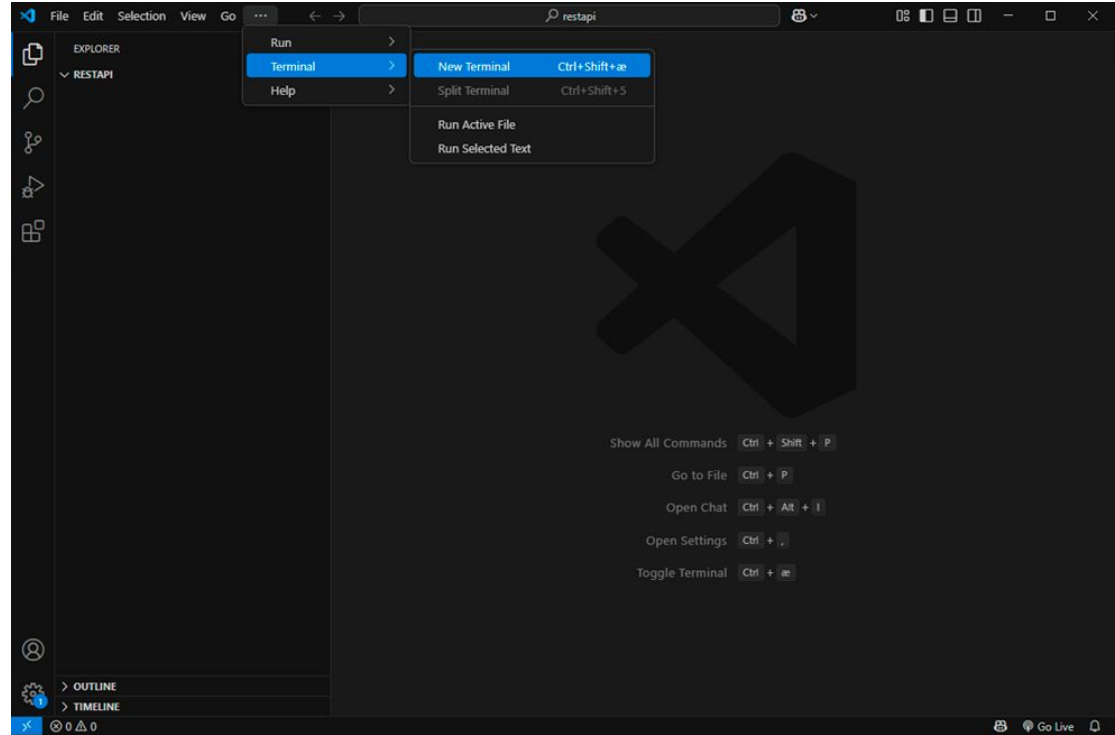


Python REST API

Setup

- Install Python in fx Microsoft Store
- Åben VS Code
- Installer Python Extension



Virtuelt miljø

- Opret og aktivér virtuelt miljø

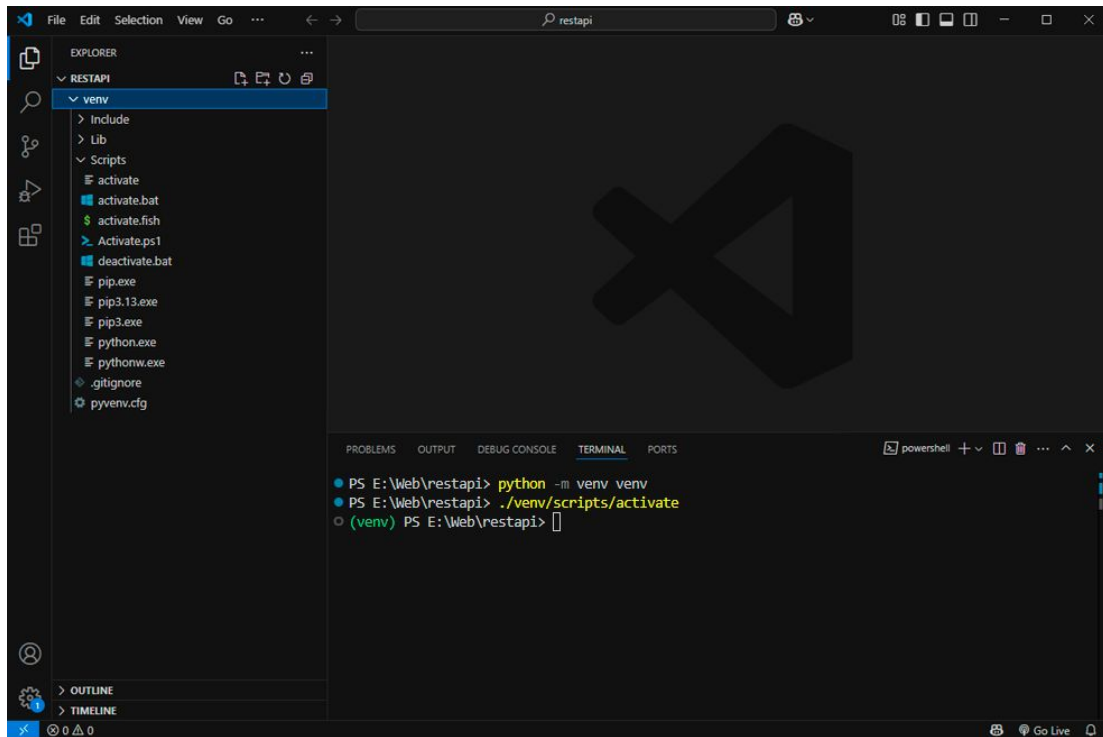
python -m venv name

- Lokal installation af Python pakker
- Isoleret pakkehåndtering

- Aktivér miljø:

