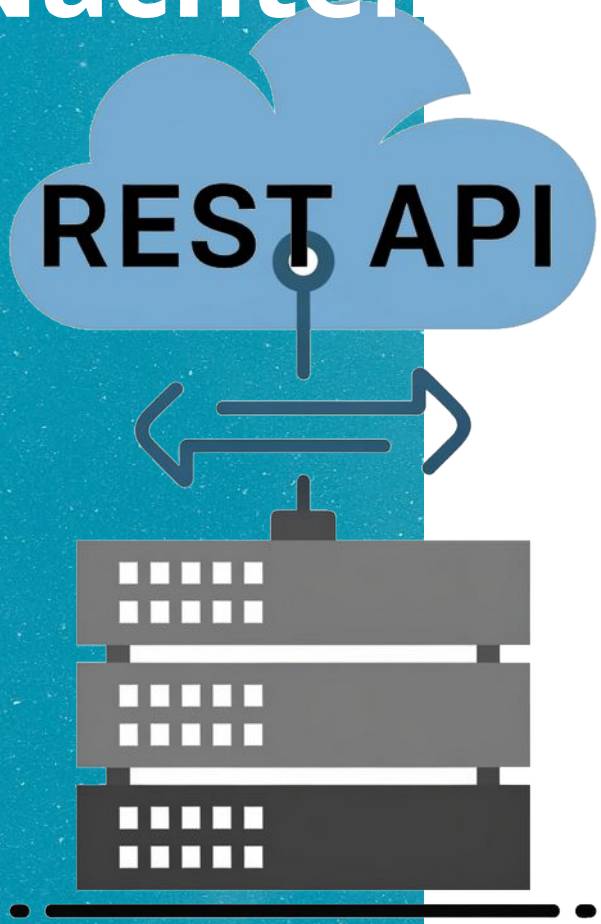


[https://gemini.google.com Flash 2.0]

- **Einfachheit**
  Leicht zu verstehen zu implementieren
- **Skalierbarkeit**
  Zustandslosigkeit ermöglicht
  eine einfache Skalierung der Serverinfrastruktur.
- **Flexibilität**
  verschiedene Datenformate
  (z.B. JSON, XML) verschiedenen Client-Typen
  (Webbrowser, mobile Apps, andere Server)
- **Lose Kopplung**
  Client und Server sind weniger stark voneinander abhängig,
  was die Wartung und Weiterentwicklung erleichtert.
- **Cacheability**
  HTTP ermöglicht das Caching von Antworten,
  was die Leistung verbessern kann.
- **Hohe Verbreitung**
  große Auswahl an Tools und Bibliotheken

SLAC 2025

Nachteil



[https://gemini.google.com Flash 2.0]

- **Over- Fetching**

Clients erhalten möglicherweise mehr Daten als benötigt

- **Under-Fetching**

müssen mehrere Anfragen senden,
um alle benötigten Daten zu erhalten

- *GraphQL ist eine mögliche Lösung für dieses Problem.*
- **Zustandslosigkeit**

kann komplexere Interaktionen erschweren

- **Keine standardisierte Fehlerbehandlung**
- **HATEOAS (Hypermedia)**

wird oft nicht vollständig implementiert

- **Sicherheitsaspekte**

Erfordert sorgfältige Implementierung
von Authentifizierung und Autorisierung.

SLAC 2025

Swagger

[https://de.wikipedia.org/wiki/Swagger_(Software)#/media/Datei:Swagger-logo.png]

Was ist Swagger?

- **Automatisierte Dokumentation**

Generierung interaktiver API-Dokumentationen,
die für Entwickler leicht verständlich sind.

- **Code-Generierung**

Ermöglicht die automatische Generierung
von Client- und Server-Stubs in verschiedenen Programmiersprachen

- **Testen der API**

Bietet Tools zum einfachen Testen von API-Endpunkten.

- **Verbesserte Zusammenarbeit**

Eine klare und standardisierte Spezifikation
erleichtert die Zusammenarbeit zwischen verschiedenen Teams.

- **API-Design**

Hilft beim Design konsistenter und gut strukturierter APIs

SLAC 2025

Swagger

[https://de.wikipedia.org/wiki/Swagger_(Software)#/media/Datei:Swagger-logo.png]

Swagger https://petstore.swagger.io/v2/swagger.json Explore

Swagger Petstore 1.0.7 OAS 2.0

[Base URL: petstore.swagger.io/v2]

https://petstore.swagger.io/v2/swagger.json

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes HTTPS

Authorize

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

Alternativen

[https://de.wikipedia.org/wiki/Swagger_(Software)#/media/Datei:Swagger-logo.png]

- **RAML (RESTful API Modeling Language)**

sprachunabhängige Spezifikationssprache für RESTful APIs

- **API Blueprint**

Eine Markdown-basierte Sprache zur Beschreibung von Web-APIs.
Fokus auf einfache Lesbarkeit und Erstellung von Mock-Servern.

- **GraphQL Schema Definition Language (SDL)**

Speziell für GraphQL-APIs.
Beschreibt die Datenstruktur und die verfügbaren Operationen.

- **Postman**

Obwohl primär ein API-Testwerkzeug,
bietet Postman auch Funktionen zur Dokumentation
und zum Teilen von API-Sammlungen.

- **Stoplight**

Eine Plattform, die verschiedene Tools für den API-Design-First-Ansat
bietet, einschließlich Spezifikationserstellung und Dokumentation.

SLAC 2025

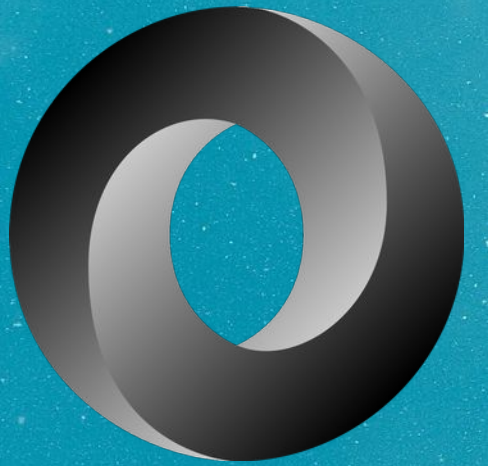
Zusammenfassung

[https://de.wikipedia.org/wiki/Swagger_(Software)#/media/Datei:Swagger-logo.png]

- **APIs**
ermöglichen die Kommunikation zwischen Softwareanwendungen.
- **REST APIs**
sind ein beliebter Architekturstil für Web Services,
der auf Einfachheit und Standardisierung basiert.
- **REST APIs** funktionieren über das HTTP-Protokoll
und nutzen standardmäßige HTTP-Methoden, URLs und Statuscodes.
- **REST APIs** bieten viele Vorteile wie Skalierbarkeit und Flexibilität,
haben aber auch Nachteile wie potenzielles Over- oder Under-Fetchin
- **Swagger/OpenAPI** ist ein Framework zur Beschreibung,
Dokumentation und zum Konsum von RESTful APIs und
erleichtert die Entwicklung und Zusammenarbeit.
- Es gibt Alternativen zu **Swagger/OpenAPI**,
die je nach Bedarf in Betracht gezogen werden können.

