UNIVERSITATEA “POLITEHNICA” BUCUREȘTI

Facultatea de Electronică Telecomunicații și Tehnologia Informației

PROIECT 3

Aplicatie web pentru cautare de hoteluri – Python  
  
*Aplicatia web va permite utilizatorului sa caute hoteluri in functie de oras, perioada selectata si numarul de persoane, folosind API-ul Hotels (https://rapidapi.com/apidojo/api/hotels4). Aplicatia trebuie sa aiba o interfata responsive si sa permita ordonarea rezultatelor dupa pret crecator/descrescator, recenzii, etc., precum si sa permita utilizatorului sa vizualizeze imaginile hotelurilor.*

Profesor coordonator

*Florea Bogdan*

Student

*Codreanu Agnia 445A*

București 2022

Cuprins

[Introducere 2](#_Toc136999749)

[Tehnologii și framework-uri utilizate 2](#_Toc136999750)

[Python 2](#_Toc136999751)

[Flask 3](#_Toc136999752)

[HTML, CSS și JavaScript 4](#_Toc136999753)

[Bootstrap 5](#_Toc136999754)

[Fluxul de lucru și implementarea 7](#_Toc136999755)

[Configurarea proiectului 8](#_Toc136999756)

[Implementarea aplicației 9](#_Toc136999757)

[Prezentarea interfeței utilizatorului 21](#_Toc136999758)

[Concluzie 24](#_Toc136999759)

[Referințe 25](#_Toc136999760)

[Anexa 25](#_Toc136999761)

[Figure 1 Abonarea la API-ul Hotels 6](#_Toc136996986)

[Figure 2 Prezentare generală a API-ului Hotels 7](#_Toc136996987)

[Figure 3 Fereastra "Create New Project" in PyCharm 8](#_Toc136996988)

[Figure 4 Pagina index.html 21](#_Toc136996989)

[Figure 5 Cautarea hotelurilor in functie de oras 22](#_Toc136996990)

[Figure 6 Pagina hotels.html 22](#_Toc136996991)

[Figure 7 Opțiuni de sortare 22](#_Toc136996992)

[Figure 8 Pagina hotel\_link.html 23](#_Toc136996993)

# Introducere

Acest proiect consta in crearea unei aplicații web in Python care sa permită utilizatorului sa caute hoteluri in funcție de oraș, perioada selectata si numărul de persoane, folosind API-ul Hotels (<https://rapidapi.com/apidojo/api/hotels4/>).

Scopul acestei aplicații este de a vă oferi o experiență ușoară și plăcută în găsirea hotelului perfect pentru călătoria dvs. Interfața aplicației este concepută pentru a fi responsive, ceea ce înseamnă că se va adapta în mod automat la diferite dimensiuni de ecran, oferindu-vă o experiență consistentă atât pe desktop, cât și pe dispozitive mobile.

Principalele caracteristici ale aplicației includ:

* Căutare avansată: Puteți introduce orașul dorit, perioada șederii și numărul de persoane pentru a căuta hoteluri disponibile în acele date.
* Filtrare și sortare: Puteți filtra rezultatele căutării după criterii precum preț, recenzii, distanță față de atracții turistice și multe altele. De asemenea, puteți sorta rezultatele în ordine crescătoare sau descrescătoare, în funcție de preferințele dvs.
* Vizualizare detalii și imagini: Puteți vizualiza informații detaliate despre fiecare hotel, inclusiv, recenzii și fotografii. Aceasta vă ajută să luați o decizie informată înainte de a face rezervarea.

Pentru a realiza aceste funcționalități, aplicația utilizează API-ul Hotels, care oferă un set complet de funcționalități pentru căutarea hotelurilor. API-ul furnizează date precise și actualizate despre hoteluri, inclusiv informații despre prețuri, facilități, recenzii și imagini.

Această documentație vă va ghida in crearea și utilizarea aplicației. Veți afla cum să interacționați cu interfața aplicației, cum să căutați hoteluri, cum să aplicați filtre și sortări.

# Tehnologii și framework-uri utilizate

Pentru a crea aplicația web, vom folosi următoarele tehnologii și framework-uri:

* ***Python***: Un limbaj de programare puternic și versatil, care va fi folosit pentru logica aplicației și comunicarea cu API-ul Hotels.
* ***Flask***: Un framework web ușor și flexibil pentru Python, care va fi utilizat pentru a construi și gestiona aplicația web. Flask permite dezvoltarea rapidă a aplicațiilor web și oferă funcționalități necesare precum rute, șabloane, gestionarea formularului și integrarea cu API-ul Hotels.
* ***HTML, CSS și JavaScript***: Aceste tehnologii de bază vor fi utilizate pentru a crea interfața utilizatorului (UI) a aplicației web și pentru a-l face responsive și atractiv.
* ***Bootstrap***: Un framework front-end popular și responsiv, care va fi utilizat pentru a adăuga stiluri și componente predefinite la interfața aplicației web.
* ***API-ul Hotels de la RapidAPI***: Acesta va fi utilizat pentru a obține date despre hoteluri, inclusiv informații despre prețuri, recenzii și imagini.

## Python

Python este un limbaj de programare interpretat, de înalt nivel și generalist, care a câștigat o popularitate considerabilă în industria software-ului. Creat de Guido van Rossum și lansat inițial în 1991, Python se remarcă prin sintaxa sa simplă și ușor de înțeles, ceea ce face ca dezvoltarea de aplicații să fie rapidă și eficientă.

Iată câteva caracteristici-cheie ale limbajului Python:

* Sintaxă clară și citibilă: Python se bazează pe un stil de scriere clar și concis, care încurajează programatorii să scrie cod care este ușor de înțeles și de întreținut. Indentarea spațiilor este obligatorie în Python, ceea ce duce la un cod bine structurat și organizat.
* Versatilitate: Python este un limbaj de programare versatil, care poate fi utilizat într-o varietate de domenii și aplicații, precum dezvoltarea web, analiza datelor, inteligența artificială, automatizarea sarcinilor și multe altele. Există o gamă largă de biblioteci și framework-uri disponibile pentru a sprijini dezvoltarea în aceste domenii.
* Interpretat și interactiv: Python este un limbaj interpretat, ceea ce înseamnă că codul Python este executat linie cu linie de către un interpretor. Aceasta facilitează testarea și depanarea codului, deoarece programatorii pot executa și experimenta cu codul imediat, fără a-l compila în prealabil.
* Bogată colecție de biblioteci: Python are o comunitate activă și generoasă, care a dezvoltat o varietate de biblioteci și module, acoperind o gamă largă de funcționalități. De exemplu, biblioteca NumPy este utilizată pentru prelucrarea eficientă a matricelor și a datelor numerice, biblioteca Pandas este folosită pentru manipularea datelor structurate, iar biblioteca TensorFlow este populară în domeniul învățării automate.
* Portabil și compatibil: Python este un limbaj portabil, ceea ce înseamnă că codul Python poate fi rulat pe mai multe platforme și sisteme de operare fără modificări majore. De asemenea, Python este compatibil cu alte limbaje de programare, permițând interoperabilitatea cu codul scris în alte limbaje.
* Comunitate puternică: Python are o comunitate vastă și activă de programatori, care contribuie la dezvoltarea și întreținerea limbajului. Există numeroase forumuri, grupuri de utilizatori și resurse online disponibile pentru a obține ajutor, a împărtăși cunoștințe și a explora noi tehnologii și tendințe în Python.
* Documentație extinsă: Python are o documentație bine structurată și extinsă, care acoperă atât limbajul însuși, cât și bibliotecile și framework-urile asociate. Acest lucru facilitează învățarea și dezvoltarea cu Python și oferă un suport solid pentru programatorii începători și avansați.