./name/scripts/activate

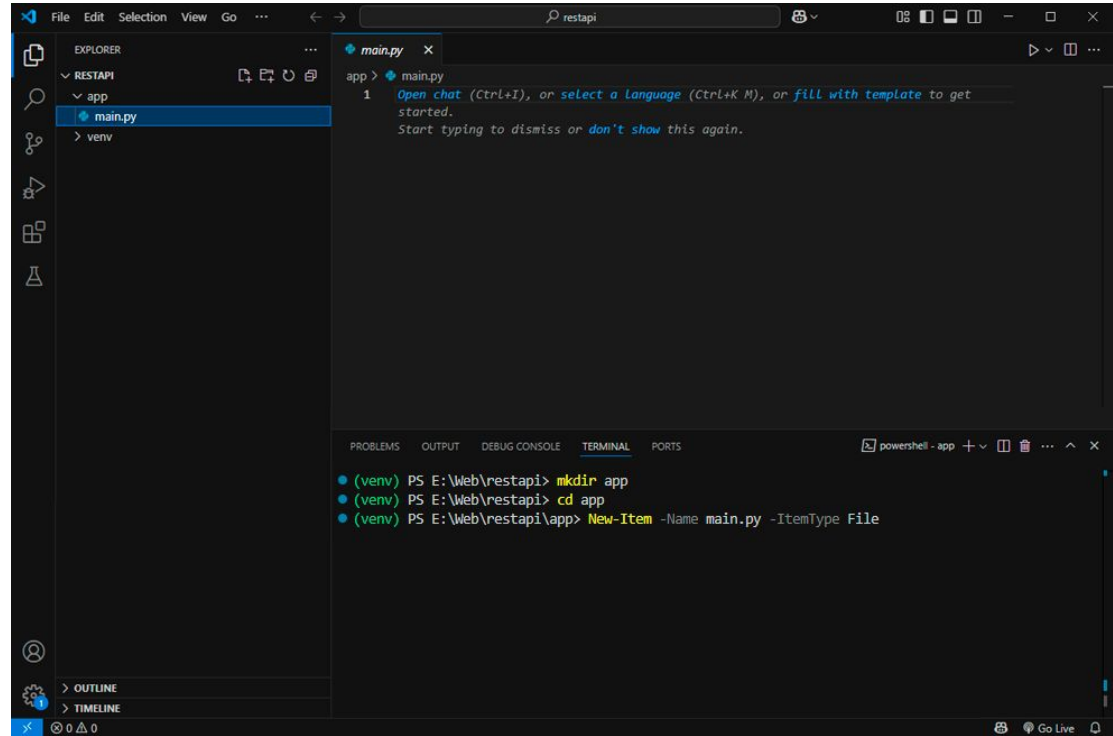
- Nu bruger vi python og pip i dette miljø



