# AAP11 RESTful APIs in Python V2.1

# Überblick

Eine API (Application Programming Interface) ist eine Black Box, an die andere Programme Anfragen (requests) stellen können, um Daten zu bekommen oder Funktionen zu nutzen.

Eine REST API ist eine API, die den Designprinzipien einer REST-Architektur folgt. REST ist die Abkürzung für „Representational State Transfer“ und bezeichnet verschiedene Regeln und Richtlinien, die festlegen, wie eine Web-API aufgebaut sein sollte.

# Was ist REST?

REST ist eine Sammlung an Prinzipien, an die man sich als Entwickler halten kann. Dabei geht es darum, wie Ressourcen/Daten repräsentiert werden, wie diese übers Netzwerk übertragen und dargestellt werden und wie sie upgedatet werden können.

Schlüssel Prinzipien von REST:

1. Ressourcenorientiert (resource-oriented)
2. Zustandslos (stateless): Clients stellen requests an Server. Server speichern keine Informationen über bisherige requests. Nach dem Request ist vor dem Request.
3. HTTP Methoden: GET, POST, PUT, DELETE, PATCH.
4. Einheitliche, konsistente URL-Syntax für die Ressourcen

|  |
| --- |
| **HTTP-Methoden**  **GET:** Retrieve data from an endpoint.  **POST:** Send data and create a new resource.  **PUT:** Update an existing resource.  **DELETE:** Remove a resource.  **PATCH:** Partial updates to a resource. |

Wenn eine Client-Anfrage über eine RESTful API gestellt wird, überträgt sie die request Daten an den Server. Diese Informationen oder Darstellungen werden in einem von mehreren Formaten über **HTTP** bereitgestellt: JSON (Javascript Object Notation), HTML, XLT, Python, PHP oder als einfacher Text. JSON ist hierbei die beliebteste Methode, weil es von Menschen als auch von Maschinen gelesen werden kann.

# Vorteile von RESTful APIs:

1. Einfache Nutzung durch HTTP-Standardmethoden wie GET, POST, etc.
2. Ressourcenorientiert und daher nutzbar von vielen verschiedenen Geräten und Betriebssystemen.
3. Bestehende Frameworks stellen Tools zur Authentifizierung und zur Kommunikation mit vielen Nutzern zur Verfügung.
4. Bestehende Bibliotheken: Entwicklung einer RESTful-API zum Beispiel in python mit Flask

(pip install Flask)

1. Öffentliches, einfaches Interface zur Interaktion mit der API
2. Trennung von Anwendungs-Logik, Datentransfer übers Internet und Darstellung
3. Skalierbarkeit: viele gleichzeitige Anfragen beantwortbar. Systeme mit steigenden Anforderungen nach oben hin erweiterbar.

Programmierung einer Flask RESTful API in python: <https://www.youtube.com/watch?v=z3YMz-Gocmw>



