Εργασία στο OpenApi, με τη βοήθεια του λογισμικού FastAPI

Δ. Ι. Βέργαδος,
Αναπληρωτής Καθηγητής,
Τμήμα Πληροφορικής,
Πανεπιστήμιο Δυτικής Μακεδονίας
(dvergados@uowm.gr)

5 Απριλίου 2025

Περίληψη

Η υποβολή της εργασία γίνεται μέσω της ηλεκτρονικής τάξης του μαθήματος.
Τα πηγαία αρχεία αυτής της εργασίας βρίσκονται στο σύνδεσμο https://netsim.cs.uowm.gr/gitlab/dvergados/fastapi-primes.

Περιεχόμενα

Κατάλογος Σχημάτων

1 Εισαγωγή

Σκοπός της εργασίας είναι να εκτελεστεί ένα σενάριο επικοινωνία Machine-to-Machine (M2M), χρησιμοποιώντας το πρότυπο OpenAPI.

Το OpenAPI https://swagger.io/specification/ είναι το de-facto standard για την επικοινωνία μεταξύ μηχανών (M2M). Είναι ένα ανοιχτό πρότυπο, το οποίο χρησιμοποιείται για τον ορισμό μιας διεπαφής μεταξύ δύο ή περισσότερων μηχανών, η οποία ακολουθεί τα πρότυπα REST API.

Σε αντίθεση με τις διεπαφές που προορίζονται για χρήση από ανθρώπους, όπου δεν απαιτείται απόλυτη ακρίβεια στον ορισμό της διεπαφής, καθώς οι χρήστες μπορούν να αντιληφθούν τον τρόπο χρήσης της εύκολα, στην περίπτωση της επικοινωνίας M2M πρέπει όλα τα στοιχεία να είναι πλήρως ορισμένα, ώστε να μπορεί ο πελάτης να λειτουργεί αυτόματα, χωρίς ανθρώπινη παρέμβαση.

Μεταξύ των στοιχείων που πρέπει να προσδιοριστούν, μεταξύ άλλων, είναι:

- Τα πρωτόκολλα επικοινωνίας που θα χρησιμοποιηθού (πχ HTTP, TLS, κτλ)
- Οι διευθύνσεις των διακομιστών που θα παρέχουν την υπηρεσία
- Οι ακριβείς διευθύνσεις - τοποθεσίες (URLs) των πόρων που θα είναι προσβάσιμοι μέσω της διεπαφής
- Οι λειτουργίες (verbs) που θα επιτρέπονται σε κάθε τοποθεσία
- Οι παράμετροι που θα είναι διαθέσιμοι σε κάθε λειτουργία, καθώς και ο τύπος τους
- Η μορφοποίηση των επιστρεφόμενων δεδομένων

Με το OpenAPI, όλες αυτές οι παράμετροι, καθώς και πολλές ακόμα, μπορούν να καθοριστούν με τρόπο που είναι σαφώς ορισμένος μέσα από το πρότυπο. Ο ορισμός αυτός έχει τη μορφή ενός αρχείου (YAML ή JSON), του οποίου η δομή είναι γραμμένη σύμφωνα με το πρότυπο.

Καθώς το OpenAPI έχει γίνει το de-facto standard, έχουν αναπτυχθεί πολλά εργαλεία τα οποία επιτρέπουν την εύκολη ανάπτυξη εφαρμογών. Ένα από αυτά τα εργαλεία είναι το FastAPI.

Το FastAPI https://fastapi.tiangolo.com/ είναι μια βιβλιοθήκη σε python, η οποία επιτρέπει την υλοποίηση server για M2M επικοινωνία σε γλώσσα python. Έχει το χαρακτηριστικό ότι ο προγραμματιστής γράφει την υλοποίηση των μεθόδων, και η βιβλιοθήκη παράγει αυτόματα το OpenApi specification που πρέπει να χρησιμοποιήσουν οι clients για να χρησιμοποιήσουν τις υπηρεσίες που υλοποιεί ο διακομιστής. Κατά συνέπεια, με τη χρήση του FastAPI είναι βέβαιο ότι η υλοποίηση θα συμβαδίζει πάντα με το αρχείο ορισμού.

2 Ζητούμενα

Σε αυτή την άσκηση θα υλοποιήσουμε επικοινωνία M2M μεταξύ ενός server και ενός client, χρησιμοποιώντας REST API, το οποίο θα οριστεί μέσω του OpenAPI.

Το παράδειγμα επικοινωνίας θα υλοποιεί μια απομακρυσμένη κλήση για τον υπολογισμό μιας μαθηματικής πράξης, συγκεκριμένα τον υπολογισμός των πρώτων παραγόντων μιας ακολουθίας αριθμών.

Η μορφοποίηση των δεδομένων που θα ανταλλάσσονται θα είναι ως εξής:

- Ο client θα στέλνει στο server ένα POST request, όπου το σώμα του μηνύματος body θα είναι ως εξής:

```
{
  "input_numbers": [ 4, 6, 10, 20]
}
```

- Ο server, θα πραγματοποιεί τον υπολογισμό των πρώτων παραγόντων όλων των αριθμών που δέχεται, και θα επιστρέφει τα αντίστοιχα αποτελέσματα με μορφοποίηση ως εξής

```
{
  "result": [
    {
      "input_number": 4,
      "prime_factors": [2, 2]
    },{
      "input_number": 6,
      "prime_factors": [2, 3]
    },{
      "input_number": 10,
      "prime_factors": [2, 5]
```

```
    },{
      "input_number": 20,
      "prime_factors": [2, 2, 5]
    }
  ]
}
```

3 Μεθοδολογία Υλοποίησης

3.1 Εγκατάσταση του απαραίτητου λογισμικού

3.1.1 Εγκατάσταση της python

Η διαδικασία εγκατάστασης της python διαφοροποιείται ανάλογα με το λειτουργικό σύστημα του υπολογιστή σας:

3.1.1.1 Linux Δε χρειάζεται να κάνουμε κάτι, η γλώσσα είναι ήδη εγκατεστημένη

Επιβεβαιώνουμε την εγκατάσταση εκτελώντας τις παρακάνω εντολές:

```
$ python3 --version
Python 3.11.2

$ pip3 --version
pip 23.0.1 from /usr/lib/python3/dist-packages/pip (python 3.11)
```

3.1.1.2 Windows Ακολουθούμε τις οδηγίες από το σύνδεσμο https://docs.python.org/3/using/windows.html Ιδιαίτερη προσοχή οι παρακάτω επιλογές να είναι επιλεγμένες κατά την εγκατάσταση

1. `Add python to environment variables`: χρειάζεται για νσ μπορούμε να τρέχουμε προγράμματα από τη γραμμή εντολών
2. `pip`: είναι απαραίτητο για να μπορούμε να εγκαταστήσουμε τις βιβλιοθήκες που χρειαζόμαστε

Επιβεβαιώνουμε την ορθότητα της εγκατάστασης με τις ακόλουθες εντολές:

```
C:\Users\Dimitrios Vergados>python --version
Python 3.8.5

C:\Users\Dimitrios Vergados>pip --version
pip 20.1.1 from c:\users\dimitrios
 ↪  vergados\appdata\local\programs\python\python38\lib\site-packages\pip (python
 ↪  3.8)
```

3.1.1.3 Macintosh Ακολουθούμε τις οδηγίες: https://docs.python.org/3/using/mac.html.

3.1.2 Εγκατάσταση επεξεργαστή κειμένου

Για τον αποτελεσματικό προγραμματισμό της εφαρμογής προτείνεται η χρήση ενός Integrated Development Environment (IDE) που υποστηρίζει python.

Προτείνεται η χρήση του `vscode`. Η εγκατάσταση του, γίνεται σύμφωνα με τις οδηγίες από εδώ: https://code.visualstudio.com/download

3.1.3 Εγκατάσταση εικονικού περιβάλλοντος (virtual environment - venv)

Τα εικονικά περιβάλλοντα (virtual environments) επιτρέπουν την εγκατάσταση βιβλιοθηκών της python, με τρόπο που διατηρεί την εγκατάσταση εντός του περιβάλλοντος, ώστε να μην επηρεάζεται μια εφαρμογή από τις βιβλιοθήκες άλλων εφαρμογών.

3.1.3.1 Δημιουργία εικονικού περιβάλλοντος

Για να δημιουργήσουμε το εικονικό περιβάλλον της εφαρμογής μας, αρχικά δημιουργούμε ένα κατάλογο (φάκελο) εργασίας, όπου θα υπάρχουν τα αρχεία μας.

Μέσα σε αυτό τον κατάλογο, εκτελούμε την εντολή

```
python3 -m venv venv # Linux
```

ή σε Windows

```
pip install virtualenv
virtualenv env
```

Αναλυτικές οδηγίες για windows υπάρχουν εδώ https://programwithus.com/learn/python/pip-virtualenv-windows.

3.1.3.2 Ενεργοποίηση εικονικού περιβάλλοντος

Αφού έχουμε δημιουργήσει το εικονικό περιβάλλον, θα πρέπει να το ενεργοποιήσουμε. Η ενεργοποίηση του εικονικού περιβάλλοντος πρέπει να γίνεται κάθε φορά που θα ανοίγουμε το τερματικό - όταν ανοίξει νέο τερματικό πρέπει να φορτωθεί και πάλι

Η ενεργοποίηση του εικονικού περιβάλλοντος γίνεται ως εξής:

```
source venv/bin/activate  # Linux
```

σε windows:

```
env\Scripts\activate.bat
```

Προϋπόθεση είναι να έχομε εισέλθει (με `cd`) στον κατάλογο (φάκελο) του project.

3.1.4 Εκτέλεση πρώτου προγράμματος python

Μέσα από τον κατάλογο του project, και αφού ενεργοποιήσουμε το εικονικό περιβάλλον, ανοίγουμε το `vscode` εκτελώντας την εντολή:

```
code .
```

Αφού ανοίξει το παράθυρο του vscode, δημιουργούμε το πρόγραμμα python με την εντολή:

```
code main.py
```

Το αρχείο main.py θα ανοίξει μέσα στο vscode.

Πληκτρολογούμε μέσα στο αρχείο το πρώτο μας πρόγραμμα python:

```
print("Hello, World!")
```

και αποθηκεύουμε το αρχείο.

Από το τερματικό, τρέχουμε το πρόγραμμα με την εντολή:

```
$ python main.py
Hello, World!
```

3.2 Δημιουργία διακομιστή με FastAPI

Οι παρακάτω οδηγίες βασίζονται στο επίσημο tutorial του `fastapi` από τη διεύθυνση https://fastapi.tiangolo.com/tutorial/

3.2.1 Εγκατάσταση απαραίτητων βιβλιοθηκών

1. Μέσα από τον κατάλογο του project, δημιουργούμε ένα αρχείο με όνομα `requirements.txt`, στο οποίο τοποθετούμε την ακόλουθη γραμμή

   ```
   fastapi[standard]==0.115.6
   ```

2. Αφού βεβαιωθούμε ότι το εικονικό περιβάλλον είναι ενεργοποιημένο, μέσα από τον κατάλογο του project, και αφού αποθηκεύσουμε το αρχείο `requirements.txt` από το βήμα 1, εκτελούμε την εντολή:

   ```
   pip install -r requirements.txt
   ```

   Η εντολή θα παράγει αρκετές γραμμές εξόδου, οι οποίες θα καταλήγουν σε κάτι ανάλογο με το παρακάτω:

```
Successfully installed MarkupSafe-3.0.2 annotated-types-0.7.0 anyio-4.8.0
 ↪ certifi-2024.12.14 click-8.1.8 dnspython-2.7.0 email-validator-2.2.0
 ↪ fastapi-0.115.6 fastapi-cli-0.0.7 h11-0.14.0 httpcore-1.0.7
 ↪ httptools-0.6.4 httpx-0.28.1 idna-3.10 jinja2-3.1.5 markdown-it-py-3.0.0
 ↪ mdurl-0.1.2 pydantic-2.10.5 pydantic-core-2.27.2 pygments-2.19.1
 ↪ python-dotenv-1.0.1 python-multipart-0.0.20 pyyaml-6.0.2 rich-13.9.4
 ↪ rich-toolkit-0.13.2 shellingham-1.5.4 sniffio-1.3.1 starlette-0.41.3
 ↪ typer-0.15.1 typing-extensions-4.12.2 uvicorn-0.34.0 uvloop-0.21.0
 ↪ watchfiles-1.0.4 websockets-14.2
```

3.2.2 Δημιουργία και εκτέλεση πρώτης δοκιμαστικής έκδοσης του server

Θα δημιουργήσουμε την πρώτη έκδοση του server τροποποιώντας το αρχείο `main.py` από τα προηγούμενα βήματα, ώστε να έχει το ακόλουθο περιεχόμενο

```python
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Για να εκτελέσουμε το server, πρέπει να βεβαιωθούμε ότι στο τερματικό βρισκόμαστε στο κατάλογο του project, ότι το εικονικό περιβάλλον είναι ενεργοποιημένο, και ότι έχουμε εγκαταστήσει τη βιβλιοθήκη fastapi από το προηγούμενο βήμα.

Εφόσον ισχύουν όλα τα παραπάνω, εκτελούμε την εντολή:

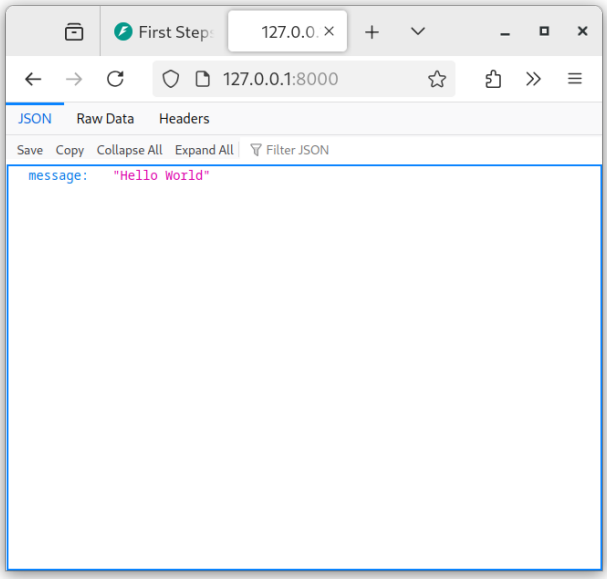
```
fastapi dev main.py
```

Θα πρέπει η εντολή να επιστρέψει έξοδο όπως η παρακάτω:

```
   FastAPI   Starting development server 

             Searching for package file structure from
             directories with __init__.py files
             Importing from
             /home/djvergad/lessons/fastapi-primes/solution/init
             ial

    module    main.py

      code   Importing the FastAPI app object from the module
             with the following code:

             from main import app

       app   Using import string: main:app

    server   Server started at http://127.0.0.1:8000
    server   Documentation at http://127.0.0.1:8000/docs

       tip   Running in development mode, for production use:
             fastapi run

             Logs:

      INFO   Will watch for changes in these directories:
             ['/home/djvergad/lessons/fastapi-primes/solution/in
             itial']
      INFO   Uvicorn running on http://127.0.0.1:8000 (Press
```

