|  |  |
| --- | --- |
| sigla_usv_nou | Universitatea „Ştefan cel Mare” - Suceava  Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor |

Proiect la Rețele de Calculatoare din cadrul cursurilor de Licenţă

Calculatoare

**PROIECT**

**Server și client DNS**

Student: Dascălu Samuel, grupa 3121A

- SUCEAVA 2022 -

**CUPRINS**

[1. INTRODUCERE](#_Toc528314028) 4

[1.1. Rezumat proiect](#_Toc528314029) 4

[1.2. Obiective](#_Toc528314031) 4

[1.3. Programe și pachete necesare](#_Toc528314031) 4

[2. DOMENIUL STUDIAT](#_Toc528314032) 5-7

[3. NOȚIUNI TEORETICE](#_Toc528314037) 5-6

[3.1. DNS](#_Toc528314031) 5-6

[3.2. Noțiuni Python necesare](#_Toc528314031) 6-7

[4. CODUL SURSĂ](#_Toc528314037) 8-10

[4.1. Pachete](#_Toc528314031) 8

[4.2. Setare nume gazdă](#_Toc528314031) 8

[4.3. Deschidere port](#_Toc528314031) 8

[4.4. Clasa MyServer](#_Toc528314031) 8

[4.5. Funcția doGet() și rezolvarea adresei](#_Toc528314031) 9

[4.6. Server](#_Toc528314031) 10

[5. REZULTAT FINAL](#_Toc528314037) 11

[6. BIBLIOGRAFIE](#_Toc528314037) 13

[7. ANEXE](#_Toc528314037) 14-15

**TABELUL cu FIGURI**

*[Fig. 1. Logo Thonny](#_Toc528314045) 4*

*[Fig. 2. Shortcut Thonny](#_Toc528314046) 4*

*[Fig. 3. Logo Browser Chrome](#_Toc528314047) 4*

*[Fig. 4. Pachetul dns.resolver](#_Toc528314048) 5*

*[Fig. 5. Exemplu de clasă](#_Toc528314049) 6*

*[Fig. 6. Exemplu de cod HTML integrat în Python](#_Toc528314049) 7*

*[Fig. 7. Tratarea excepțiilor Python](#_Toc528314049) 7*

*[Fig. 8. Pachete importate](#_Toc528314049) 8*

*[Fig. 9. Creare nume gazdă](#_Toc528314049) 8*

*[Fig. 10. Deschiderea portului](#_Toc528314049) 8*

*[Fig. 11. Declarare Clasa MyServer](#_Toc528314049) 8*

*[Fig. 12. Funția doGet() partea 1](#_Toc528314049) 9*

*[Fig. 13. Funția doGet() partea 2](#_Toc528314049) 9*

*[Fig. 14. Funția doGet() partea 3 1](#_Toc528314049)0*

*[Fig. 15. Menținerea server-ului și închiderea lui](#_Toc528314049) 10*

*[Fig. 16. Adresă corectă(pagina web) 11](#_Toc528314049)*

*[Fig. 17. Apăsare buton fără adresă(pagina web) 11](#_Toc528314049)*

*[Fig. 18. Adresă inexistentă(pagina web) 11](#_Toc528314049)*

*[Fig. 19. Harta afișată după introducere adresă (rezultat final) 1](#_Toc528314049)2*

1. **INTRODUCERE:**
   1. **Rezumat proiect:**

Proiectul constă în identificarea IP-ului corespunzător unei adrese literare(exemplu - google.com) și afișarea coordonatelor IP-ului server-ului, în browser, pe o hartă de tip Google Maps. Proiectul are, astfel, ca scop, demonstrarea funcționalității conceptului de server și client DNS.

* 1. **Obiective:**
* Crearea unui server web(http) în limbaj de programare Python;
* Deschiderea unui port pentru server;
* Importarea facilităților DNS și GEOCODER(aflarea coordonatelor unui IP);
* Folosirea limbajului HTML pentru editarea paginii web;
* Afișarea pe pagina server-ului http a unui buton care așteaptă scrierea adresei dorite de către client;
* Afișarea pe pagina web, pe o hartă Google Maps, a locației IP-ului server-ului găsit.

