
Evaluación N ° 2

Desarrollo de entorno DEVNET y APIS

NOMBRE: Bastián Martorell, Javier Riquelme
CARRERA: Ing. en Telecomunicaciones, Conectividad y Redes
ASIGNATURA: Redes Avanzadas I
PROFESOR: Mario Villanueva Alveal
FECHA: 08/05/2024

Tabla de contenido

Evaluación N ° 2	1
Desarrollo de entorno DEVNET y APIS	1
1 Introducción	3
2 Creación y desarrollo de entorno DevNet	4
2.1 Inicio del repositorio de Git:	4
2.2 Confirmación de la configuración:	5
2.3 Creación de un archivo de texto con nombre y apellido	6
2.4 Modificación del archivo y posterior confirmación en ambas partes	7
2.5 Modificación de datos	9
2.6 Nuevo Branch, fusión de ramas y nuevo contenido	11
2.7 Enlace repositorio: https://github.com/Javierhrg/RedesAvanzadasI-2024	13
3 Creación de aplicación en Python	14
3.1 Desarrollo de la primera API	15
3.2 Ejemplo de funcionamiento	16
3.3 Desarrollo de segunda API	17
3.4 Ejemplo de funcionamiento	19
4 Conclusión	20

1 Introducción

En el actual panorama de desarrollo de software, la habilidad para establecer entornos de desarrollo eficaces y confiables es esencial para el éxito de cualquier proyecto. Este informe explora el proceso de creación y prueba del entorno de desarrollo DevNet, enfocándose en dos aplicaciones específicas diseñadas para proporcionar soluciones prácticas y efectivas.

El análisis se centra primero en la creación de una aplicación que ofrece la geolocalización de las ubicaciones de origen y destino ingresadas por el usuario. Su objetivo es facilitar la obtención de información geográfica relevante, como latitud, longitud, ciudad, región y país, sin generar mensajes de error molestos para el usuario. Esta aplicación promete una experiencia fluida y precisa a través de un diseño intuitivo y una integración sin problemas de APIs de geolocalización.

El segundo aspecto crucial del estudio se enfoca en otra aplicación, esta vez orientada a calcular la distancia entre diferentes ciudades de América Latina. Utilizando la API de GraphHopper, esta aplicación permite calcular la distancia en kilómetros y millas entre cualquier par de ciudades seleccionadas por el usuario. Esta funcionalidad proporciona información valiosa para planificar viajes y rutas, demostrando la versatilidad y utilidad de las herramientas disponibles en el entorno DevNet.

A lo largo del informe, se examina detenidamente el proceso de desarrollo y prueba de ambas aplicaciones, resaltando las técnicas, herramientas y mejores prácticas utilizadas para garantizar su funcionamiento óptimo. Además, se exploran los beneficios potenciales que estas soluciones pueden aportar a las organizaciones, desde la mejora de la experiencia del usuario hasta el aumento de la eficiencia operativa y la competitividad en el mercado.

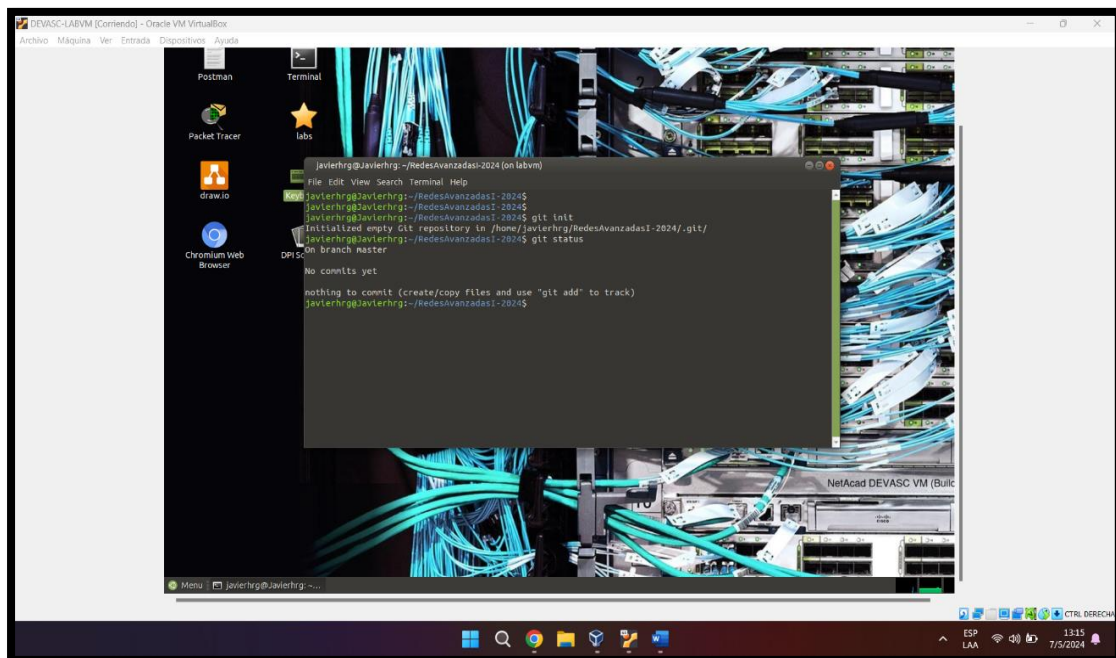
Con un enfoque en la innovación, la calidad y la satisfacción del cliente, este informe ofrece una visión integral del potencial del entorno de desarrollo DevNet para impulsar el éxito empresarial en el mundo digital actual.

2 Creación y desarrollo de entorno DevNet

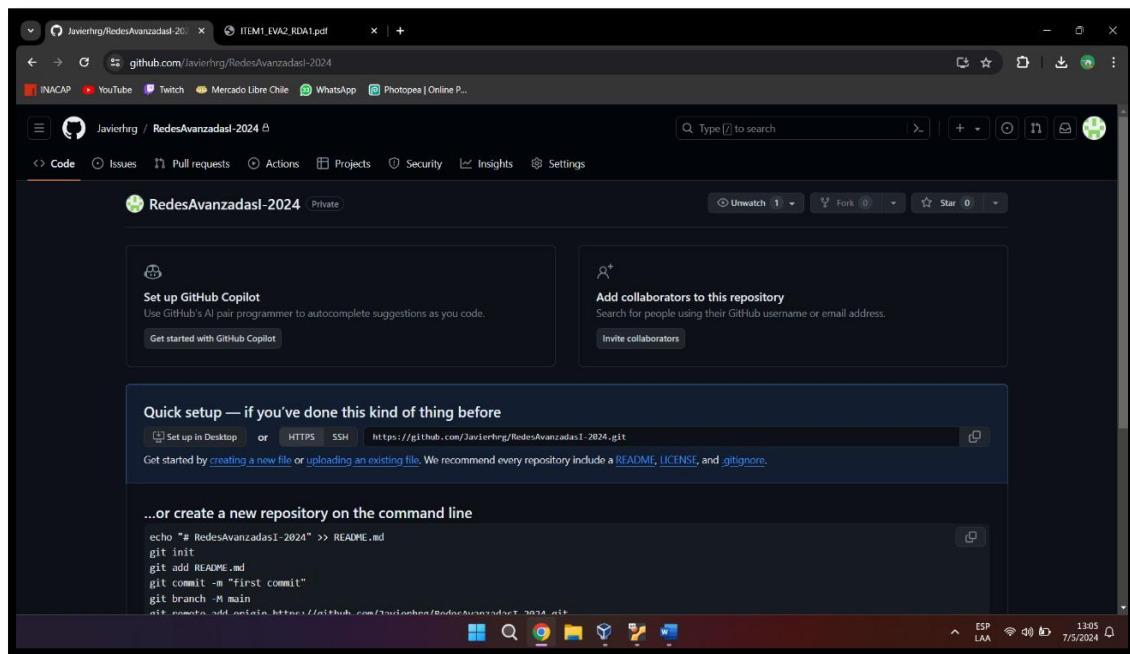
Para iniciar, resulta esencial establecer un repositorio de Git para el proyecto. Esto posibilitará rastrear los cambios en el código y colaborar eficazmente con otros miembros del equipo de desarrollo. A continuación, se detallan los pasos requeridos para iniciar el repositorio y configurar el nombre y correo electrónico del usuario.

