

# Flask

## .Partea II. Șabloane în Flask

Ce ne aşteaptă?

1. Bazele şabloanelor HTML
2. Şabloanele variabilelor
3. Şabloanele pentru controlul fluxului
4. Şabloanele de moştenire
5. Funcţia `url_for`
6. Şabloanele formularelor

# 1. Bazele șabloanelor HTML

## Esența esența șabloanelor HTML

- În prima parte a cursului, funcțiile view au returnat pagini HTML prin intermediul unor stringuri Python.
- În realitate se dorește ca funcțiile view să returneze șabloane HTML interpretate ( fișiere HTML numite și templates ce reprezintă o pagină web)
- Flask automat va căuta șabloanele HTML în directoriul templates.
- Pentru interpretarea șabloanelor se importă funcția `render_template` din flask și se returnează un fișier `.html` din funcția view

## Scriptul paginii HTML

- Se creează un folder templates în folderul de program
- În acest folder se creează un document HTML numit pag1.html ce va afișa 2 mesaje:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Bun venit pe aceasta pagina</h1>
    <p>Este o pagina HTML experimentală în cursul Flask</p>
  </body>
</html>
```

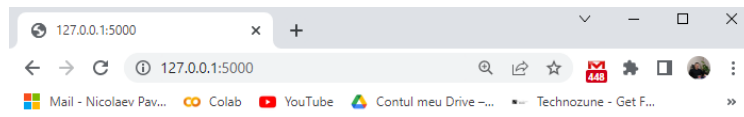
## Scriptul Python

- Se modifică conținutul fișierului app1.py

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

```
@app.route('/')  
def index():  
    return render_template('pag1.html')
```

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



## Bun venit pe aceasta pagina

Este o pagina HTML experimentală în cursul Flask

## 2. Șabloanele variabilelor

### Eșența șabloanelor variabilelor

- Utilizând funcția `render_template` se pot interpreta fișierele HTML în aplicația Flask dar nu se utilizează oportunitățile limbajului Python
- De multe ori se dorește a se utiliza cod Python pentru a modifica și actualiza variabile și logica programului și apoi aceste informații se transmit șablonului.
- Pentru aceasta se utilizează mecanismul de șablon Jinja ce permite înserarea variabilelor din cod Python în fișiere HTML.
- Sintaxa de înserare a variabilelor:  
`{{nume_variabila}}`
- În aceste șabloane se pot însera stringuri, liste, dicționare și altele
- Se setează parametrii în funcția `render_template` a fișierului Python și apoi se utilizează acești parametri înserându-se în șablonul `{{ }}`

## Scriptul paginii HTML cu șablon Jinja

- Se modifică fișierul pag1.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Bun venit pe aceasta pagina {{nume}}</h1>
    <p>Este o pagina HTML experimentală în cursul Flask</p>
  </body>
</html>
```

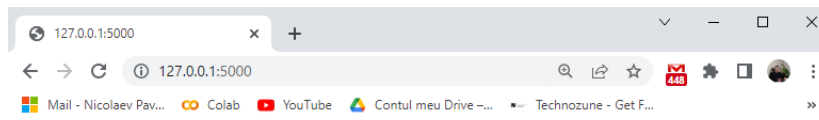
## Scriptul Python

### ■ Se modifică conținutul fișierului app1.py

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

@app.route('/')
def index():
    nume='Pavel'
    return render_template('pag1.html', nume=nume)

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



**Bun venit pe aceasta pagina Pavel**

Este o pagina HTML experimentală în cursul Flask



### 3. Șabloane pentru controlul fluxului

#### Esența șabloanelor pentru controlul fluxului

- Cu șabloanele Jinja în codul HTML pot fi înserate variabile Python folosind sintaxa `{{ nume_variabila }}`.
- În codul HTML se pot însera și declarații de control a fluxului și anume bucle for și declarații if utilizând șabloane de control a fluxului
- Sintaxa acestor șabloane utilizează simbolurile `{% %}`
- Se consider exemplul în care se transmit elementele unei liste Python (`lista_mea`) într-o listă HTML utilizând secvența striptului

```
<ul>
```

```
    {% for element in lista_mea %}
```

```
    <li> {{ element }} </li>
```

```
    {% endfor %}
```

```
</ul>
```

