CC - Laborator I/II

Scop:

- Crearea unui site web, accesibil prin Internet
  - Accesarea site-ului prin port: 200.100.23.260:8130
  - Bonus: Accesarea site-ului folosind un hostname / **o cale** www.mywebsite.com / (200.100.23.260/**path**)
- Folosirea formularelor (Web Forms)
  - Simple (request.form)
  - Bonus: Folosind biblioteci (FlaskForm)
- Scrieti un site web cu doua pagini si un o bara de navigare:
  - Pagina 1: Lista de carti (pentru fiecare carte se va afisa o lista verticala care sa contina pe primul rand **autorul**, pe al doilea rand **editura** si pe al treilea rand **numarul de pagini**);
  - Pagina 2: Adaugare carte (un formular care sa contina 4 campuri: nume carte, autor, editura, numar pagini si la submisie se va adauga cartea intr-o lista);

Functionalitati extra:

- Verificati tipul campurilor (ex: numar pagini != o suta trei);
- Verificati daca exista deja o carte cu datele introduse si in acest caz afisati o eroare;
- Salvati cartile intr-un fisier text. Cand adaugati o carte, salvati-o in fisier. Extra: folositi un fisier binar pentru performanta;
- Implementati un REST API (folosind JSON) pentru adaugare / vizualizare / actualizare / stergere carti (CRUD); de exemplu:
  - POST /api/books ← Create
  - GET /api/books ← Read all
  - GET /api/book/<id> ← Read one
  - PUT /api/book/<id> ← Update
  - DELETE /api/books ← Delete all
  - DELETE /api/book/<id> ← Delete one
- Scrieti pagini care sa contina formulare pentru actiunile de mai sus si sa foloseasca REST API-ul pentru a le implementa;
- In loc sa folositi fisiere, folositi o baza de data (de ex. SQLite), in care sa aveti: Book (pk, author_fk, name, pages, publishing_house), Author(pk, name); Creati un API v2 care sa foloseasca baza de date

Pasul 1:

- Instalare software pe o masina virtuala:
  - $ ssh -i /path/to/private-key website-vm
  - [$ sudo apt-get install nginx python python3-venv]
- Crearea unui virtual environment pentru Python si instalare Flask
  - $ mkdir website && cd website
  - $ python3 -m venv .venv
  - $ source .venv/bin/activate
  - $ pip install flask
- Rulare si testare web server:
  - Urmariti tutorialul
  - Creati un fisier **website.py** in directorul **website** (modificati partea cu rosu astfel incat sa aveti un numar intre 8000 si 8099; daca la rulare apare o problema, schimbati portul):

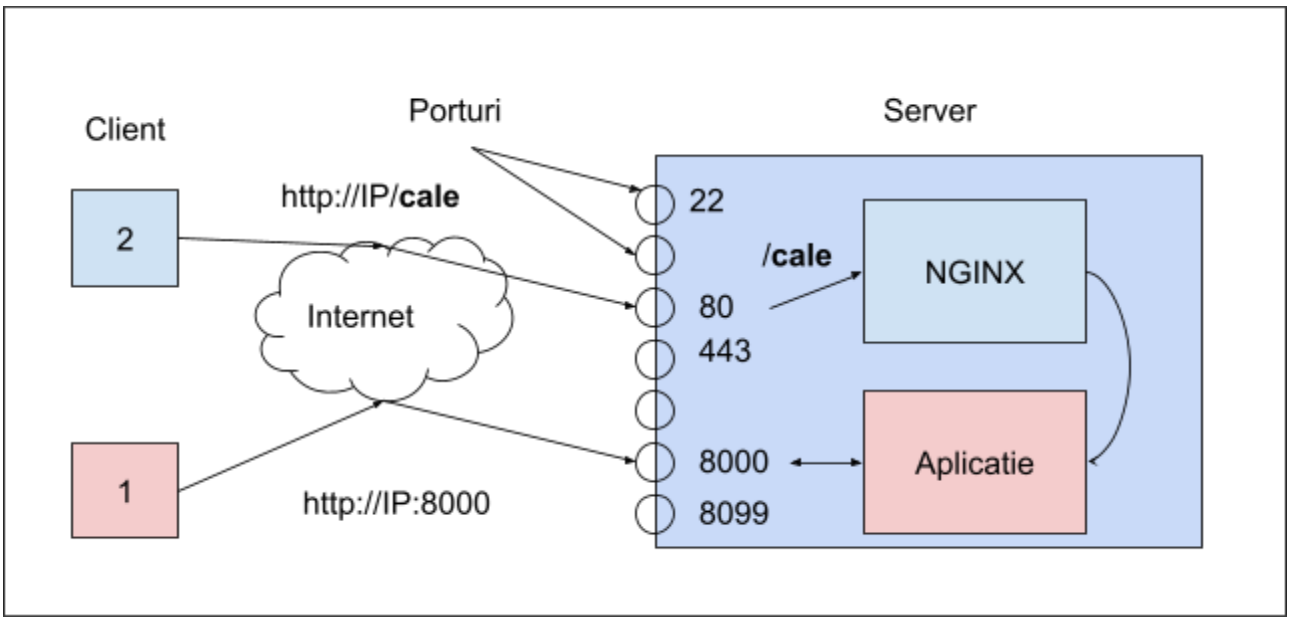
    ```
    from flask import Flask
    app = Flask(__name__)