2.1 Inicio del repositorio de Git:

- Se abre la terminal o línea de comandos en el directorio raíz del proyecto.
- Se ejecuta el comando "git init". Esto inicializa un nuevo repositorio de Git en el directorio actual.

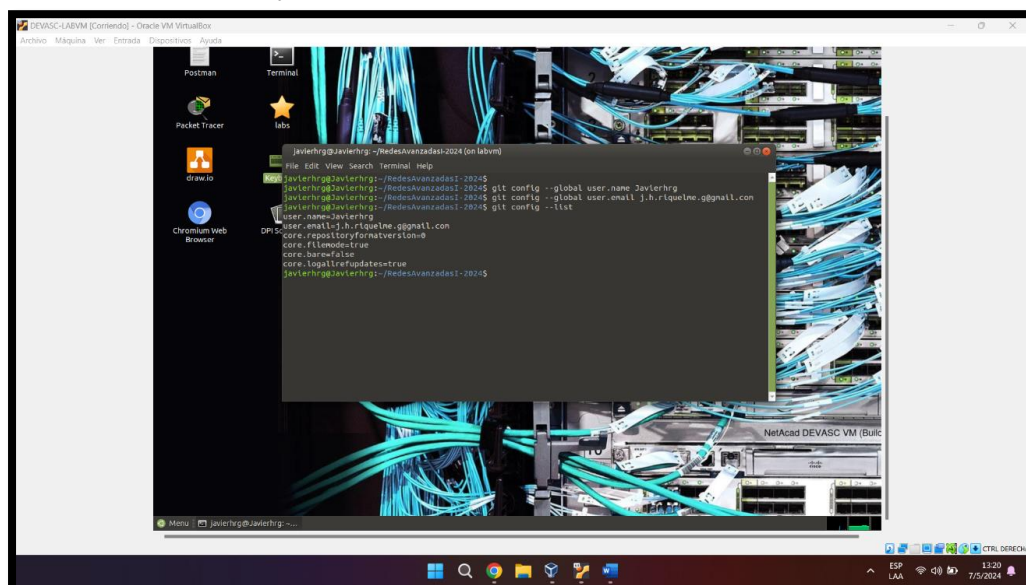


- Después de crear el repositorio local, se creará el repositorio remoto en la página de GitHub con la cuenta correspondiente iniciada anteriormente, al cual se le asignará el nombre "RedesAvanzadasI-2024".



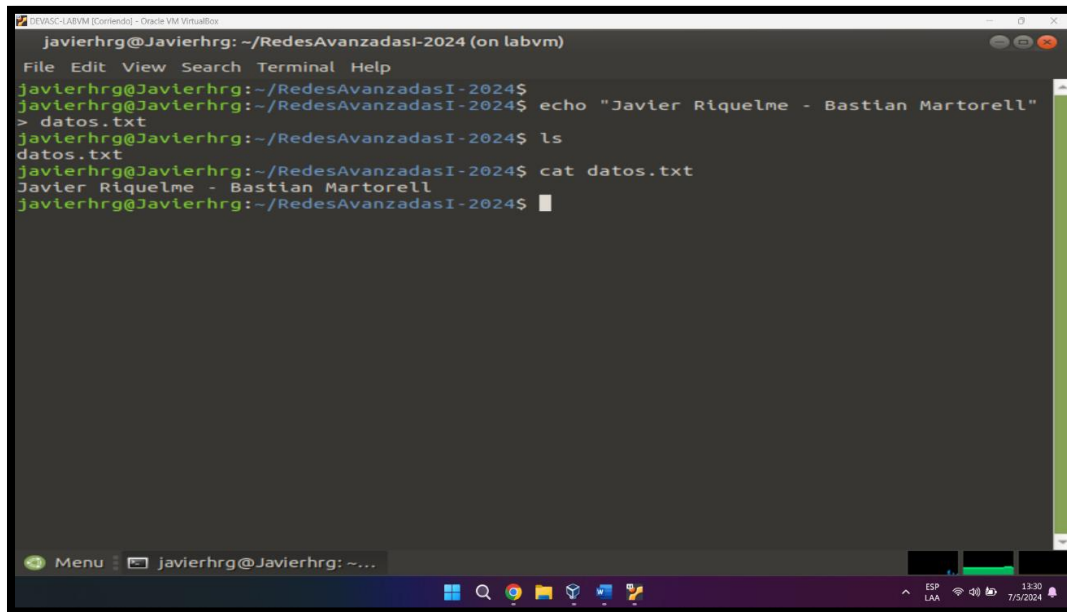
2.2 Confirmación de la configuración:

- Una vez configurado tu nombre y correo electrónico, puedes verificar que la configuración se haya realizado correctamente utilizando el comando "git config --list". Este comando mostrará una lista de todas las configuraciones actuales de Git en tu sistema, incluyendo tu nombre de usuario y correo electrónico.