* 1. **Programe și pachete necesare:**

- Limbajul de programare folosit în realizarea acestui proiect este **Python**, deci pentru un astfel de cod este nevoie de un program compatibil cu Python. În acest caz, este folosit **Thonny**. Se mai pot folosi programe precum Atom, PyCharm, etc.

F*ig. 1. Logo Thonny* F*ig. 2. Shortcut Thonny*

- Programul Thonny este o platformă bună pentru începători, fiind ușor de instalat pe Windows, Mac sau Linux la adresa https://thonny.org/.

- Proiectul se folosește de afișarea pe o pagină web, deci este nevoie de **conexiune la internet** și un **brower**(Chrome, Mozilla, etc.)



F*ig. 3. Logo Browser Chrome*

- Ca pachete de Python, se vor folosi: **HTTPServer**, **time**, **dns.resolver**, **geocoder**.

1. **DOMENIUL STUDIAT**

- Fiecare “site” de pe Internet are alocată o **adresă IP**. Folosirea de către utilizatorii obișnuiți a acestor adrese este dificilă și poate genera erori. Astfel, este necesar un protocol care să facă corespondența dintre numele diverselor componente de rețea și adresele lor IP. Această problemă este rezolvată de către protocolul DNS (Domain Names System).

- La nivelul comunicației sunt utilizate exclusiv adresele IP, adresele literale fiind doar o manieră de prezentare prietenoasă a adreselor locațiilor. Identificarea în rețele pe bază de adrese IP este reglementată prin principalul protocol de comunicație la nivel Internet (nivel rețea în modelul ISO/OSI), protocolul Internet, Internet Protocol - IP. Gestiunea adreselor IP se face pe criterii de localizare fizică, criterii geografice.

-Proiectul are la centrul său conceptul de DNS.

- Pachetul din programare care introduce protocolul DNS cu care se realizează corespondența dintre rețea și adresa IP este **dns.resolver**.



F*ig. 4. Pachetul dns.resolver*

1. **NOȚIUNI TEORETICE**

**3.1. DNS**

-Un **sistem de nume de domeniu** (abreviat **DNS**, în engleză *Domain Name System*) este un [sistem distribuit](https://ro.wikipedia.org/w/index.php?title=Sistem_distribuit&action=edit&redlink=1" \o "Sistem distribuit — pagină inexistentă) de păstrare și interogare a unor date arbitrare într-o structură ierarhică. Cea mai cunoscută aplicație a DNS este structură ierarhică. Cea mai cunoscută aplicație a DNS este gestionarea domeniilor în [Internet](https://ro.wikipedia.org/wiki/Internet" \o "Internet).

- Caracteristicile sistemului de nume (DNS) sunt:

* folosește o structură ierarhizată;
* deleagă autoritatea pentru nume;
* [baza de date](https://ro.wikipedia.org/wiki/Baz%C4%83_de_date" \o "Bază de date) cu numele și adresele IP este distribuită.

- La începutul existenței Internetului, când numărul de calculatoare legate în rețea era relativ mic, a existat o tabelă realizată de organizația NIC (Network Information Center), care făcea **corespondența dintre adresa IP și numele gazdei**. Numele de gazde erau șiruri de caractere, fără a avea o organizare ierarhică.

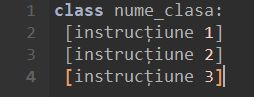
- Tipic, procesul de rezoluție a numelor se desfășoară astfel:

1. Name resolverul primește de la o aplicație client TCP/IP un nume; acesta formulează o interogare primului server de nume din lista serverelor;
2. Serverul de nume (DNS) determină daca este mandatat (autorizat) pentru domeniul respectiv (dacă există configurată o zonă DNS care conține numele respectiv);
3. Dacă este autorizat, transmite răspunsul clientului;
4. Dacă nu, transmite o interogare altui server de nume pentru un răspuns autorizat; obține răspunsul autorizat și transmite clientului un răspuns neautorizat; totodată stochează răspunsul local pentru a răspunde la alte cereri pentru același nume.
5. Resolverul de nume transmite răspunsul aplicației utilizator și îl păstrează într-un cache pentru o anumită perioadă;
6. Dacă name resolverul nu primește un răspuns într-un anumit timp, transmite cererea următorului server de nume din listă. Când lista este epuizată, va genera o eroare.