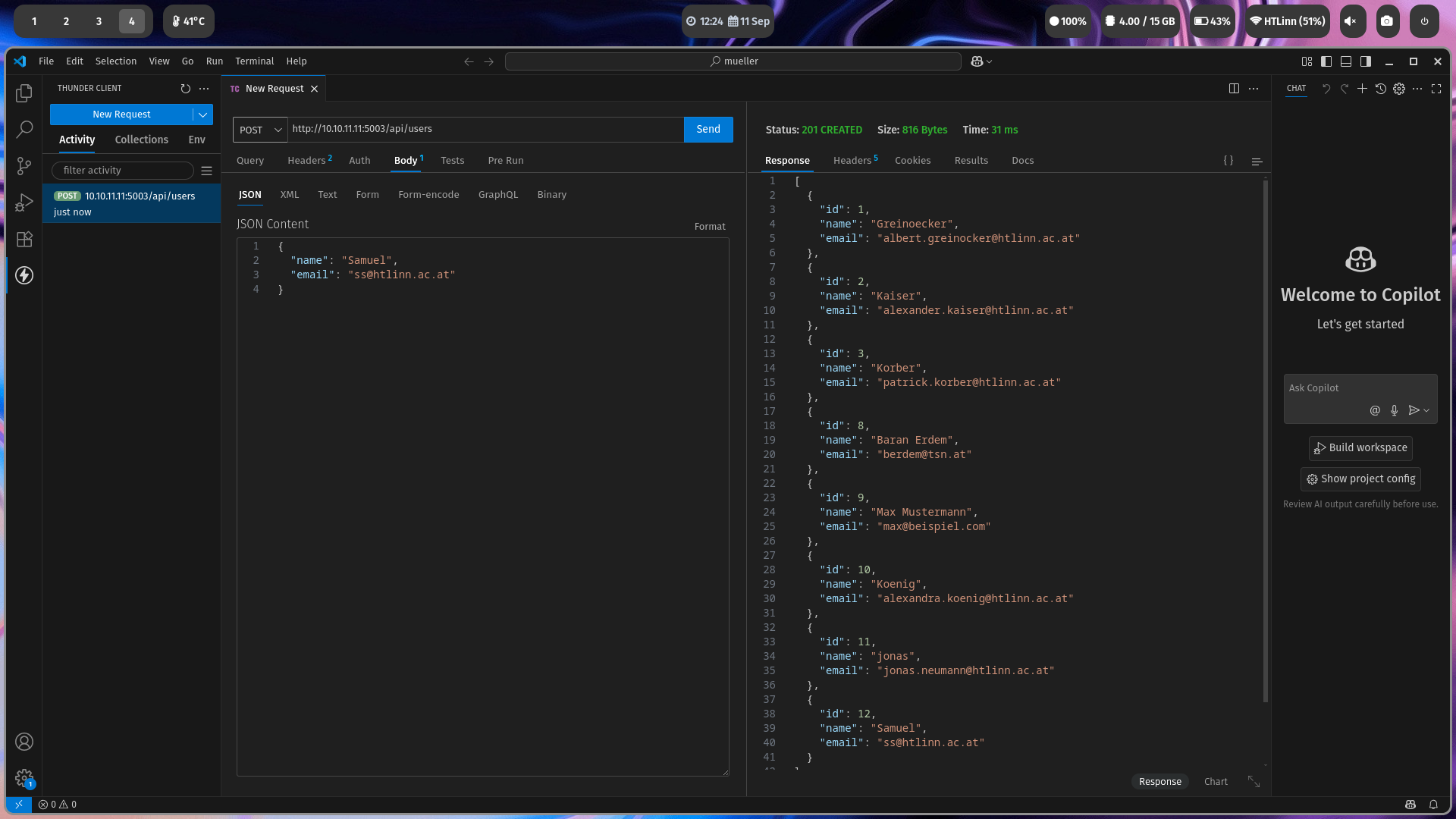
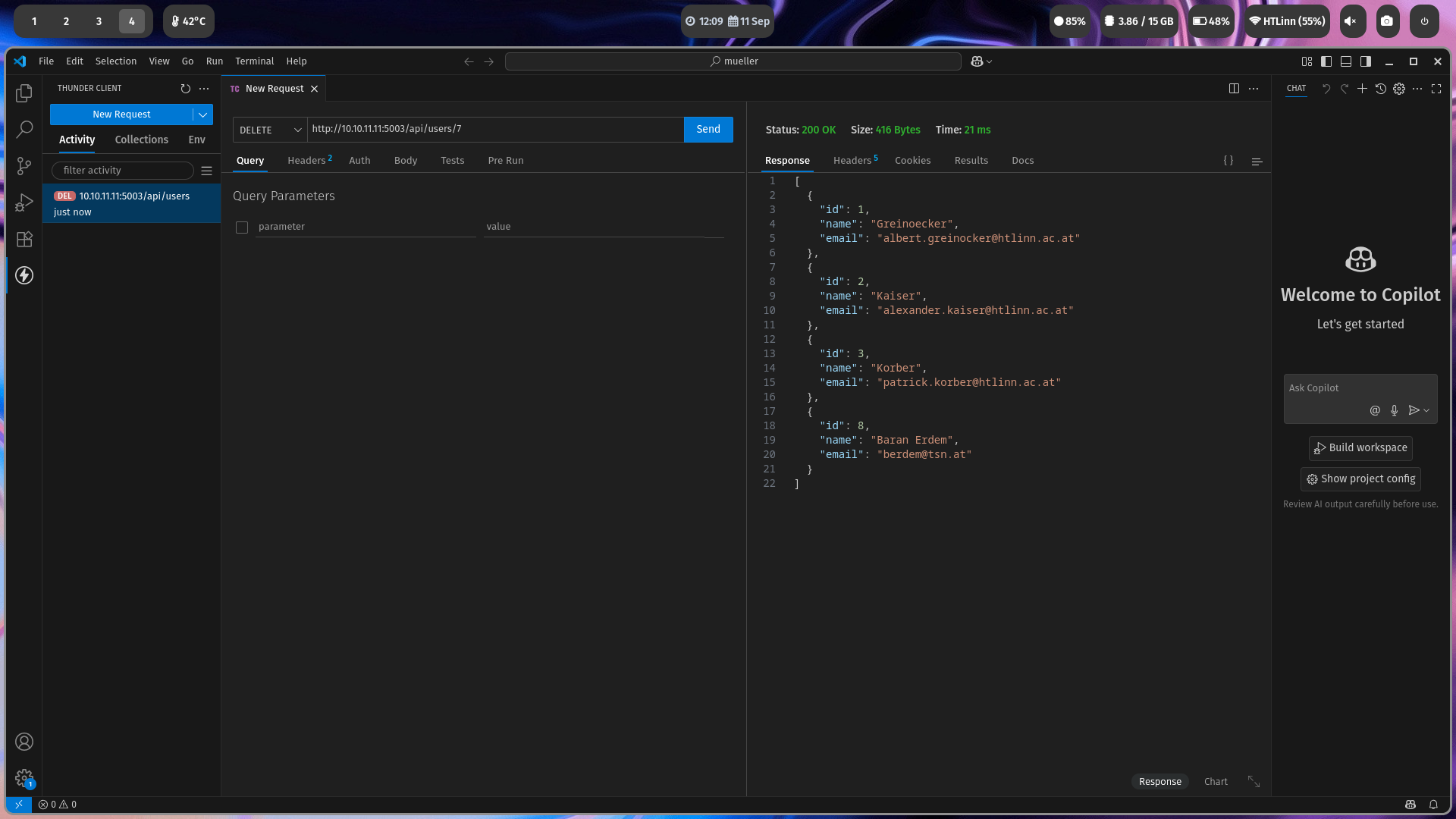
Der Flask Server von api.py aus dem Video läuft bereits auf dem Server 10.10.11.11 und ist unter dem Port 5003 erreichbar. Die API erreichst du also mit der URL <http://10.10.11.11:5003/api/users>