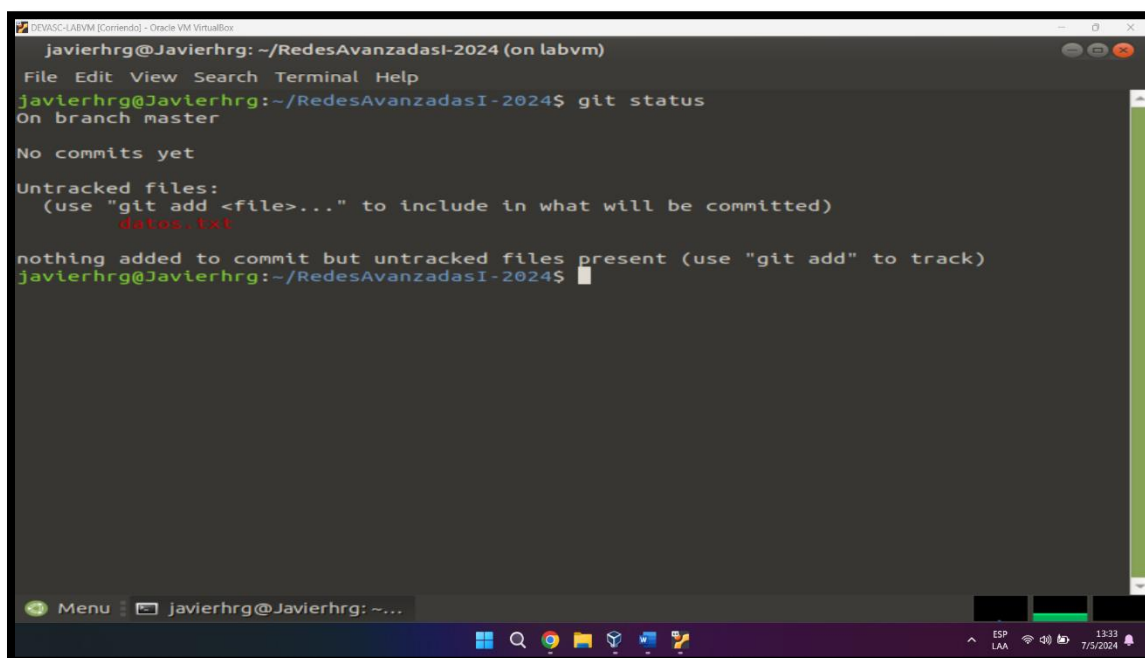
2.3 Creación de un archivo de texto con nombre y apellido

- Para este paso, se utilizará el comando "echo" para agregar los nombres y apellidos de los integrantes y así continuar con el desarrollo del trabajo. Una vez agregados los nombres, se guardarán estos datos en un archivo ".txt" que se designará como "datos.txt".



```
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024 (on labvm)
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ echo "Javier Riquelme - Bastian Martorell"
> datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ ls
datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ cat datos.txt
Javier Riquelme - Bastian Martorell
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
```

- Después de haber registrado los nombres de los integrantes en "datos.txt", verificamos el estado de Git donde nos indica que debemos agregar el archivo "datos.txt".



```
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024 (on labvm)
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git status
On branch master

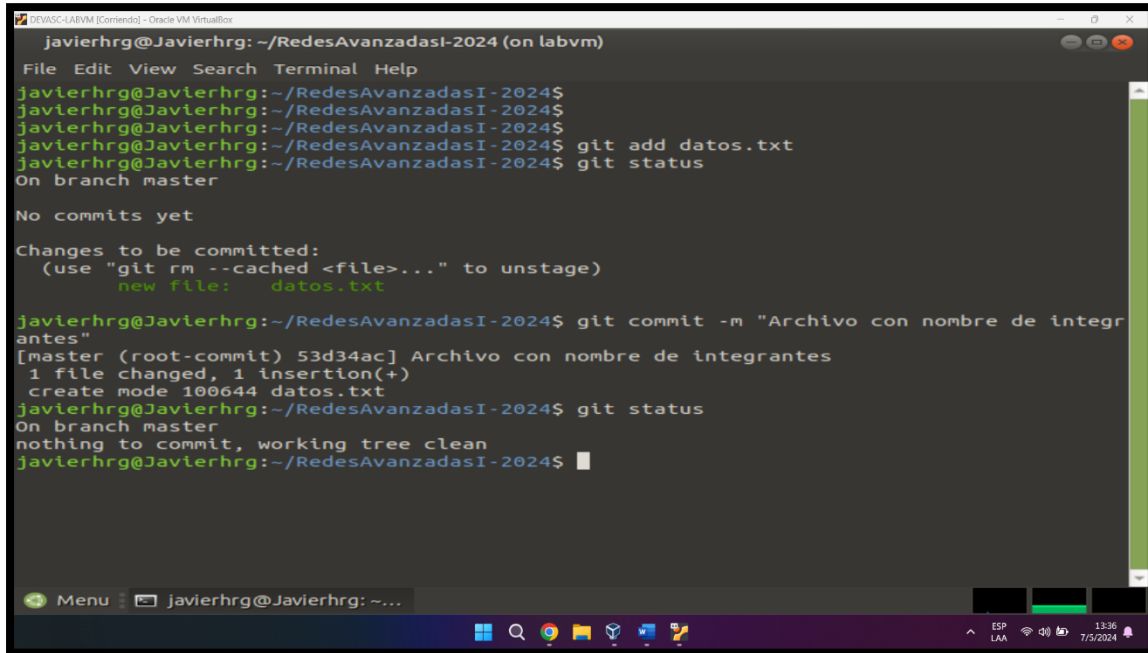
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    datos.txt

nothing added to commit but untracked files present (use "git add" to track)
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
```

2.4 Modificación del archivo y posterior confirmación en ambas partes

- Para agregar el archivo "datos.txt", se aplicará el comando "git add datos.txt". Al mismo tiempo, para comprobar que está en orden, se utilizará "git status". Después, se realizará un "git commit -m" donde se dejará un comentario para verificar y describir lo realizado. Por último, se ejecutará nuevamente "git status" para asegurarse de que el archivo se haya agregado correctamente.