Python este un limbaj de programare puternic și flexibil, care este folosit pe scară largă în industrie și în comunitatea de dezvoltatori. Datorită simplității sale, Python este adesea alegerea preferată pentru începători, dar poate fi, de asemenea, utilizat eficient în proiecte complexe și de amploare [4].

## Flask

Flask este un framework web minimalist și flexibil scris în limbajul de programare Python. A fost creat pentru a permite dezvoltarea rapidă a aplicațiilor web și se concentrează pe simplitate și extensibilitate. Iată câteva caracteristici și concepte cheie ale Flask:

* Rutare: Flask facilitează definirea rutelor URL și asocierea acestora cu funcții Python numite "viziuni". Atunci când un utilizator accesează o anumită rută, Flask va apela viziunea asociată și va returna răspunsul corespunzător.
* Șabloane: Flask integrează un sistem de șabloane care permite generarea de pagini web dinamice. Șabloanele combină codul Python cu HTML pentru a permite manipularea și afișarea datelor în mod dinamic.
* Manipularea formularului: Flask simplifică preluarea datelor din formularele HTML și procesarea acestora în cadrul aplicației. Acest lucru include validarea datelor, gestionarea erorilor și salvarea datelor într-o bază de date sau folosirea lor în alte scopuri.
* Extensibilitate: Flask este conceput să fie extensibil, ceea ce înseamnă că dezvoltatorii pot adăuga funcționalități suplimentare prin intermediul unor extensii sau module de terțe părți. Există o gamă largă de extensii disponibile pentru Flask, care acoperă domenii precum autentificare, baze de date, formulare, sesiuni și multe altele.
* Configurare simplă: Flask oferă o configurație flexibilă și ușor de utilizat. Dezvoltatorii pot seta variabile de configurare pentru a personaliza comportamentul aplicației lor, cum ar fi chei secrete, setări de bază de date, etc.
* Dezvoltare locală și scalabilitate: Flask vine cu un server web de dezvoltare încorporat, ceea ce înseamnă că dezvoltatorii pot rula și testa aplicațiile lor într-un mediu local înainte de a le implementa pe un server real. Flask poate fi, de asemenea, scalat pentru a face față unor aplicații web mai mari și mai complexe prin intermediul tehnologiilor și a practicilor adecvate.

Flask este considerat unul dintre cele mai populare și ușor de învățat framework-uri web în limbajul Python. Are o comunitate activă de dezvoltatori care contribuie cu extensii și documentație pentru a face dezvoltarea aplicațiilor web mai ușoară și mai eficientă [3].

## HTML, CSS și JavaScript

HTML, CSS și JavaScript sunt trei tehnologii fundamentale pentru dezvoltarea paginilor web moderne. Fiecare dintre ele îndeplinește un rol specific în crearea și stilizarea conținutului web interactiv.

HTML (HyperText Markup Language) este limbajul de marcare utilizat pentru structurarea și organizarea conținutului unei pagini web. Este folosit pentru a defini elemente precum titluri, paragrafe, imagini, linkuri și multe altele. HTML utilizează o sintaxă bazată pe etichete și elemente înierbate, care permit crearea unei structuri logice a paginii web. Browser-ele interpretează apoi acest cod HTML și îl afișează în mod corespunzător utilizatorilor [5].

CSS (Cascading Style Sheets) este un limbaj de stilizare care este utilizat pentru a aplica aspectul vizual al paginilor web. Cu ajutorul CSS, poți defini culori, fonturi, dimensiuni, layout-uri și alte proprietăți de stilizare pentru elementele HTML. CSS funcționează prin selectarea elementelor HTML și aplicarea de reguli de stil specificate într-un fișier separat sau în cadrul paginii HTML însăși. Aceasta permite o separare clară între conținutul și stilul paginii web, permițând o mai mare flexibilitate în gestionarea aspectului vizual [6].

JavaScript este un limbaj de programare de înalt nivel care oferă posibilitatea de a face paginile web interactive și dinamice. Este folosit pentru a adăuga funcționalități interactive, cum ar fi manipularea și modificarea conținutului HTML, gestionarea evenimentelor (de exemplu, un clic pe un buton), validarea datelor, interacțiunea cu serverele și multe altele. JavaScript poate fi inclus direct în codul HTML sau poate fi scris în fișiere separate și apoi încărcat și rulat în cadrul paginii [7].

Aceste trei tehnologii - HTML, CSS și JavaScript - lucrează împreună pentru a crea experiențe web captivante și interactivitate pentru utilizatori. HTML oferă structura și conținutul paginii, CSS oferă stilizarea și aspectul vizual, iar JavaScript adaugă funcționalitate și interactivitate. Combinând aceste tehnologii, dezvoltatorii web pot crea aplicații web puternice și atrăgătoare.

## Bootstrap

Bootstrap este un framework front-end popular și puternic utilizat pentru dezvoltarea rapidă și ușoară a paginilor web responsive. A fost creat inițial de către Twitter și este acum un proiect open-source.

Caracteristica principală a Bootstrap este sistemul său de grilă (grid system), care permite împărțirea paginilor în coloane și rânduri, oferind astfel un control flexibil asupra layout-ului paginilor. Această abordare bazată pe grilă permite dezvoltatorilor să creeze rapid și ușor un design responsiv, care se adaptează automat la diferite dimensiuni de ecran și dispozitive, cum ar fi computerele desktop, tabletele și telefoanele mobile.

Bootstrap vine cu o varietate de componente și stiluri predefinite, cum ar fi butoane, formulare, navigație, bare de progres, carusel, tabele și multe altele. Aceste componente pot fi utilizate direct și personalizate pentru a se potrivi nevoilor specifice ale unei aplicații sau pagini web.

