

---

## Evaluación N°2 “DevNet y APIS”

---

NOMBRE: Juan Huenuhueque, Joaquín Rivera

CARRERA: Ingeniería en Telecomunicaciones, Conectividad y Redes

ASIGNATURA: Redes Avanzadas I

PROFESOR: Mario Villanueva Alveal

FECHA: 08/05/2024

## Índice

<b>Evaluación N°2 “DevNet y APIS”</b>	<b>1</b>
1    Introducción.....	3
2    Creación y prueba del entorno de desarrollo DevNet.....	4
2.1    Inicio del Repositorio de Git:.....	4
2.2    Creación de Directorio de Trabajo y Repositorio Local: .....	4
2.3    Creación de Archivo de Texto con Nombre y Apellido: .....	5
2.4    Confirmación del Archivo en Git: .....	5
2.5    Modificaciones al Archivo y Confirmación en Git: .....	6
2.6    Creación de un Nuevo Branch: .....	6
2.7    Nuevo Cambio en el Archivo y Confirmación en el Branch: .....	7
2.8    Fusión del Branch con la Rama Master:.....	7
2.9    Envío del Archivo a GitHub: .....	8
3    Cree una aplicación que entregue la geolocalización de las ubicaciones de un origen y destino dada por el usuario. Muestre la latitud, Longitud, ciudad, región y país. Evite mensajes de error. ....	9
3.1    Aplicación geocalización de ubicaciones .....	9
3.2    Aplicación que calcule la distancia entre distintas ciudades de cualquier país de Latinoamérica en kilómetros utilizando la API GraphHopper .....	10
4    Conclusión .....	15

## 1 Introducción

El presente trabajo se enfoca en la creación y prueba del entorno de desarrollo DevNet, utilizando herramientas como Git para el control de versiones y la colaboración entre desarrolladores. A través de una serie de pasos prácticos, se establecerá un flujo de trabajo que permita gestionar eficientemente el código fuente y registrar los cambios realizados durante el desarrollo del proyecto.

Desde la configuración inicial del repositorio hasta la fusión de ramas y la entrega del trabajo final en GitHub, cada paso será detallado y ejecutado con precisión. Asimismo, se hará énfasis en la importancia de la comunicación y la transparencia en el proceso de desarrollo, aspectos esenciales para mantener un equipo cohesionado y orientado hacia el logro de los objetivos propuestos.

En este contexto, se presenta la siguiente tarea: el desarrollo de una aplicación en Python que aborde dos aspectos fundamentales en la gestión de ubicaciones geográficas y rutas de viaje. En primer lugar, se requerirá una funcionalidad que permita al usuario obtener la geolocalización detallada de un origen y destino especificados. Esta característica, además de proporcionar información sobre la latitud, longitud, ciudad, región y país, se diseñará para ser intuitiva y libre de errores, garantizando una experiencia fluida para el usuario.

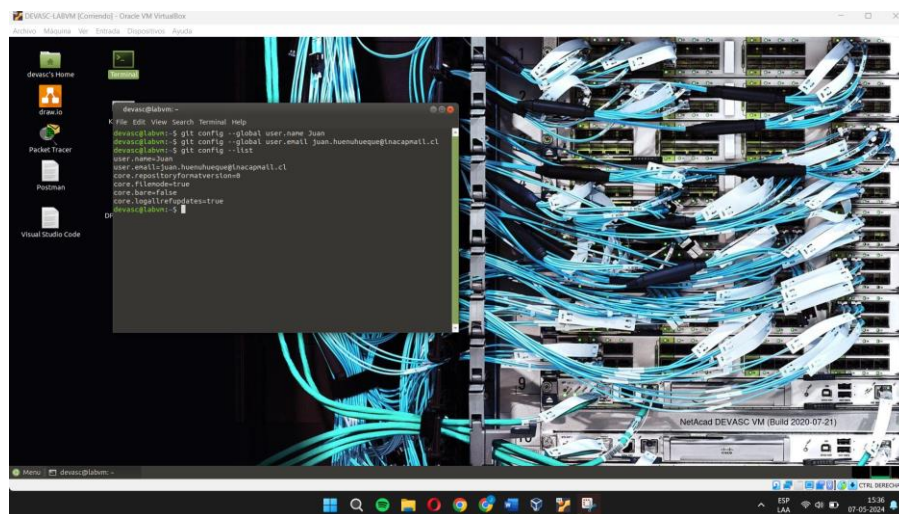
Por otro lado, la segunda parte de la aplicación se centrará en el cálculo de la distancia entre distintas ciudades de Latinoamérica utilizando la API de GraphHopper. Esta funcionalidad permitirá al usuario ingresar las ciudades de origen y destino, seleccionar el medio de transporte deseado y obtener información precisa sobre la distancia en kilómetros y millas, así como la duración estimada del viaje en horas, minutos y segundos.

En resumen, el desarrollo de esta aplicación no solo busca satisfacer necesidades prácticas relacionadas con la geolocalización y el transporte, sino también promover una interacción amigable y efectiva con el entorno geográfico, aprovechando las capacidades de Python y las API disponibles para ofrecer soluciones innovadoras y útiles.

## 2 Creación y prueba del entorno de desarrollo DevNet

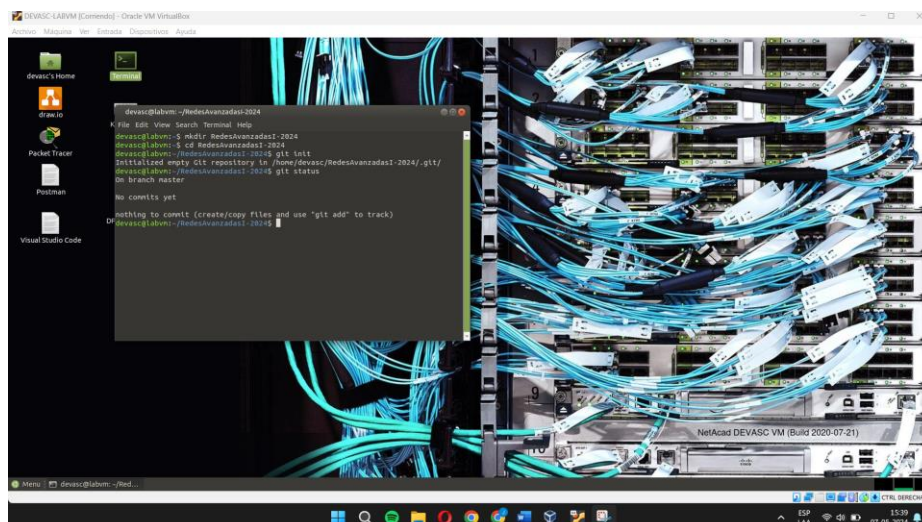
### 2.1 Inicio del Repositorio de Git:

En el primer paso de nuestro proceso de configuración del entorno de desarrollo DevNet, procedimos a iniciar un repositorio de Git utilizando el comando 'git init'. Posteriormente, configuramos nuestro nombre de usuario y correo electrónico asociado al repositorio utilizando los comandos 'git config --global user.name' y 'git config --global user.email'. Confirmamos la correcta configuración utilizando 'git config --list'.