```
        CTRL+C to quit)
INFO    Started reloader process [138309] using WatchFiles
INFO    Started server process [138329]
INFO    Waiting for application startup.
INFO    Application startup complete.
```

Σε αυτό το σημείο μπορούμε να ανοίξουμε τους συνδέσμους που φαίνονται παραπάνω στο browser, και θα δούμε έξοδο όπως στις Εικόνες 1 και 2:



Σχήμα 1: Σύνδεσμος http://127.0.0.1:8000

Αν στο δεύτερο σύνδεσμο επιλέξουμε το κουμπί «`Try it out`» μπορούμε να στείλουμε ένα request, και ο server θα μας απατήσει με το αντικείμενο (Εικόνα 3)

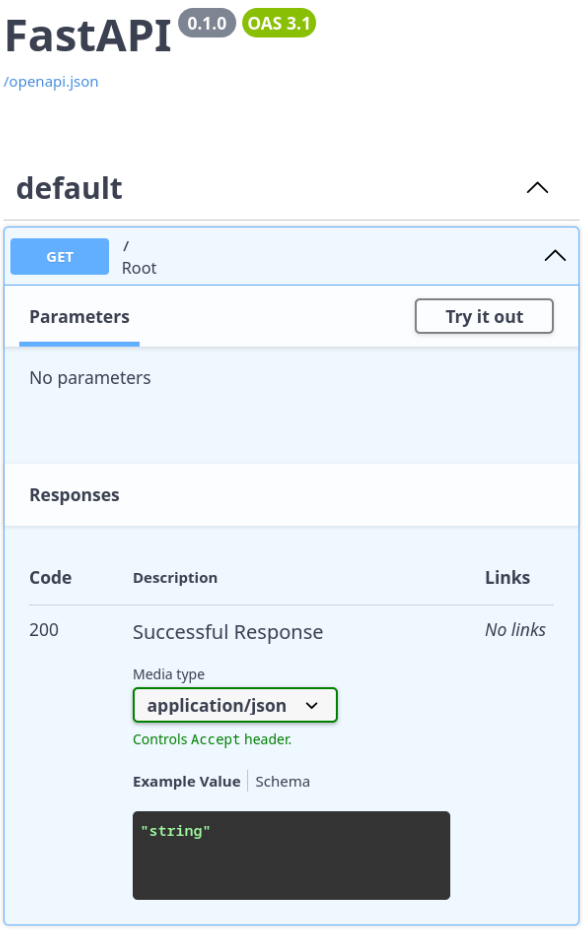
```
{
  "message": "Hello World"
}
```

3.2.3 Δημιουργία API endpoint για τον υπολογισμό πρώτων παραγόντων

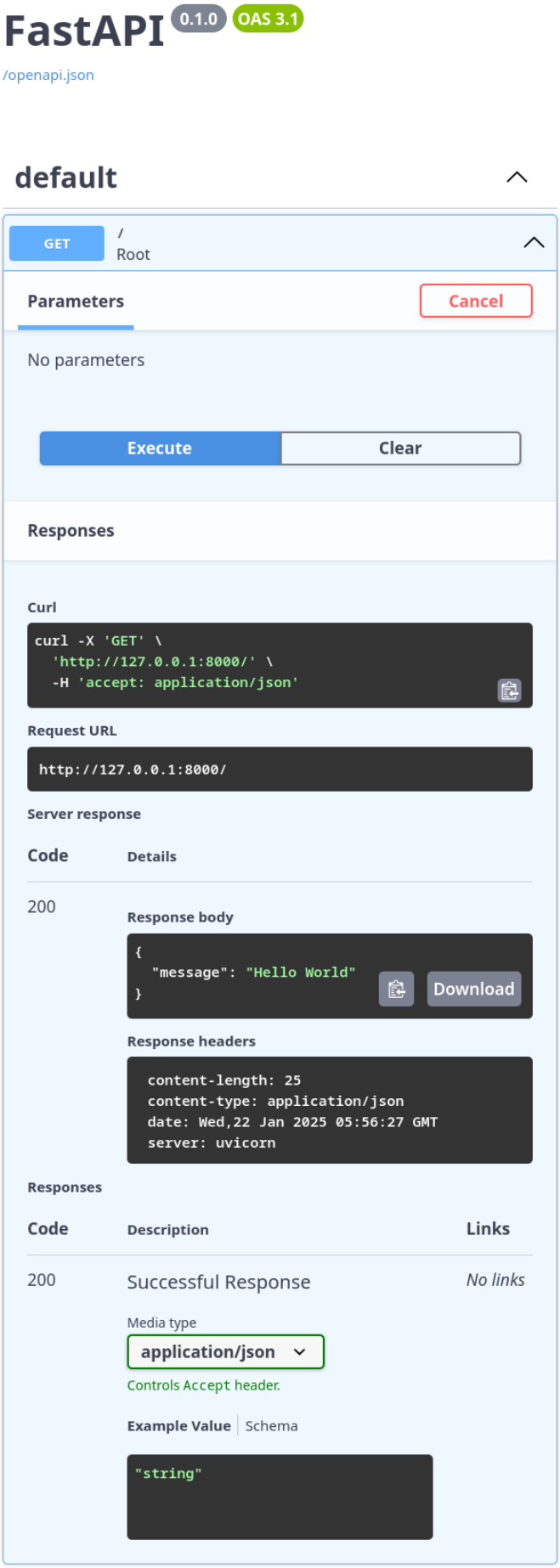
Πρώτα θα ορίσουμε τη συνάρτηση που θα κληθεί από το νέο endpoint `/primes`, καθώς και τη μορφή του αντικειμένου που δέχεται η συνάρτηση:

Τροποποιούμε το αρχείο main.py ως εξής:

```
from fastapi import FastAPI
from pydantic import BaseModel, conint