**3.2. Noțiuni Python abordate**

**Clase.**

- O clasă este creată folosind cuvântul cheie **class**. Atributele şi metodele clasei sunt listate într-un bloc indentat.



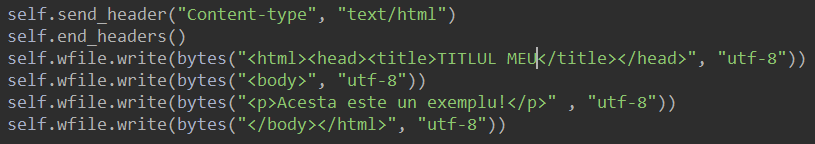
F*ig. 5. Exemplu de clasă.*

- Clasele şi metodele au o diferenţă specifică faţă de funcţiile obişnuite - ele trebuie să aibă un prefix suplimentar care trebuie adăugat la începutul listei de parametri, dar nu trebuie să-i daţi o valoare când apelaţi metoda, Python o va furniza. Această variabilă specială se referă la obiectul însuşi (engl. - self) şi prin convenţie este numită **self**. Când ne referim la membrii clasei, vom folosi self.membru,

într-un mod asemănător cu folosirea "this" din Java.

**Cod HTML.**

- Codul HTML(limbaj folosit în proiect) cu care se editează pagina web direct din codul sursă Python trebuie convertit cu o metodă, astfel încât compilatorul să îl poată interpreta. În rest, codul HTML se scrie la fel ca într-un Notepad obișnuit. Exemplu de cod:

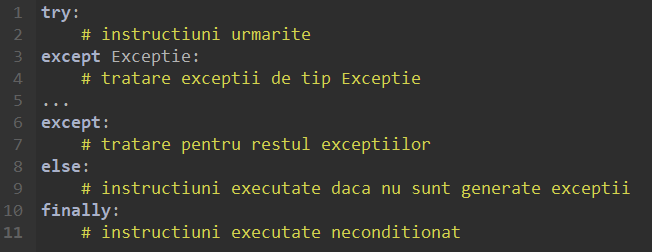


F*ig. 6. Exemplu de cod HTML integrat în Python*

**Excepții.**

- Excepțiile oferă un mecanism eficient de identificare și rezolvare a erorilor care apar în timpul execuției unui program. În limbajul Python există posibilitatea tratării excepțiilor prin stabilirea unei căi alternative de continuare a execuției programului.

- Limbajul Python oferă o soluție eficientă de rezolvare a excepțiilor care apar într-un program prin intermediul mecanismului de tratare a excepțiilor. Implementarea acestei soluții se face folosind construcții de tipul **try .. except .. finally**.



F*ig. 7. Tratarea excepțiilor Python*

1. **CODUL SURSĂ**

**4.1. Pachete**

- Pachetele implementate în cod conțin metodele necesare pentru:

- creare server web (http server)

- rezolvarea DNS (dns.resolver)

- aflarea coordonatelor după IP (geocoder)