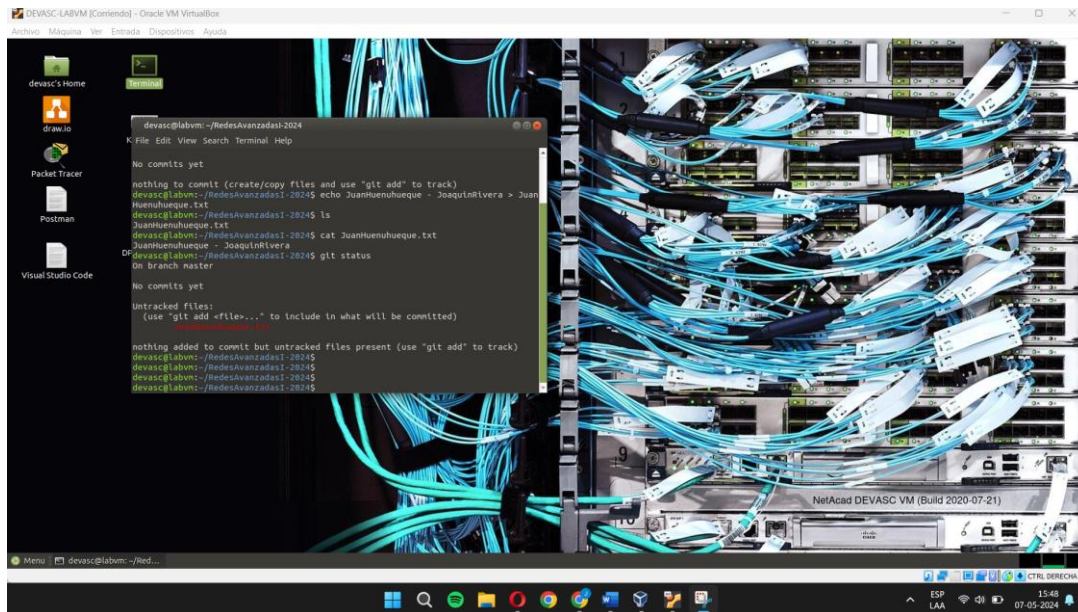
### 2.2 Creación de Directorio de Trabajo y Repositorio Local:

Continuando con el proceso, creamos un directorio de trabajo denominado 'RedesAvanzadas1-2024' y lo inicializamos como nuestro repositorio local mediante el comando 'git init'. Verificamos el estado actual del repositorio utilizando 'git status' para asegurarnos de que todo esté en orden.



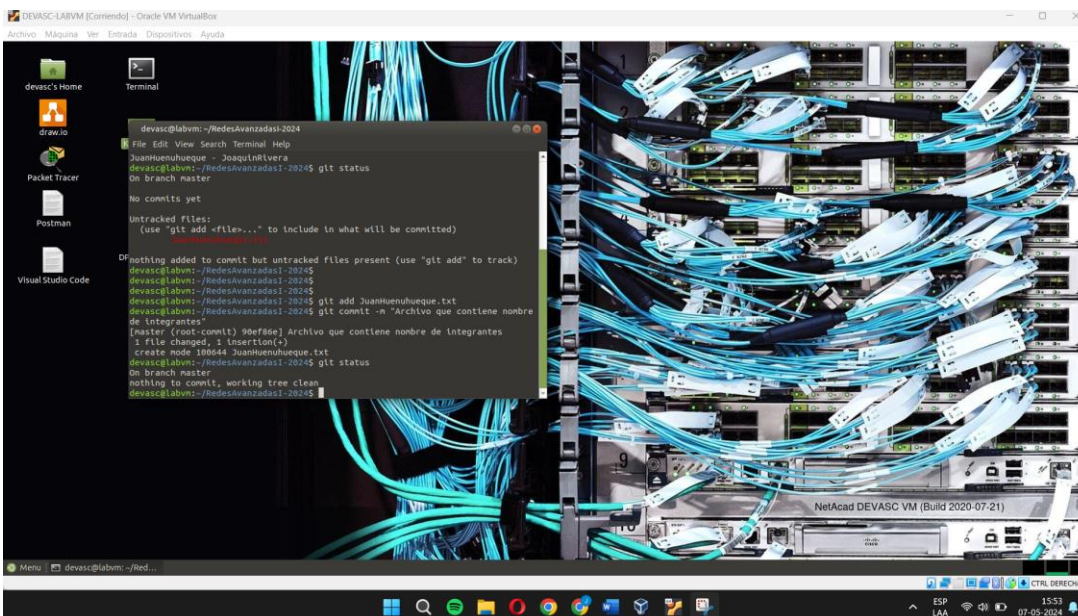
## 2.3 Creación de Archivo de Texto con Nombre y Apellido:

Para mantener un registro claro de nuestras actividades, creamos un archivo de texto llamado 'nombre\_apellido.txt', donde incluimos nuestro nombre y apellido. Este archivo servirá como parte fundamental de nuestra estructura de trabajo.



## 2.4 Confirmación del Archivo en Git:

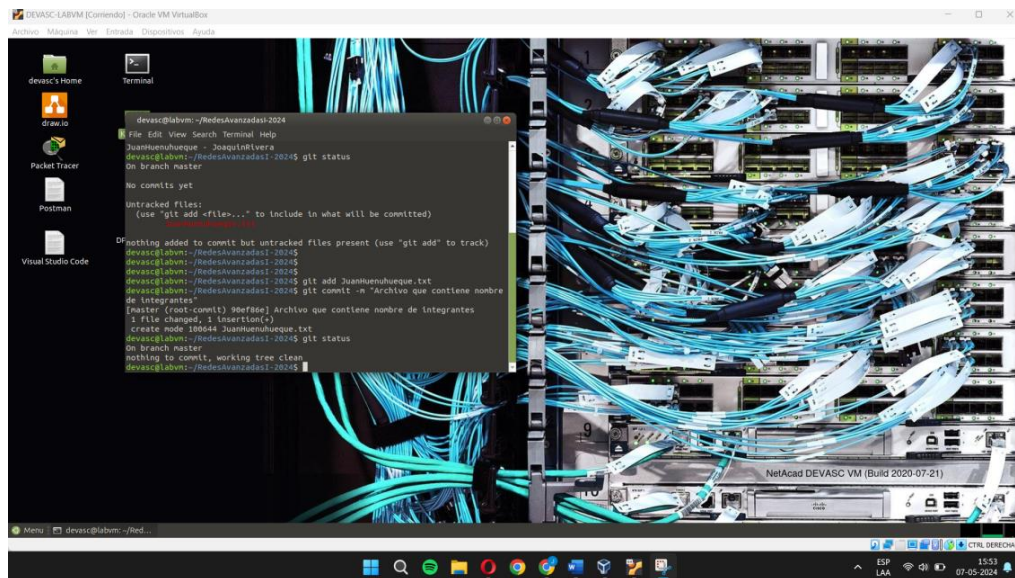
Una vez creado el archivo 'nombre\_apellido.txt', procedimos a confirmarlo en Git utilizando el comando 'git add' seguido de 'git commit'. A través de este proceso, garantizamos que Git rastree los cambios realizados en el archivo y registre adecuadamente nuestras acciones.