SLAC 2025

json

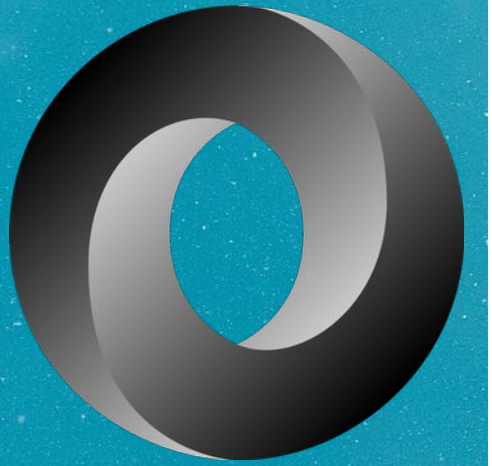


[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

- **JSON** ist ein leichtgewichtiges Datenformat, das für den einfachen Datenaustausch zwischen Anwendungen entwickelt wurde.
- Es ist sowohl für Menschen einfach zu lesen und zu schreiben, als auch für Maschinen einfach zu parsen und zu generieren.
- Ursprünglich von JavaScript abgeleitet, ist es heute sprachunabhängig und wird von vielen Programmiersprachen unterstützt

SLAC 2025

Vorteil



[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

- **Einfachheit**

Klare und einfache Syntax

- **Gute Lesbarkeit**

Für Menschen gut lesbar und verständlich.

- **Leichtgewichtig**

Weniger Overhead als andere Datenformate wie XML

- **Sprachunabhängig**

Kann in nahezu jeder modernen Programmiersprache verwendet wer

- **Effiziente Verarbeitung**

Viele Bibliotheken in verschiedenen Sprachen bieten effiziente Möglichkeiten zum Parsen und Generieren von JSON.

- **Direkte Abbildung auf Datenstrukturen**

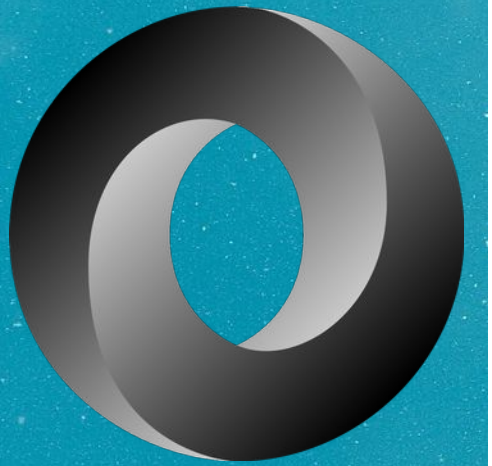
Kann einfach in gängige Datenstrukturen wie Objekte (Dictionaries/M und Arrays (Listen) in Programmiersprachen umgewandelt werden.

- **Weite Verbreitung**

De-facto-Standard für den Datenaustausch in Webanwendungen und A

SLAC 2025

Nachteil

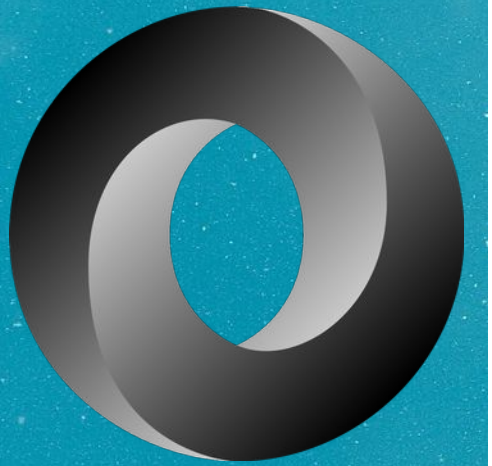


[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

- **Weniger ausdrucksstark als XML**:
Bietet keine integrierte Möglichkeit für Attribute, Kommentare oder komplexe Strukturierungen wie XML-Schemas.
- **Keine Unterstützung für Verknüpfungen oder Referenzen**:
Das Verknüpfen von Daten innerhalb des Dokuments kann umständli sein.
- **Keine eingebaute Validierung**:
Es gibt keine standardmäßige Möglichkeit, die Struktur und Datentypen eines JSON-Dokuments zu definieren und zu validieren (obwohl es externe Schemas gibt).
- **Begrenzte Datentypen**:
Unterstützt nur eine begrenzte Anzahl von Datentypen (primitive Typen, Objekte und Arrays).

SLAC 2025

Aufbau



[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

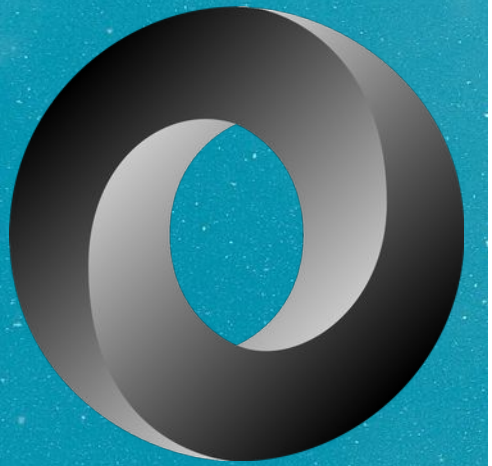
Objekte

Eine ungeordnete Sammlung von Schlüssel-Wert-Paaren.
Schlüssel sind Strings (in doppelten Anführungszeichen),
und Werte können primitive Datentypen (String, Zahl, Boolean, null),
andere Objekte oder Arrays sein.

```
{
  "name": "Max Mustermann",
  "age": 30,
  "city": "Leipzig"
}
```