from typing import List

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}

class RequestObjet(BaseModel):
    input_numbers: List[conint(gt=1, lt=100)]

@app.post("/primes/")
async def primes(request_object: RequestObjet):
    return {"message": "Hello World"}
```



Σχήμα 2: Σύνδεσμος http://127.0.0.1:8000/docs



Σχήμα 3: Επιστρεφόμενη τιμή μετά από την αποστολή request στο server

Τώρα αν δούμε το σύνδεσμο http://localhost:8000/docs#/default/primes_primes__post θα πρέπει να εμφανιστεί σελίδα όπως στην Εικόνα 4.

Μπορούμε να στείλουμε αντικείμενα και πάλι μέσα από το UI, αλλά η Επιστρεφόμενη τιμή θα είναι πάντα η ίδια, καθώς δεν έχουμε ορίσει ακόμα κάποια απάντηση.

3.2.4 Ορισμός τύπου επιστρεφόμενης τιμής

Προσθέτουμε τον παρακάτω κώδικα για να ορίσουμε τη μορφοποίηση του επιστρεφόμενου αντικειμένου.

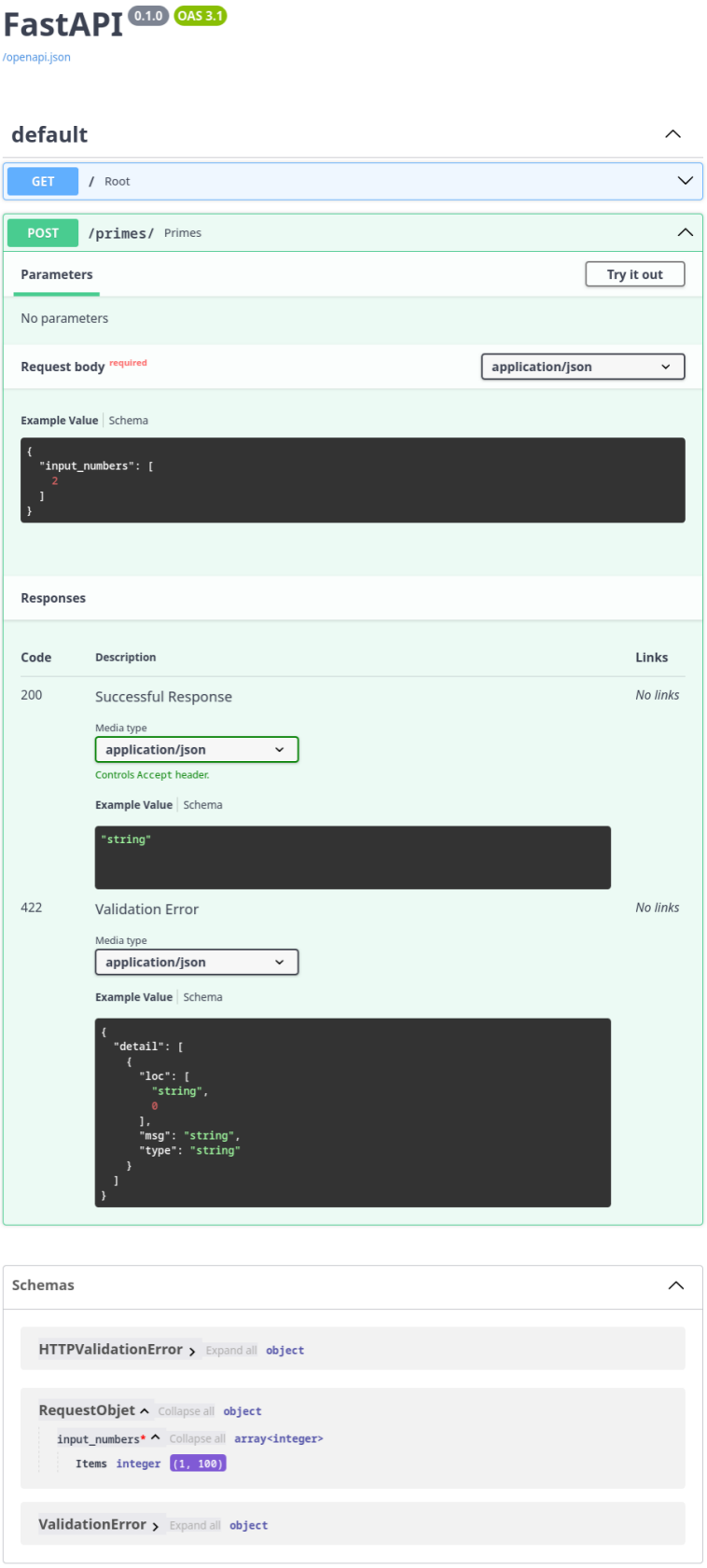
```
class ResponseItem(BaseModel):
    input_number: int
    prime_factors: List[int]