****

F*ig. 8. Pachete importate*

**4.2. Setare nume gazdă**

8

F*ig. 9. Creare nume gazdă*

**4.3. Deschidere port server**

**8**

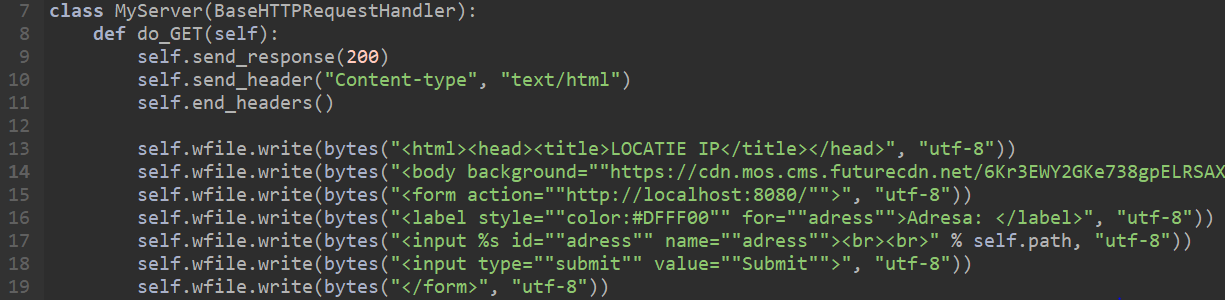
F*ig. 10. Deschiderea portului*

**4.4. Clasa MyServer**

**13**

F*ig. 11. Declarare Clasa MyServer*

**4.5. Funcția doGet() și rezolvarea adresei**

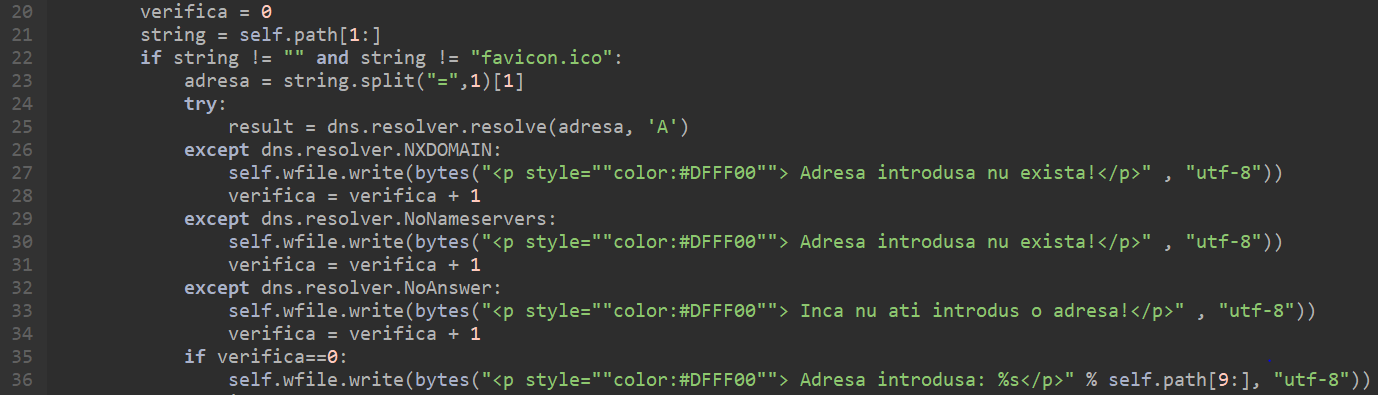
****

F*ig. 12. Funcția doGet() partea 1*

-Funcția **doGet(self)** din cadrul clasei **MyClass** rezolvă un request de la browser spre server și începe cu introducerea prin cod HTML a elementelor ce vor apărea prima dată pe pagina web în fața clientului.

- Linia 11 se ocupă cu înștiințarea programului că se va folosi cod HTML.

- La deschiderea paginii web vor fi afișate spațiul de introducere a adresei dorite a clientului(**self.path**) și butonul de **Submit**, care va declanșa o cerere de tip http către server.



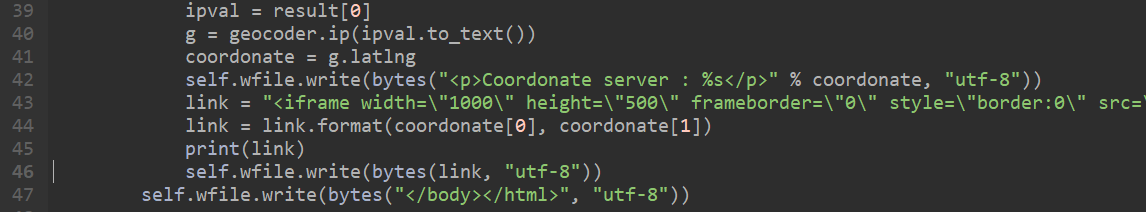
F*ig. 13. Funcția doGet() partea 2*

- În codul de mai sus este rezolvată adresa introdusă, prin protocolul DNS, și se află IP-ul. Apoi se tratează tipurile de greșeli pe care le poate face clientul la introducerea adresei.