SLAC 2025

Aufbau



[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

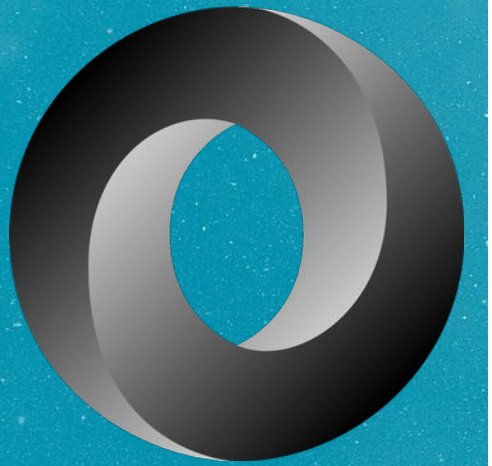
Array

Eine geordnete Liste von Werten.
Werte können primitive Datentypen, Objekte oder andere Arrays sein.

```
[
  "Apfel",
  "Banane",
  "Kirsche"
]
```

SLAC 2025

Aufbau



[hhttps://en.m.wikipedia.org/wiki/File:JSON_vector_logo.svg]

Kommbination

Objekte und Arrays können beliebig ineinander verschachtelt sein, um komplexe Datenstrukturen darzustellen.

```
{
 "students": [
  {
   "name": "Anna Schmidt",
   "grades": [90, 85, 92]
  },
  {
   "name": "Peter Müller",
   "grades": [78, 88, 80]
  }
 ],
 "classAverage": 85.5
```

SLAC 2025

Tools

Rest API

curl
Vielzahl von Protokollen,
einschließlich HTTP. Ideal für automatisierte Tests und Skripte.
wget
für einfache GET-Anfragen an APIs
Webbrowser
GET-Anfragen in die Adressleiste (mehr Entwicklertools)
API-Testclients
Benutzerfreundliche grafische Oberflächen
zum Erstellen, Senden und Untersuchen

SLAC 2025

Tools

json

jq
Vielzahl von Protokollen, einschließlich HTTP.
Ideal für automatisierte Tests und Skripte.

jp
für einfache GET-Anfragen an APIs

jello
GET-Anfragen in die Adressleiste (mehr Entwicklertools)

jello

```
echo '{"foo":"bar","baz":[1,2,3], "users": [{ "vname": "bob"},{"vname": "Alice"}] }' | jello -s
_ = {};
_.foo = "bar";
_.baz = [];
_.baz[0] = 1;
_.baz[1] = 2;
_.baz[2] = 3;
_.users = [];
_.users[0] = {};
_.users[0].vname = "bob";
_.users[1] = {};
_.users[1].vname = "Alice";
```

SLAC 2025

Tools

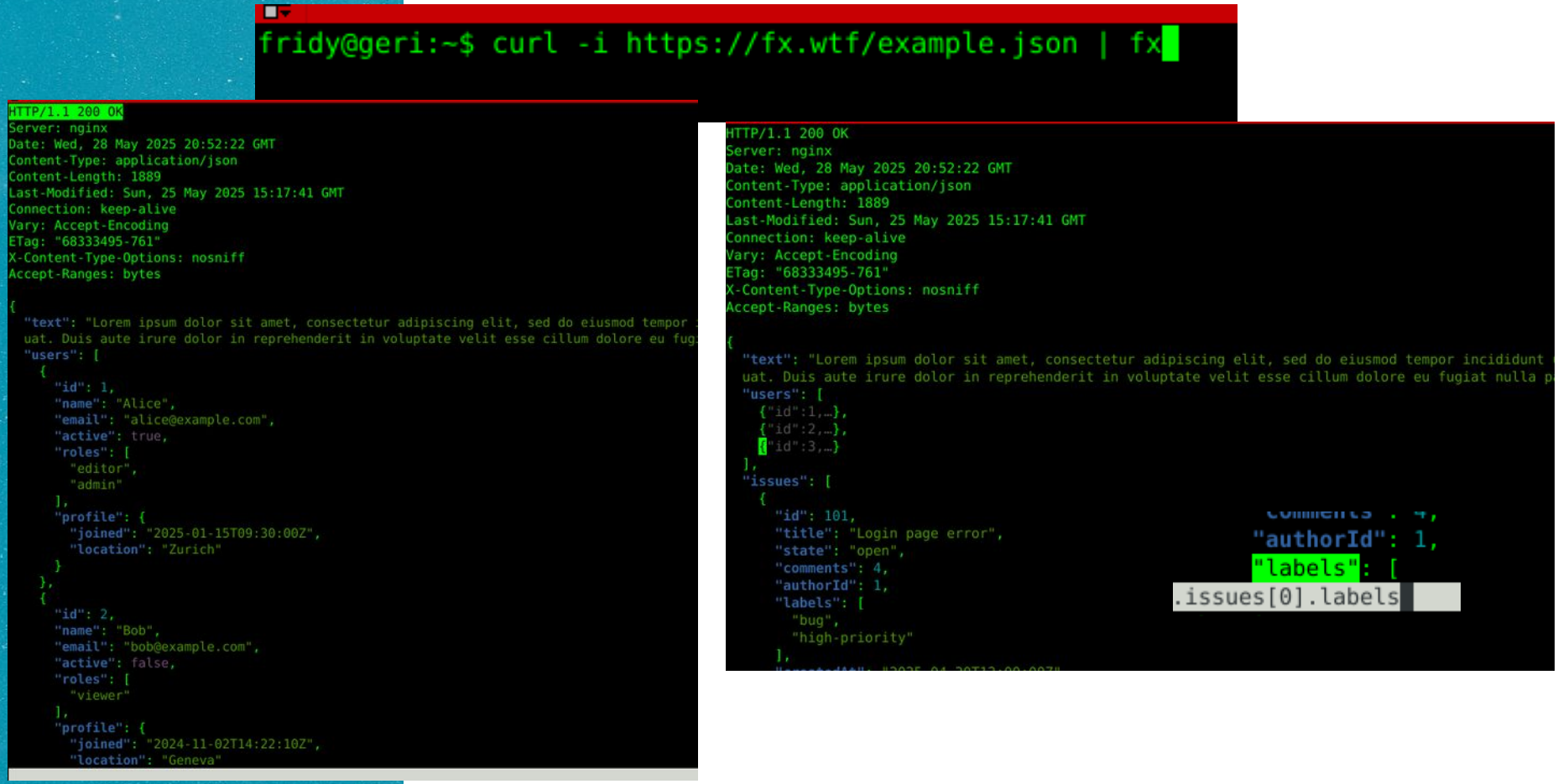
json

- **jq**
  zum Filtern, Transformieren und Bearbeiten von JSON-Daten
- **jp**
  zum Filtern
- **jello**
  filters JSON
- **emuto**
  JSON processing too
- **fx**
  JSON processing tool (auch Interactiv)