## 2.5 Modificaciones al Archivo y Confirmación en Git:

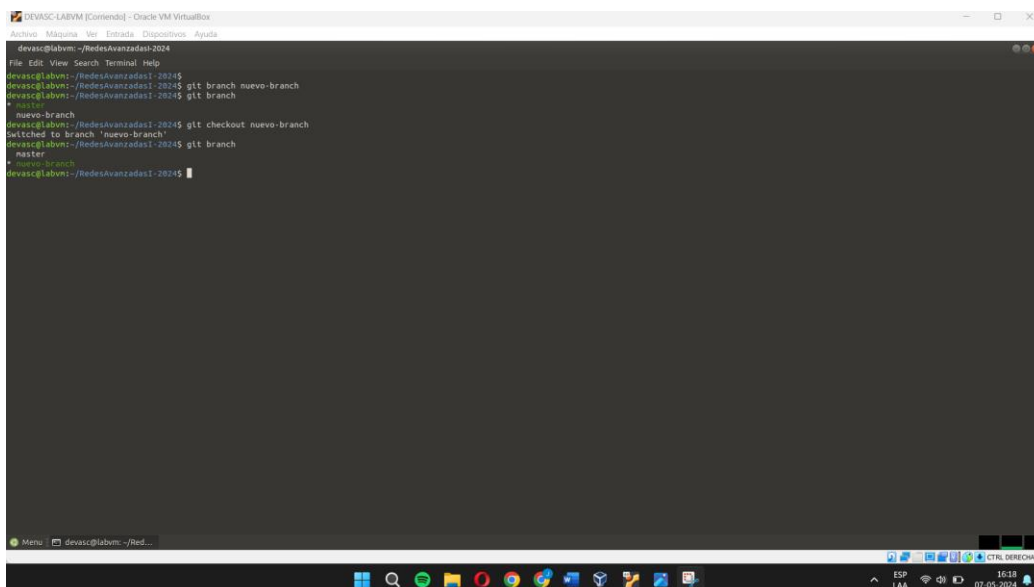
Para demostrar el flujo de trabajo en Git, realizamos modificaciones al archivo 'nombre\_apellido.txt', añadiendo información adicional. Posteriormente, confirmamos estos cambios utilizando los comandos 'git add' y 'git commit', manteniendo un historial claro de las modificaciones efectuadas.



```
devasc@labvm:~/RedesAvanzadas1-2024$ git status
On branch master
No commits yet
Untracked files:
  (use "git add -f" to include in what will be committed)
  JuanHuenahueque.txt
nothing added to commit but untracked files present (use "git add" to track)
devasc@labvm:~/RedesAvanzadas1-2024$ git add JuanHuenahueque.txt
devasc@labvm:~/RedesAvanzadas1-2024$ git commit -m "Archivo que contiene nombre de integrantes"
[master (root-commit) 98ef86e] Archivo que contiene nombre de integrantes
 1 file changed, 1 insertion(+)
 create mode 100644 JuanHuenahueque.txt
devasc@labvm:~/RedesAvanzadas1-2024$ git status
On branch master
nothing to commit, working tree clean
devasc@labvm:~/RedesAvanzadas1-2024$
```

## 2.6 Creación de un Nuevo Branch:

Dando paso a una práctica fundamental en el desarrollo colaborativo, creamos un nuevo branch denominado 'nuevo-branch'. Verificamos la creación exitosa de este branch y cambiamos a él para continuar nuestras operaciones de desarrollo.

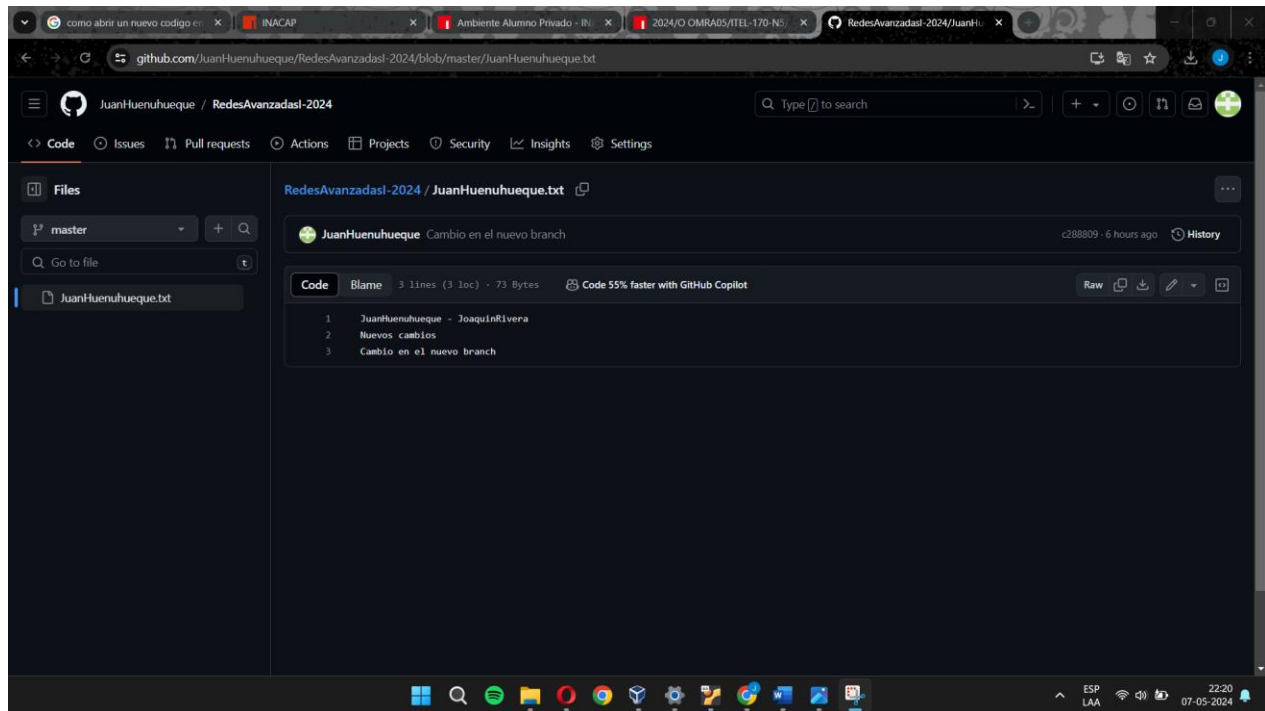


```
devasc@labvm:~/RedesAvanzadas1-2024$ git branch nuevo-branch
devasc@labvm:~/RedesAvanzadas1-2024$ git checkout nuevo-branch
Switched to branch 'nuevo-branch'
devasc@labvm:~/RedesAvanzadas1-2024$ git branch
* nuevo-branch
  master
devasc@labvm:~/RedesAvanzadas1-2024$
```



## 2.9 Envío del Archivo a GitHub:

Finalmente, para mantener una copia de nuestro trabajo en un repositorio remoto, configuramos un repositorio en GitHub. Verificamos en GitHub que los cambios se hayan reflejado correctamente en el repositorio remoto.