- Variabila **string** va conține numele adresei introduse. Condițiile din primul *if* tratează situația în care butonul Submit este apăsat fără a se introduce o adresă (*string* este *NULL*) sau dacă *string* conține adresa ,*,favicon.ico*”, adresă accesată în plus de Thonny în urma introducerii adresei de către client.

- Se folosesc excepțiile, greșelile potențiale ale clientului fiind interpretate de către codul Python ca pe niște erori, acestea fiind scrise în dreptul cuvântului cheie **except**. Astfel, sunt luate în calcul cazurile în care clientul introduce o adresă inexistentă sau nu introduce nicio adresă.

- Variabila **verifica** se ocupă cu verificarea cazului în care s-a introdus o adresă corectă, se afișează un mesaj corespunzător, și se poate continua cu algoritmul.



F*ig. 14. Funcția doGet() partea 3*

- În realitate, prin rezolvarea adresei cu metoda *dns.resolver.resolve* este posibilă salvarea în variabila **result** a o multitudine de IP-uri. În cazul de față, **Ipval** va conține primul IP identificat.

- Variabila **g** va conține datele oferite de pachetul **geocoder** în rezolvarea unui IP. Singurele date care ne interesează, însă, sunt coordonatele IP-ului(g.latlng), pe care le vom salva în vectorul **coordonate**, care va conține latitudinea și longitudinea. De exemplu, pentru “google.com”, vectorul va conține [42.6975, 23.3241].

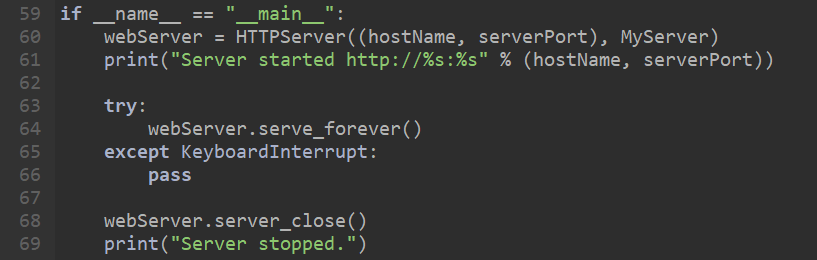
- **Link** conține cod HTML pentru un element de tip *iframe* care incorporeazăharta Google Maps cu noile coordonate introduse și o cheie API necesară accesării unui astfel de serviciu oferit de Google, afișându-se pe pagina web harta navigabilă, cu un pin în locația respectivă.

*Link = "<iframe width=\"1000\" height=\"500\" frameborder=\"0\" style=\"border:0\"src=\"https://www.google.com/maps/embed/v1/place?key=”*

*cheia API”=****{0},{1}****\" allowfullscreen> </iframe>"*

- **Link** conține în loc de coordonate, “*{0},{1}”*, aceasta fiind în strânsă legătură cu comanda *link = link.format(coordonate[0], coordonate[1])* , deoarece: {0}=coordonate[0] și{1}=coordonate[1].

**4.6. Server**

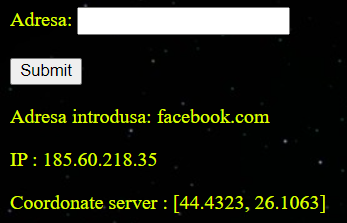


F*ig. 15. Menținerea server-ului și închiderea lui*

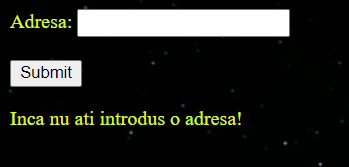
- La linia 60 se creează un obiect de tipul MyServer ce va conține elementele server-ului, adică numele gazdei, portul folosit. Se va afișa în terminalul programului dacă serverul a fost creat cu succes(“Server started”).

- În ultimele linii de cod se se menține serverul deschis continuu, dar poate fi și oprit din terminalul programului, prin tastele Ctrl+c.