SLAC 2025

Tools

fx



SLAC 2025

Motivation

>1200 Switche in Nagvis

```
import time
from geopy.geocoders import GoogleV3
geolocator = GoogleV3()
fobj_in = open("/home/afritzsche/Schreibtisch/nagios/python/adress.txt")
fobj_out = open("/home/afritzsche/Schreibtisch/nagios/python/adress2.txt", "w")
for line in fobj_in:
    address, (latitude, longitude) = geolocator.geocode(line)
    print(address, latitude, longitude)
    fobj_out.write(address, (latitude, longitude))
    time.sleep(10)

fobj_in.close()
fobj_out.close()
```

SLAC 2025

Motivation

Redmine

```
while read service
do
    echo -e "---\n service $service "
    if [[ $(curl -s -u ${wikiuser}:${wikipasswort}
    ${wikiserver}/doku.php?id=doku:monitoring:cmk:services:$(sed 's/ /%20/g; s/\"//g' <<<$service) |
    egrep -c "Dieses Thema existiert noch nicht|muss überarbeitet werden") -gt 0 ]]
    then
        if [[ $(egrep -c -i "postgres|ldap|mysql|maria" <<<$service) -gt 0 ]]
        then
            uid=3
            #DB Verantwortliche
        elif [[ $(egrep -c -i "pop|smtp|imap" <<<$service) -gt 0 ]]
        then
            uid=74
            #Email Verantwortliche
        else
            uid=5
            #DeepInn
        fi
        subject="Dokumentation für chekmk check für Service $(sed 's/\"//g' <<<$service) anlegen"
        description="den checkmk check für Service  $(sed 's/\"//g' <<<$service) unter
        ${wikiserver}/doku.php?id=doku:monitoring:cmk:services:$(sed 's/ /%20/g; s/\"//g' <<<$service)
        dokumentieren"
        curl -i -H 'Content-Type: application/xml' -X POST --data
        '<issue><project_id>65</project_id><tracker_id>8</tracker_id><subject>'"$subject"
        '</subject><description>'"$description"'</description><assigned_to_id>'"$uid"
        '</assigned_to_id><custom_fields type="array"><custom_field
        id="1"><value>Test</value></custom_field></custom_fields></issue>' -u ${redmineuser}:
        ${redminepaswort} ${redmineserver}/issues.xml
    fi

done < ${serviceliste}
```

Motivation

Redmine

Redmine

Übersicht | Download | Aktivität | Roadmap | Tickets | News | Wiki | Foren | Repository

Guide » Developer Guide »

Redmine API

Redmine exposes some of its data through a REST API. This API provides access and basic CRUD operations (create, update, dele
XML and JSON formats.

API Description

Resource	Status	Notes	Availability
Issues	Stable		1.0
Projects	Stable		1.0
Project Memberships	Alpha		1.4
Users	Stable		1.1
Time Entries	Stable		1.1
News	Prototype	Prototype implementation for `index` only	1.1
Issue Relations	Alpha		1.3
Versions	Alpha		1.3
Wiki Pages	Alpha		2.2

[https://www.redmine.org/projects/redmine/wiki/rest_api]

SLAC 2025

Motivation

Redmine

https://www.redmine.org/projects/redmine/wiki/rest_api

https://petstore.swagger.io/

https://editor.swagger.io/

https://github.com/Morpheus235/redmine-swagger-api

Motivation

Redmine

https://www.redmine.org/projects/redmine/wiki/rest_api

https://demo.redminecloud.net/

https://demo.redminecloud.net/issues.json?issue_id=762

SLAC 2025

Motivation

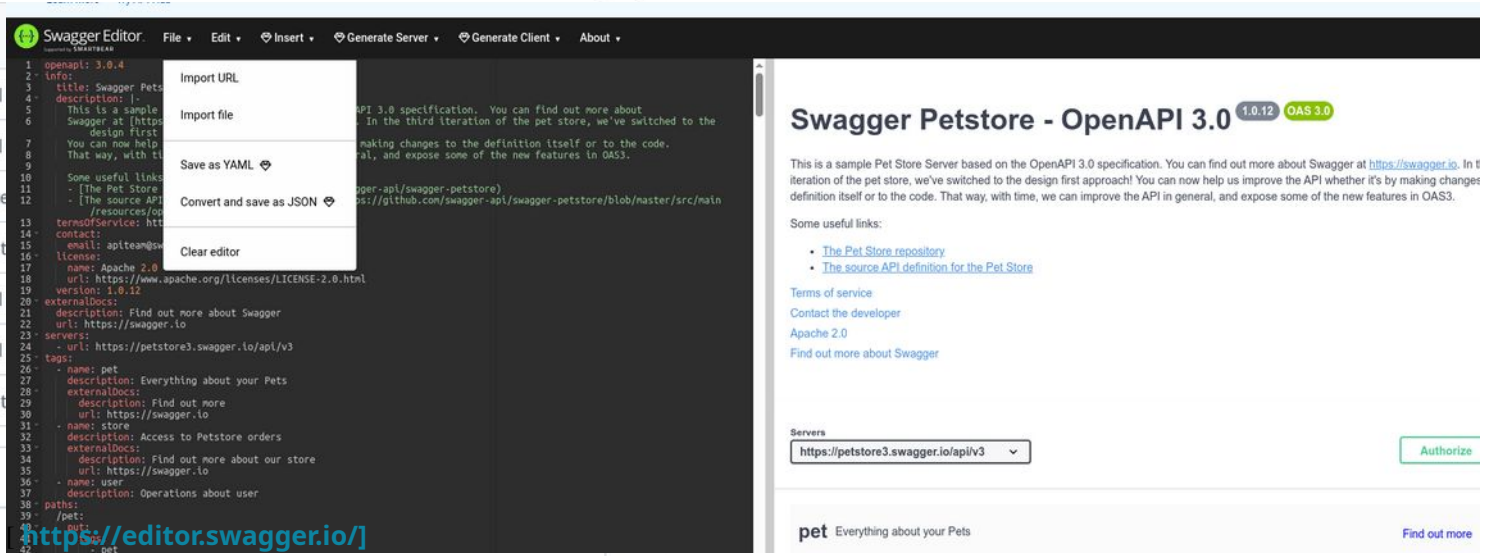
Redmine

master · 2 Branches · 0 Tags · Go to file · Code · Abo

Morpheus235 Delete swagger-github-pages.png

.github/workflows	Initial
dist	Initial
screenshots	Delete
README.md	Updat
index.html	Initial
swagger-ui.version	Initial
swagger.yaml	Updat

README



[https://editor.swagger.io/]

Redmine Swagger API

A swagger api definition for the redmine rest API. Work in progress

Github Page Swagger-UI

[https://github.com/Morpheus235/redmine-swagger-api]