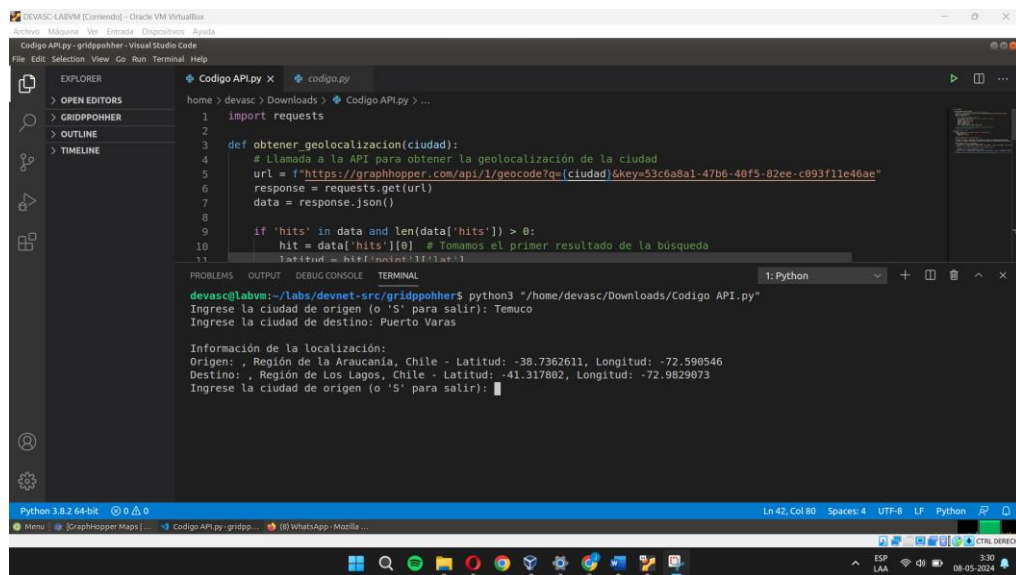
Adjunto link del repositorio: <https://github.com/JuanHuenhueque/RedesAvanzadasI-2024.git>



- 3 Cree una aplicación que entregue la geolocalización de las ubicaciones de un origen y destino dada por el usuario. Muestre la latitud, Longitud, ciudad, región y país. Evite mensajes de error.

### 3.1 Aplicación geocalización de ubicaciones

Para este punto procederemos a mostrar el código correspondiente a la geolocalización de las ubicaciones escogidas por el usuario, donde se pueden apreciar la latitud, longitud, ciudad, región y país.

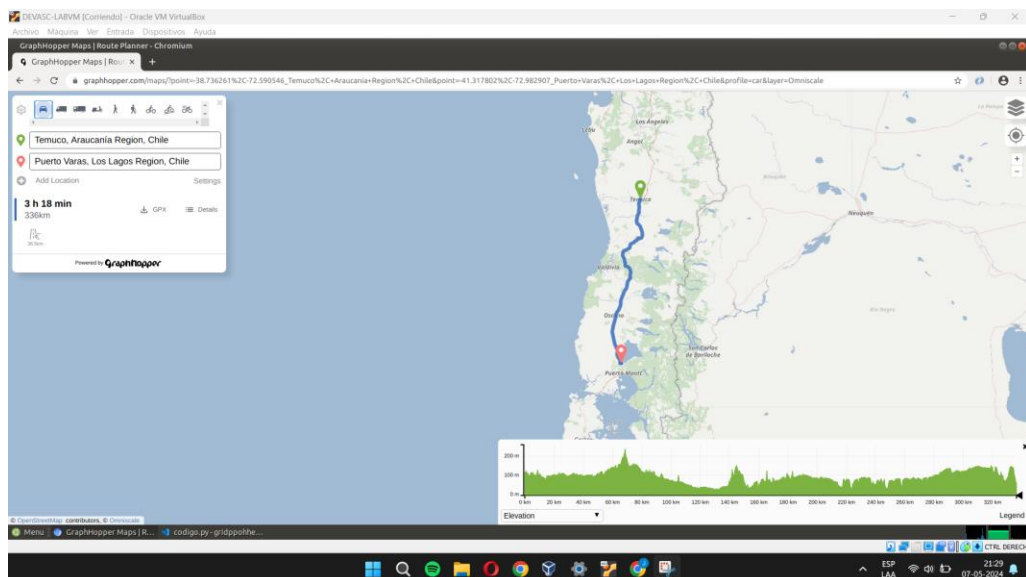


```

1 import requests
2
3 def obtener_geolocalizacion(ciudad):
4     # llamada a la API para obtener la geolocalización de la ciudad
5     url = f"https://graphhopper.com/api/1/geocode?q={ciudad}&key=53c6a8a1-47b6-40f5-82ee-c093f11c46ae"
6     response = requests.get(url)
7     data = response.json()
8
9     if 'hits' in data and len(data['hits']) > 0:
10        hit = data['hits'][0] # Tomamos el primer resultado de la búsqueda
11        latitud = hit['location']['lat']
12
13 devasc@labvm: ~/labs/devnet-src/gridppohher$ python3 "/home/devasc/Downloads/Codigo API.py"
Ingrese la ciudad de origen (o 'S' para salir): Temuco
Ingrese la ciudad de destino: Puerto Varas

Información de la localización:
Origen: , Región de la Araucanía, Chile - Latitud: -38.7362611, Longitud: -72.598546
Destino: , Región de Los Lagos, Chile - Latitud: -41.317882, Longitud: -72.9829073
Ingrese la ciudad de origen (o 'S' para salir):
  