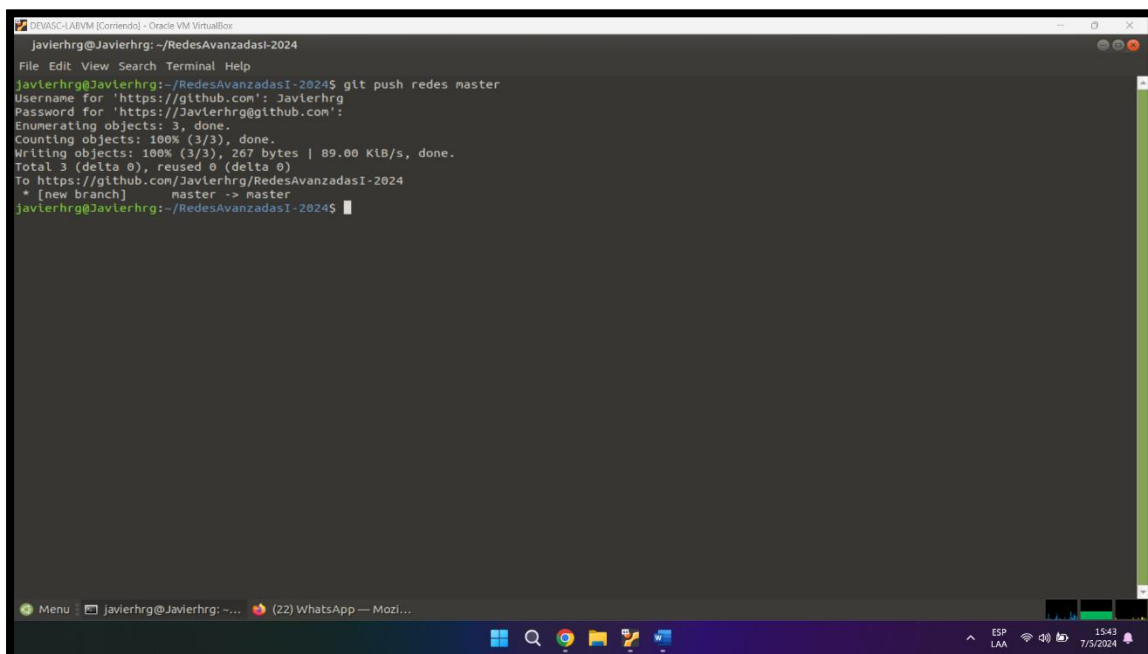
```
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024 (on labvm)
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git add datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   datos.txt

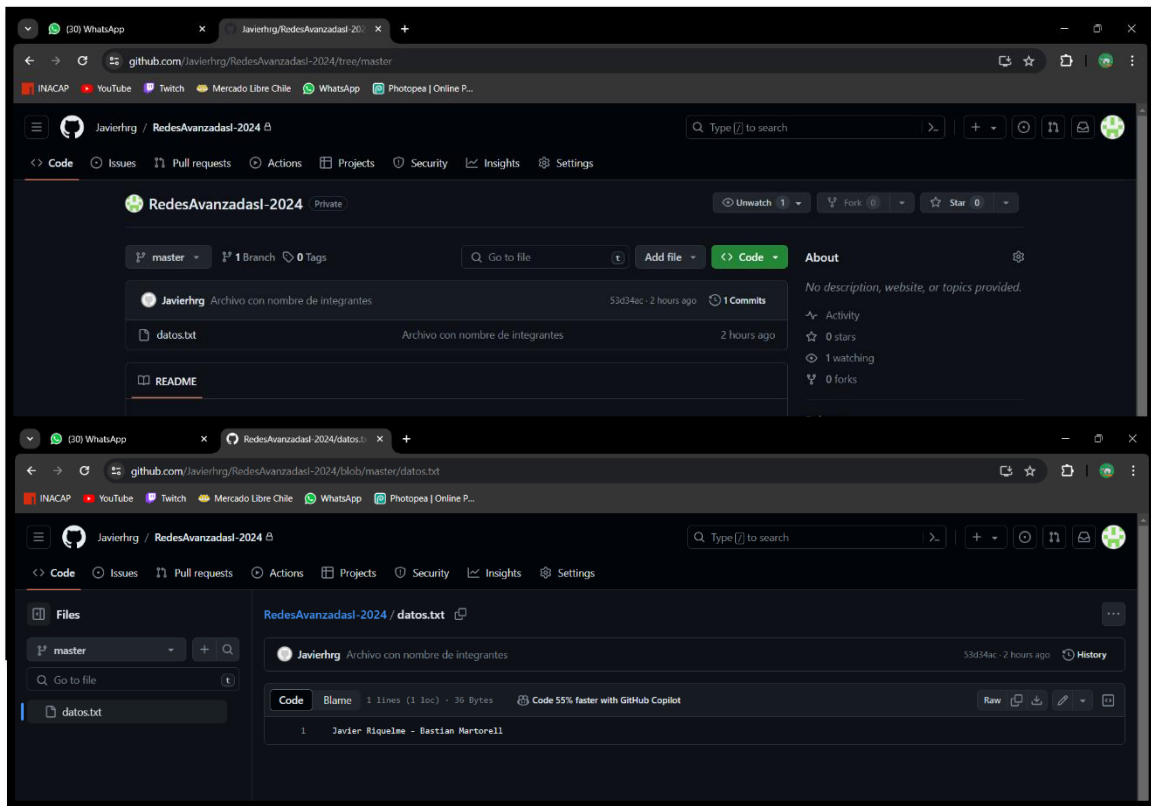
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git commit -m "Archivo con nombre de integrantes"
[master (root-commit) 53d34ac] Archivo con nombre de integrantes
 1 file changed, 1 insertion(+)
 create mode 100644 datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git status
On branch master
nothing to commit, working tree clean
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
```

- En este paso, se utiliza el comando "push" para subir el archivo "datos.txt" desde el repositorio local de Git hacia el repositorio remoto.



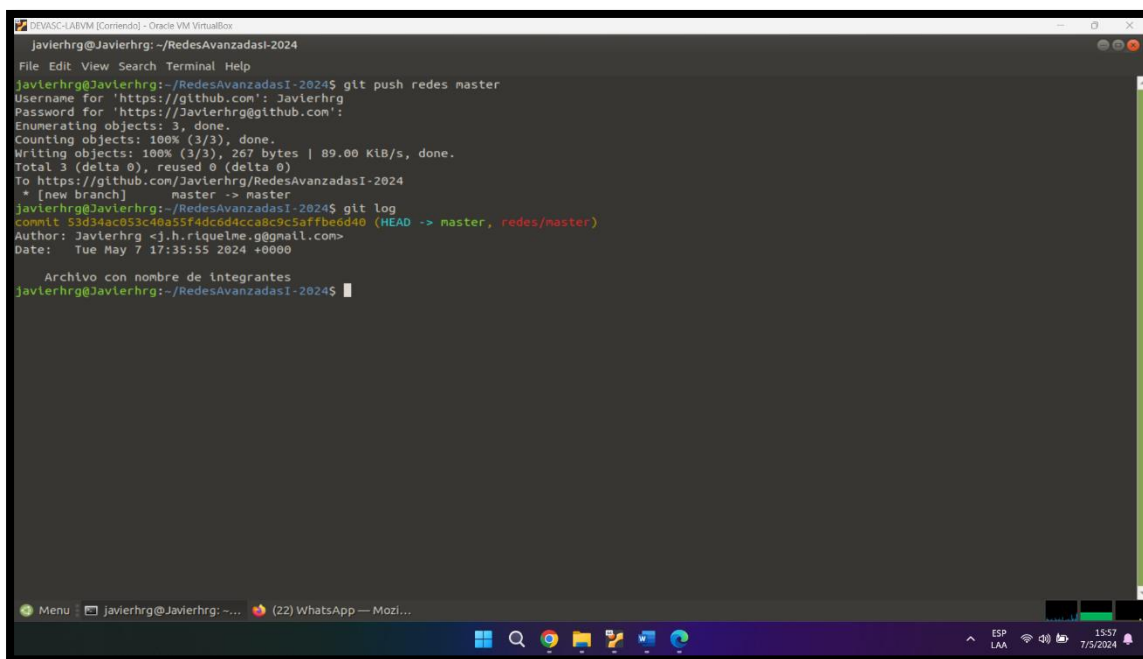
```
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git push redes master
Username for 'https://github.com': Javierhrg
Password for 'https://Javierhrg@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 267 bytes | 89.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Javierhrg/RedesAvanzadasI-2024
 * [new branch] master -> master
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
```

- Confirmación del cambio en la página de GitHub.



2.5 Modificación de datos

- Para modificar los datos, se debe utilizar el siguiente comando: "git push redes master". Esto indica a Git que suba al repositorio remoto, en la rama master, el archivo denominado "datos.txt", seguido de "git log".