class ResponseObject(BaseModel):
    result: List[ResponseItem]

@app.post("/primes/")
async def primes(request_object: RequestObjet) -> ResponseObject:
    return {"message": "Hello World"}
```

Δεν ξεχνάμε να προσθέσουμε τον κώδικα `-> ResponseObject` στον ορισμό της συνάρτησης `primes` ώστε να ορίσουμε ότι η συνάρτηση επιστρέφει αυτό τον τύπο.

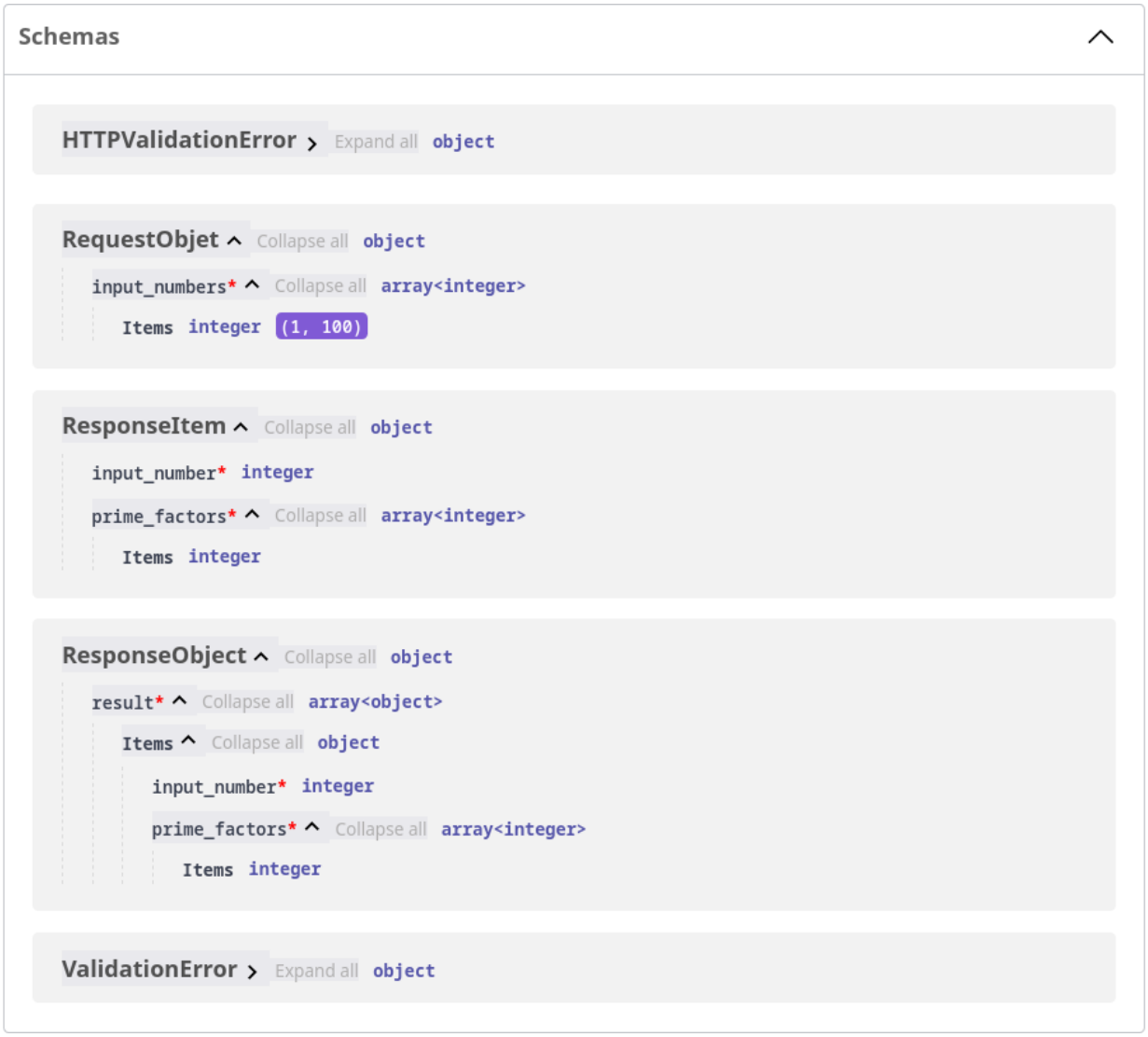
Τώρα αν ανανεώσουμε τη σελίδα http://localhost:8000/docs#/default/primes_primes__post θα πρέπει να δούμε το UI έχει ενημερωθεί με το αντικείμενο που επιστρέφεται όπως στην εικόμα 5.

Αν όμως στείλουμε request, θα μας εμφανιστεί σφάλμα στο server, καθώς δεν έχουμε ορίσει ακόμα το αντικείμενο που επιστρέφει το API, οπότε η επιστρεφόμενης τιμή δεν είναι σύμφωνα με τον ορισμό του API (Εικόνα 6)

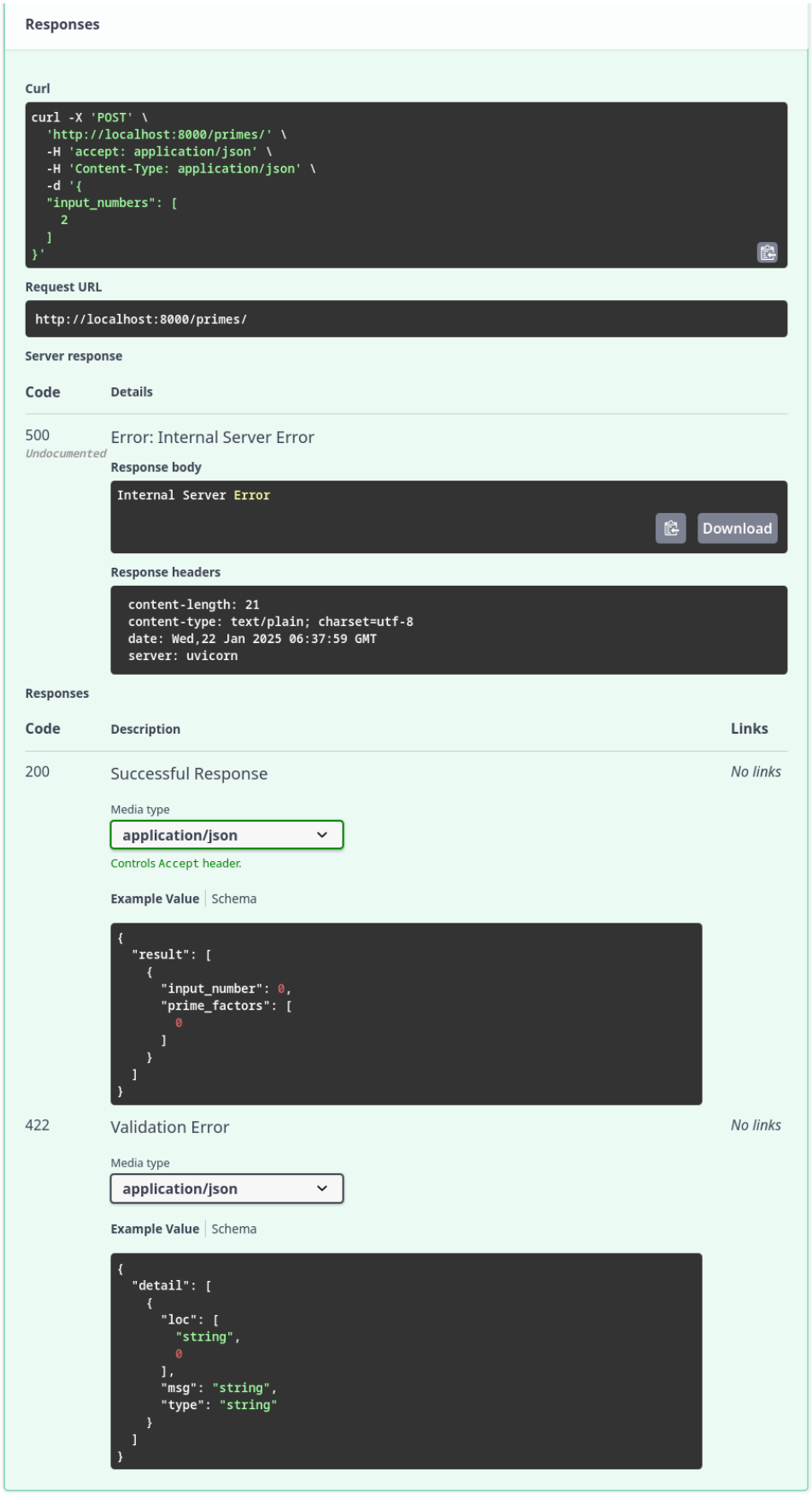
```
      INFO   127.0.0.1:36728 - "POST /primes/ HTTP/1.1" 500
     ERROR   Exception in ASGI application
Traceback (most recent call last):
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 409, in run_asgi
    result = await app(  # type: ignore[func-returns-value]
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 60, in __call__
    return await self.app(scope, receive, send)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 1054, in __call__
    await super().__call__(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 113, in __call__
    await self.middleware_stack(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 187, in __call__
    raise exc
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 165, in __call__
    await self.app(scope, receive, _send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 62, in __call__
    await wrap_app_handling_exceptions(self.app, conn)(scope, receive, send)