```

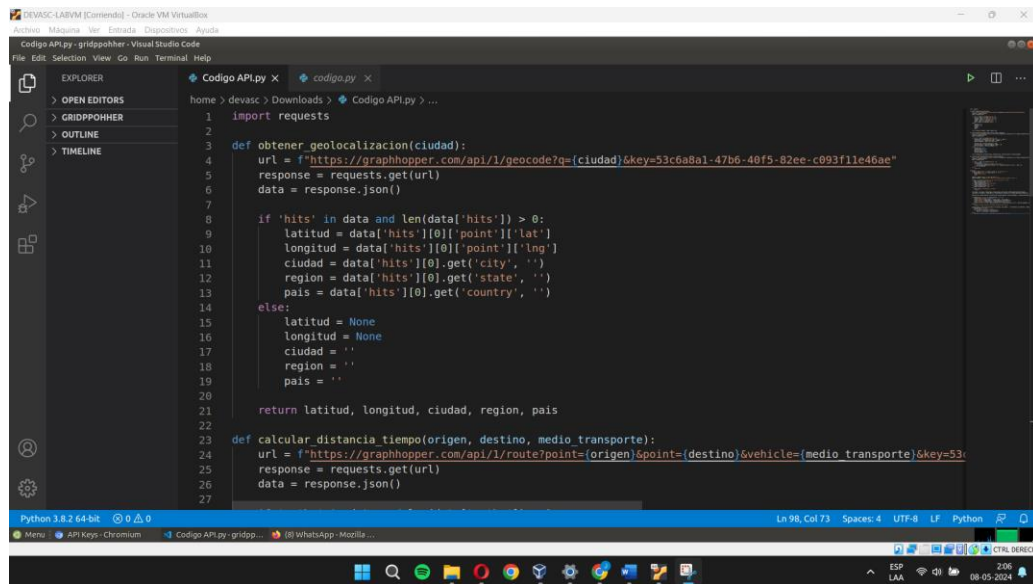
Para esta imagen se muestra la parte grafica dentro del GraphHopper, donde se observa la loc1 que es Temuco y la loc2 que corresponde Puerto Varas.



### 3.2 Aplicación que calcule la distancia entre distintas ciudades de cualquier país de Latinoamérica en kilómetros utilizando la API GraphHopper

La aplicación desarrollada ofrece una solución integral para calcular la distancia entre ciudades latinoamericanas, proporcionando información detallada sobre el tiempo de viaje, instrucciones de la ruta ingresada y además da a conocer la opción de transporte elegida por el usuario (auto, bicicleta, a pie), todo esto formulando interacción con el mismo.