# Scriptul paginii HTML cu șabloane de control a fluxului

## ■ Se modifică fișierul pag1.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Bun venit pe aceasta pagina</h1>
    <p>Este o pagina HTML experimentală în cursul Flask ce include</p>
    <ul>
      {% for element in lista_mea %}
      <li>{{ element }} </li>
      {% endfor %}
    </ul>
    {% if 'Baze SQL în Flask' in lista_mea %}
      <p>Cursul include și informații de conectare la baze de date</p>
    {% else %}
      <p>Cursul nu conține informații de conectare la baza de date</p>
    {% endif %}
  </body>
</html>
```

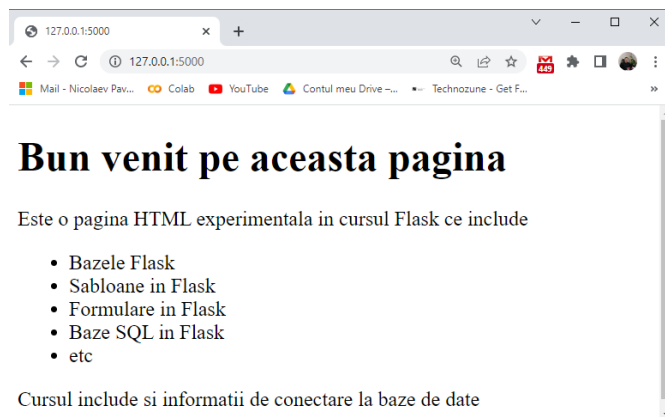
## Scriptul Python

### ■ Se modifică conținutul fișierului app1.py

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

@app.route('/')
def index():
    lista_mea = ['Bazele Flask', 'Sabloane in Flask', 'Formulare in Flask',
                'Baze SQL in Flask', 'etc' ]
    return render_template('pag1.html', lista_mea=lista_mea)

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



## 4. Șabloane de moștenire

### Esența șabloanelor de moștenire

- În exemple de mai sus pentru accesarea fiecărei pagini web a fost creat un fișier HTML și o funcție view corespunzătoare
- Însă în cadrul unei aplicații web, paginile conțin o multitudine de informații comune de exemplu bara de navigare, footer, etc..
- Pentru a nu introduce repet una și aceeași informație în mai multe pagini se utilizează șabloane de moștenire
- Se creează un șablon .html (numit sablon.html) ce va conține informația utilă și altor pagini
- Apoi se utilizează șabloanele `{% extend "base.html" %}` și `{% block %}` pentru a include această informație în alte pagini.

## Șabloane de moștenire

sablon.html



{%block content%

{% endblock %}



pag1.html

{%extends "sablon.html"%}

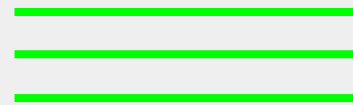
{%block content%



{% endblock %}



Rezultat



## Pagini HTML cu șabloane de moștenire

- Se creează pagina html cu șablonul de moștenire `sablon.html`:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exemple Flask</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous">
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
        JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
  </head>
  <body>
    <nav class="navbar navbar-dark bg-primary">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Cursul Flask </a>
      </div>
    </nav>
    {% block content %}
    {% endblock %}
  </body>
</html>
```

## Pagini HTML accesate în aplicație

### ■ Se modifică conținutul fișierului app1.html

```
{% extends "sablon.html" %}

{% block content %}
    <h1>Bun venit pe aceasta pagina</h1>
    <p>Este o pagina HTML experimentală în cursul Flask</p>
{% endblock %}
```

### ■ Se creează fișierul app2.html cu conținutul

```
{% extends "sablon.html" %}

{% block content %}
    <h1>Bun venit pe aceasta pagina {{nume.capitalize()}}</h1>
    <p>Este o pagina HTML experimentală în cursul Flask</p>
{% endblock %}
```

## Scriptul Python

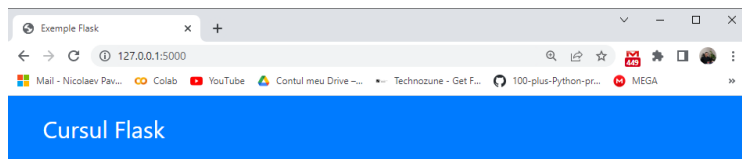
### ■ Se modifică conţinutul fişierului app1.py

