## Lab 10: Integrate a REST API in a Python Application

### Case Study

CodeSphere Technologies, a mid-sized software solutions firm focused on transportation and logistics automation, faced growing demand for real-time, route-based data services within its applications. Their existing systems lacked API integration skills necessary to retrieve and display live routing information, limiting the development of smart logistics features in their software.

To resolve this, the company initiated a DevNet training module aimed at upskilling developers in REST API integration using Python and modern IDEs like Visual Studio Code (VS Code). This lab served as a key hands-on opportunity to practice live API calls and JSON data handling.

### Business Challenge

The organization’s developers lacked proficiency in integrating external APIs to automate route calculations, which affected their ability to offer intelligent transport features to clients. This hindered scalability and responsiveness in client-facing applications, especially those relying on dynamic navigation data.

To overcome this, the organization onboarded you, a DevNet-certified specialist, to build and test REST API-based solutions to enhance application capabilities and align with cloud-first development goals.

### Solution

In this lab, you will develop a Python-based application that communicates with the MapQuest Directions API to retrieve and process route data. The lab covers the following areas:

1. Access GraphHopper’s REST APIs
2. Build Python applications
3. Parse JSON data and display results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **// Access GraphHopper’s REST APIs**  1. Makesure your DEVASC VM is up and running.  2. Go to **https://www.graphhopper.com/** and **Sign up / log in**    3. Navigate to **API Keys** in your dashboard. Select **Create New key** under **API Keys.** Copy your key for use in the script **7d3ed570-4556-4b32-bb74-7b6ac10f96ce.**    // **Build Geocoding Application**  1. To create the project folder and python file,open the terminal and run the following commands to create and navigate to the project directory, then launch Visual Studio Code:   |  | | --- | | mkdir -p ~/labs/devnet-src/graphhopper  cd ~/labs/devnet-src/graphhopper  code . |   2. Once VS Code opens, create a new Python file named **graphhopper\_parse-json\_1.py** in the current directory.    3. Add required Imports. **Save** and run **python3 graphhopper\_parse-json\_1.py** to verify no errors are present.   |  | | --- | | import requests  import urllib.parse |     4. Now build the URL for geocoding. In place of **key** place **"YOUR\_API\_KEY".** Then save and run **python3 graphhopper\_parse-json\_1.py** in terminal**.** You should see a status code **200** and a JSON response.Look for hits[0]["point"]["lat"] and ["lng"].   |  | | --- | | geocode\_url = "https://graphhopper.com/api/1/geocode?"  loc1 = "Washington, D.C."  key = "a8ce77d7-6f63-46e7-8f22-95b172a06758"  url = geocode\_url + urllib.parse.urlencode({"q": loc1, "limit": "1", "key": key})  response = requests.get(url)  print("Status Code:", response.status\_code)  print(response.json()) |     **// Parse JSON Data and Display Results**  1.In this part, you will create a Python script to send a URL request to the GraphHopper Directions API. You will then test your API call. 2. Create another python file with name graphhopper\_parse-json\_2.py. and then use the following code.  |  | | --- | | import urllib.parse  import requests  # User Inputs  loc1 = input("Starting Location: ")  if loc1.lower() in ["quit", "q"]:  exit()  loc2 = input("Destination: ")  if loc2.lower() in ["quit", "q"]:  exit()  # API Key  key = "a8ce77d7-6f63-46e7-8f22-95b172a06758"  # Geocode URL  geo\_url = "https://graphhopper.com/api/1/geocode?"  loc1\_url = geo\_url + urllib.parse.urlencode({"q": loc1, "limit": "1", "key": key})  loc2\_url = geo\_url + urllib.parse.urlencode({"q": loc2, "limit": "1", "key": key})  # Geocode Requests  resp1 = requests.get(loc1\_url).json()  resp2 = requests.get(loc2\_url).json()  # Extract lat/lng  point1 = resp1["hits"][0]["point"]  point2 = resp2["hits"][0]["point"]  lat1, lng1 = point1["lat"], point1["lng"]  lat2, lng2 = point2["lat"], point2["lng"]  # Directions API Call  route\_url = "https://graphhopper.com/api/1/route?"  route\_url += urllib.parse.urlencode([  ("point", f"{lat1},{lng1}"),  ("point", f"{lat2},{lng2}"),  ("vehicle", "car"),  ("locale", "en"),  ("instructions", "true"),  ("key", key)  ], doseq=True)  print("URL:", route\_url)  # Request Directions  route\_resp = requests.get(route\_url)  print("Status Code:", route\_resp.status\_code)  if route\_resp.status\_code == 200:  data = route\_resp.json()  distance = data["paths"][0]["distance"] / 1000 # meters to km  time = data["paths"][0]["time"] / 60000 # ms to min  print(f"\nTrip Distance: {distance:.2f} km")  print(f"Trip Time: {time:.2f} minutes")  print("="\*40)  for instr in data["paths"][0]["instructions"]:  print(f"{instr['text']} ({instr['distance']:.2f} m)")  print("="\*40)  else:  print("Request failed. Check key or inputs.") |   3. Now, run the script to verify its working using command **python3 graphhopper\_parse-json\_2.py**    4. Enter the following when asked location:  Starting Location: Washington, D.C.  Destination: New York  It will provide you with clear directions to your location, trip distance and trip time.   |  | | --- | | URL: https://graphhopper.com/api/1/route?point=38.8950368%2C-77.0365427&point=40.7127281%2C-74.0060152&vehicle=car&locale=en&instructions=true&key=a8ce77d7-6f63-46e7-8f22-95b172a06758  Status Code: 200  Trip Distance: 363.40 km  Trip Time: 234.01 minutes  ========================================  **Continue (70.68 m)**  Turn sharp left onto South Drive (48.52 m)  Turn left onto West Executive Avenue Northwest (9.93 m)  Turn right onto State Place Northwest (123.97 m)  Turn right onto 17th Street Northwest (681.54 m)  Turn right onto K Street Northwest (1071.54 m)  Turn left onto 11th Street Northwest (135.75 m)  Turn right onto L Street Northwest (55.51 m)  Turn slight right onto Massachusetts Avenue Northwest (5230.60 m)  Continue onto New York Avenue Northeast (2817.06 m)  Keep left onto Baltimore-Washington Parkway and drive toward Baltimore (44332.47 m)  Keep right and take I 895 North toward Baltimore Harbor Tunnel Thruway (15527.83 m)  Keep left onto Express Toll Lanes (13800.58 m)  Keep right (66115.16 m)  Continue onto Delaware Turnpike and drive toward EZ-Pass Only (15944.84 m)  Keep right onto Delaware Turnpike and take I 295, NJTP toward Delaware Memorial Bridge, New York, New Jersey (10515.58 m)  Keep left onto New Jersey Turnpike and drive toward NJ Turnpike, Atlantic City (3981.11 m)  Keep left (75720.78 m)  Continue onto New Jersey Turnpike (88271.33 m)  Keep right and take I 78, US 1, US 9 toward Newark Airport, Holland Tunnel (662.51 m)  Keep right and take I 78 East toward Bayonne, Jersey City, Holland Tunnel (13282.70 m)  Keep left onto Holland Tunnel and drive toward Holland Tunnel, Manhattan (3554.34 m)  Turn slight right onto Canal Street (524.23 m)  Turn right onto Lafayette Street (854.90 m)  Turn right (62.27 m)  Arrive at destination (0.00 m)  ========================================  devasc@labvm:~/labs/devnet-src/graphhopper$ | |