În plus, Bootstrap oferă și un set extins de clase CSS, care permite stilizarea rapidă și consistentă a elementelor HTML. Aceste clase facilitează aspectul vizual coerent și ușor de gestionat al întregului site.

Bootstrap este dezvoltat folosind HTML, CSS și JavaScript, astfel încât poate fi integrat în proiectele existente fără probleme. Framework-ul include, de asemenea, suport pentru jQuery, o bibliotecă JavaScript populară, care oferă funcționalități suplimentare și interacțiuni dinamice.

Prin utilizarea Bootstrap, dezvoltatorii web beneficiază de avantajele unui design responsiv și a unui aspect vizual atractiv, fără a fi necesară scrierea de cod de la zero. Framework-ul oferă, de asemenea, documentație detaliată și exemple de utilizare, facilitând astfel procesul de învățare și implementare [10].

RapidAPI

RapidAPI este o platformă care permite dezvoltatorilor să acceseze și să utilizeze diverse API-uri într-un mod simplificat și unificat. Iată câteva aspecte cheie despre RapidAPI:

* Marketplace-ul API-urilor: RapidAPI oferă un marketplace bogat de API-uri, care acoperă o gamă largă de domenii, cum ar fi social media, analize, plată, inteligență artificială și multe altele. Dezvoltatorii pot explora și accesa aceste API-uri dintr-un singur loc, fără a mai trebui să caute și să integreze manual fiecare API în parte.
* Integrare și documentație simplificate: Platforma RapidAPI facilitează integrarea API-urilor prin intermediul unui sistem unificat de autentificare și apeluri de solicitare. Dezvoltatorii pot accesa rapid chei de autentificare, documentație și exemple de cod pentru fiecare API disponibil în marketplace. Acest lucru reduce timpul și efortul necesare pentru a integra și a utiliza API-urile.
* Testare și depanare simplificate: RapidAPI oferă un set de instrumente pentru testarea și depanarea API-urilor. Dezvoltatorii pot trimite cereri de test și pot vizualiza răspunsurile, pentru a se asigura că integrarea API-ului este corectă și funcționează conform așteptărilor.
* Monitorizare și analiză: Platforma RapidAPI permite monitorizarea și analiza utilizării API-urilor. Dezvoltatorii pot vizualiza statistici și metrice relevante, cum ar fi răspunsul API-urilor, timpul de răspuns și limitările de rată. Aceste informații pot ajuta la optimizarea performanței și la identificarea potențialelor probleme.
* Suport pentru mai multe limbaje de programare: RapidAPI este compatibil cu mai multe limbaje de programare, inclusiv Python, JavaScript, Ruby, Java și altele. Acest lucru oferă dezvoltatorilor flexibilitate în a utiliza API-urile în mediul lor preferat de dezvoltare.
* Comunitate și suport: RapidAPI oferă o comunitate activă de dezvoltatori și suport tehnic, unde utilizatorii pot discuta, pune întrebări și împărtăși cunoștințe legate de integrarea API-urilor. De asemenea, RapidAPI oferă documentație detaliată și tutoriale pentru a ajuta dezvoltatorii în procesul de integrare și utilizare a API-urilor.

RapidAPI este o soluție puternică pentru dezvoltatorii care doresc să utilizeze și să integreze API-uri în aplicațiile lor. Prin intermediul platformei RapidAPI, dezvoltatorii pot accelera procesul de dezvoltare și pot beneficia de o gamă largă de servicii și funcționalități predefinite, fără a fi nevoie să dezvolte de la zero fiecare funcționalitate în parte [8].

Un API REST (Representational State Transfer) este un stil arhitectural pentru dezvoltarea de servicii web care utilizează protocolul HTTP. API-urile REST sunt construite pe principiul comunicării între client și server prin intermediul cererilor HTTP (GET, POST, PUT, DELETE) și a răspunsurilor HTTP (de obicei în format JSON sau XML). API-urile REST sunt frecvent utilizate pentru a crea aplicații web și mobile.

CRUD (Create, Read, Update, Delete) reprezintă cele patru operații de bază pentru gestionarea datelor în sistemele software. Aceste operații permit crearea, citirea, actualizarea și ștergerea datelor într-o bază de date sau într-un sistem de stocare.

În contextul unui API REST, funcționalitățile CRUD pot fi implementate în felul următor:

* Create (POST): Permite adăugarea de noi resurse în sistem.
* Read (GET): Permite citirea și obținerea de informații despre resursele existente.
* Update (PUT/PATCH): Permite actualizarea datelor unei resurse existente.
* Delete (DELETE): Permite ștergerea unei resurse existente.

Pentru a utiliza funcționalitățile CRUD într-un API REST, se trimit cereri HTTP adecvate către endpoint-urile corespunzătoare. De exemplu, pentru a crea o nouă resursă, se trimite o cerere POST la un anumit endpoint, care conține datele noii resurse. Pentru a citi informații despre o resursă, se trimite o cerere GET la endpoint-ul corespunzător. Actualizarea și ștergerea resurselor se realizează prin cereri PUT/PATCH și DELETE, respectiv.

În general, API-urile REST care oferă funcționalități CRUD sunt proiectate pentru a permite manipularea și gestionarea datelor într-un mod simplu și intuitiv, respectând principiile și convențiile REST [9].

API-ul Hotels oferit de RapidAPI este utilizat pentru a obține informații despre hoteluri și pentru a efectua căutări de hoteluri pe baza criteriilor specificate.

Pentru a utiliza API-ul, se face o cerere HTTP la endpoint-urile corespunzătoare pentru a obține datele dorite. Endpoint-urile pot include căutarea hotelurilor în funcție de locație, dată de check-in/check-out, număr de camere și alte parametri relevanți. De asemenea, există endpoint-uri pentru a obține detalii despre un anumit hotel, inclusiv informații despre camere, facilități, imagini și altele.

Pentru a accesa API-ul Hotels prin RapidAPI, trebuie să aveți un cont pe platforma RapidAPI și să obțineți o cheie de autentificare (API key) pentru a include în cererile dvs. API key este utilizată pentru a autentifica cererile dvs. și a vă asigura că aveți acces la resursele API-ului.

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Figure 1 Abonarea la API-ul Hotels

O imagine care conține text, captură de ecran, Font, număr

Descriere generată automat

Figure 2 Prezentare generală a API-ului Hotels

După obținerea cheii de autentificare, aceasta poate fi inclusă în antetul cererilor HTTP, în mod obișnuit prin intermediul unui câmp de antet numit "X-RapidAPI-Key". Aceasta indică API-ul că sunteți un utilizator autorizat și permite efectuarea cererilor.