SLAC 2025

Motivation

Redmine CLI

https://github.com/MrJeffLarry/redmine-cli

https://github.com/mightymatth/arcli

https://pypi.org/project/redmine-ci/

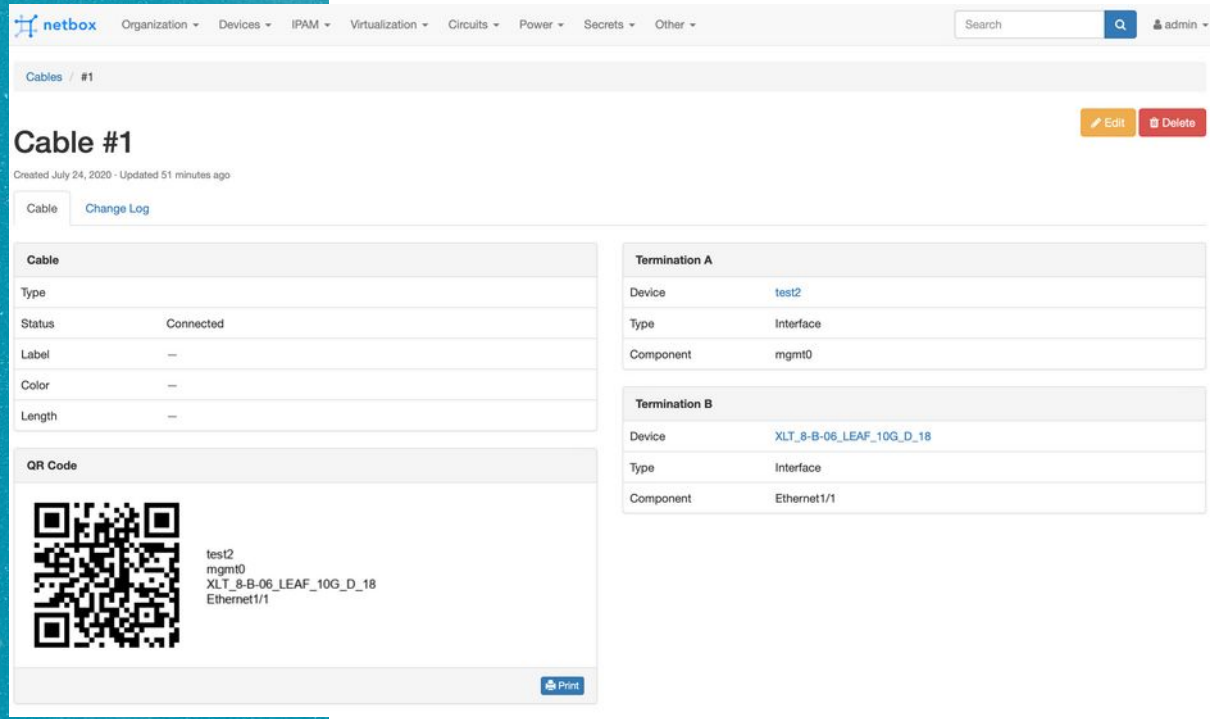
https://pypi.org/project/redman/

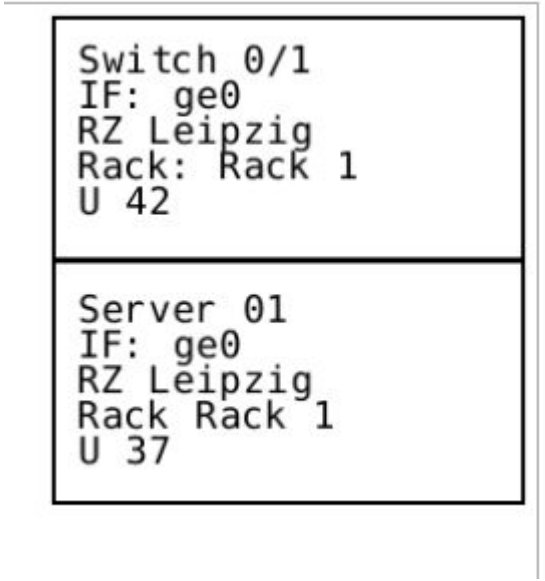
https://pypi.org/project/git-redmine/

SLAC 2025

Motivation

Netbox Labels





SLAC 2025

Motivation

Netbox Labels

https://labelary.com/viewer.html

CT~~CD,~CC^~CT~
^XA~TA000~JSN^LT0^MNW^MTT^PON^PMN^LH0,0^JMA^PR5,5~SD15^
JUS^LRN^CI0^XZ
^XA
^MMT
^PW551
^LL0256
^LS0
^FT35,233^AAB,18,10^FH\^FDSwitch 0/1^FS
^FT53,233^AAB,18,10^FH\^FDIF: ge0^FS

^FT71,233^AAB,18,10^FH\^FDRZ Leipzig ^FS
^FT89,233^AAB,18,10^FH\^FDRack: Rack 1^FS
^FT107,233^AAB,18,10^FH\^FDU 42^FS

^FT162,233^AAB,18,10^FH\^FDServer 01^FS
^FT180,233^AAB,18,10^FH\^FDIF: ge0^FS

^FT198,233^AAB,18,10^FH\^FDRZ Leipzig^FS
^FT216,233^AAB,18,10^FH\^FDRack Rack 1^FS

SLAC 2025

Motivation

Netbox Labels

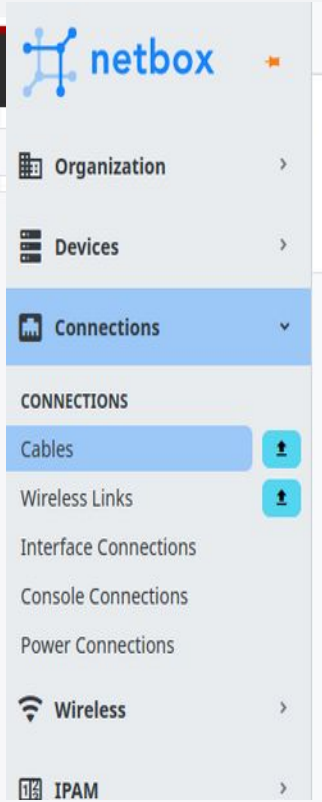
Nicht sicher | 0.0.0.0:8000/api/dcim/cables/1/