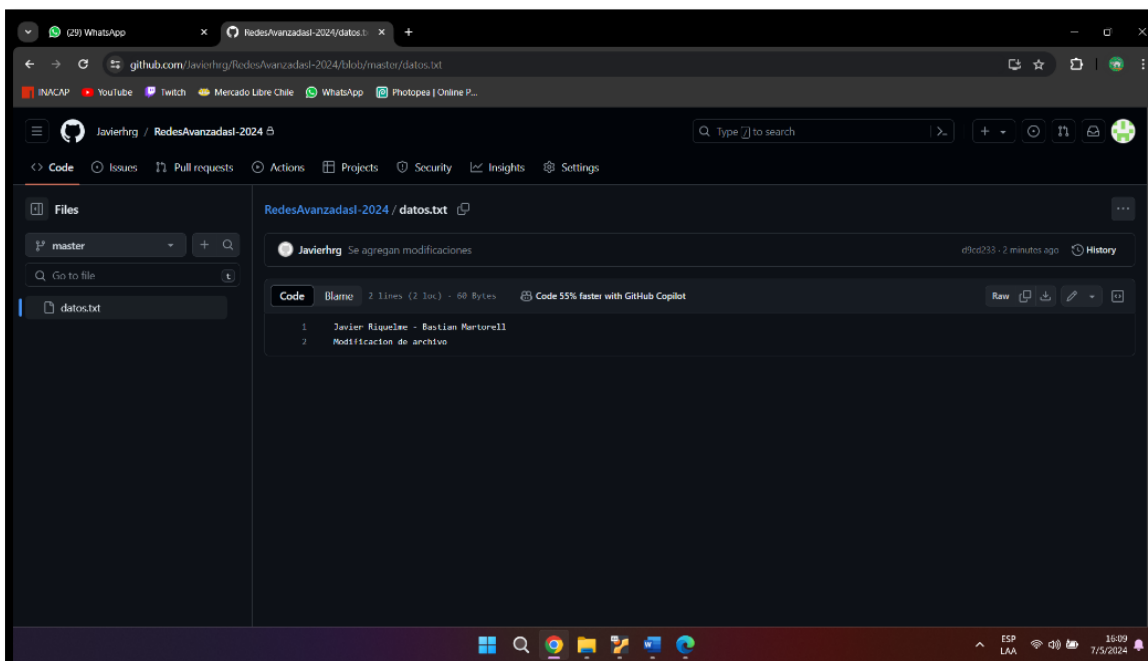


```

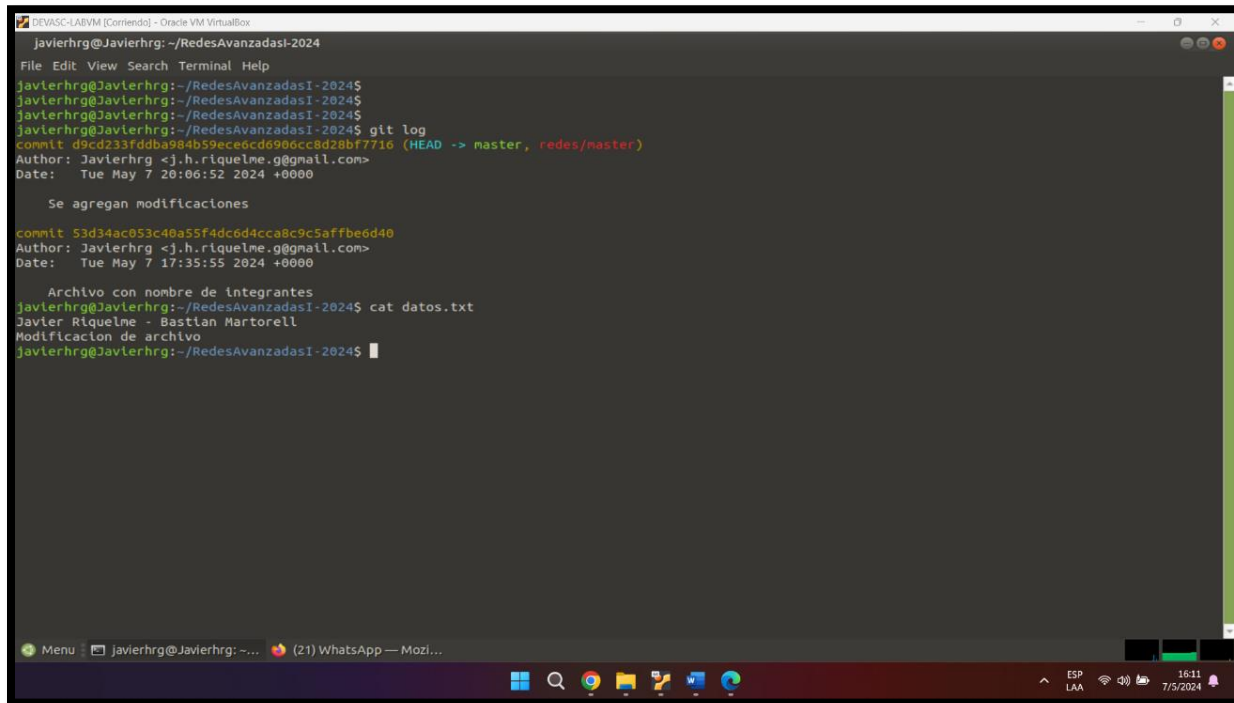
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git push redes master
Username for 'https://github.com': Javierhrg
Password for 'https://Javierhrg@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 267 bytes | 89.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Javierhrg/RedesAvanzadasI-2024
 * [new branch] master -> master
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git log
commit 53d34ac053c40a55f4dc6d4cca8c9c5affbe0d40 (HEAD -> master, redes/master)
Author: Javierhrg <j.h.rigue@ne.ig@gmail.com>
Date: Tue May 7 17:35:55 2024 +0000

    Archivo con nombre de integrantes
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
  
```

- Después, continuaremos con la confirmación del cambio en el repositorio remoto de GitHub.



- Y también se verificará el cambio en el repositorio local.



```

DEVASC-LABVM [Corriendo] - Oracle VM VirtualBox
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git log
commit d9cd233fddb984b59ece6cd6906cc8d28bf7716 (HEAD -> master, redes/master)
Author: Javierhrg <j.h.riquelme.g@gmail.com>
Date: Tue May 7 20:06:52 2024 +0000

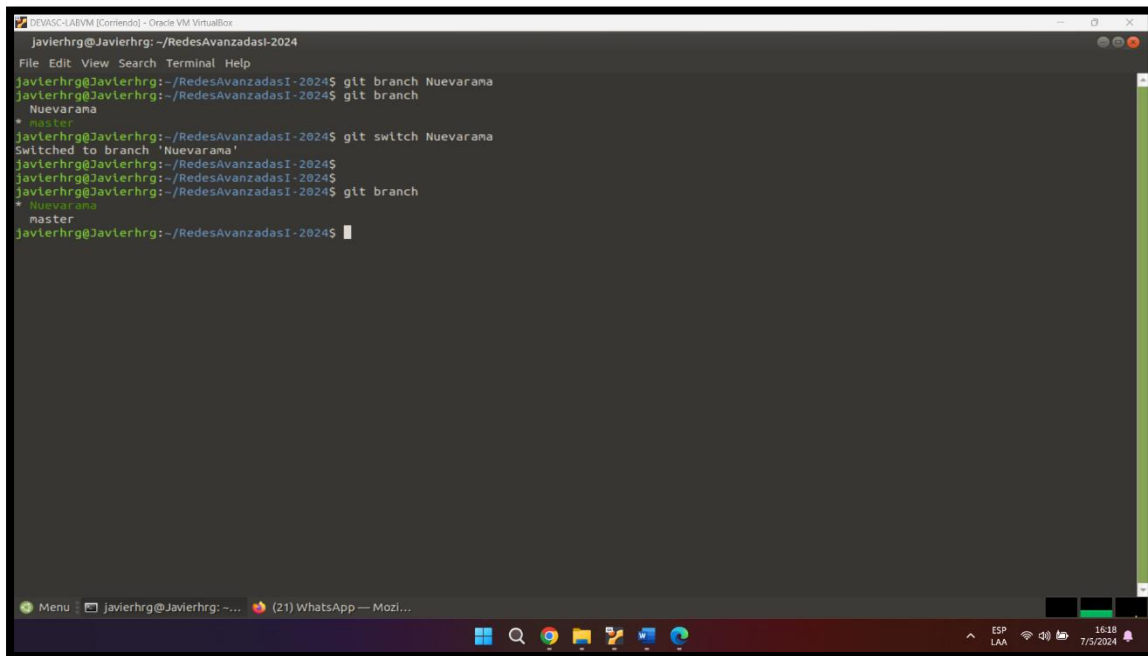
    Se agregan modificaciones

commit 53d34ac053c40a55f4dc6d4cca8c9c5affbe6d40
Author: Javierhrg <j.h.riquelme.g@gmail.com>
Date: Tue May 7 17:35:55 2024 +0000

    Archivo con nombre de integrantes
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ cat datos.txt
Javier Riquelme - Bastian Martorell
Modificacion de archivo
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
  