Când primiți răspunsul API-ului Hotels, acesta conține datele cerute, care pot fi procesate și afișate în aplicația dvs. web, cum ar fi afișarea rezultatelor căutării de hoteluri sau detalii despre un anumit hotel.

În codul prezentat, se utilizează modulul Python "requests" pentru a efectua cereri HTTP către endpoint-urile API-ului Hotels. Se specifică URL-ul endpoint-ului, se adaugă parametri relevanți și se trimite cererea HTTP pentru a obține răspunsul API-ului. Răspunsul este apoi procesat și utilizat pentru a afișa datele relevante în șabloanele HTML ale aplicației web.

# Fluxul de lucru și implementarea

Procesul de creare a aplicației web va consta în următorii pași:

### Configurarea proiectului

Proiectul Flask este destinat dezvoltării productive a aplicațiilor Flask. PyCharm se ocupă de crearea structurii specifice de directoare și a setărilor.

Pașii de configurare:

1. Deschideți PyCharm și selectați "Create New Project" în ecranul de pornire.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Figure 3 Fereastra "Create New Project" in PyCharm

1. În fereastra "New Project", în partea stângă, faceți clic pe "Flask" în lista de tipuri de proiecte.
2. În fereastra "New Project", alegeți locația și numele proiectului. Puteți opta să creați și un mediu virtual pentru proiect.
3. Asigurați-vă că ați selectat interpretorul corect Python pentru proiect. Dacă nu aveți un interpretor configurat, faceți clic pe "New..." și selectați interpretorul dorit.
4. Faceți clic pe "Create" pentru a finaliza crearea proiectului.

PyCharm creează o aplicație și produce o structură de directoare specifică, pe care o puteți explora în fereastra instrumentului Proiect. Pe lângă asta, PyCharm creează un script Python stub cu numele app.py, care oferă un simplu exemplu „Hello, World!”.

Puteți rula aplicația Flask făcând clic pe butonul de rulare din bara de instrumente sau utilizând combinația de taste Shift + F10. Aplicația va fi disponibilă la adresa http://localhost:5000 sau la o altă adresă specificată în consolă.

În PyCharm, puteți vizualiza și gestiona pachetele în fereastra de instrumente Python Packages și în setările interpretorului Python. Fereastra de instrumente Python Packages oferă cea mai rapidă și ordonată modalitate de a vizualiza și instala pachete pentru interpretorul Python selectat în prezent. Această fereastră este activată în mod implicit și o puteți găsi în grupul inferior al ferestrelor de instrumente. În orice moment, o puteți deschide utilizând meniul principal: View | Tool Windows | Python Packages.

Fereastra de instrumente Python Packages afișează pachetele instalate și pachetele disponibile în depozitele de pachete PyPI și Conda. Utilizați câmpul de căutare pentru a filtra lista de pachete disponibile. Instalați pachetele necesare pentru aplicație. Puteți utiliza următoarea comandă în terminalul PyCharm:

pip install flask requests

Pe măsură ce dezvoltați aplicația dvs., PyCharm vă va ajuta cu completarea automată a codului, depanarea și alte funcționalități utile.

### Implementarea aplicației

#### Creați un fișier Python (de exemplu, app.py) în directorul proiectului.

#### Importați modulele necesare:

from flask import Flask, render\_template, request

import json

from flask\_datepicker import datepicker

import datetime

import ast

import requests

#### Configurați aplicația Flask:

app = Flask(\_\_name\_\_)

#### Definiți rutele aplicației:

##### Ruta pentru pagina principală:

@app.route('/')  
def index():  
 return render\_template('index.html')

În codul prezentat, se definește o rută către URL-ul de bază "/" folosind decoratoarea @app.route('/'). Acesta este un decorator Flask care indică că funcția de mai jos va fi apelată atunci când se accesează URL-ul specificat.

Funcția index() este definită pentru ruta de bază și are rolul de a afișa șablonul HTML "index.html". Aceasta utilizează funcția render\_template() pentru a încărca și a afișa șablonul specificat.

Funcția render\_template() este oferită de Flask și este folosită pentru a procesa și afișa șabloanele HTML. Aceasta primește ca argument numele șablonului și poate primi și argumente suplimentare care pot fi folosite în șablon pentru a personaliza afișarea.

Astfel, atunci când se accesează URL-ul de bază, funcția index() este apelată și returnează șablonul HTML "index.html" către client, care va fi afișat în browser.

##### Ruta pentru căutarea hotelurilor:

@app.route('/hotels', methods=['GET', 'POST'])  
def hotels():  
 url = "https://hotels4.p.rapidapi.com/properties/v2/list"  
  
 check\_in = request.args.get('check-in')  
 check\_out = request.args.get('check-out')  
 #num\_rooms = request.args.get('num-rooms')  
 num\_travellers = request.args.get('num-travellers')  
 # print(num\_rooms, num\_travellers)  
 destination = request.args.get('destination')  
  
 #print(destination)  
  
 destination\_dict = {  
 'latitude': float(destination.split(' ')[0]),  
 'longitude': float(destination.split(' ')[1])  
 }  
 #print(destination\_dict)  
 check\_in\_dict = {  
 'day': int(check\_in.split('-')[2]),  
 'month': int(check\_in.split('-')[1]),  
 'year': int(check\_in.split('-')[0])  
 }  
  
 check\_out\_dict = {  
 'day': int(check\_out.split('-')[2]),  
 'month': int(check\_out.split('-')[1]),  
 'year': int(check\_out.split('-')[0])  
 }  
  
 # Check if a sorting filter is specified in the request  
 filter\_value = request.args.get('sort')  
 if filter\_value==None:  
 filter\_value="RECOMMENDED"  
  
 headers = {  
 "content-type": "application/json",  
 "X-RapidAPI-Key": "5fbc045366msh8a95618fc9afb3ap19795ejsn138b5df7840b",  
 "X-RapidAPI-Host": "hotels4.p.rapidapi.com"  
 }  
 payload = {  
 "currency": "USD",  
 "eapid": 1,  
 "locale": "en\_US",  
 "siteId": 300000001,  
 "destination": {"coordinates": destination\_dict},  
 "checkInDate": check\_in\_dict,  
 "checkOutDate": check\_out\_dict,  
 "rooms": [  
 {  
 "adults": int(num\_travellers),  
 "children": []  
 }  
 ],  
 "resultsStartingIndex": 0,  
 "resultsSize": 200,  
 "sort": filter\_value,  
 "filters": {"price": {  
 "max": 700,  
 "min": 10  
 }}  
 }  
 print(payload)  
 response = requests.request("POST", url, json=payload, headers=headers).json()  
  
 return render\_template('hotels.html', properties=response)

Codul prezentat definește o rută către URL-ul "/hotels" utilizând decoratoarea @app.route('/hotels', methods=['GET', 'POST']). Acesta indică că funcția de mai jos va fi apelată atunci când se accesează acest URL și permite atât metodele GET, cât și POST.