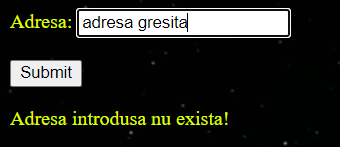
1. **REZULTAT FINAL.**



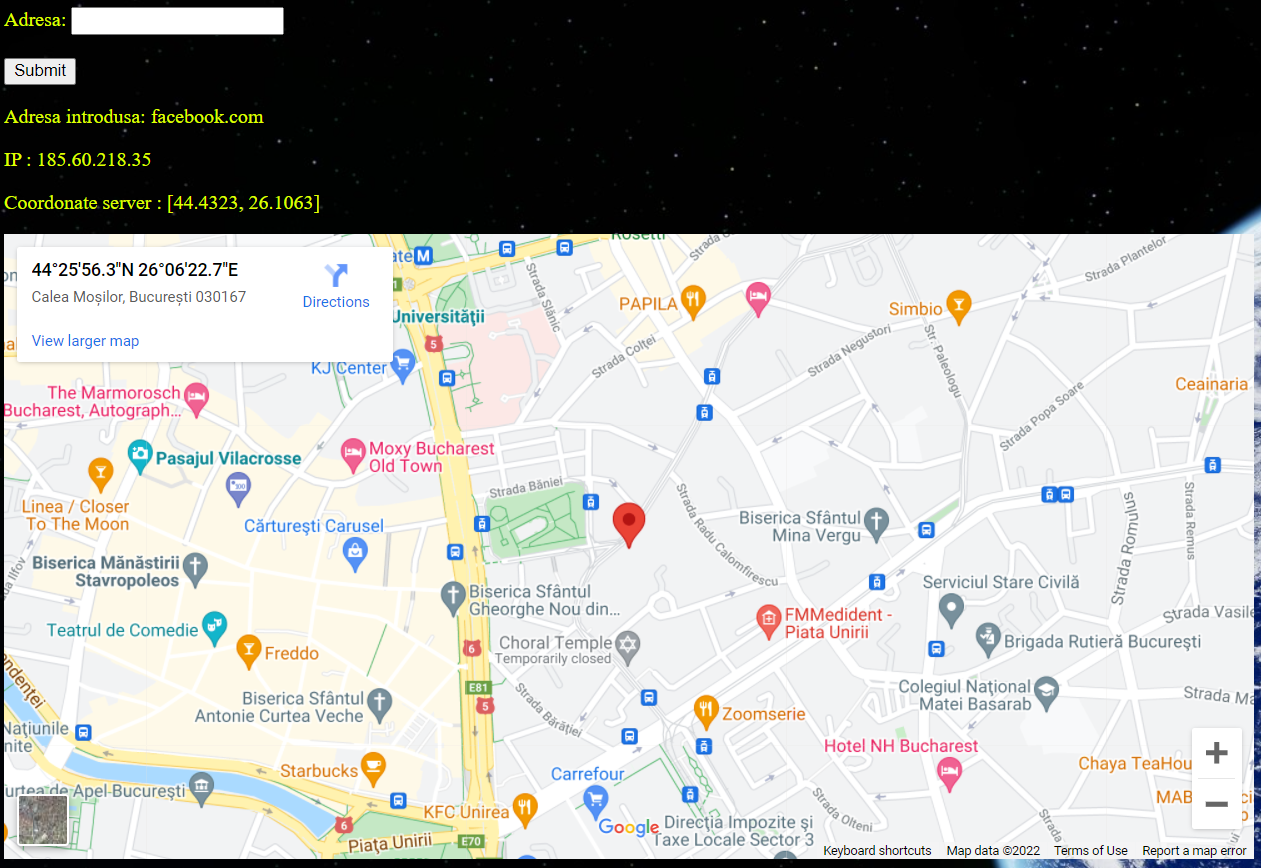
F*ig. 16. Adresa corectă(pagina web)*



F*ig. 17. Apăsare buton fără adresă(pagina web)*



F*ig. 18. Adresă inexistentă(pagina web)*



F*ig. 19. Harta afișată după introducere adresă (rezultat final)*

**Posibile îmbunătățiri**

- La acest proiect se mai poate adăuga și opțiunea să poată fi căutată *locația direct după IP*, iar când locația este găsită, pe pagina web să apară *informații și poze cu orașul/țara respectivă*.