### RESTful API nutzen

Installiere die **Thunderclient** extension in VS-Code und öffne sie (Blitz Symbol in der linken Leiste).

Klicke auf „New Request“ und führe GET, POST und DELETE requests durch! Dokumentiere die Ergebnisse, in dem du ein Video Tutorial erstellst mit der „Aufzeichnen“ Funktion des Windows-Snipping Tools.



## Weitere Aufgabenstellungen:

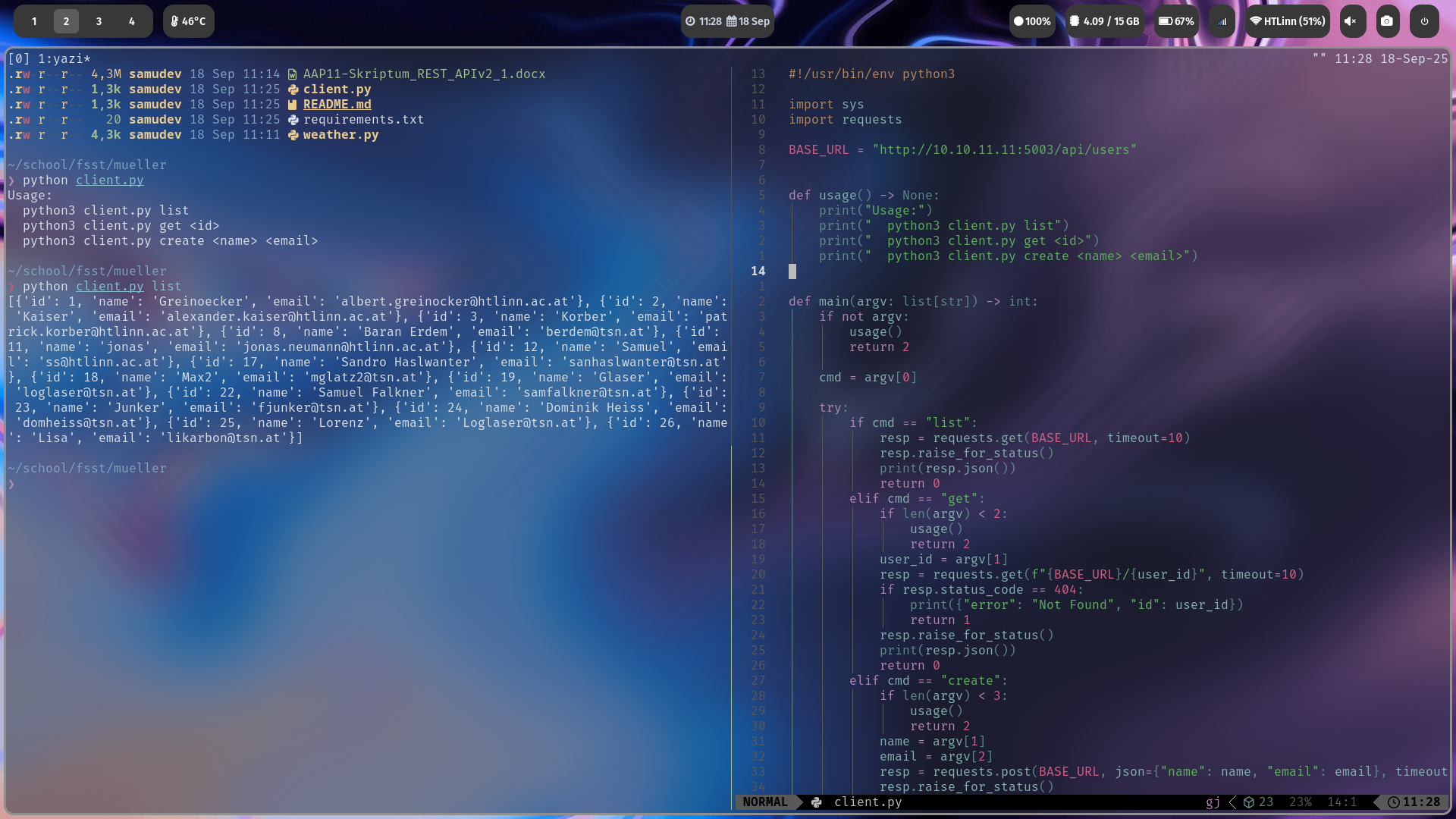
Bitte wähle nach eigenem Geschmack eine oder mehrere der Programmieraufgaben aus und dokumentiere diese hier im Arbeitsauftrag.