Opret projektstruktur

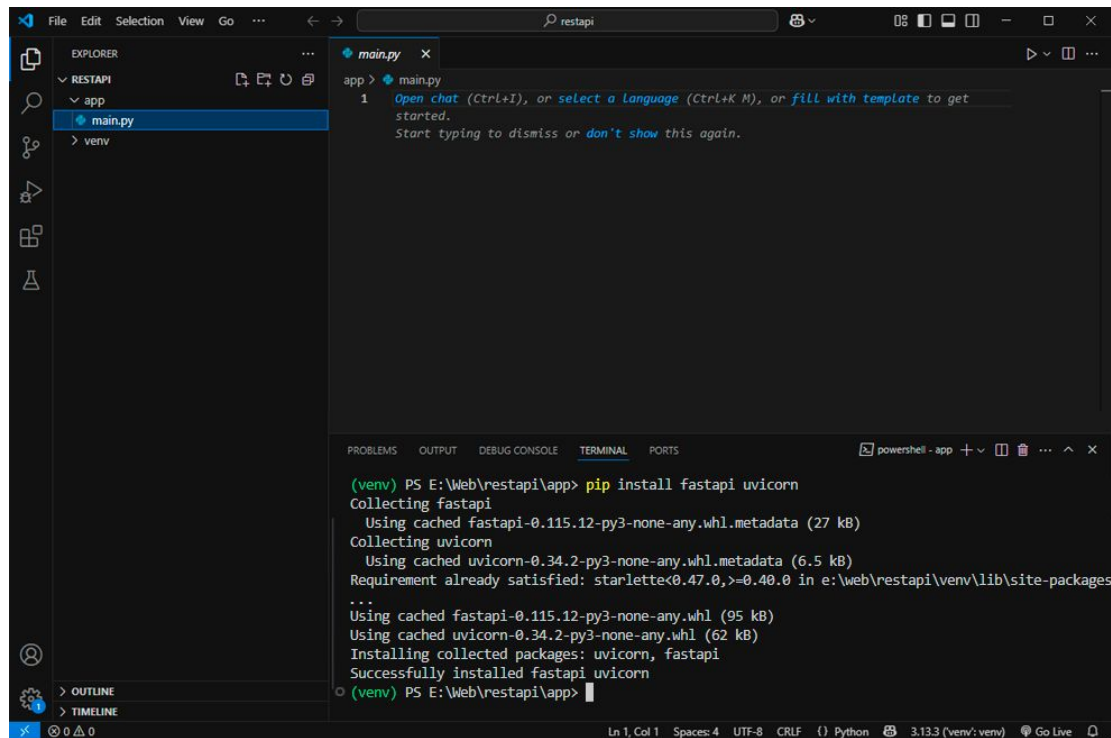
- app mappe
- main.py fil ← API kode

*New-Item -Name main.py
-ItemType File*



Fast API

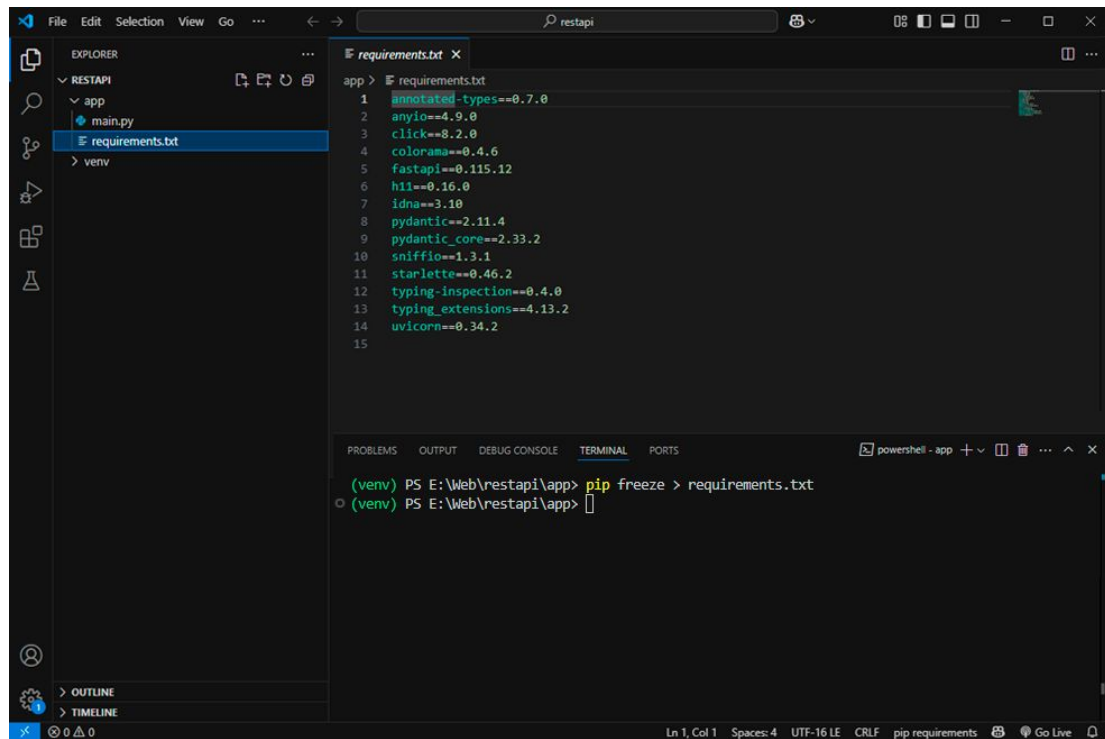
- Installer fast API pakker
 - `pip install fastapi uvicorn`*
 - Asynkron kode
 - `async`, `await`
 - Automatisk datavalidering
 - Auto tjek om dataene passer vores beskrivelse
 - Automatisk dokumentation
 - Genererer swagger UI til vores endpoint
 - `http://localhost:8000/docs`



```
restapi
EXPLORER
  RESTAPI
    app
      main.py
      > venv
main.py
1 Open chat (Ctrl+I), or select a language (Ctrl+K M), or fill with template to get started.
  Start typing to dismiss or don't show this again.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS E:\Web\restapi\app> pip install fastapi uvicorn
Collecting fastapi
  Using cached fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting uvicorn
  Using cached uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: starlette<0.47.0,>=0.40.0 in e:\web\restapi\venv\lib\site-packages
...
Using cached fastapi-0.115.12-py3-none-any.whl (95 kB)
Using cached uvicorn-0.34.2-py3-none-any.whl (62 kB)
Installing collected packages: uvicorn, fastapi
Successfully installed fastapi uvicorn
(venv) PS E:\Web\restapi\app>
```