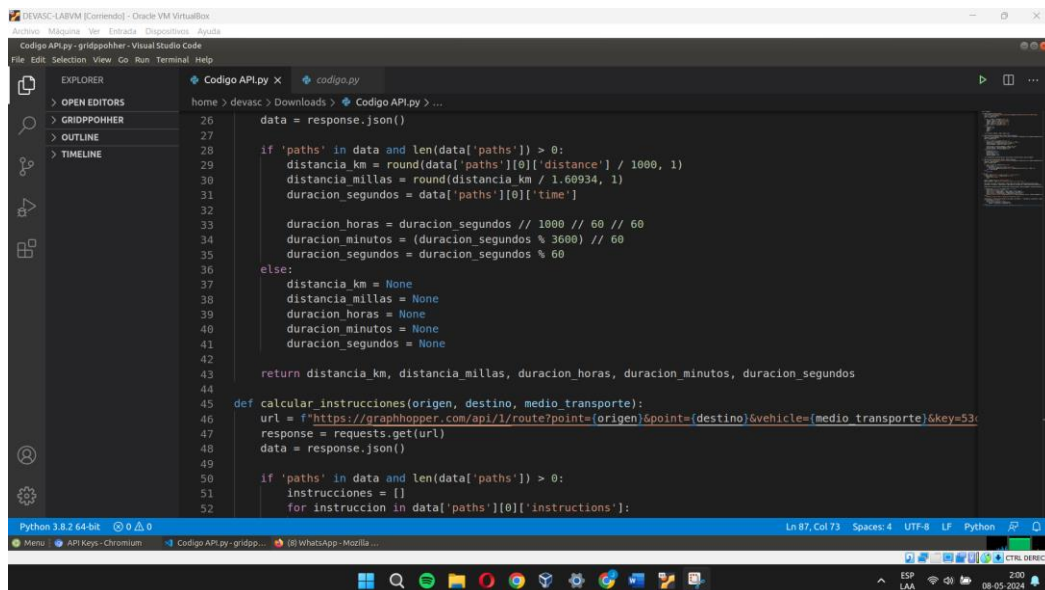
A continuación, se mostrarán capturas de pantalla del código implementado:



```

1 import requests
2
3 def obtener_geolocalizacion(ciudad):
4     url = f"https://graphhopper.com/api/1/geocode?q={ciudad}&key=53c6a8a1-47b6-40f5-82ee-c093f11e46ae"
5     response = requests.get(url)
6     data = response.json()
7
8     if 'hits' in data and len(data['hits']) > 0:
9         latitud = data['hits'][0]['point']['lat']
10        longitud = data['hits'][0]['point']['lng']
11        ciudad = data['hits'][0].get('city', '')
12        region = data['hits'][0].get('state', '')
13        pais = data['hits'][0].get('country', '')
14    else:
15        latitud = None
16        longitud = None
17        ciudad = ''
18        region = ''
19        pais = ''
20
21    return latitud, longitud, ciudad, region, pais
22
23 def calcular_distancia_tiempo(origen, destino, medio_transporte):
24     url = f"https://graphhopper.com/api/1/route?point={origen}&point={destino}&vehicle={medio_transporte}&key=53c6a8a1-47b6-40f5-82ee-c093f11e46ae"
25     response = requests.get(url)
26     data = response.json()
27

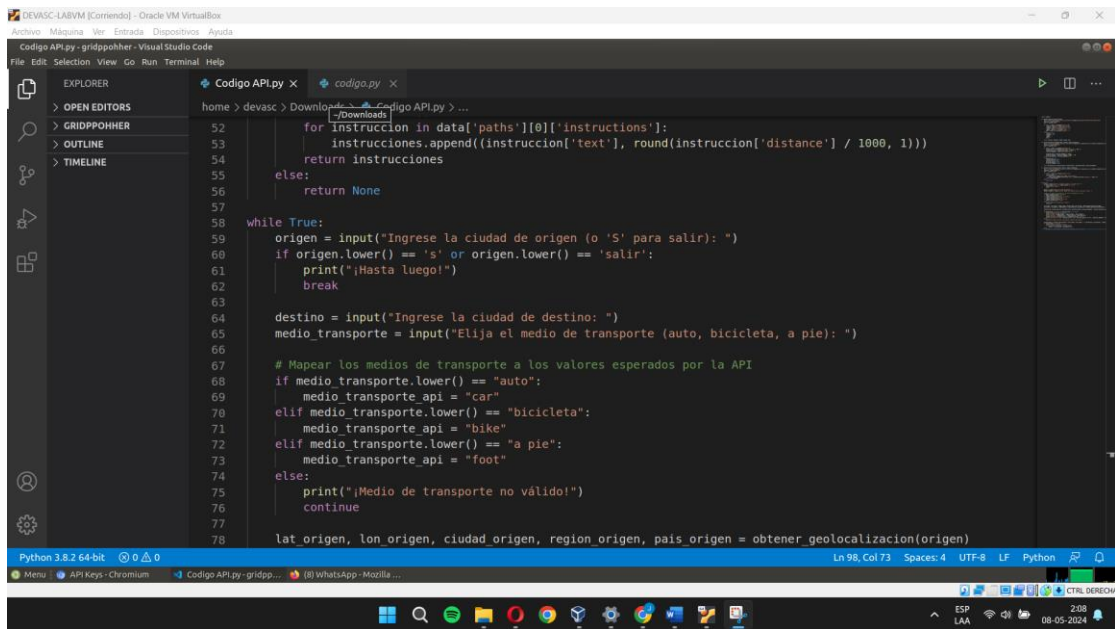
```



```

26 data = response.json()
27
28 if 'paths' in data and len(data['paths']) > 0:
29     distancia_km = round(data['paths'][0]['distance'] / 1000, 1)
30     distancia_millas = round(distancia_km / 1.60934, 1)
31     duracion_segundos = data['paths'][0]['time']
32
33     duracion_horas = duracion_segundos // 1000 // 60 // 60
34     duracion_minutos = (duracion_segundos % 3600) // 60
35     duracion_segundos = duracion_segundos % 60
36 else:
37     distancia_km = None
38     distancia_millas = None
39     duracion_horas = None
40     duracion_minutos = None
41     duracion_segundos = None
42
43 return distancia_km, distancia_millas, duracion_horas, duracion_minutos, duracion_segundos
44
45 def calcular_instrucciones(origen, destino, medio_transporte):
46     url = f"https://graphhopper.com/api/1/route?point={origen}&point={destino}&vehicle={medio_transporte}&key=53c6a8a1-47b6-40f5-82ee-c093f11e46ae"
47     response = requests.get(url)
48     data = response.json()
49
50 if 'paths' in data and len(data['paths']) > 0:
51     instrucciones = []
52     for instruccion in data['paths'][0]['instructions']:
53

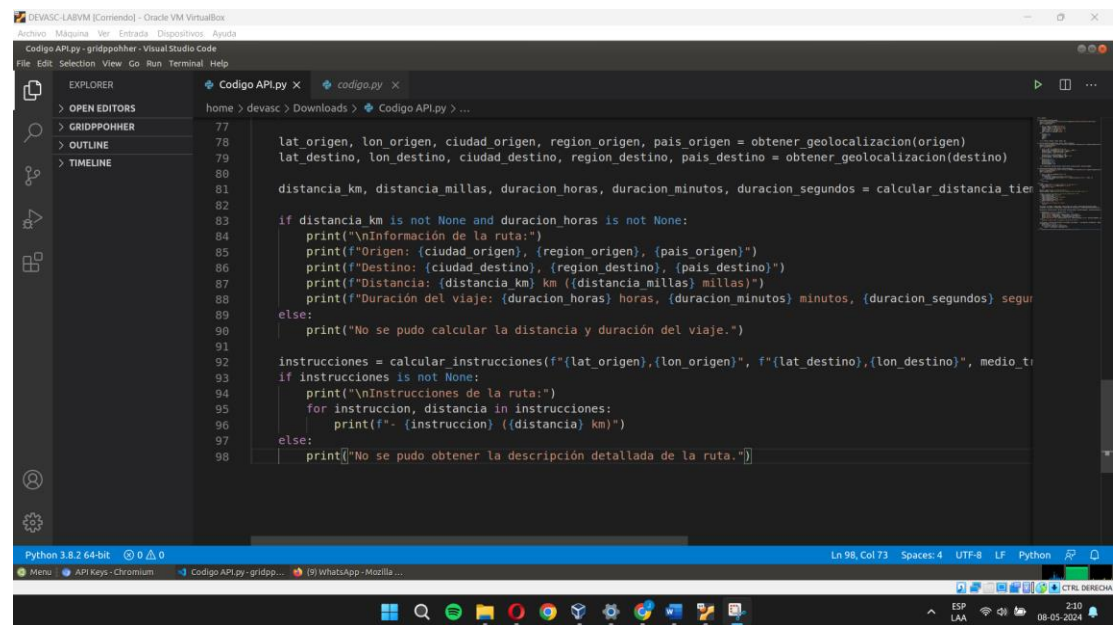
```



```

home > devasc > Downloads > Codigo API.py > ...
52     for instruccion in data['paths'][0]['instructions']:
53         instrucciones.append(instruccion['text'], round(instruccion['distance'] / 1000, 1))
54     return instrucciones
55 else:
56     return None
57
58 while True:
59     origen = input("Ingrese la ciudad de origen (o 'S' para salir): ")
60     if origen.lower() == 's' or origen.lower() == 'salir':
61         print("¡Hasta luego!")
62         break
63
64     destino = input("Ingrese la ciudad de destino: ")
65     medio transporte = input("Elija el medio de transporte (auto, bicicleta, a pie): ")
66
67     # Mapear los medios de transporte a los valores esperados por la API
68     if medio transporte.lower() == "auto":
69         medio transporte api = "car"
70     elif medio transporte.lower() == "bicicleta":
71         medio transporte api = "bike"
72     elif medio transporte.lower() == "a pie":
73         medio transporte api = "foot"
74     else:
75         print("Medio de transporte no válido!")
76         continue
77
78     lat origen, lon origen, ciudad origen, region origen, pais origen = obtener_geolocalizacion(origen)