- O funcție care ar demonstra conectivitatea prin rețele ar fi *trace-route,* ceea ce la acest proiect ar putea fi reprezentat printr-o animație pe hartă a conectării succesive la diverse servere.

1. **BIBLIOGRAFIE**
2. #https://pythonbasics.org/webserver/
3. #https://ipinfo.io/
4. #https://www.dnspython.org/examples/
5. #https://docs.mapbox.com/help/glossary/static-images-api/
6. #https://www.w3schools.com/tags/tryit.asp?filename=tryhtml\_input\_test
7. #https://www.tutorialspoint.com/python\_network\_programming/python\_dns\_look\_up.htm
8. #https://developers.google.com/maps/documentation/embed/embedding-map#place\_mode
9. https://thonny.org/

**7. ANEXE**

**Fișierul DNS.py**

from http.server import BaseHTTPRequestHandler, HTTPServer

import dns.resolver

import geocoder

hostName = "localhost"

serverPort = 8080

class MyServer(BaseHTTPRequestHandler):

def do\_GET(self):

self.send\_response(200)

self.send\_header("Content-type", "text/html")

self.end\_headers()

self.wfile.write(bytes("<html><head><title>LOCATIE IP</title></head>", "utf-8"))

self.wfile.write(bytes("<body background=""https://cdn.mos.cms.futurecdn.net/6Kr3EWY2GKe738gpELRSAX.jpg"">", "utf-8"))

self.wfile.write(bytes("<form action=""http://localhost:8080/"">", "utf-8"))

self.wfile.write(bytes("<label style=""color:#DFFF00"" for=""adress"">Adresa: </label>", "utf-8"))

self.wfile.write(bytes("<input %s id=""adress"" name=""adress""><br><br>" % self.path, "utf-8"))

self.wfile.write(bytes("<input type=""submit"" value=""Submit"">", "utf-8"))

self.wfile.write(bytes("</form>", "utf-8"))

verifica = 0

string = self.path[1:]

if string != "" and string != "favicon.ico":

adresa = string.split("=",1)[1]

try:

result = dns.resolver.resolve(adresa, 'A')

except dns.resolver.NXDOMAIN:

self.wfile.write(bytes("<p style=""color:#DFFF00""> Adresa introdusa nu exista!</p>" , "utf-8"))

verifica = verifica + 1

except dns.resolver.NoNameservers:

self.wfile.write(bytes("<p style=""color:#DFFF00""> Adresa introdusa nu exista!</p>" , "utf-8"))

verifica = verifica + 1

except dns.resolver.NoAnswer:

self.wfile.write(bytes("<p style=""color:#DFFF00""> Inca nu ati introdus o adresa!</p>" , "utf-8"))

verifica = verifica + 1

if verifica==0:

self.wfile.write(bytes("<p style=""color:#DFFF00""> Adresa introdusa: %s</p>" % self.path[9:], "utf-8"))

for ipval in result:

self.wfile.write(bytes("<p style=""color:#DFFF00""> IP : %s</p>" % ipval.to\_text(), "utf-8"))

ipval = result[0]

g = geocoder.ip(ipval.to\_text())

coordonate = g.latlng

self.wfile.write(bytes("<p style=""color:#DFFF00""> Coordonate server : %s</p>" % coordonate, "utf-8"))

link = "<iframe width=\"1000\" height=\"500\" frameborder=\"0\" style=\"border:0\" src=\"https://www.google.com/maps/embed/v1/place?key=AIzaSyCP2Rn6dZ9OKtmx4Au2c74hYRdT3r1HNEs&q={0},{1}\" allowfullscreen> </iframe>"

link = link.format(coordonate[0], coordonate[1])

print(link)

self.wfile.write(bytes(link, "utf-8"))

self.wfile.write(bytes("</body></html>", "utf-8"))

if \_\_name\_\_ == "\_\_main\_\_":

webServer = HTTPServer((hostName, serverPort), MyServer)

print("Server started http://%s:%s" % (hostName, serverPort))

try:

webServer.serve\_forever()

except KeyboardInterrupt:

pass

webServer.server\_close()

print("Server stopped.")