```

2.6 Nuevo Branch, fusión de ramas y nuevo contenido

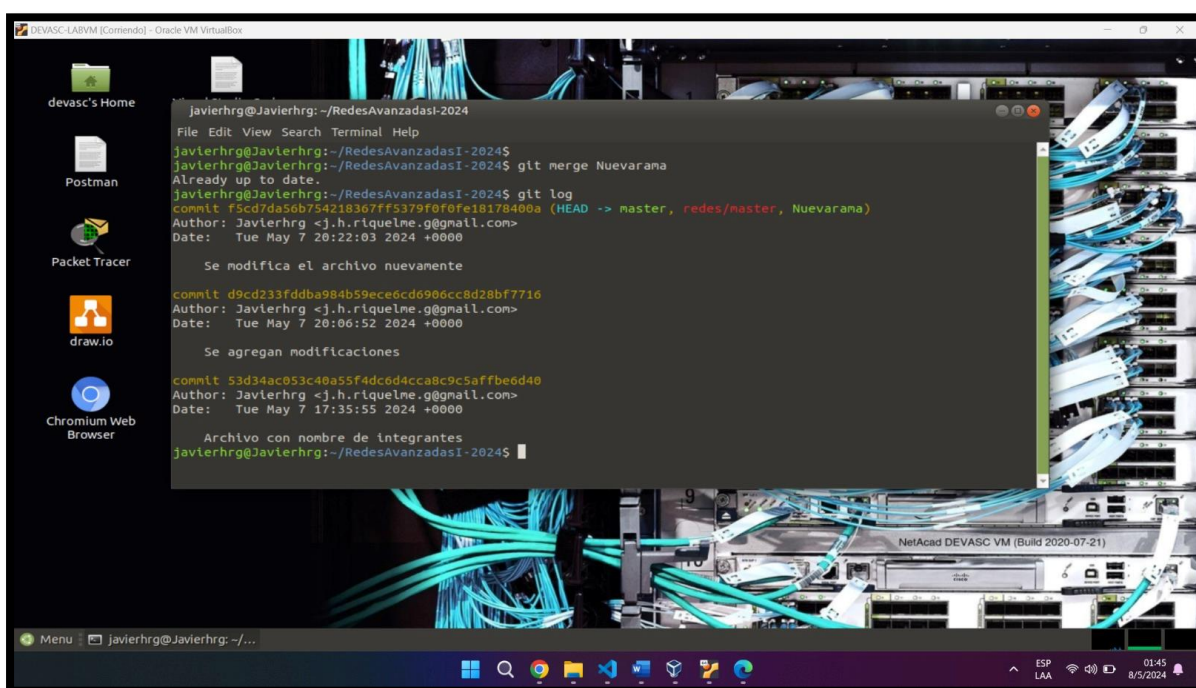
- Para este punto, se ejecutará el comando "git branch", que servirá para crear una nueva rama, y también nos permitirá cambiarnos entre las mismas con el comando "git switch". Después, se creará una nueva rama escribiendo "git branch Nuevaarama".



```

javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git branch Nuevaarama
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git branch
* master
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git switch Nuevaarama
Switched to branch 'Nuevaarama'
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git branch
* Nuevaarama
  master
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
  
```

- Después de crear la nueva rama, se utilizará el comando "git merge" con la rama "Nuevaarama" para fusionarla con la otra rama "master".



```

javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git merge Nuevaarama
Already up to date.
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git log
commit f5cd7da56b754218367ff5379f0f0fe18178400a (HEAD -> master, redes/master, Nuevaarama)
Author: Javierhrg <j.h.rtuellme.g@gmail.com>
Date: Tue May 7 20:22:03 2024 +0000

    Se modifica el archivo nuevamente

commit d9cd233fddba984b59ece6cd6906cc8d28bf7716
Author: Javierhrg <j.h.rtuellme.g@gmail.com>
Date: Tue May 7 20:06:52 2024 +0000

    Se agregan modificaciones

commit 53d34ac053c40a55f4dc6d4cca8c9c5affbe6d40
Author: Javierhrg <j.h.rtuellme.g@gmail.com>
Date: Tue May 7 17:35:55 2024 +0000

    Archivo con nombre de integrantes
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
  
```

- Después, se modifica nuevamente el archivo "datos.txt" utilizando el comando "echo" para agregar la nueva modificación. Para verificar que el cambio se haya efectuado, se ejecutará "cat datos.txt". Luego, se agregarán los nuevos datos y se continuará con "git add" y "git commit -m".

```

DEVASC-LABVM [Comandos] - Oracle VM VirtualBox
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ ls
datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ cat datos.txt
Javier Riquelme - Bastian Martorell
Modificacion de archivo
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ echo "Modificacion de archivo N°2" >> datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ cat datos.txt
Javier Riquelme - Bastian Martorell
Modificacion de archivo
Modificacion de archivo N°2
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git add datos.txt
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git commit -m "Se modifica el archivo nuevamente"
[Nuevarama f5cd7da] Se modifica el archivo nuevamente
1 file changed, 1 insertion(+)