```



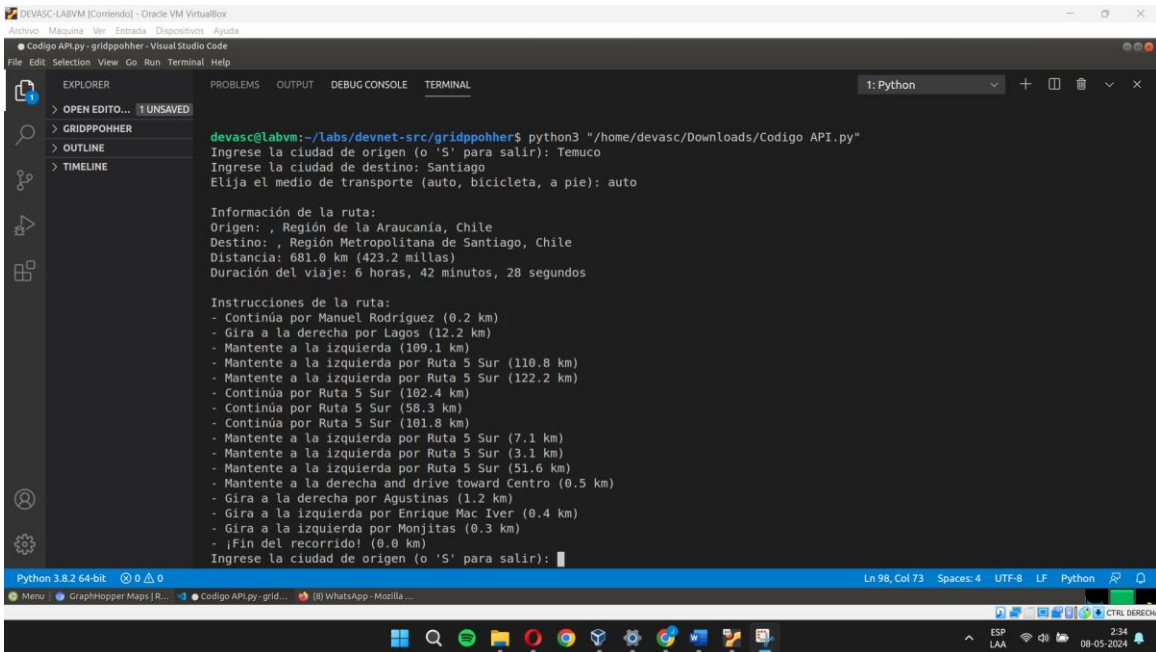
```

77     lat origen, lon origen, ciudad origen, region origen, pais origen = obtener_geolocalizacion(origen)
78     lat destino, lon destino, ciudad destino, region destino, pais destino = obtener_geolocalizacion(destino)
79
80     distancia km, distancia millas, duracion horas, duracion minutos, duracion segundos = calcular_distancia_tiem
81
82
83     if distancia km is not None and duracion horas is not None:
84         print("\nInformación de la ruta:")
85         print(f"Origen: {ciudad origen}, {region origen}, {pais origen}")
86         print(f"Destino: {ciudad destino}, {region destino}, {pais destino}")
87         print(f"Distancia: {distancia km} km ({distancia millas} millas)")
88         print(f"Duración del viaje: {duracion horas} horas, {duracion minutos} minutos, {duracion segundos} segundos")
89     else:
90         print("No se pudo calcular la distancia y duración del viaje.")
91
92     instrucciones = calcular_instrucciones(f"{lat origen},{lon origen}", f"{lat destino},{lon destino}", medio transporte)
93     if instrucciones is not None:
94         print("\nInstrucciones de la ruta:")
95         for instruccion, distancia in instrucciones:
96             print(f"- {instruccion} ({distancia} Km)")
97     else:
98         print("No se pudo obtener la descripción detallada de la ruta.")

```

En las siguientes imágenes se puede apreciar como el código arroja una pregunta al usuario, donde le pide que ingrese una ciudad de origen y una ciudad de destino en la ruta, luego de esto le pregunta el medio de transporte por el cual el usuario va a realizar el viaje, que en este caso será en auto.

Finalizando el código arrojará la información de la ruta que corresponde a: Origen, destino, distancia, duración del viaje. Para luego finalizar con las instrucciones correspondientes del viaje, mostrándole al usuario el trayecto que debe realizar hacia su destino.

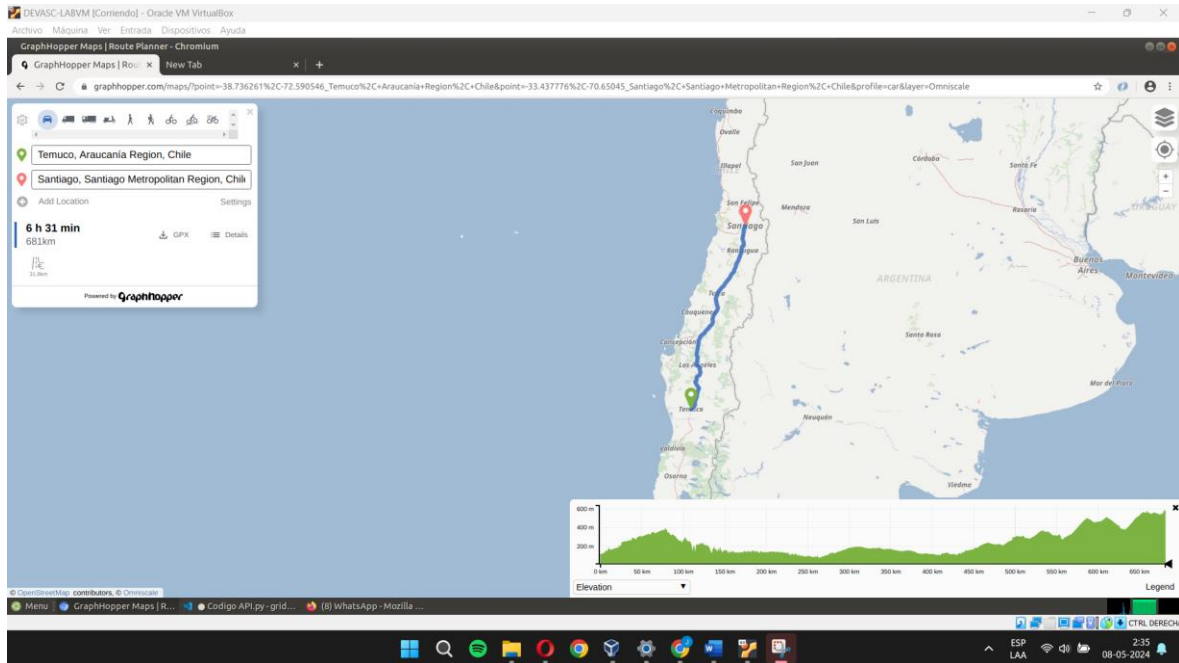


```
devasc@labvm:~/labs/devnet-src/gridppohher$ python3 "/home/devasc/Downloads/Codigo API.py"
Ingrese la ciudad de origen (o 'S' para salir): Temuco
Ingrese la ciudad de destino: Santiago
Elija el medio de transporte (auto, bicicleta, a pie): auto

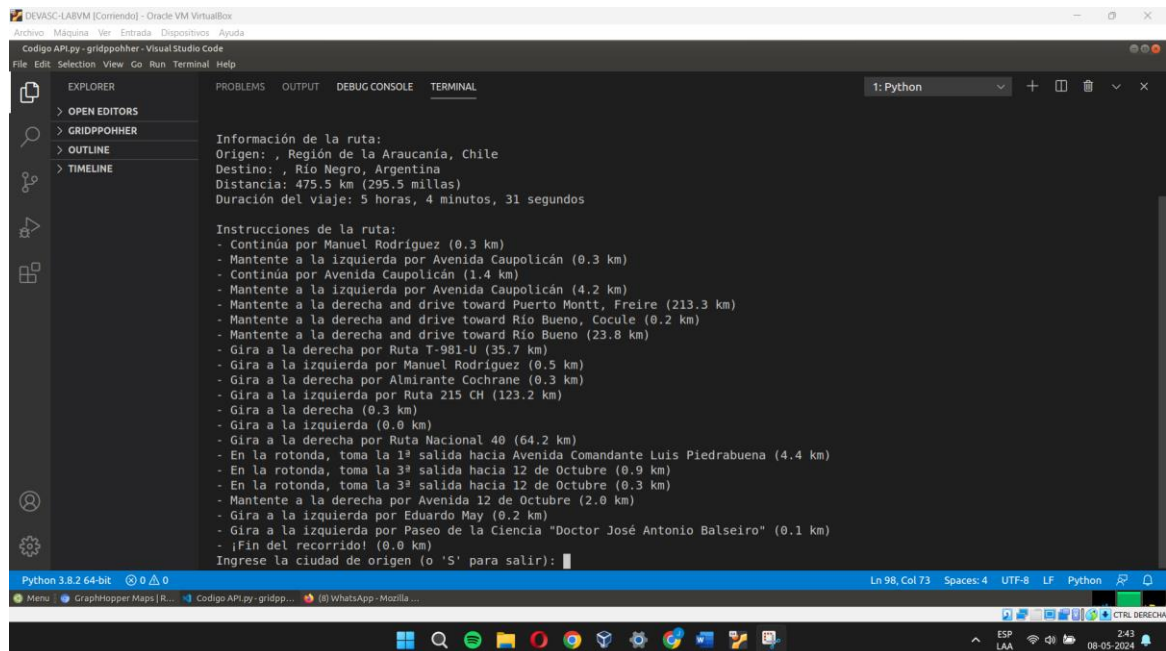
Información de la ruta:
Origen: , Región de la Araucanía, Chile
Destino: , Región Metropolitana de Santiago, Chile
Distancia: 681.0 km (423.2 millas)
Duración del viaje: 6 horas, 42 minutos, 28 segundos

Instrucciones de la ruta:
- Continúa por Manuel Rodríguez (0.2 km)
- Gira a la derecha por Lagos (12.2 km)
- Mantente a la izquierda (109.1 km)
- Mantente a la izquierda por Ruta 5 Sur (110.8 km)
- Mantente a la izquierda por Ruta 5 Sur (122.2 km)
- Continúa por Ruta 5 Sur (102.4 km)
- Continúa por Ruta 5 Sur (58.3 km)
- Continúa por Ruta 5 Sur (101.8 km)
- Mantente a la izquierda por Ruta 5 Sur (7.1 km)
- Mantente a la izquierda por Ruta 5 Sur (3.1 km)
- Mantente a la izquierda por Ruta 5 Sur (51.6 km)
- Mantente a la derecha and drive toward Centro (0.5 km)
- Gira a la derecha por Agustinas (1.2 km)
- Gira a la izquierda por Enrique Mac Iver (0.4 km)
- Gira a la izquierda por Monjitas (0.3 km)
- ¡Fin del recorrido! (0.0 km)
Ingrese la ciudad de origen (o 'S' para salir):
```