Miljøstyring

- Requirements



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'RESTAPI' with a subdirectory 'app' containing 'main.py' and 'requirements.txt'. The 'requirements.txt' file is selected and its contents are displayed in the main editor. The file contains a list of Python dependencies with version constraints. Below the editor, the TERMINAL panel is active, showing a PowerShell prompt in a virtual environment. The user has entered the command 'pip freeze > requirements.txt' to save the current installed dependencies to the requirements file.

```
requirements.txt
1 annotated-types==0.7.0
2 anyio==4.9.0
3 click==8.2.0
4 colorama==0.4.6
5 fastapi==0.115.12
6 h11==0.16.0
7 idna==3.10
8 pydantic==2.11.4
9 pydantic_core==2.33.2
10 sniffio==1.3.1
11 starlette==0.46.2
12 typing-inspection==0.4.0
13 typing_extensions==4.13.2
14 uvicorn==0.34.2
15

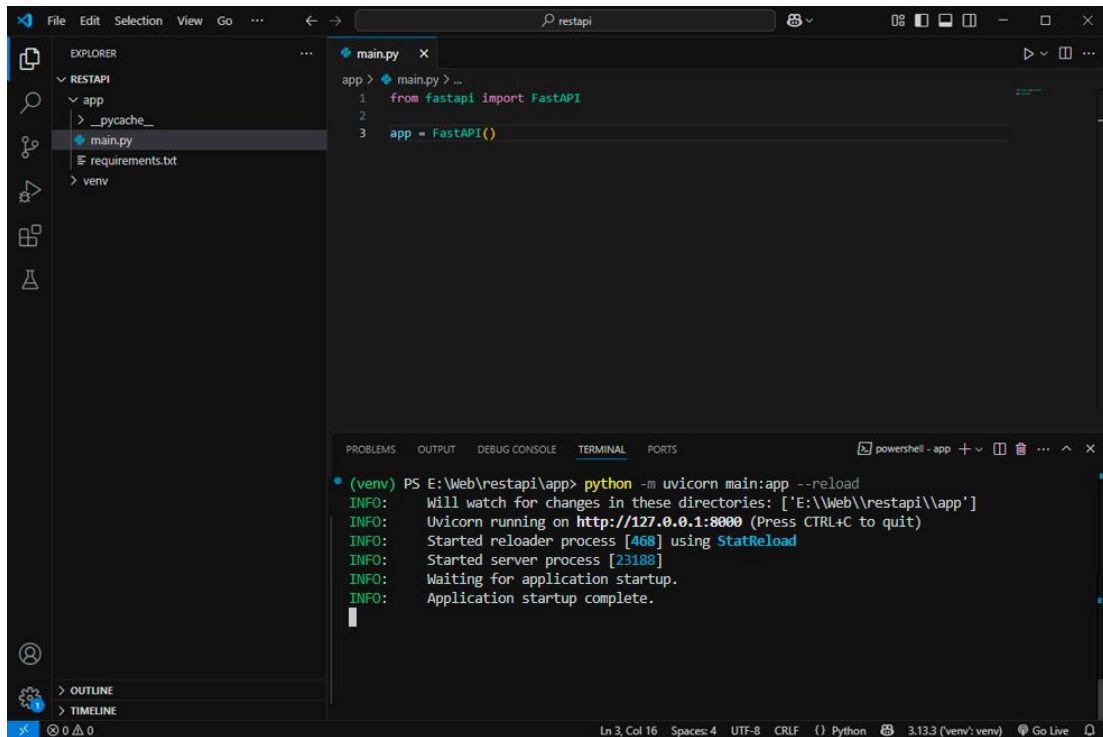
(venv) PS E:\Web\restapi\app> pip freeze > requirements.txt
(venv) PS E:\Web\restapi\app>
```

Test API

- Import dependencies
- Opret Fast API app
- Run API (+ auto genindlæs server)

```
python -m uvicorn  
main:app --reload
```

```
fastapi dev
```



The screenshot shows a Visual Studio Code editor window with a project named 'restapi'. The Explorer sidebar on the left shows a folder 'app' containing files like '__pycache__', 'main.py', 'requirements.txt', and 'venv'. The main editor area shows the 'main.py' file with the following code:

```
1 from fastapi import FastAPI
2
3 app = FastAPI()
```