Funcția hotels() este definită pentru ruta "/hotels" și are rolul de a trata cererile primite pentru căutarea hotelurilor. În cod, se realizează următoarele acțiuni:

1. Se specifică URL-ul către care se va efectua cererea API: "https://hotels4.p.rapidapi.com/properties/v2/list".
2. Se obțin parametrii din cererea HTTP trimiși de client folosind request.args.get(). Parametrii includ data de check-in (check-in), data de check-out (check-out), numărul de călători (num-travellers) și destinația (destination).
3. Se construiesc dicționare de date necesare pentru cererea către API, cum ar fi coordonatele destinației, data de check-in și data de check-out. Acestea sunt extrase din parametrii obținuți anterior.
4. Se verifică dacă există o valoare de sortare specificată în cerere. Dacă nu există, se setează valoarea implicită ca "RECOMMENDED".
5. Se definesc antetele cererii HTTP care includ tipul conținutului și cheile API necesare pentru a accesa API-ul Hotels. De asemenea, se definește payload-ul cererii care conține informațiile necesare pentru căutarea hotelurilor.
6. Se efectuează o cerere POST către URL-ul specificat, folosind payload-ul și antetele definite anterior. Răspunsul primit este convertit în format JSON.
7. Se returnează șablonul HTML "hotels.html", împreună cu răspunsul primit din cererea API, pentru a afișa rezultatele căutării hotelurilor.

Astfel, atunci când se accesează URL-ul "/hotels", funcția hotels() este apelată și se vor efectua cereri către API-ul Hotels pentru a obține rezultatele căutării hotelurilor.

##### Ruta pentru afisarea informatiilor despre hotel:

@app.route('/hotel\_link', methods=['GET', 'POST'])  
def hotel\_link():  
 url = "https://hotels4.p.rapidapi.com/properties/v2/detail"  
  
 hotel\_id = request.args.get('id')  
  
 payload = {  
 "currency": "USD",  
 "eapid": 1,  
 "locale": "en\_US",  
 "siteId": 300000001,  
 "propertyId": hotel\_id  
 }  
 print(payload)  
 headers = {  
 "content-type": "application/json",  
 "X-RapidAPI-Key": "5fbc045366msh8a95618fc9afb3ap19795ejsn138b5df7840b",  
 "X-RapidAPI-Host": "hotels4.p.rapidapi.com"  
 }  
  
 response = requests.request("POST", url, json=payload, headers=headers).json()  
 #data = json.loads(response)  
  
 return render\_template('hotel\_link.html', hotel\_link=response['data']['propertyInfo'])

Codul prezentat definește o rută către URL-ul "/hotel\_link" utilizând decoratoarea @app.route('/hotel\_link', methods=['GET', 'POST']). Acesta indică că funcția de mai jos va fi apelată atunci când se accesează acest URL și permite atât metodele GET, cât și POST.

Funcția hotel\_link() este definită pentru ruta "/hotel\_link" și are rolul de a trata cererile primite pentru obținerea detaliilor despre un anumit hotel. În cod, se realizează următoarele acțiuni:

1. Se specifică URL-ul către care se va efectua cererea API: "https://hotels4.p.rapidapi.com/properties/v2/detail".
2. Se obține ID-ul hotelului din cererea HTTP trimisă de client folosind request.args.get('id').
3. Se construiește payload-ul cererii către API, care include moneda, identificatorul site-ului, ID-ul hotelului și alte informații necesare.
4. Se definește antetul cererii HTTP care include tipul conținutului și cheile API necesare pentru a accesa API-ul Hotels.
5. Se efectuează o cerere POST către URL-ul specificat, folosind payload-ul și antetul definite anterior. Răspunsul primit este convertit în format JSON.
6. Se returnează șablonul HTML "hotel\_link.html", împreună cu informațiile despre hotelul obținute din răspunsul cererii API, pentru a afișa detaliile hotelului.

Astfel, atunci când se accesează URL-ul "/hotel\_link", funcția hotel\_link() este apelată și se vor efectua cereri către API-ul Hotels pentru a obține detaliile despre hotelul specificat prin ID-ul furnizat.

#### Implementați șabloanele HTML în directorul "templates".

##### index.html: conține formularul de căutare a hotelurilor.

<!DOCTYPE html>

<html>

<head>

<link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-select/1.13.5/css/bootstrap-select.min.css" rel="stylesheet">  
<link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.2.1/css/bootstrap.min.css" rel="stylesheet">  
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.2.1/js/bootstrap.bundle.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-select/1.13.5/js/bootstrap-select.min.js"></script>

</head>

<body>

<form class="form-inline d-flex justify-content-around" action="/hotels">  
 <div class="form-group">  
 <select class="selectpicker" data-live-search="true" name="destination">  
  
</select>  
 </div>  
<script >  
 $(***document***).ready(function() {  
 $('.selectpicker').selectpicker();  
  
 $('.bs-searchbox > input').on('input', function() {  
 var query = $(this).val();  
 var url = 'https://example.com/api/endpoint?q=' + query;  
  
 $.ajax({  
 url: 'https://hotels4.p.rapidapi.com/locations/v3/search',  
 method: 'GET',  
 headers: {  
 'x-rapidapi-host': 'hotels4.p.rapidapi.com',  
 'x-rapidapi-key': '703ad93641msh095a4d680095257p14fa96jsn3577c7c039bb'  
 },  
 data: {  
 q: query,  
 locale: 'new york',  
 langid :'1033',  
 siteid: '300000001'  
 },  
 success: function(data) {  
 // Populate the dropdown with the new options  
 var options = [];  
 $.each(data.sr, function(index, item) {  
 options.push('<option value="' + item.coordinates.lat + ' '+ item.coordinates.long+'">' + item.regionNames.displayName + '</option>');  
 });  
 $('.selectpicker').html(options.join(''));  
 $('.selectpicker').selectpicker('refresh');  
 },  
 error: function(jqXHR, textStatus, errorThrown) {  
 ***console***.error(errorThrown);  
 }  
 });  
 });  
});  
  
</script>  
 <div class="form-group">  
 <input type="date" class="btn btn-lg btn-light mx-1" placeholder="Check-in Date" name="check-in" aria-describedby="button-addon2" aria-label="check-in" min="" required>  
</div>  
<div class="form-group">  
 <input type="date" class="btn btn-lg btn-light mx-1" placeholder="Check-out Date" name="check-out" aria-describedby="button-addon2" aria-label="check-out" min="{{ min\_checkout\_date }}" required>  
</div>  
  