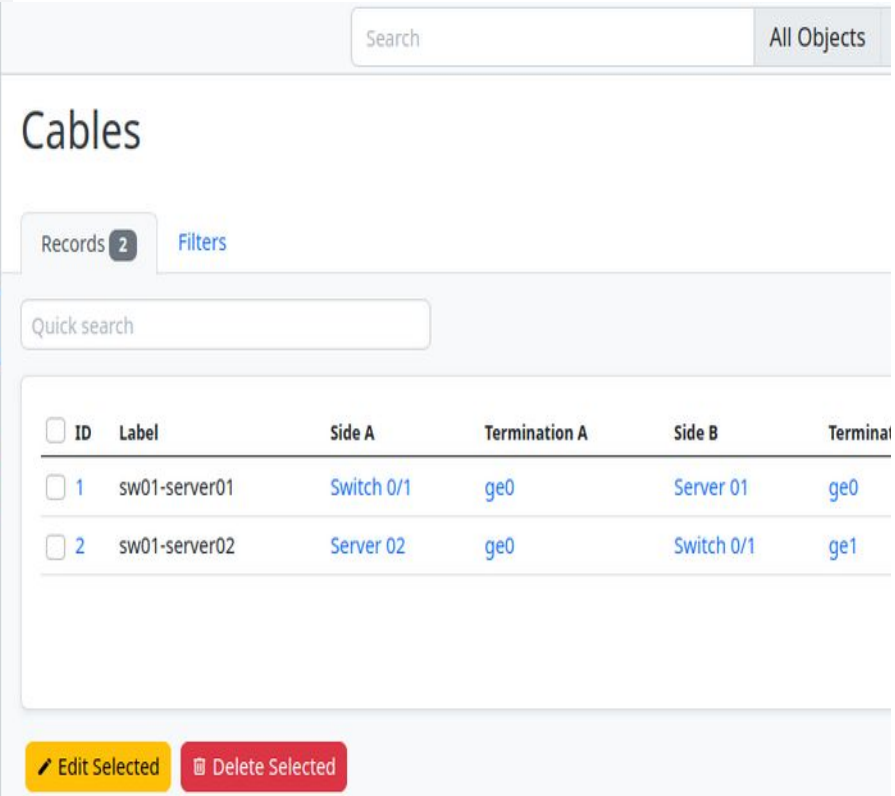
NetBox

GET /api/dcim/cables/1/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept


```
{
    "id": 1,
    "url": "http://0.0.0.0:8000/api/dcim/cables/1/",
    "display": "sw01-server01",
    "termination_a_type": "dcim.interface",
    "termination_a_id": 29,
    "termination_a": {
        "id": 29,
        "url": "http://0.0.0.0:8000/api/dcim/interfaces/29/",
        "display": "ge0",
        "device": {
            "id": 2,
            "url": "http://0.0.0.0:8000/api/dcim/devices/2/",
            "display": "Switch 0/1",
            "name": "Switch 0/1"
        },
        "name": "ge0",
        "cable": 1,
        "_occupied": true
    },
    "termination_b_type": "dcim.interface",
    "termination_b_id": 57,
    "termination_b": {
        "id": 57,
        "url": "http://0.0.0.0:8000/api/dcim/interfaces/57/",
```



SLAC 2025

Motivation

Netbox Labels

https://demo.netbox.dev/

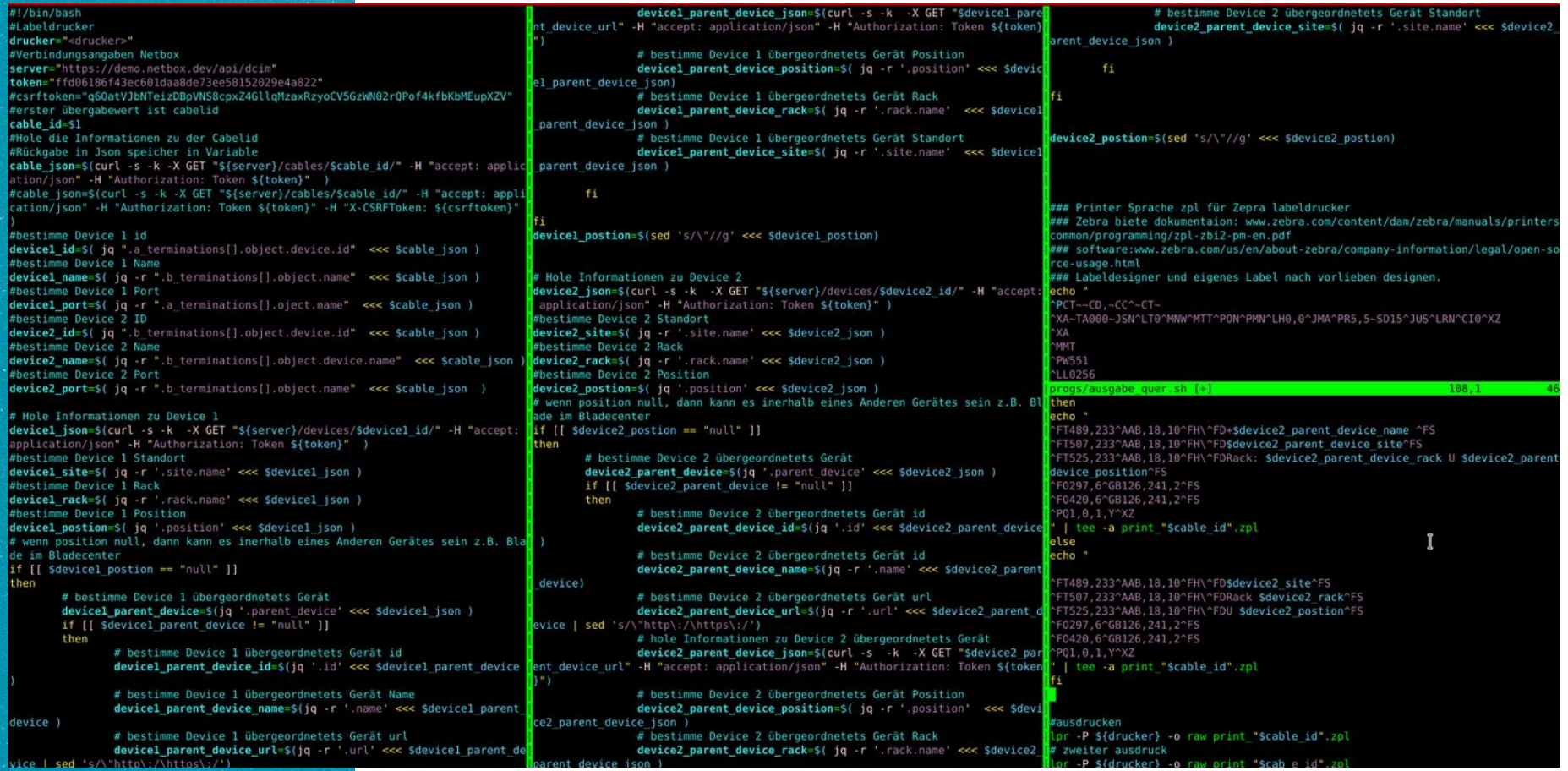
https://demo.netbox.dev/api/schema/swagger-ui/#/