En esta captura de pantalla se muestran las horas y minutos en las que se demora un viaje en auto desde Temuco hacia Santiago. Todo esto en la página GraphHopper para así verificar que el código está correcto.

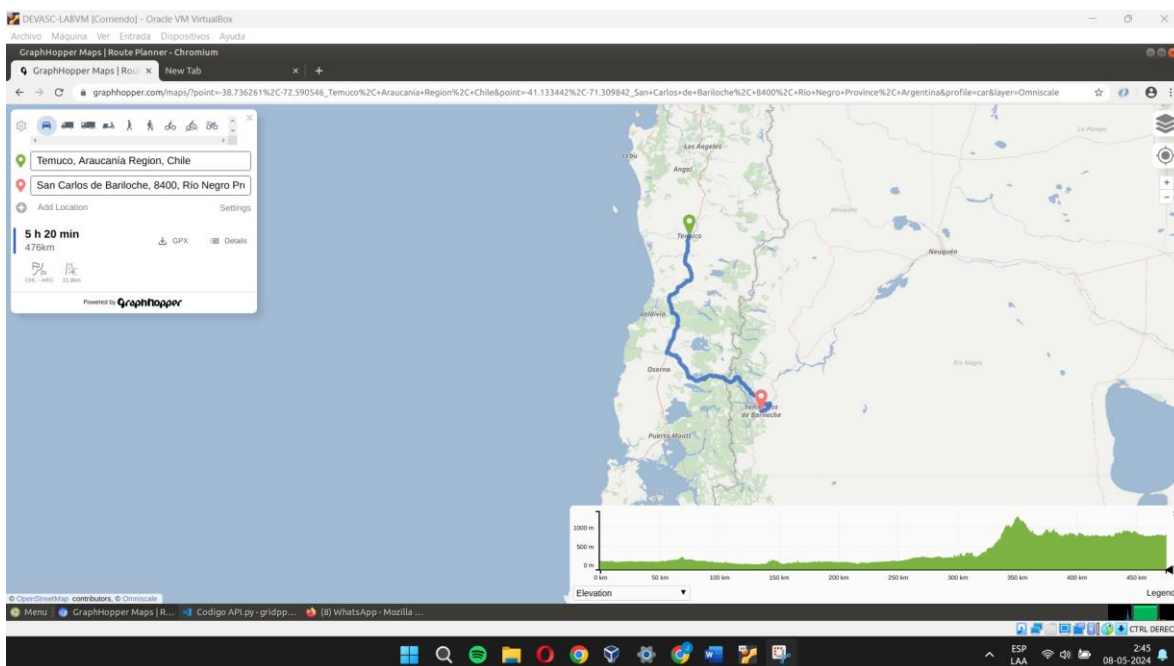


En la siguiente imagen tenemos otro ejemplo fuera de Chile, donde se aprecia un viaje desde Temuco, hacia la ciudad de Bariloche que está ubicado en el país Argentina.





Finalizando con la interfaz grafica del viaje desde Temuco hacia Bariloche, con una duración de 5 horas con 20 minutos.



## 4 Conclusión

Para concluir este trabajo, en el primer ítem, hemos creado y probado el entorno de desarrollo DevNet siguiendo una serie de pasos que incluyeron la configuración inicial del repositorio Git, la creación de un directorio de trabajo, la manipulación de archivos mediante Git, la creación y fusión de ramas, así como el envío del trabajo a GitHub para su revisión.

En el segundo ítem, hemos desarrollado una aplicación en Python que cumple con las siguientes funcionalidades: proporcionar la geolocalización de un origen y destino dados por el usuario, calcular la distancia entre ciudades de Latinoamérica utilizando la API GraphHopper con opción de elegir el medio de transporte, mostrar la duración del viaje en horas, minutos y segundos, presentar los valores con precisión de un decimal, y ofrecer la opción de salir del programa. Además, hemos incluido la narrativa del viaje en español para una experiencia más completa.

En conclusión, esta informe nos ha permitido aplicar y fortalecer nuestros conocimientos en el desarrollo de entornos de trabajo colaborativo y en la creación de aplicaciones prácticas utilizando herramientas de geolocalización y APIS externas