```



Σχήμα 4: Εμφάνιση του swagger UI με τις παραμέτρους του API



Σχήμα 5: Ορισμός επιστρεφόμενης τιμής



Σχήμα 6: Σφάλμα μη ορθής επιστρεφόμενης τιμής

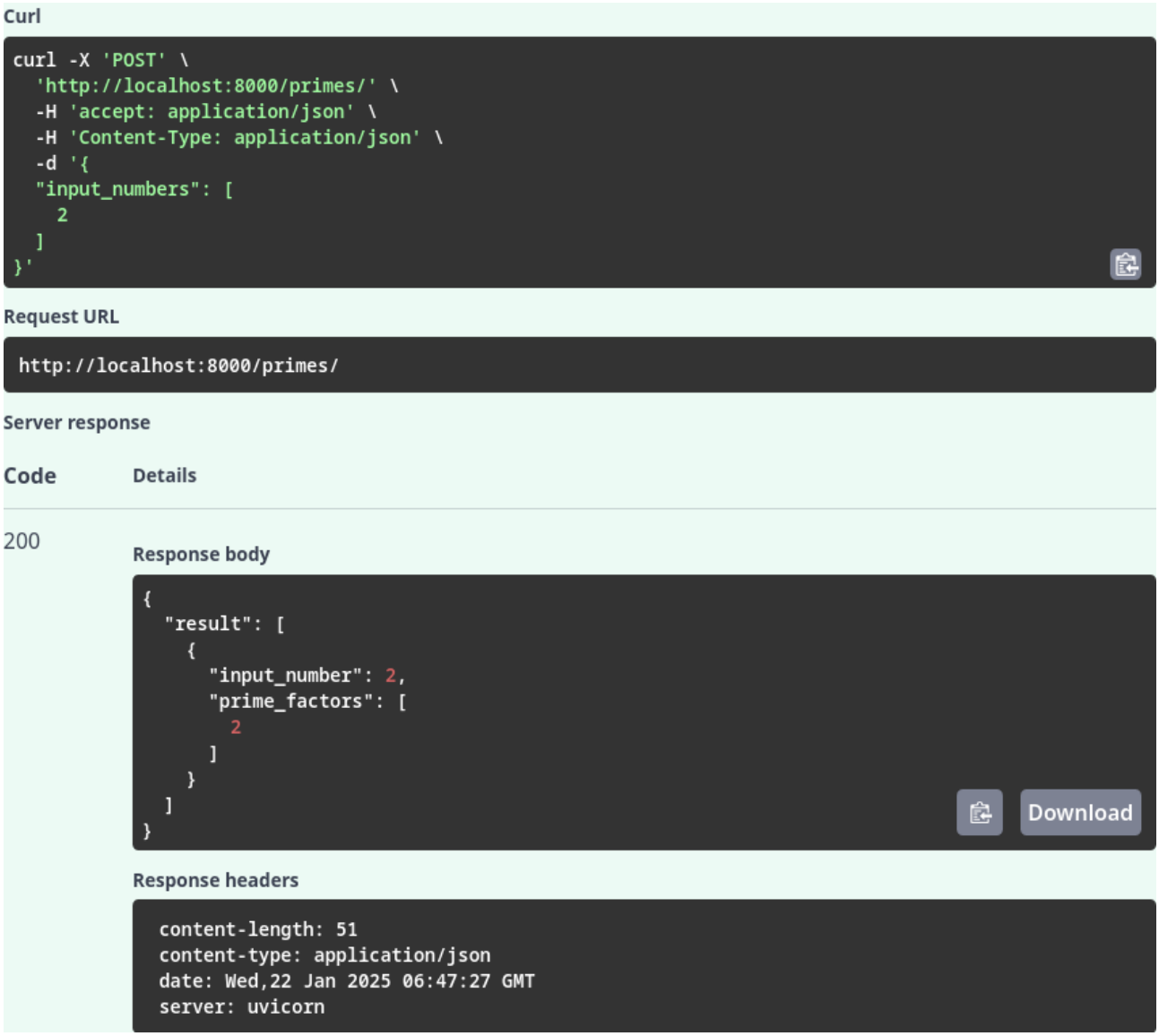
```
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 53, in wrapped_app
    raise exc
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 42, in wrapped_app
    await app(scope, receive, sender)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 715, in __call__
    await self.middleware_stack(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 735, in app
    await route.handle(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 288, in handle
    await self.app(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 76, in app
    await wrap_app_handling_exceptions(app, request)(scope, receive, send)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 53, in wrapped_app
    raise exc
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 42, in wrapped_app
    await app(scope, receive, sender)
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 73, in app
    response = await f(request)
               ^^^^^^^^^^^^^^^^
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 327, in app
    content = await serialize_response(
              ^^^^^^^^^^^^^^^^^^^^^^^^^
  File
↪  "/home/djvergad/lessons/fastapi-primes/solution/initial/venv/lib/python3.11/site-packages
↪  line 176, in serialize_response
    raise ResponseValidationError(
fastapi.exceptions.ResponseValidationError: 1 validation errors:
  {'type': 'missing', 'loc': ('response', 'result'), 'msg': 'Field required',
↪  'input': {'message': 'Hello World'}}
```

Διορθώνουμε το σφάλμα ορίζοντας μια hard-coded επιστρεφόμενη τιμή σύμφωνα με το πρότυπο που ορίσαμε.