### Programmierung eines RESTful Clients:

Erstelle ein Python-Programm mit der Request-Bibliothek mit dem du am obigen Server

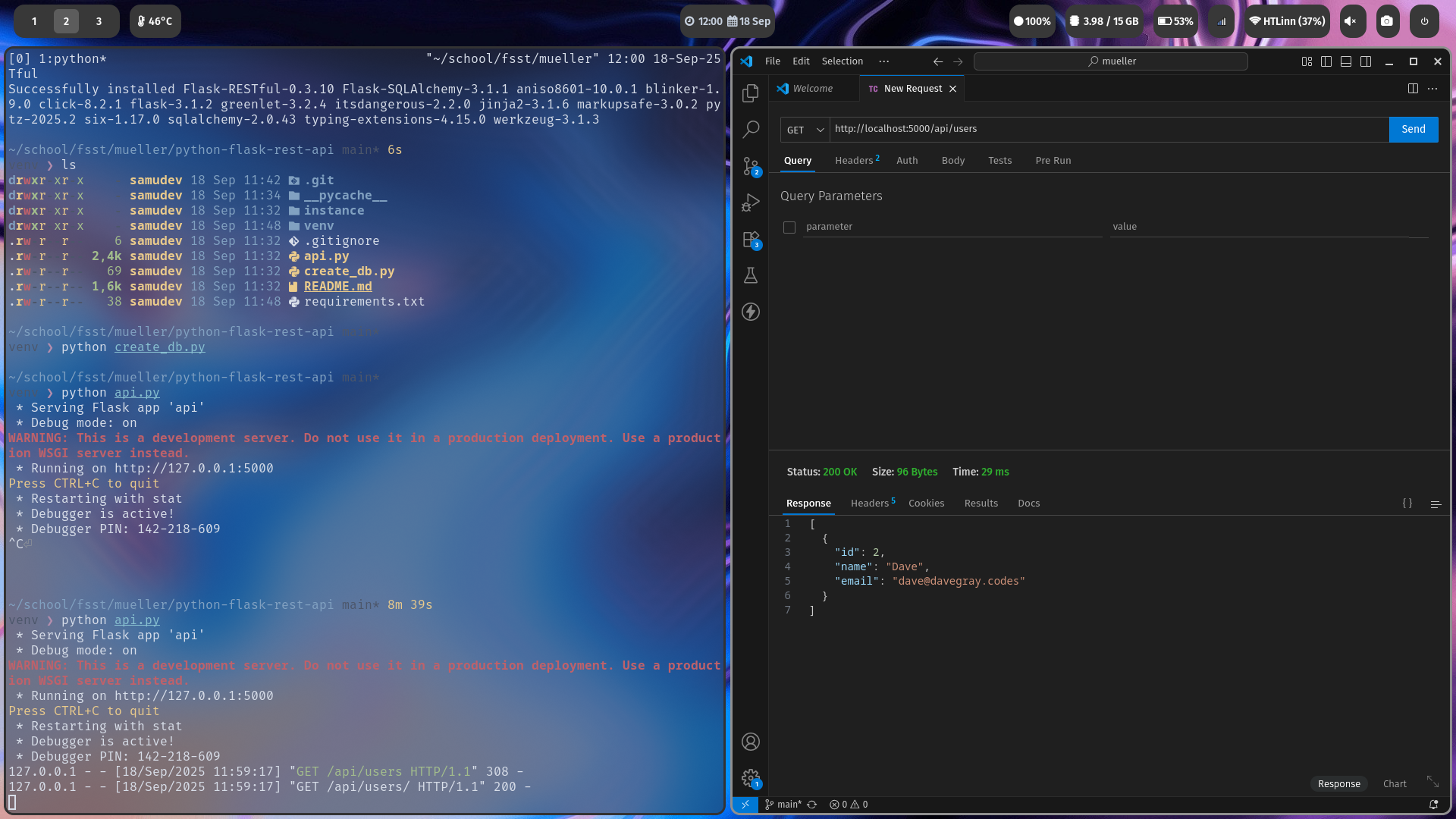
* Einen Datensatz anlegst
* Einen Datensatz ausgibst
* Datensätze suchst