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

@app.route('/')
def pag1():
    return render_template('pag1.html')

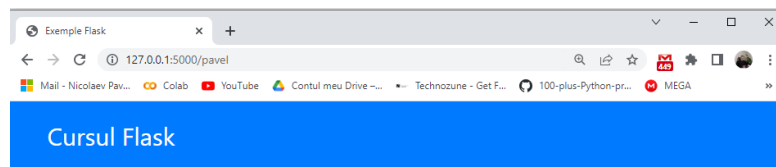
@app.route('/<nume>')
def pag2(nume):
    return render_template('pag2.html', nume=nume)

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



Bun venit pe aceasta pagina

Este o pagina HTML experimentală în cursul Flask



Bun venit pe aceasta pagina Pavel

Este o pagina HTML experimentală în cursul Flask



## 5. Funcția url\_for

### Rolul funcției url\_for

- Funcția url\_for permite conexiune simplificată între paginile web ale aplicației
- Funcția url\_for se include cu sablonul variabilei ca valoarea a atributului href a tag-ului a
- Drept parametru al funcție se utilizează numele funcție view de acces al paginii

```
href="{{url_for('pag1')}}"
```

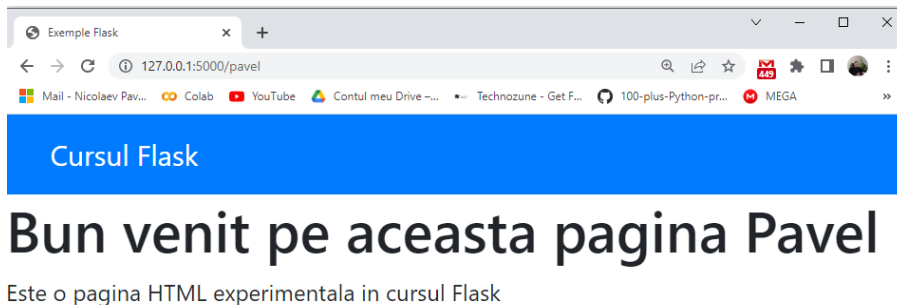
## Exemplu script cu includerea url\_for

### ■ Se modifică conținutul fișierului sablon.html

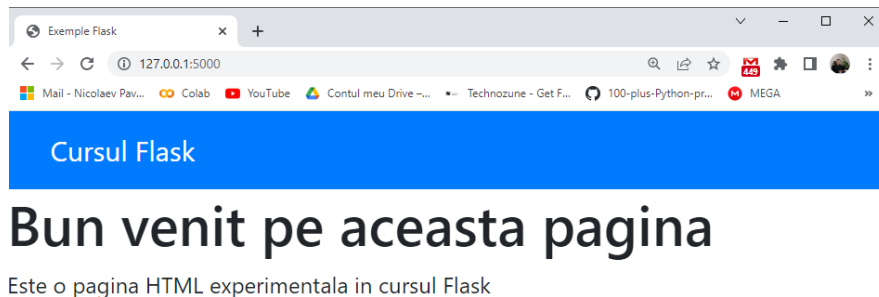
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exemple Flask</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous">
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
        JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
  </head>
  <body>
    <nav class="navbar navbar-dark bg-primary">
      <div class="container-fluid">
        <a: class="navbar-brand" href="{url_for('pag1')}}">Cursul Flask </a>
      </div>
    </nav>
    {% block content %}
    {% endblock %}
  </body>
</html>
```

## Accesarea între pagini html

- În browser se accesează `http://127.0.0.1:5000/pavel`



- Pe această pagină se accesează link-ul Cursul Flask



## 6. Șabloanele formularelor

### Pagini HTML cu șabloane de moștenire

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exemple Flask</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmY1" crossorigin="anonymous"></script>
  </head>
  <body>
    <nav class="navbar navbar-dark bg-primary">
      <div class="container-fluid">
        <a class="navbar-brand" href="{{url_for('acasa')}}"> Cursul Flask </a>
      </div>
    </nav>
    {% block content %}

    {% endblock %}
  </body>
</html>
```

## Pagina HTML de pornire

- Se creează un șablon .html numit acasa.html pentru pagina de pornire

```
{% extends "sablon.html" %}

{% block content %}
    <div class="jumbotron">
        <h1>Bun venit pe pagina cursului Flask</h1>
        <p>Autentifică-te pentru a accesa cursul Flask </p>
        <a href="{{url_for('autentificare')}}"> Autentificare curs Flask </a>
    </div>

{% endblock %}
```