 <!--  
<div class="form-group">  
  
 <input type="number" class="btn btn-lg btn-light mx-1" id="num-rooms" name="num-rooms" min="1" value="1" max="8">  
</div> -->  
<div class="form-group">  
  
 <input type="number" class="btn btn-lg btn-light mx-1" id="num-travellers" name="num-travellers" min="1" value="1" max="14">  
</div>  
<div class="form-group">  
 <button class="btn btn-primary" type="submit" id="button-addon2">Search</button>  
</div>  
  
<script>  
 const ***checkInInput*** = ***document***.querySelector('input[name="check-in"]');  
 const ***checkOutInput*** = ***document***.querySelector('input[name="check-out"]');  
  
 const ***today*** = new ***Date***().toISOString().split('T')[0];  
 ***checkInInput***.min = ***today***;  
  
 ***checkInInput***.addEventListener('change', function () {  
 ***checkOutInput***.min = this.value;  
 ***checkOutInput***.value = '';  
 });  
  
 ***checkOutInput***.addEventListener('change', function () {  
 const checkInDate = new ***Date***(***checkInInput***.value);  
 const checkOutDate = new ***Date***(this.value);  
  
 if (checkOutDate <= checkInDate) {  
 const nextDay = new ***Date***(checkInDate.getTime() + 24 \* 60 \* 60 \* 1000);  
 this.value = nextDay.toISOString().split('T')[0];  
 }  
 });  
</script>  
</form>

</body>  
</html>

Codul HTML prezentat este un șablon pentru pagina "index.html" care conține un formular de căutare a hotelurilor. Să explicăm secțiunile importante ale acestui cod:

1. În secțiunea <form>, se definește un formular cu clasa "form-inline" și cu acțiunea setată la "/hotels". Acest formular va fi utilizat pentru a trimite datele către ruta "/hotels" atunci când utilizatorul apasă butonul "Search".
2. În interiorul formularului, există mai multe elemente <div> care definesc diferite componente ale formularului, cum ar fi câmpurile select, câmpurile de introducere a datelor și butonul de căutare.
3. În secțiunea <script>, se utilizează JavaScript (biblioteca jQuery) pentru a gestiona interacțiunea utilizatorului și pentru a realiza o cerere AJAX către API-ul Hotels pentru a obține destinațiile disponibile în funcție de introducerea utilizatorului.
4. După ce utilizatorul introduce o valoare în câmpul de căutare, se face o cerere către API-ul Hotels prin intermediul funcției $.ajax(). Se trimit parametrii necesari pentru căutare, inclusiv valoarea introdusă de utilizator și cheile API.
5. Răspunsul primit de la API-ul Hotels este prelucrat și se generează opțiuni pentru câmpul select utilizând funcția $.each(). Opțiunile reprezintă destinațiile disponibile și sunt adăugate în câmpul select utilizând funcția $('.selectpicker').html(options.join(''));. Apoi, se actualizează câmpul select folosind $('.selectpicker').selectpicker('refresh');.
6. În secțiunea <script> de jos, se efectuează unele manipulări JavaScript pentru a seta datele minime pentru câmpurile de introducere a datelor "check-in" și "check-out". Se verifică, de asemenea, dacă data de check-out introdusă este înainte de data de check-in și, în caz afirmativ, se setează data de check-out la următoarea zi a datei de check-in.

În ansamblu, acest cod HTML reprezintă formularul de căutare a hotelurilor în care utilizatorul poate selecta destinația, data de check-in, data de check-out și numărul de călători. După completarea formularului și apăsarea butonului "Search", datele sunt trimise către ruta "/hotels" pentru a efectua căutarea și afișa rezultatele.

##### hotels.html: afișează rezultatele căutării sub formă de listă de hoteluri.

<!DOCTYPE html>  
<html>  
<head>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" type="text/css">  
 <link rel="stylesheet" href="https://static.pingendo.com/bootstrap/bootstrap-4.3.1.css">  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
</head>  
<body>

<form class="form-inline d-flex justify-content-around">  
  
 <div class="form-group" >  
 <div class="col-md-6">  
 <label>What we are paid impacts our sort order</label></div>  
 <div class="col-md-2"><label for="filter-select">Sort by:</label></div>  
 <div class="col-md-4">  
 <select class="form-control" id="filter-select" name="sort">  
 <option value="RECOMMENDED">Recommended</option>  
 <option value="PRICE\_LOW\_TO\_HIGH">Price: low to high</option>  
 <option value="PRICE\_HIGH\_TO\_LOW">Price: high to low</option>  
 <option value="DISTANCE">Distance from city centre</option>  
 <option value="REVIEW\_RELEVANT">Guest rating + our choices</option>  
 <option value="PROPERTY\_CLASS">Property class</option>  
 </select>  
</div>  
  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script>  
$(***document***).ready(function() {  
 // Get the current "sort" parameter from the URL  
 var currentSort = new URLSearchParams(***window***.location.search).get("sort");  
  
 // Set the selected option in the dropdown based on the current "sort" value  
 if (currentSort) {  
 $('#filter-select').val(currentSort);  
 }  
  
 // Handle filter selection  
 $('#filter-select').change(function() {  
 var selectedFilter = $(this).val();  
  
 // Get the current URL and add/update the "sort" parameter  
 var url = new URL(***window***.location.href);  
 url.searchParams.set("sort", selectedFilter);  
  
 // Redirect to the updated URL  
 ***window***.location.href = url.href;  
 });  
});  
</script>  
 </div></form>  
 </div></div>  
 <div id="hotels-list">  
 {% for property in properties['data']['propertySearch']['properties']|batch(3) %}  
  
 <div class="row">  
 {% for row in property %}  
 <div class="col-lg-4 p-3">  
 <div class="card bg-dark text-center text-light">  
 <div class="card-body p-4">  
 <img class="img-fluid d-block" src="{{row['propertyImage']['image']['url']}}" style="width: 100%; height: 250px;">  
 <div class="card-custom">  
 <h3 style="font-size: 1.20rem; text-align: center;"><a class="mx-auto" style="color: #007bff;" href="/hotel\_link?id={{ row['id'] }}">{{ row['name'] }}</a></h3>  
 <br>  
 </div>  
 {% for item in row['price']['options'] %}  
 <h2 class="my-3"> <b>{{ item['formattedDisplayPrice'] }}</b> </h2>  
 {% endfor %}  
 <p >Rating: {{ row['reviews']['score'] }}/10 ({{ row['reviews']['total'] }} reviews)</p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 {% endfor %}  
 </div>

</body>  
</html>

Codul HTML prezentat este un șablon pentru pagina "hotels.html" care afișează rezultatele căutării hotelurilor într-o listă de carduri.