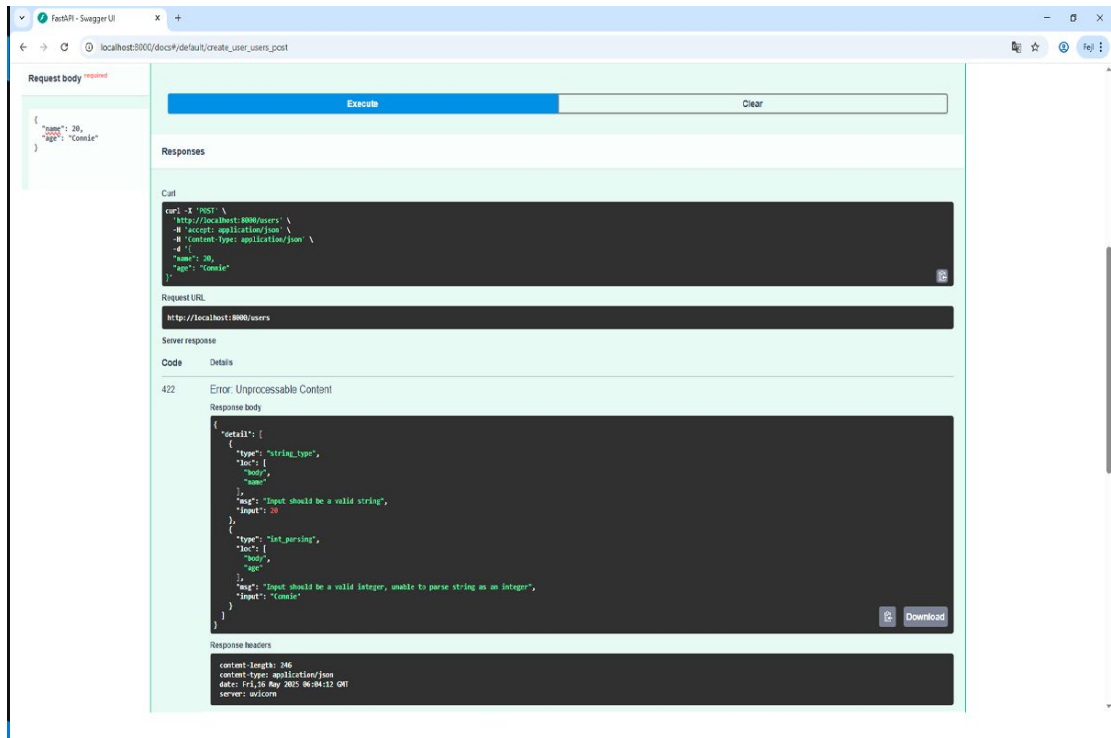
Below the editor, the TERMINAL panel is open, showing the command prompt output for running the application:

```
(venv) PS E:\Web\restapi\app> python -m uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['E:\Web\restapi\app']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [468] using StatReload
INFO: Started server process [23188]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

The status bar at the bottom indicates the current file is 'Ln 3, Col 16', using 'UTF-8' encoding, 'CRLF' line endings, and 'Python' language. It also shows the Python version as '3.13.3 (venv: venv)' and a 'Go Live' button.

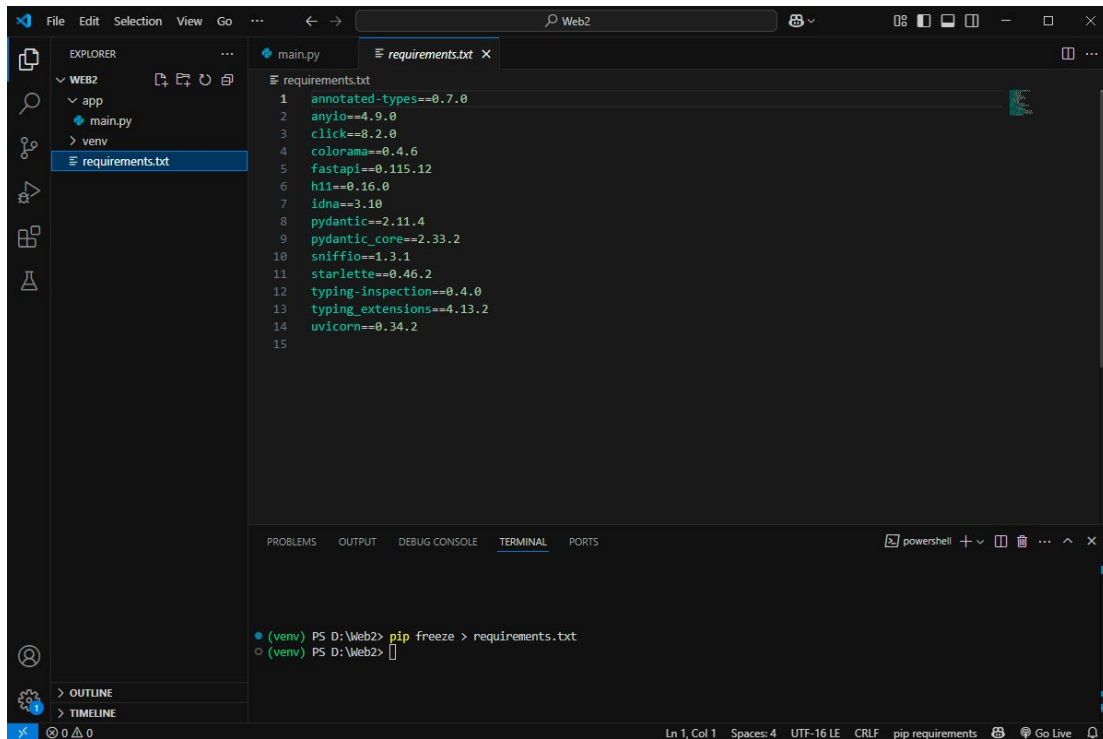
Test API

- Auto dokumentation
 - localhost:8000/docs
- Auto datavalidering
 - Send ugyldig forespørgsel



Ekstra

- *pip install fastapi[standard]*
 - fastapi dev
- requirements
 - fastapi
 - uvicorn
 - pydantic



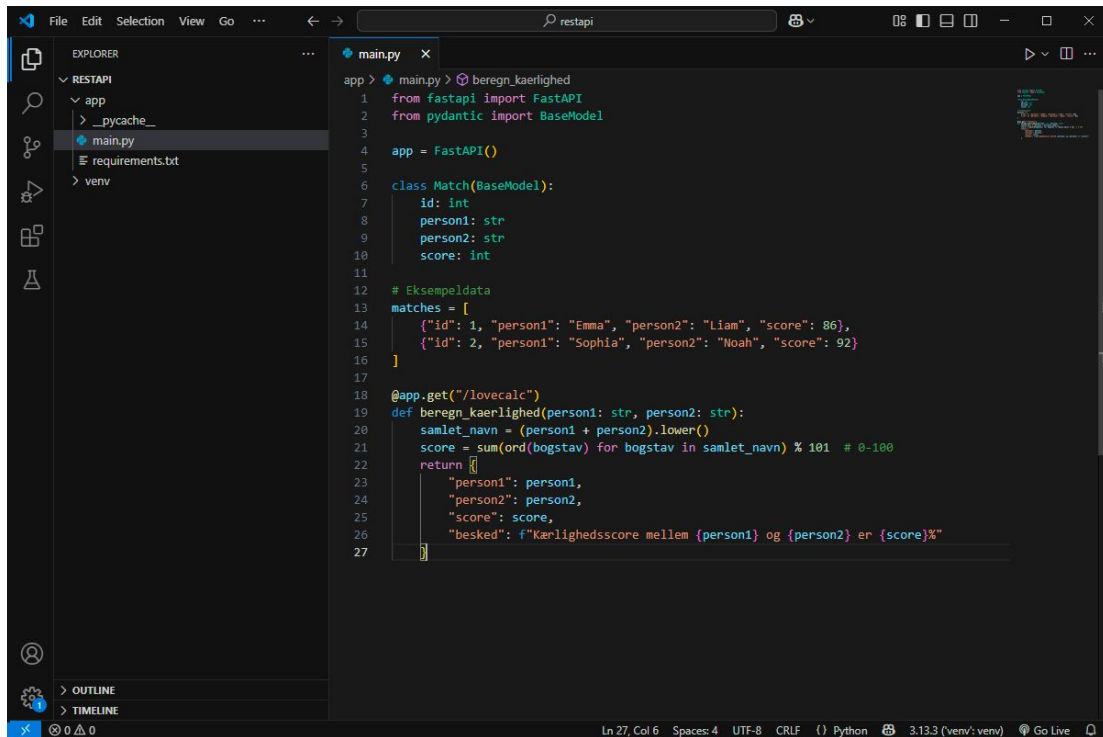
The screenshot shows the Visual Studio Code interface with a project named 'Web2'. The Explorer sidebar on the left shows the file structure: 'Web2' (folder), 'app' (folder), 'main.py' (file), and 'venv' (folder). The 'requirements.txt' file is selected and open in the editor. The file contains the following dependencies:

```
1 annotated-types==0.7.0
2 anyio==4.9.0
3 click==8.2.0
4 colorama==0.4.6
5 fastapi==0.115.12
6 h11==0.16.0
7 idna==3.10
8 pydantic==2.11.4
9 pydantic_core==2.33.2
10 sniffio==1.3.1
11 starlette==0.46.2
12 typing-inspection==0.4.0
13 typing_extensions==4.13.2
14 uvicorn==0.34.2
15
```

The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt. The command `pip freeze > requirements.txt` has been executed, and the output is visible in the terminal.

GET

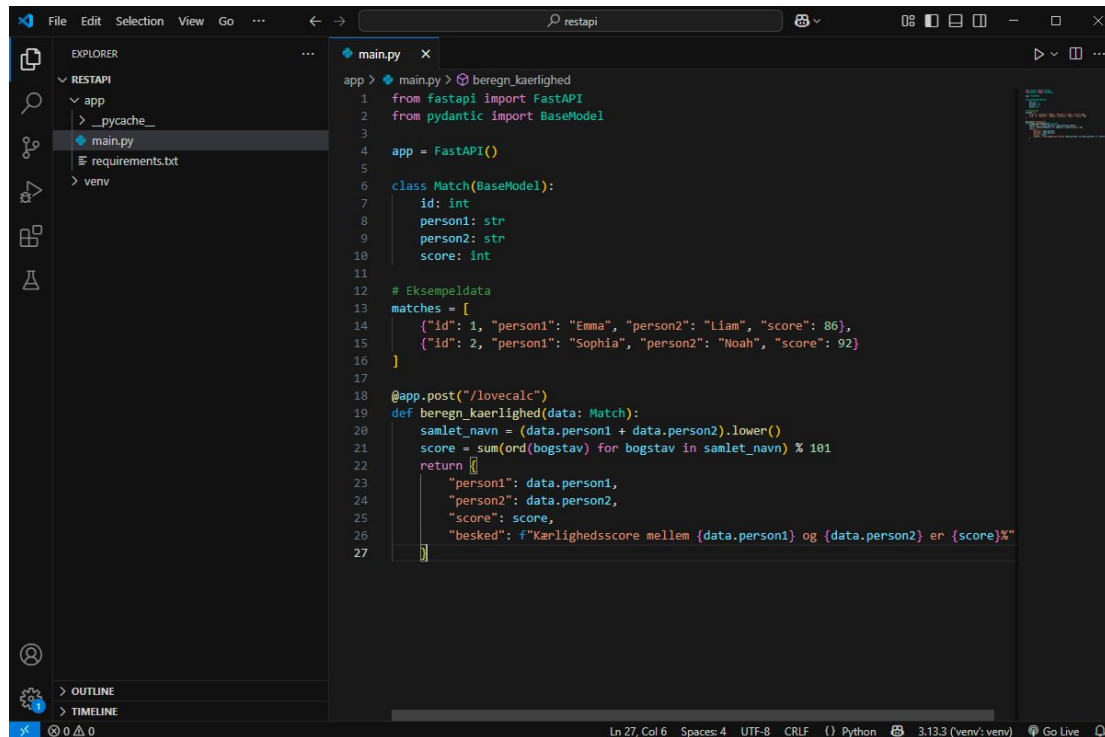
- GET /matches
- GET /lovecalc