```

- Luego de realizar las modificaciones, se procederá a subir los cambios al repositorio remoto de GitHub.

```

DEVASC-LABVM [Comandos] - Oracle VM VirtualBox
javierhrg@Javierhrg: ~/RedesAvanzadasI-2024
File Edit View Search Terminal Help
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git push redes master
Username for 'https://github.com': Javierhrg
Password for 'https://Javierhrg@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Javierhrg/RedesAvanzadasI-2024
d9cd233..f5cd7da master -> master
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$ git log
commit f5cd7da56b754218367ff5379f0f0fe18178400a (HEAD -> master, redes/master, Nuevarama)
Author: Javierhrg <j.h.riquelme.g@gmail.com>
Date: Tue May 7 20:22:03 2024 +0000

    Se modifica el archivo nuevamente

commit d9cd233fddba984b59ece6cd6906cc8d28bf7716
Author: Javierhrg <j.h.riquelme.g@gmail.com>
Date: Tue May 7 20:06:52 2024 +0000

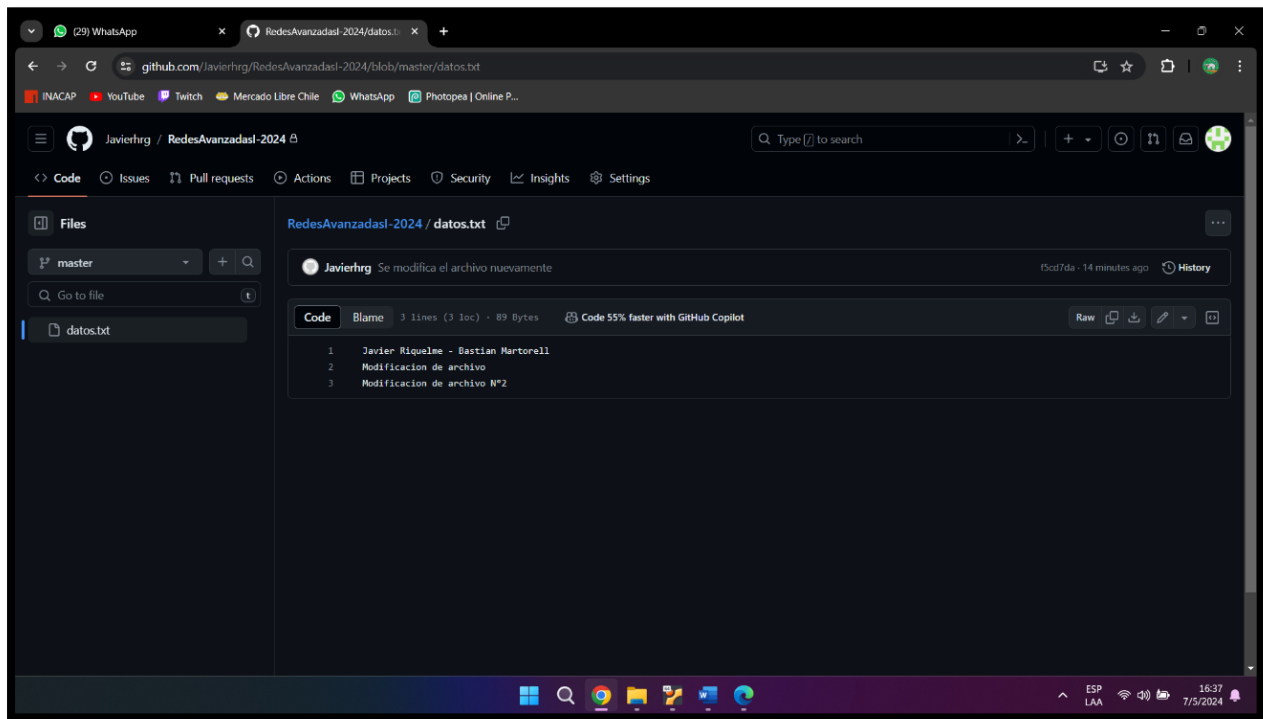
    Se agregan modificaciones

commit 53d34ac053c40a55f4dc6d4cca8c9c5affbe6d40
Author: Javierhrg <j.h.riquelme.g@gmail.com>
Date: Tue May 7 17:35:55 2024 +0000

    Archivo con nombre de integrantes
javierhrg@Javierhrg:~/RedesAvanzadasI-2024$

```

- Finalmente, se revisará el cambio en la página de GitHub.



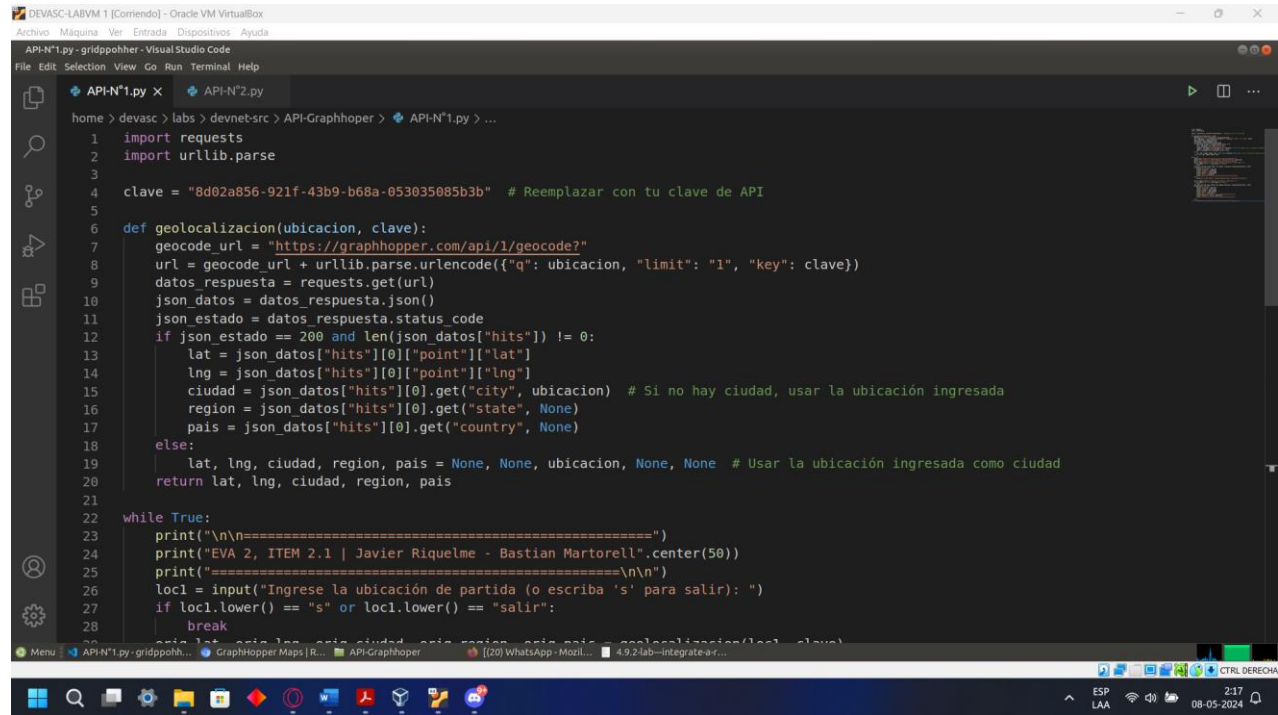
2.7 Enlace repositorio: <https://github.com/Javierhrg/RedesAvanzadasI-2024>

3 Creación de aplicación en Python

En respuesta a la solicitud de creación de una aplicación en Python, se ha desarrollado un programa multifuncional que aborda dos escenarios distintos pero complementarios. En primer lugar, la aplicación proporciona la geolocalización de un origen y destino específicos ingresados por el usuario, mostrando información detallada como latitud, longitud, ciudad, región y país, con el objetivo de ofrecer una herramienta útil y precisa para la ubicación de lugares. Posteriormente, la aplicación aprovecha la API de GraphHopper para calcular la distancia entre ciudades de cualquier país de Latinoamérica en kilómetros, ofreciendo también la opción de visualizar esta distancia en millas. Además, permite al usuario elegir el medio de transporte, muestra la duración estimada del viaje en horas, minutos y segundos con valores redondeados a un decimal, e imprime la narrativa del viaje en español, todo ello en un entorno interactivo que facilita su uso. La aplicación está diseñada para evitar mensajes de error y proporcionar una experiencia fluida y completa al usuario, incluyendo la opción de salir del programa mediante la entrada de la letra "S" o "Salir". Con estas características, la aplicación busca brindar una solución integral para la localización y planificación de viajes dentro de Latinoamérica, ofreciendo datos precisos y relevantes en un formato accesible y amigable.