```python
@app.post("/primes/")
async def primes(request_object: RequestObjet) -> ResponseObject:
    return ResponseObject(
        result=[
            ResponseItem(
                input_number=2,
                prime_factors=[
                    2
                ]
            )
```

```
        ]
    )
```

Τώρα η το UI επιστρέφει την τιμή που ορίσαμε χωρίς σφάλμα (Εικόνα 7).



Σχήμα 7: Hard-coded απάντηση από το server

3.2.5 Υλοποίηση συνάρτησης υπολογισμού πρώτων παραγόντων

Μια συνάρτηση υπολογισμού πρώτων παραγόντων είναι η παρακάτω:

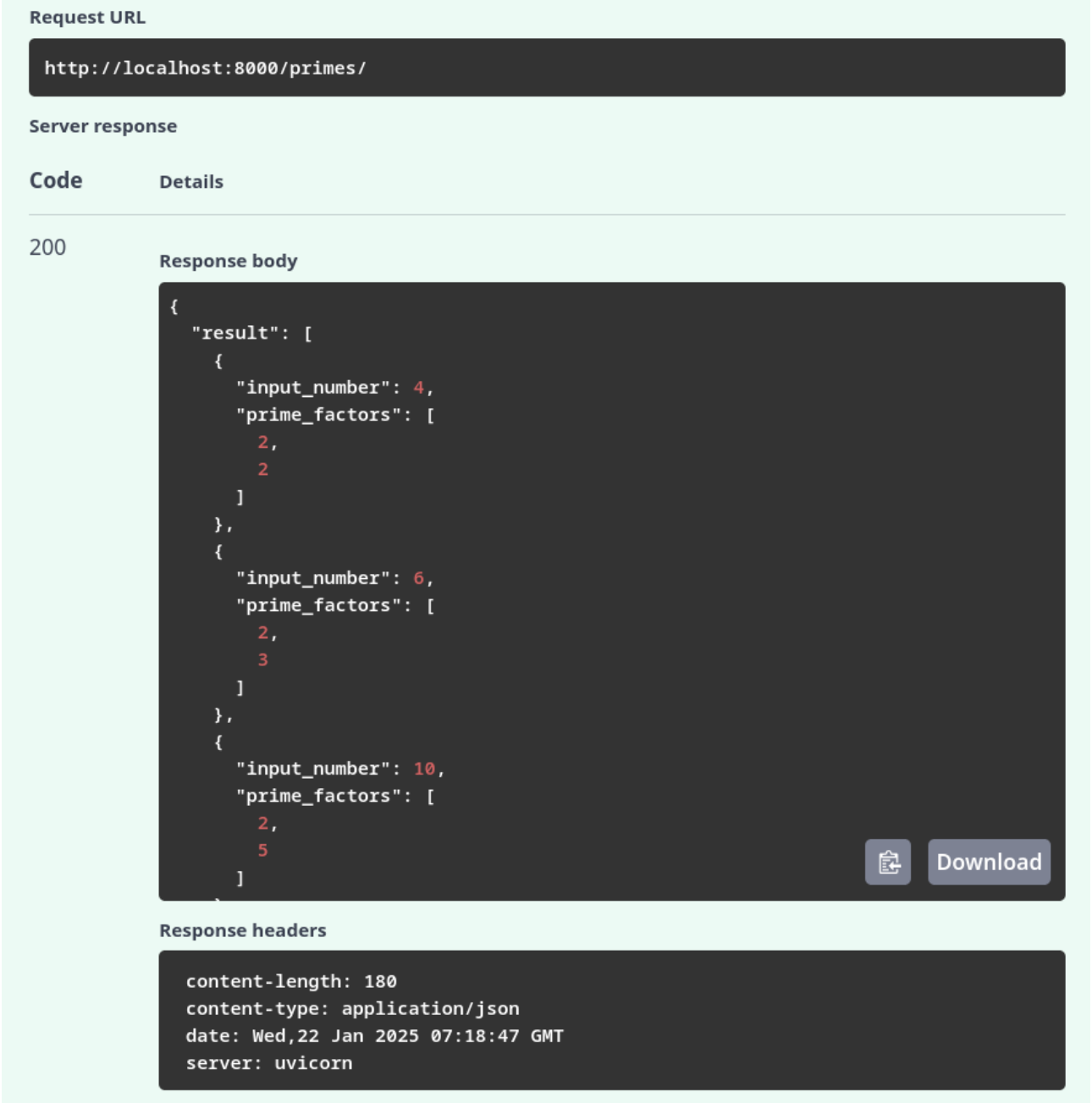
```
def prime_factors(number: int):
    result = []
    i = 2
    while i <= number:
        if number % i == 0:
            result += [i]
            number //= i
        else:
            i+=1
    return result
```

Καλούμε τη συνάρτηση από τη συνάρτηση `primes`:

```
@app.post("/primes/")
```

```python
async def primes(request_object: RequestObjet) -> ResponseObject:
    return ResponseObject(
        result=[
            ResponseItem(
                input_number=number,
                prime_factors=prime_factors(number)
            )
            for number in request_object.input_numbers
        ]
    )
```

Τώρα αν ξαναστείλου το request πρέπει να πάρουμε τη σωστή απάντηση (Εικόνα 8)



Σχήμα 8: Παράδεγμα ορθής απάντησης από το server

3.3 Δημιουργία του client

Δημιουργούμε ένα νέο python αρχείο με όνομα `client.py`, με τα ακόλουθα περιεχόμενα:

```python
import requests
import json
import sys

api_url = "http://localhost:8000/primes"

obj = {'input_numbers': [int(n) for n in sys.argv[1:]]}

response = requests.post(api_url, json=obj)
result = response.json()

print(json.dumps(result, indent=2))
```

Ενώ ο server τρέχει, εκετελούμε το αρχείο με την ακόλουθη εντολή:

```
python client.py 4 50 24 95
```

Θα πρέπει να δούμε αποτέλεσμα όπως το παρακάτω:

```
$ python client.py 4 50 24 95
{
  "result": [
    {
      "input_number": 4,
      "prime_factors": [
        2,
        2
      ]
    },
    {
      "input_number": 50,
      "prime_factors": [
        2,
        5,
        5
      ]
    },
    {
      "input_number": 24,
      "prime_factors": [
        2,
        2,
        2,
        3
      ]
    },
    {
      "input_number": 95,
      "prime_factors": [
        5,
        19
      ]
    }
  ]
}
```

4 Υποβολή εργασίας

Το τελικό παραδοτέο της εργασίας είναι ένα `zip` αρχείο, που περιλαμβάνει τουλάχιστον:

1. Τον κώδικα του σεναρίου που εκτελέστηκε (χωρίς τον υποκατάλογο `venv`)

2. Ένα σύντομο report που να περιγράφει τη διαδικασία που εκτελέσατε για την υλοποίηση της άσκησης, με παραδείγματα κώδικα και screenshots

4.1 Ζητούμενα

1. Ενσωμάτωση documentation strings στο UI
2. Επαλήθευση της ορθής δομής των request και response από το client
3. Υλοποίηση μηχανισμού authentication για το API