```
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3
4 app = FastAPI()
5
6 class Match(BaseModel):
7     id: int
8     person1: str
9     person2: str
10    score: int
11
12 # Eksempeldata
13 matches = [
14     {"id": 1, "person1": "Emma", "person2": "Liam", "score": 86},
15     {"id": 2, "person1": "Sophia", "person2": "Noah", "score": 92}
16 ]
17
18 @app.get("/lovecalc")
19 def beregn_kaerlighed(person1: str, person2: str):
20     samlet_navn = (person1 + person2).lower()
21     score = sum(ord(bogstav) for bogstav in samlet_navn) % 101 # 0-100
22     return {
23         "person1": person1,
24         "person2": person2,
25         "score": score,
26         "besked": f"K erlighedsscore mellem {person1} og {person2} er {score}%"
27     }
```

POST

- POST /lovecalc



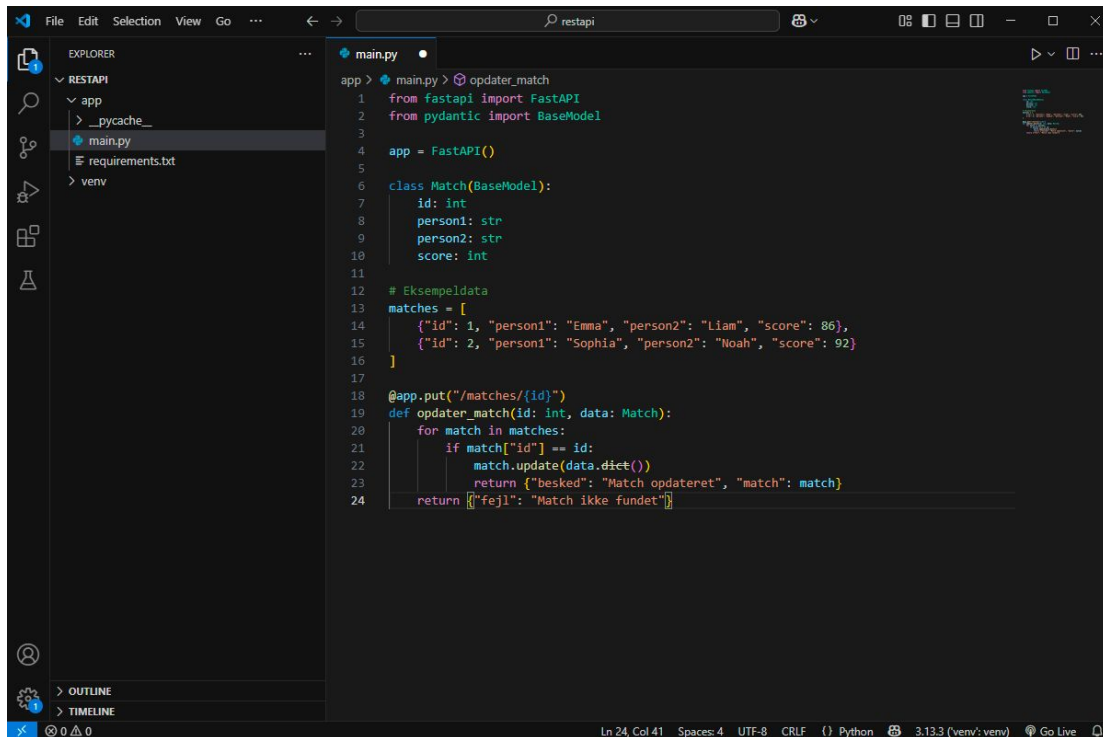
The screenshot shows a Visual Studio Code editor window with a project named 'RESTAPI'. The Explorer sidebar on the left shows the file structure: 'app' (containing '__pycache__'), 'main.py', 'requirements.txt', and 'venv'. The main editor window displays the code in 'main.py'.