https://demo.netbox.dev/api/dcim/cables/23/

SLAC 2025

Motivation

Netbox Labels



```
#!/bin/bash
#Labeldrucker
drucker="<drucker>"
#Verbindungsangaben Netbox
server="https://demo.netbox.dev/api/dcim"
token="ffd06186f43ec601daa8de73ee58152029e4a822"
#csrftoken="q6OatVJbNTeizDBpVNS8cpxZ4GllqMzaxRzyoCV5GzWN02rQPof4kfbKbMEupXZV"
#erster übergabewert ist cabelid
cable_id=$1
#Hole die Informationen zu der Cabelid
#Rückgabe in Json speicher in Variable
cable_json=$(curl -s -k -X GET "${server}/cables/$cable_id/" -H "accept: application/json" -H "Authorization: Token ${token}"  )
#cable_json=$(curl -s -k -X GET "${server}/cables/$cable_id/" -H "accept: application/json" -H "Authorization: Token ${token}" -H "X-CSRFToken: ${csrftoken}" )
#bestimme Device 1 id
device1_id=$( jq ".a_terminations[].object.device.id"  <<< $cable_json )
#bestimme Device 1 Name
device1_name=$( jq -r ".b_terminations[].object.name"  <<< $cable_json )
#bestimme Device 1 Port
device1_port=$( jq -r ".a_terminations[].oject.name"  <<< $cable_json )
#bestimme Device 2 ID
device2_id=$( jq ".b_terminations[].object.device.id"  <<< $cable_json )
#bestimme Device 2 Name
device2_name=$( jq -r ".b_terminations[].object.device.name"  <<< $cable_json )
#bestimme Device 2 Port
device2_port=$( jq -r ".b_terminations[].object.name"  <<< $cable_json  )

# Hole Informationen zu Device 1
device1_json=$(curl -s -k  -X GET "${server}/devices/$device1_id/" -H "accept: application/json" -H "Authorization: Token ${token}"  )
#bestimme Device 1 Standort
device1_site=$( jq -r '.site.name' <<< $device1_json )
#bestimme Device 1 Rack
device1_rack=$( jq -r '.rack.name' <<< $device1_json )
#bestimme Device 1 Position
device1_postion=$( jq '.position' <<< $device1_json )
# wenn position null, dann kann es inerhalb eines Anderen Gerätes sein z.B. Blade im Bladecenter
if [[ $device1_postion == "null" ]]
then
        # bestimme Device 1 übergeordnetets Gerät
        device1_parent_device=$(jq '.parent_device' <<< $device1_json )
        if [[ $device1_parent_device != "null" ]]
        then
                # bestimme Device 1 übergeordnetets Gerät id
                device1_parent_device_id=$(jq '.id' <<< $device1_parent_device )
                # bestimme Device 1 übergeordnetets Gerät Name
                device1_parent_device_name=$(jq -r '.name' <<< $device1_parent_device )
                # bestimme Device 1 übergeordnetets Gerät url
                device1_parent_device_url=$(jq -r '.url' <<< $device1_parent_device | sed 's/\"http\:/\https\:/')
                device1_parent_device_json=$(curl -s -k  -X GET "$device1_parent_device_url" -H "accept: application/json" -H "Authorization: Token ${token}")
                # bestimme Device 1 übergeordnetets Gerät Position
                device1_parent_device_position=$( jq -r '.position' <<< $device1_parent_device_json)
                # bestimme Device 1 übergeordnetets Gerät Rack
                device1_parent_device_rack=$( jq -r '.rack.name'  <<< $device1_parent_device_json )
                # bestimme Device 1 übergeordnetets Gerät Standort
                device1_parent_device_site=$( jq -r '.site.name'  <<< $device1_parent_device_json )
        fi
fi
device1_postion=$(sed 's/\"//g' <<< $device1_postion)


# Hole Informationen zu Device 2
device2_json=$(curl -s -k  -X GET "${server}/devices/$device2_id/" -H "accept: application/json" -H "Authorization: Token ${token}" )
#bestimme Device 2 Standort
device2_site=$( jq -r '.site.name' <<< $device2_json )
#bestimme Device 2 Rack
device2_rack=$( jq -r '.rack.name' <<< $device2_json )
#bestimme Device 2 Position
device2_postion=$( jq '.position' <<< $device2_json )
# wenn position null, dann kann es inerhalb eines Anderen Gerätes sein z.B. Blade im Bladecenter
if [[ $device2_postion == "null" ]]
then
        # bestimme Device 2 übergeordnetets Gerät
        device2_parent_device=$(jq '.parent_device' <<< $device2_json )
        if [[ $device2_parent_device != "null" ]]
        then
                # bestimme Device 2 übergeordnetets Gerät id
                device2_parent_device_id=$(jq '.id' <<< $device2_parent_device )
                # bestimme Device 2 übergeordnetets Gerät id
                device2_parent_device_name=$(jq -r '.name' <<< $device2_parent_device)
                # bestimme Device 2 übergeordnetets Gerät url
                device2_parent_device_url=$(jq -r '.url' <<< $device2_parent_device | sed 's/\"http\:/\https\:/')
                # hole Informationen zu Device 2 übergeordnetets Gerät
                device2_parent_device_json=$(curl -s  -k  -X GET "$device2_parent_device_url" -H "accept: application/json" -H "Authorization: Token ${token}")
                # bestimme Device 2 übergeordnetets Gerät Position
                device2_parent_device_position=$( jq -r '.position'  <<< $device2_parent_device_json )
                # bestimme Device 2 übergeordnetets Gerät Rack
                device2_parent_device_rack=$( jq -r '.rack.name' <<< $device2_parent_device_json )
                # bestimme Device 2 übergeordnetets Gerät Standort
                device2_parent_device_site=$( jq -r '.site.name' <<< $device2_parent_device_json )
        fi
fi


device2_postion=$(sed 's/\"//g' <<< $device2_postion)



### Printer Sprache zpl für Zepra labeldrucker
### Zebra biete dokumentaion: www.zebra.com/content/dam/zebra/manuals/printers/common/programming/zpl-zbi2-pm-en.pdf
### software:www.zebra.com/us/en/about-zebra/company-information/legal/open-source-usage.html
### Labeldesigner und eigenes Label nach vorlieben designen.
echo "
^PCT~~CD,~CC^~CT~
^XA~TA000~JSN^LT0^MNW^MTT^PON^PMN^LH0,0^JMA^PR5,5~SD15^JUS^LRN^CI0^XZ
^XA
^MMT
^PW551
^LL0256
progs/ausgabe_quer.sh [+]                                    108,1         46
then
echo "
^FT489,233^AAB,18,10^FH\^FD+$device2_parent_device_name ^FS
^FT507,233^AAB,18,10^FH\^FD$device2_parent_device_site^FS
^FT525,233^AAB,18,10^FH\^FDRack: $device2_parent_device_rack U $device2_parent_device_position^FS
^FO297,6^GB126,241,2^FS
^FO420,6^GB126,241,2^FS
^PQ1,0,1,Y^XZ
" | tee -a print_"$cable_id".zpl
else
echo "

^FT489,233^AAB,18,10^FH\^FD$device2_site^FS
^FT507,233^AAB,18,10^FH\^FDRack $device2_rack^FS
^FT525,233^AAB,18,10^FH\^FDU $device2_postion^FS
^FO297,6^GB126,241,2^FS
^FO420,6^GB126,241,2^FS
^PQ1,0,1,Y^XZ
" | tee -a print_"$cable_id".zpl
fi


#ausdrucken
lpr -P ${drucker} -o raw print_"$cable_id".zpl
# zweiter ausdruck
lpr -P ${drucker} -o raw print_"$cable_id".zpl
```