## Pagina HTML de autentificare

- Se creează un șablon .html numit autentificare.html care va include formularul de autentificare (datele formularului vor fi doar citite nu și salvate undeva)

```
{% extends "sablon.html" %}

{% block content %}
    <div class="jumbotron">
        <h1>Bun venit pe pagina de autentificare pentru cursul Flask</h1>
        <p>Completează formularul. Datele nu se vor salva </p>
        <form action="{{url_for('multumire')}}">
            <label for='nume'> Nume: </label>
            <input type="text" name="nume">
            <label for='prenume'> Prenume: </label>
            <input type="text" name="prenume">
            <input type="submit" value="Transmite formularul">
        </form>
    </div>
{% endblock %}
```

## Pagina HTML de mulțumire

- Se creează un șablon .html numit multumire.html pentru afișarea unui mesaj de mulțumire cu specificarea numelui și prenumelui

```
{% extends "sablon.html" %}

{% block content %}
    <div class="jumbotron">
        <h1> {{nume}} {{prenume}}, Vă mulțumim pentru autentificare!</h1>
    </div>

{% endblock %}
```

## Pagina HTML de eroare

- Se creează un șablon .html numit eroare\_404.html pentru afișarea unui mesaj de eroare atunci când pagina accesată nu există

```
{% extends "sablon.html" %}
```

```
{% block content %}
```

```
<div class="jumbotron">
```

```
<p>Ne pare rău! Pagina solicitata nu a fost gasită</p>
```

```
</div>
```

```
{% endblock %}
```



## Scriptul Python

### ■ Se modifică conținutul fișierului app1.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def acasa():
    return render_template('acasa.html')

@app.route('/autentificare')
def autentificare():
    return render_template('autentificare.html')

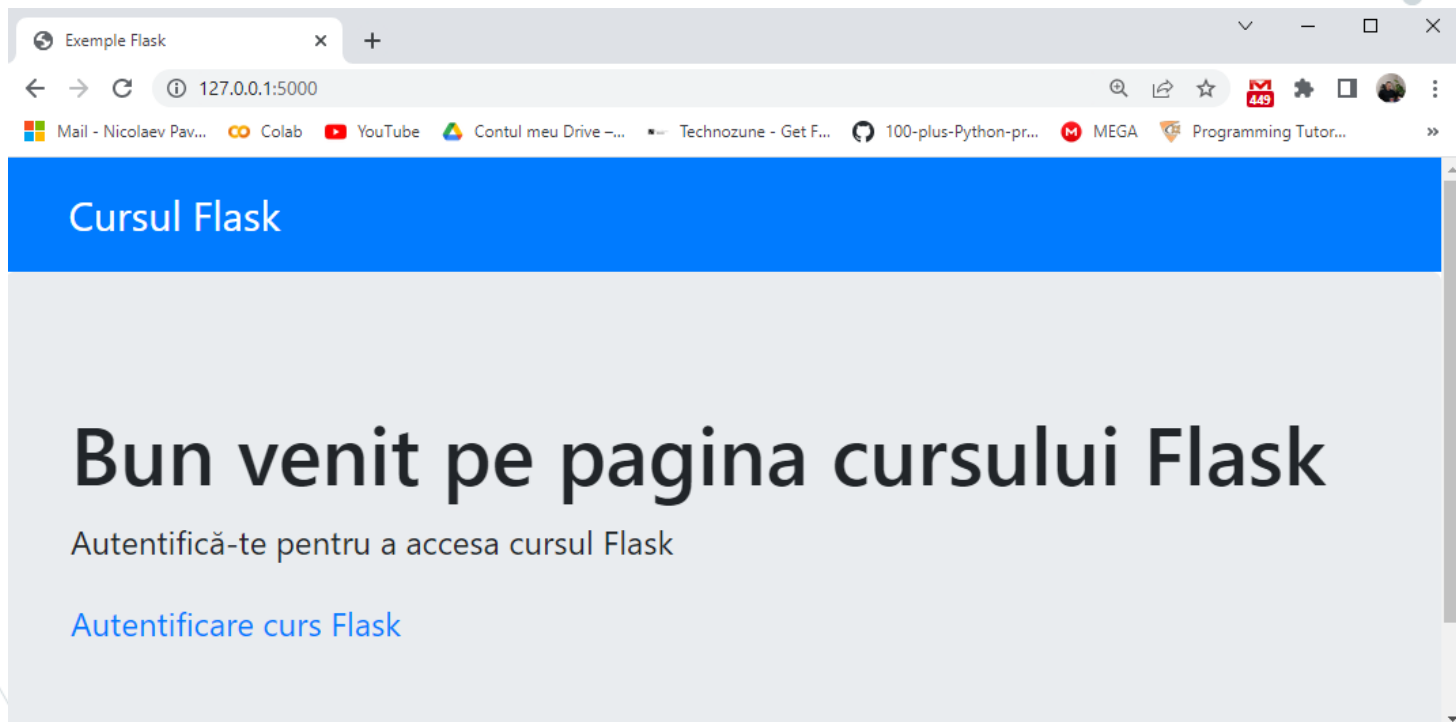
@app.route('/multumire')
def multumire():
    nume = request.args.get('nume')
    prenume = request.args.get('prenume')
    return render_template('multumire.html', nume=nume, prenume=prenume)