Să explicăm secțiunile importante ale acestui cod:

1. În secțiunea <form>, se definește un formular cu clasa "form-inline" pentru a permite utilizatorului să selecteze opțiunea de sortare a rezultatelor. Aceasta este realizată utilizând un câmp select cu opțiunile de sortare disponibile.
2. În secțiunea <script>, se utilizează JavaScript (biblioteca jQuery) pentru a gestiona selecția utilizatorului și pentru a actualiza parametrul "sort" din URL-ul curent. Când utilizatorul selectează o opțiune de sortare, se modifică URL-ul curent prin adăugarea/actualizarea parametrului "sort" și se realizează o redirecționare către noul URL.
3. În secțiunea <div id="hotels-list">, se utilizează o structură de bucle pentru a itera prin fiecare hotel din lista de proprietăți (properties). Aceasta permite afișarea fiecărui hotel într-un card.
4. În interiorul cardului, se afișează imaginea hotelului, numele hotelului și prețul hotelului. Aceste informații sunt preluate din obiectul row care reprezintă un hotel în lista de proprietăți.
5. În secțiunea <a> (linkul către detalii), se utilizează o cale relativă "/hotel\_link" pentru a redirecționa utilizatorul către ruta "hotel\_link" pentru a afișa detalii suplimentare despre hotelul respectiv. ID-ul hotelului este transmis prin intermediul parametrului "id" în URL.

În ansamblu, acest cod HTML reprezintă o pagină care afișează rezultatele căutării hotelurilor sub formă de carduri. Utilizatorul poate selecta o opțiune de sortare dintr-un câmp select și rezultatele vor fi actualizate în funcție de opțiunea selectată. Fiecare card afișează informații despre hotel, inclusiv imaginea, numele, prețul și ratingul. Utilizatorul poate face clic pe un hotel pentru a accesa mai multe detalii despre acel hotel.

##### hotel\_link.html: afișează detaliile unui hotel specific.

<!DOCTYPE html>  
<html>  
<head>  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" type="text/css">  
 <link rel="stylesheet" href="https://static.pingendo.com/bootstrap/bootstrap-4.3.1.css">  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css" />  
</head>  
<body>

<div class="col-6">  
 <h3 class="mb-0"> {{ hotel\_link['summary']['name'] }} </h3>  
 <span>  
 {{ hotel\_link['summary']['overview']['propertyRating']['accessibility'] }}  
 </span>  
 <br>  
 <span>  
 {% for i in range(hotel\_link['summary']['overview']['propertyRating']['rating']|int) %}  
 <i class="fas fa-star"></i>  
 {% endfor %}  
 {% if hotel\_link['summary']['overview']['propertyRating']['rating'] % 1 >= 0.5 %}  
 <i class="fas fa-star-half-alt"></i>  
{% endif %}  
 </span>  
 <p class="my-3">{{ hotel\_link['summary']['tagline'] }}</p>  
 <p class="my-3">Rating: {{ hotel\_link['reviewInfo']['summary']['overallScoreWithDescriptionA11y']['value'] }}</p>  
  
 </div>

<div class="col-4 text-right">  
  
 <iframe src='https://maps.google.com/maps?q={{ hotel\_link['summary']['map']['markers'][0]['mapMarker']['latLong']['latitude'] }},%20{{ hotel\_link['summary']['map']['markers'][0]['mapMarker']['latLong']['longitude'] }}&amp;z=12&amp;t=k&amp;output=embed' scrolling="no" frameborder="0" height="230" width="150%" class="rounded mx-auto" align-self-center></iframe>  
  
</div>

<div class="text-center col-md-12">  
 <h1>{{ hotel\_link['summary']['name'] }} Gallery</h1>  
</div>

<div id="carouselExampleIndicators" class="carousel slide carousel-fade" data-ride="carousel">  
  
 <div class="carousel-inner w-75 mx-auto">  
 {% for image in hotel\_link['propertyGallery']['images'] %}  
 {% if image==hotel\_link['propertyGallery']['images'][0] %}  
 <div class="carousel-item active"> <img class="d-block img-fluid w-100" src="{{ image['image']['url']}}" style="height: 500px; width: 100%; object-fit: cover;">  
  
 {% elif image!=hotel\_link['propertyGallery']['images'][0] %}  
 <div class="carousel-item"> <img class="d-block img-fluid w-100" src="{{ image['image']['url']}}" style="height: 500px; width: 100%; object-fit: cover;">  
 {% endif %}  
  
 </div>  
{% endfor %}  
  
 </div>  
 <a style="background-color: black;" class="carousel-control-prev " href="#carouselExampleIndicators" role="button" data-slide="prev">  
 <span class="carousel-control-prev-icon w-100 mx-auto" aria-hidden="true"></span>  
 <span class="sr-only">Previous</span>  
 </a>  
 <a style="background-color: black;" class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">  
 <span class="carousel-control-next-icon w-100 mx-auto" aria-hidden="true"></span>  
 <span class="sr-only">Next</span>  
 </a>  
 </div>  
 </div>

</body>  
</html>

Acest cod HTML reprezintă un șablon pentru afișarea detaliilor despre un hotel. Să îl explicăm:

1. Codul include biblioteci externe CSS și de fonturi, care sunt importate utilizând tag-urile <link> în secțiunea <head>.
2. Secțiunea <body> începe cu un element de container <div class="col-6"> care conține informații despre hotel, cum ar fi numele acestuia, accesibilitatea, ratingul, sloganul și scorul general.
3. Mai jos, există un alt <div class="col-4 text-right"> care afișează o fereastră iframe Google Maps cu locația hotelului, bazată pe coordonatele latitudine și longitudine.
4. Urmează un <div class="text-center col-md-12"> care conține un titlu pentru galeria de imagini a hotelului.
5. În interiorul acestui <div>, există un element de tip carusel <div id="carouselExampleIndicators" class="carousel slide carousel-fade" data-ride="carousel"> care afișează o prezentare de diapozitive cu imagini.
6. Imaginile sunt iterate folosind o buclă {% for image in hotel\_link['propertyGallery']['images'] %}. Pentru prima imagine, se creează un div cu clasa carousel-item active, iar pentru imaginile ulterioare se generează elemente div cu clasa carousel-item.
7. În fiecare div.carousel-item, un tag <img> afișează imaginea hotelului utilizând atributul src cu URL-ul obținut din {{ image['image']['url'] }}.
8. La sfârșitul codului, există două controale de navigare pentru carusel, permițând utilizatorului să navigheze la imaginea anterioară și următoare.