    @app.route("/")
    def hello_world():
        return "<p>Hello, world!</p>"

    if __name__ == '__main__':
        app.run(host='0.0.0.0', port=80XX)
    ```

  - (.venv) $ python website.py
  - Testare:
    - Alt terminal -> SSH -> curl localhost:80**XX**
    - In browser: http://<IP-WEBSITE>:80**XX**
- EXTRA:

  Avand in vedere faptul ca nu este singurul website care ruleaza pe server nu putem pune portul 80 (pentru porturile < 1024 aveti nevoie de permisiuni de administrator - root si daca rulati pe portul 80 ceilalti studenti nu vor putea rula site-ul lor pe portul 80 - o singura aplicatie poate asculta pe un port la un moment dat)

  Din acest motiv vom folosi un Reverse Proxy pentru a putea partaja portul 80 intre mai multe site-uri web. De obicei, aceasta partajare se

face folosind campul **Host** trimis in request-ul HTTP, dar noi vom folosi calea pentru a gasi site-ul corect.



- Dupa ce va asigurati ca aplicatia voastra functioneaza pe portul ales (cazul 1), trebuie sa configurati programul NGINX sa stie sa redirectioneze **/cale** catre aplicatia corecta. Pentru acest lucru, trebuie sa:
  - Creati in directorul /etc/nginx/sites-available/ un fisier text cu informatii despre site-ul vostru
    (ex: $ cat /etc/nginx/sites-available/template)
  - Modificati fisierul de configurare al nginx-ului:
    /etc/nginx/sites-available/default in care trebuie sa adaugati ruta pentru **/cale:**

```
location /cale {
        rewrite /cale/?(.*) /$1 break;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://127.0.0.1:8000;
}
```

- Acum puteti sa va conectati in browser la http://<IP-WEBSITE>**/cale** si programul NGINX va redirectiona conexiunea catre aplicatia voastra.

Pasul 2:

- Aplicatii web folosind Flask; Idei de baza:
  - ❖ La pasul 1 am vazut ca putem returna direct cod de HTML si acesta este transformat de browser(*return "<h1>Hello</h1>"*).
  - ❖ De obicei este greu sa facem acest lucru direct din limbajul de programare intrucat putem avea structuri complexe cu loop-uri, conditii etc.
  - ❖ In Flask se definesc **templates** care contin codul HTML si la rularea codului partile care se modifica sunt injectate in codul HTML.
  - ❖ Exemplu:

```
$ tree website/
website/
├── main.py
└── templates
    └── test_page.html

$ cat website/templates/test_page.html
<!DOCTYPE html>
<html>
  <body>
    <h1>Simple page</h1>
    <p>Passed data: {{ data }}</p>
  </body>
</html>

$ cat website/main.py
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def hello_world():
    data = 123
    return render_template("test_page.html", data=data)

if __name__ == '__main__':
    app.run(port=8000)
```

- Pagina web:

  Simple page

  Passed data: 123

- Urmariti tutorialele de pe Digital Ocean legate de Flask
  - Tutorial prima aplicatie Flask
  - Tutorial despre template-uri
  - **Tutorial despre formulare simple**
  - Tutorial despre formulare folosind biblioteca FlaskWTF

Pasul 3: Modificati aplicatia cu formulare de la pasul precedent (formulare simple / formulare folosind FlaskWTF) pentru aplicatia ceruta anterior.