@app.errorhandler(404)
def page_not_found(e):
    return render_template('eroare_404.html')

if __name__ == '__main__':
    app.run(debug=True)
```

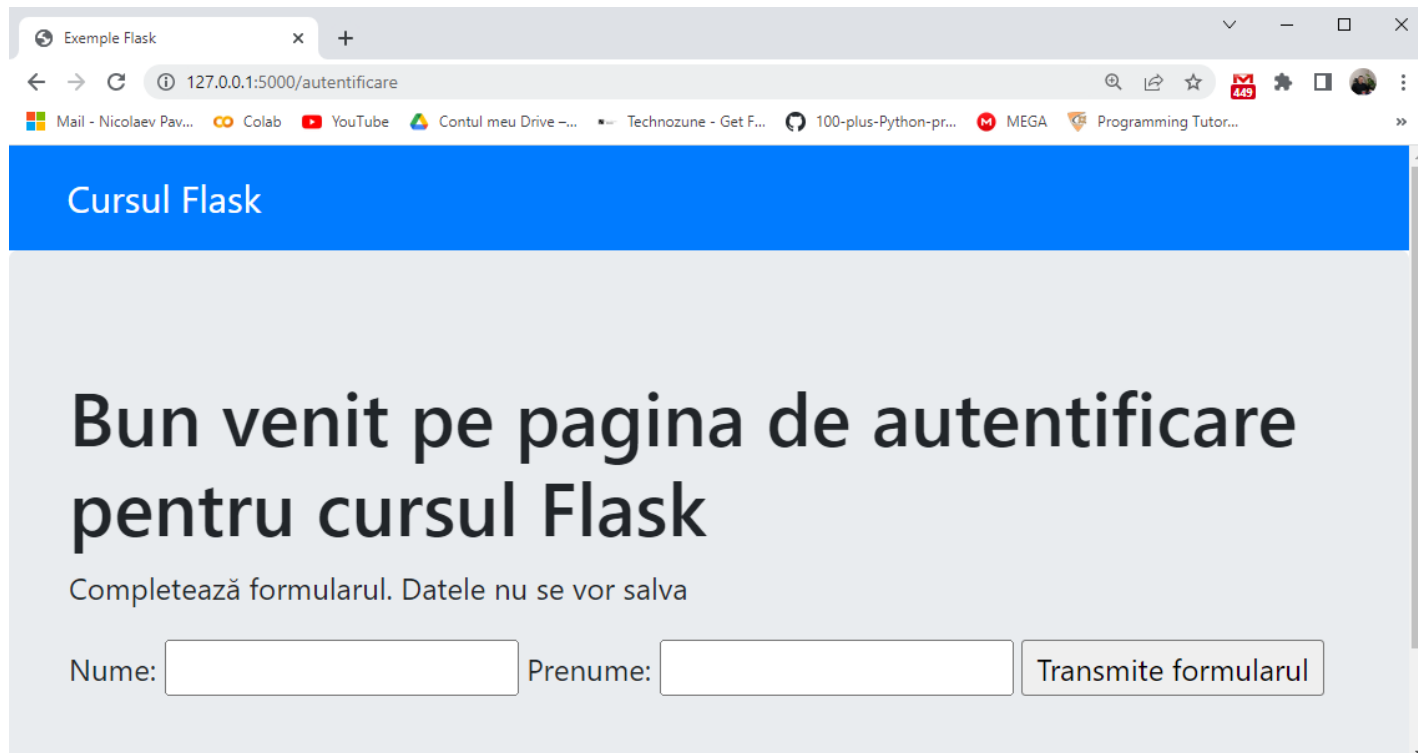
## Pagina web de pornire

- În browser se introduce adresa `http://127.0.0.1:5000`



## Pagina web de autentificare

- Se accesează link-ul Autentificare curs Flask de pe pagina de pornire



Exemple Flask

127.0.0.1:5000/autentificare