În ansamblu, acest șablon HTML combină informațiile despre hotel, integrarea Google Maps și o galerie de imagini pentru a oferi o afișare vizuală și informativă a detaliilor hotelului.

#### Stilizarea elementelor web folosind CSS în directorul "static".

.btn-light {  
 color: #999;  
 background-color: #f8f9fa;  
 border-color: #f8f9fa;  
 font-size: 1rem;  
 padding: 0.375rem 0.75rem;  
 line-height: 1.5;  
 border-radius: 0.25rem;  
 transition: color .15s ease-in-out,background-color .15s ease-in-out,border-color .15s ease-in-out,box-shadow .15s ease-in-out;  
}  
.btn-primary:hover {  
 color: #999;  
}  
.bootstrap-select>.dropdown-toggle {  
 width: 200px; /\* change this to the desired width \*/  
  
}  
.row\_custom{  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
  
}  
  
.text-primary {  
 color: #007bff **!important**;  
}  
.card-custom {  
 width: 100%;  
 height: 30px;  
  
 }  
  
.bg-dark {  
 background-color: #151515 **!important**;  
}

Codul CSS furnizat definește diferite stiluri pentru elementele dintr-o pagină web. Voi explica fiecare parte a codului în continuare:

1. .btn-light: Această clasă definește stilurile pentru un buton cu clasa .btn-light. Acesta are culoarea textului #999, culoarea de fundal #f8f9fa, culoarea chenarului #f8f9fa, dimensiunea fontului 1rem, un anumit padding, linia de bază 1.5, un border-radius 0.25rem și o tranziție animată pentru schimbarea culorilor.
2. .btn-primary:hover: Această clasă definește stilurile pentru un buton cu clasa .btn-primary atunci când se face hover (când cursorul mouse-ului este deasupra butonului). Acesta are culoarea textului #999.
3. .bootstrap-select>.dropdown-toggle: Această clasă definește stilurile pentru un element dropdown din biblioteca Bootstrap Select. Setează lățimea elementului la 200px. Este important să menționez că acest cod poate fi specific pentru un anumit plugin sau bibliotecă și nu este direct legat de CSS-ul de bază al Bootstrap.
4. .row\_custom: Această clasă definește stilurile pentru un container de tip "row" cu clasa .row\_custom. Setează afișarea flexibilă, direcția de coloană, alinierea centrală și centrată a elementelor din interior.
5. .text-primary: Această clasă definește stilurile pentru text cu clasa .text-primary. Setează culoarea textului la #007bff !important. Adăugarea !important asigură că această valoare de culoare va primi prioritate în fața altor stiluri definite pentru același element.
6. .card-custom: Această clasă definește stilurile pentru un element de tip card cu clasa .card-custom. Setează o lățime de 100% și o înălțime de 30px.
7. .bg-dark: Această clasă definește stilurile pentru un element cu clasa .bg-dark. Setează culoarea de fundal la #151515 !important. Similar cu clasa .text-primary, adăugarea !important asigură prioritatea stilului pentru culoarea de fundal.

Acestea sunt doar câteva exemple de stiluri definite în codul CSS furnizat. Aceste clase pot fi utilizate în fișierele HTML pentru a aplica stilurile definite la elementele specifice folosind atributul class.

#### Rulați aplicația:

Adăugați următoarele linii la sfârșitul fișierului app.py:

if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

Executați fișierul app.py pentru a porni serverul de dezvoltare Flask.

Accesați aplicația în browser la adresa http://localhost:5000 pentru a testa funcționalitățile.

# Prezentarea interfeței utilizatorului

O imagine care conține text, captură de ecran, nor, cer

Descriere generată automatO imagine care conține text, captură de ecran, Font, alb

Descriere generată automat

Figure 4 Pagina index.html

O imagine care conține text, captură de ecran, Software multimedia, software

Descriere generată automat

Figure 5 Cautarea hotelurilor in functie de oras

O imagine care conține text, captură de ecran, Software multimedia, software

Descriere generată automat

Figure 6 Pagina hotels.html

O imagine care conține text, captură de ecran

Descriere generată automat

Figure 7 Opțiuni de sortare

O imagine care conține text, software, captură de ecran, Software multimedia

Descriere generată automat

O imagine care conține bazin de înot, captură de ecran, piscină, apă

Descriere generată automat

Figure 8 Pagina hotel\_link.html

# Concluzie

În concluzie, procesul de creare a aplicației web a fost unul complex, dar cu ajutorul unor instrumente și tehnologii precum Flask, Bootstrap și API-ul Hotels, am reușit să dezvoltăm o aplicație funcțională și atractivă.

Am început prin configurarea proiectului, instalarea dependențelor și crearea structurii de bază a aplicației. Am utilizat Flask pentru a gestiona rutele și a comunica cu API-ul Hotels, pentru a obține și manipula datele necesare.

Aplicația noastră beneficiază de o interfață utilizator prietenoasă și bine stilizată datorită Bootstrap. Prin utilizarea Bootstrap, am putut beneficia de o varietate de componente predefinite, care ne-au ajutat să dezvoltăm rapid și eficient pagini web responsive și bine stilizate. Pingendo, cu interfața sa vizuală intuitivă, ne-a permis să lucrăm cu ușurință la stilizarea paginilor noastre, fără a fi necesară scrierea manuală a codului CSS sau HTML.

Am implementat funcționalități importante, cum ar fi căutarea, sortarea și afișarea rezultatelor, precum și vizualizarea imaginilor hotelurilor. Am asigurat și responsivitatea aplicației, astfel încât utilizatorii să poată accesa și utiliza aplicația de pe diferite dispozitive.

În timpul procesului de dezvoltare, am acordat o atenție deosebită testării și depanării, asigurându-ne că aplicația funcționează corect și că erorile sunt gestionate în mod adecvat.

Pe lângă funcționalitățile principale, am păstrat și flexibilitatea de a adăuga funcționalități suplimentare în viitor, pentru a satisface nevoile și cerințele utilizatorilor.

# Referințe

[1] <https://www.jetbrains.com/help/pycharm/creating-flask-project.html>

[2] <https://rapidapi.com/blog/how-to-build-an-api-in-python/>

[3] <https://flask.palletsprojects.com/en/2.3.x/>

[4] <https://en.wikipedia.org/wiki/Python_(programming_language)>

[5] <https://en.wikipedia.org/wiki/HTML>

[6] <https://en.wikipedia.org/wiki/CSS>

[7] <https://en.wikipedia.org/wiki/JavaScript>

[8] <https://rapidapi.com/blog/build-food-website/>

[9] <https://en.wikipedia.org/wiki/Representational_state_transfer>

[10] <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>

# Anexa

Codul se găsește la adresa <https://github.com/AgniaCodreanu/Proiect3>