SLAC 2025

Motivation Netbox Labels

```
cable_json=$(curl -s -k -X GET "${server}/cables/$cable_id/" -H "accept: application/json" -H "Authorization: Token ${token}"  )
#bestimme Device 1 id
device1_id=$( jq ".a_terminations[].object.device.id"  <<< $cable_json )
#bestimme Device 1 Name
device1_name=$( jq -r ".b_terminations[].object.name"  <<< $cable_json )
```

SLAC 2025

Motivation

Netbox CLI

https://netboxlabs.com/docs/netbox/administration/netbox-shell/

https://github.com/nbcli/nbcli

https://nbcli.codeberg.page/latest/

SLAC 2025

Motivation

Netbox API

https://docs.ansible.com/ansible/latest/collections/netbox/netbox/netbox_cable_module.html#ansible-collections-netbox-netbox-netbox-netbox-cable-module

https://docs.ansible.com/ansible/latest/collections/netbox/netbox/nb_inventory_inventory.html#ansible-collections-netbox-netbox-nb-inventory-inventory

https://docs.ansible.com/ansible/latest/collections/netbox/netbox/nb_lookup_lookup.html#ansible-collections-netbox-netbox-nb-lookup-lookup

Motivation

Jira API

https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/#expansion

https://www.postman.com

SLAC 2025

Motivation bitbucket API

```
for i in nfs xxx xxxxx;
do
  liste=$(host_search -cmajor_app=$i | sed  ':a;N;$!ba;s/# \n//g; s/#//');
  if [[ -n $liste ]];
  then
       new_branch="feature/migiere_hosts_von_app_${i}";
       pull="migiere_hosts_von_app_${i}_1";
       titel="migriere hosts von app $i";
       description="script hostmigieren";
       git -C /home/afritzsche/ansible215_test switch -f main
       for server in $liste;
       do
              migration -l $server  -f -b $new_branch;
              git -C /home/afritzsche/ansible215_test add host_vars/$server/*;
              git -C /home/afritzsche/ansible215_test commit -m 'add $server';
       done;
       echo '#########';
       echo "git -C /home/afritzsche/ansible215_test push --set-upstream origin $new_branch";
       git -C /home/afritzsche/ansible215_test push --set-upstream origin $new_branch;
       if [[$? -gt 0 ]];
       then
         echo "Abbruch";
         break;
       fi;

       echo " curl -k --insecure -X POST -H \"Content-Type: application/json\" --header 'Accept:
       application/json' -u andreas.fritzsche@xxxxx.de
       https://git.xxxx.de/rest/api/1.0/projects/ProjektA/repos/ansible215/pull-requests -d ' {
       \"title\": \"$titel\", \"description\": \"$description\", \"state\": \"OPEN\", \"open\": true,
       \"closed\": false, \"fromRef\": { \"id\": \"refs/heads/$new_branch\", \"repository\": {
       \"slug\": \"ansible215\", \"name\": \"$new_branch\", \"project\": { \"key\": \"ProjektA\" } } },
       \"toRef\": { \"id\": \"refs/heads/main\", \"repository\": { \"slug\": \"ansible215\", \"name\":
       \"main\", \"project\": { \"key\": \"ProjektA\" } } }, \"locked\": false, \"reviewers\": [ {
       \"user\": { \"name\":\"Max.Mueller@xxxxx.de\", \"emailAddress\":\"Max.Mueller@xxxxx.de\",
       \"id\":15381, \"displayName\":\"Max Mueller/xxxxx\", \"active\":true,
       \"slug\":\"Max.Mueller_xxxxx.de\", \"type\":\"NORMAL\", \"links\":{ \"self\":[
       {\"href\":\"https://git.xxxx.de/users/Max.Mueller_xxxxx.de\"}]},
       \"avatarUrl\":\"https://git.xxxx.de/users/Max.Mueller_xxxxx.de/avatar.png?s=192&v=1696587440389\"}
        ] ] }' " | bash;
       fi  ; done
```

SLAC 2025

Motivation

bitbucket API

https://developer.atlassian.com/cloud/bitbucket/rest/intro/#authentication

https://www.postman.com

SLAC 2025

Da ist ein Wort der Warnung angebracht:

Da das S in SLAC für Sicherheit steht!!!

Zugangsdaten besser nicht in Skripten und niemals Veröffentlichen

- auf Zugangsdaten scannen(git hook)

- Zugangsdaten auslagern
  file... gitignore
  Passworttressor

- Verschlüsseln (git-crypt, git-secrets ...)

- ...

Wohin mit den Zugangsdaten SLAC 2026 ;-)

SLAC 2025

Da ist ein Wort der Warnung angebracht:

Da das S in SLAC für Sicherheit steht!!!

API absichern

- **Nicht nur API KEY → Authentifizierung**
- **API Gateway**
- **RBAC**
- **Verschlüsseln (https)**
- **...**

API, Jetzt endlich Sicher SLAC 2026 ;-)


Fragen?

SLAC 2025

https://gitlab.com/1fridy/apis_fuer_administrierende

SLAC 2025

Danke

CC by SA, Andreas Fritzsche