Mail - Nicolaev Pav... Colab YouTube Contul meu Drive... Technozone - Get F... 100-plus-Python-pr... MEGA Programming Tutor...

### Cursul Flask

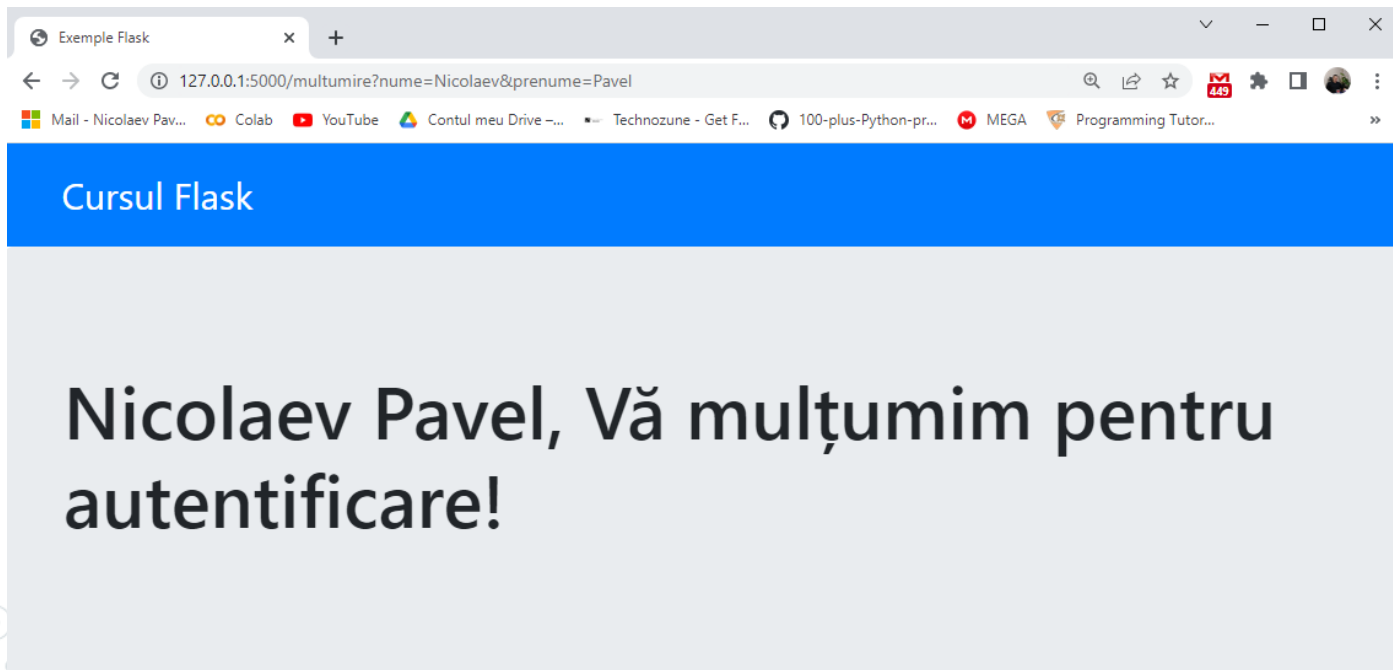
# Bun venit pe pagina de autentificare pentru cursul Flask

Completează formularul. Datele nu se vor salva

Nume:  Prenume:

## Pagina web de mulțumire

- Se completează și se transmite formularul din pagina de autentificare



## Pagina web de eroare

- În browser se introduce adresa <http://127.0.0.1:5000/> urmată de o extensie inexistentă