```
app > main.py > beregn_kaerlighed
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3
4 app = FastAPI()
5
6 class Match(BaseModel):
7     id: int
8     person1: str
9     person2: str
10    score: int
11
12 # Eksempeldata
13 matches = [
14     {"id": 1, "person1": "Emma", "person2": "Liam", "score": 86},
15     {"id": 2, "person1": "Sophia", "person2": "Noah", "score": 92}
16 ]
17
18 @app.post("/lovecalc")
19 def beregn_kaerlighed(data: Match):
20     samlet_navn = (data.person1 + data.person2).lower()
21     score = sum(ord(bogstav) for bogstav in samlet_navn) % 101
22     return {
23         "person1": data.person1,
24         "person2": data.person2,
25         "score": score,
26         "besked": f"K rlighedsscore mellem {data.person1} og {data.person2} er {score}%"
27     }
```

The status bar at the bottom indicates the cursor is at line 27, column 6, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.3, and the 'venv' environment.

PUT

- PUT /matches{id}

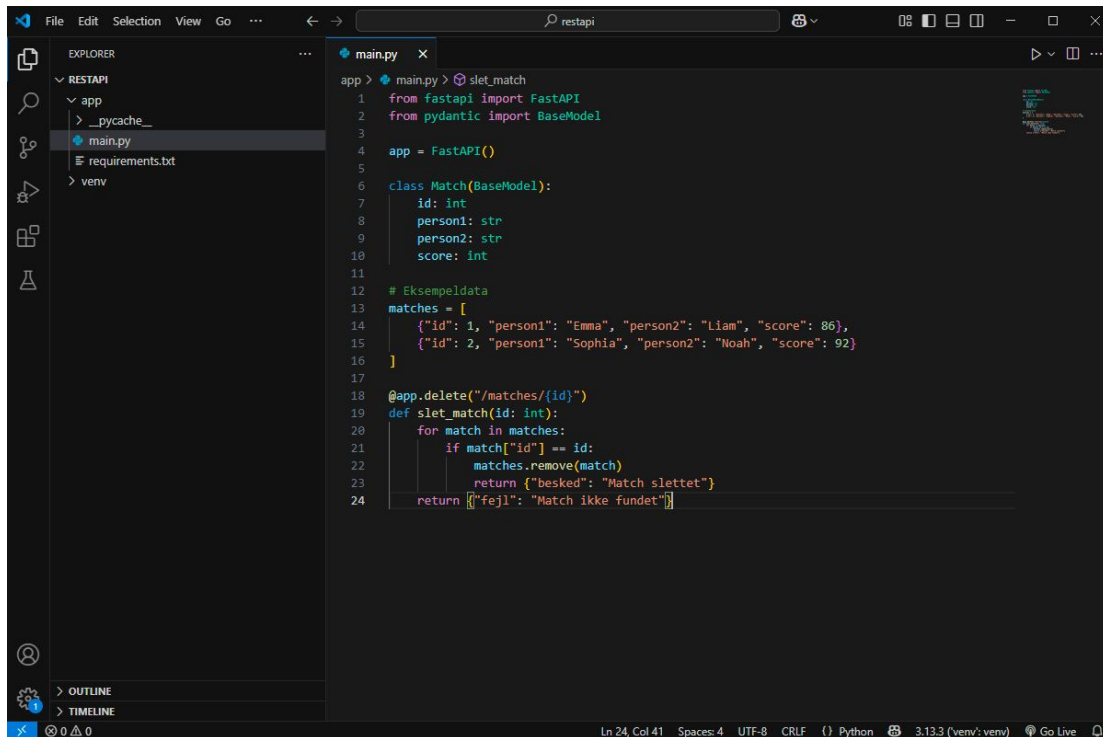


The screenshot shows a VS Code editor with a REST API client on the left and a Python FastAPI server code on the right. The REST API client is named 'restapi' and shows a PUT request to '/matches{id}'. The Python code is in 'main.py' and defines a FastAPI application with a 'Match' model and an 'opdater_match' endpoint.

```
app > main.py > opdater_match
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3
4 app = FastAPI()
5
6 class Match(BaseModel):
7     id: int
8     person1: str
9     person2: str
10    score: int
11
12 # Eksempeldata
13 matches = [
14     {"id": 1, "person1": "Emma", "person2": "Liam", "score": 86},
15     {"id": 2, "person1": "Sophia", "person2": "Noah", "score": 92}
16 ]
17
18 @app.put("/matches/{id}")
19 def opdater_match(id: int, data: Match):
20     for match in matches:
21         if match["id"] == id:
22             match.update(data.dict())
23             return {"besked": "Match opdateret", "match": match}
24     return {"fej1": "Match ikke fundet"}
```

DELETE

- DELETE /matches{id}



The screenshot shows a VS Code editor with a REST API client on the left and a Python FastAPI endpoint on the right. The REST API client is configured for a REST API named 'restapi' with a base URL of 'http://localhost:8000'. The endpoint is 'DELETE /matches/{id}'. The Python code defines a FastAPI application with a 'Match' model and a 'slet_match' endpoint that deletes a match from a list of matches.

```
1 from fastapi import FastAPI
2 from pydantic import BaseModel
3
4 app = FastAPI()
5
6 class Match(BaseModel):
7     id: int
8     person1: str
9     person2: str
10    score: int
11
12 # Eksempeldata
13 matches = [
14     {"id": 1, "person1": "Emma", "person2": "Liam", "score": 86},
15     {"id": 2, "person1": "Sophia", "person2": "Noah", "score": 92}
16 ]
17
18 @app.delete("/matches/{id}")
19 def slet_match(id: int):
20     for match in matches:
21         if match["id"] == id:
22             matches.remove(match)
23             return {"besked": "Match slettet"}
24     return {"fej1": "Match ikke fundet"}
```

Dokumentation

- <https://fastapi.tiangolo.com/> ← Dokumentation til at bygge API
- <https://docs.python.org/3/> ← Python dokumentation og kommandoer
- <https://docs.python.org/3/library/venv.html> ← Virtuelt miljø
- <https://medium.com/geekculture/javascript-vs-python-syntax-cheatsheet-9bc7c59599c6> ← js vs python cheatsheet