Hinweise dazu findest du z.B. im RealPython Tutorial:  
<https://realpython.com/api-integration-in-python/#rest-and-python-consuming-apis>   
oder in meiner ChatGPT Recherche:  
<https://chatgpt.com/share/68c274c7-a1e4-8012-8ef3-fa8c33006d17>



### Realisierung des Servers aus dem Youtube-Tutorial am eigenen PC:

Siehe Git:  
<https://github.com/gitdagray/python-flask-rest-api>  
Dokumentiere dabei deine Schritte und Probleme.



Probleme beim installieren von requirements, sonst ohne probleme

### Verwende eine andere öffentliche RestAPI für ein Service deiner Wahl



Dokumentiere dabei deine Schritte und Probleme.

GEOCODE\_URL = "https://geocoding-api.open-meteo.com/v1/search"

FORECAST\_URL = "https://api.open-meteo.com/v1/forecast"

Chatgpt hat mir diese empfohlen, keine Probleme

## Kontrollfragen zum Verständnis:

* Welche Vorteile haben RESTful APIs? Standard http methoden
* Erkläre eine den Aufbau einer RESTful URL zur Abfrage des zweiten Datensatzes. Get /users/2
* Welche im allgemeinen Features von HTTPS können bei RESTful APIs „gratis“ mit verwendet werden? Verschlüsselung, server verifizierung, vertraulichkeit
* Was bedeutet stateless? Speichert keine zustände

Welche Vorteile bietet dies in Bezug auf die Serverarchitektur? Einfache skalierung

* Zu obigem Web-Tutorial:
  + Wie funktioniert ein Flask-Webserver?  
    Wann wird der gestartet? Läuft der immer? Wie erfolgt die Behandlung einzelner URLs? Web Server, bei python app.py, läuft solange die python datei läuft, mit routen oder eigenen Befehlen
  + Aus welcher Bibliothek stammt SQLAlchemy?  
    Was leistet diese Bibliothek? Python bibliothek, arbeit mit Datenbanken wie MySQL
  + SCHWIERIG: Was ist ein Decorator und was macht der @app.route() bzw. @marshal\_with() Decorator ist eine Funktion die eine Funktion erweitert, [@app.rout](http://@app.rout)e verbinded eine URL mit einer Route, marshal\_with formatiert automatisch

## Lerntagebuch:

Was habe ich wann und wie lange gemacht?

11.9 thunderclient, eigenen Restclient schreiben

18.9 eigenen Restclient fertigen, öffentliche API benutzen

Was hat Probleme bereitet? Typ festlegen (thunderclient)

Wobei habe ich am meisten gelernt?bei öffenlichen API’s benutzen

**Quellen:**

Aufbauend auf den Lesetext von <https://www.redhat.com/de/topics/api/what-is-a-rest-api>

©KaisA Mit deepseek und chatgpt 4.0 überarbeitet  
©MuelW Erweiterung um Programmieraufgaben und Verständnisfragen