3.1 Desarrollo de la primera API

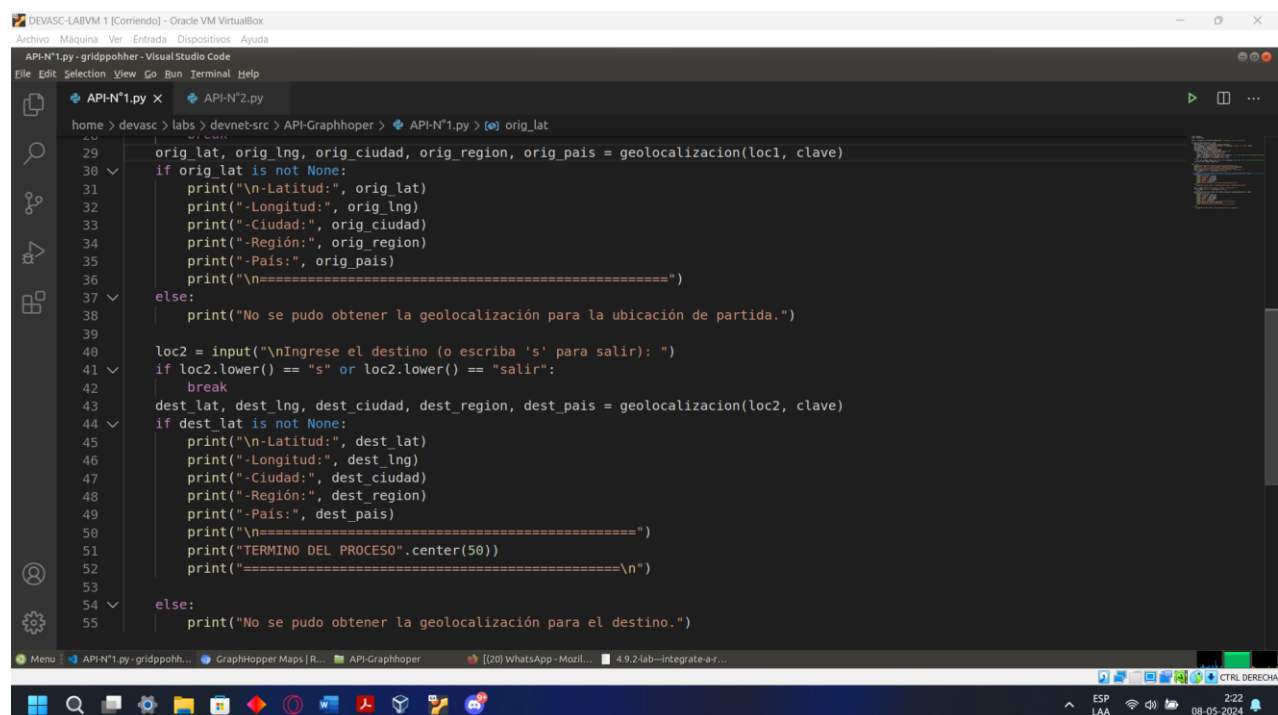
El presente informe analiza un script en Python diseñado para obtener la geolocalización de ubicaciones específicas utilizando la API de GraphHopper. Este script permite a los usuarios ingresar una ubicación de origen y una ubicación de destino, y posteriormente obtiene información detallada sobre cada una de ellas, incluyendo la latitud, longitud, ciudad, región y país.



```

home > devasc > labs > devnet-src > API-Graphhopper > API-N°1.py > ...
1 import requests
2 import urllib.parse
3
4 clave = "8d02a856-921f-43b9-b68a-053035085b3b" # Reemplazar con tu clave de API
5
6 def geolocalizacion(ubicacion, clave):
7     geocode_url = "https://graphhopper.com/api/1/geocode?"
8     url = geocode_url + urllib.parse.urlencode({"q": ubicacion, "limit": "1", "key": clave})
9     datos_respuesta = requests.get(url)
10    json_datos = datos_respuesta.json()
11    json_estado = datos_respuesta.status_code
12    if json_estado == 200 and len(json_datos["hits"]) != 0:
13        lat = json_datos["hits"][0]["point"]["lat"]
14        lng = json_datos["hits"][0]["point"]["lng"]
15        ciudad = json_datos["hits"][0].get("city", ubicacion) # Si no hay ciudad, usar la ubicación ingresada
16        region = json_datos["hits"][0].get("state", None)
17        pais = json_datos["hits"][0].get("country", None)
18    else:
19        lat, lng, ciudad, region, pais = None, None, ubicacion, None, None # Usar la ubicación ingresada como ciudad
20    return lat, lng, ciudad, region, pais
21
22 while True:
23     print("\n\n=====")
24     print("EVA 2, ITEM 2.1 | Javier Riquelme - Bastian Martorell".center(50))
25     print("=====")
26     loc1 = input("Ingrese la ubicación de partida (o escriba 's' para salir): ")
27     if loc1.lower() == "s" or loc1.lower() == "salir":
28         break
29     orig_lat, orig_lng, orig_ciudad, orig_region, orig_pais = geolocalizacion(loc1, clave)

```



```

29 orig_lat, orig_lng, orig_ciudad, orig_region, orig_pais = geolocalizacion(loc1, clave)
30 if orig_lat is not None:
31     print("\n-Latitud:", orig_lat)
32     print("\n-Longitud:", orig_lng)
33     print("\n-Ciudad:", orig_ciudad)
34     print("\n-Región:", orig_region)
35     print("\n-País:", orig_pais)
36     print("\n=====")
37 else:
38     print("No se pudo obtener la geolocalización para la ubicación de partida.")
39
40 loc2 = input("\nIngrese el destino (o escriba 's' para salir): ")
41 if loc2.lower() == "s" or loc2.lower() == "salir":
42     break
43 dest_lat, dest_lng, dest_ciudad, dest_region, dest_pais = geolocalizacion(loc2, clave)
44 if dest_lat is not None:
45     print("\n-Latitud:", dest_lat)
46     print("\n-Longitud:", dest_lng)
47     print("\n-Ciudad:", dest_ciudad)
48     print("\n-Región:", dest_region)
49     print("\n-País:", dest_pais)
50     print("\n=====")
51     print("TERMINO DEL PROCESO".center(50))
52     print("=====")
53 else:
54     print("No se pudo obtener la geolocalización para el destino.")
55

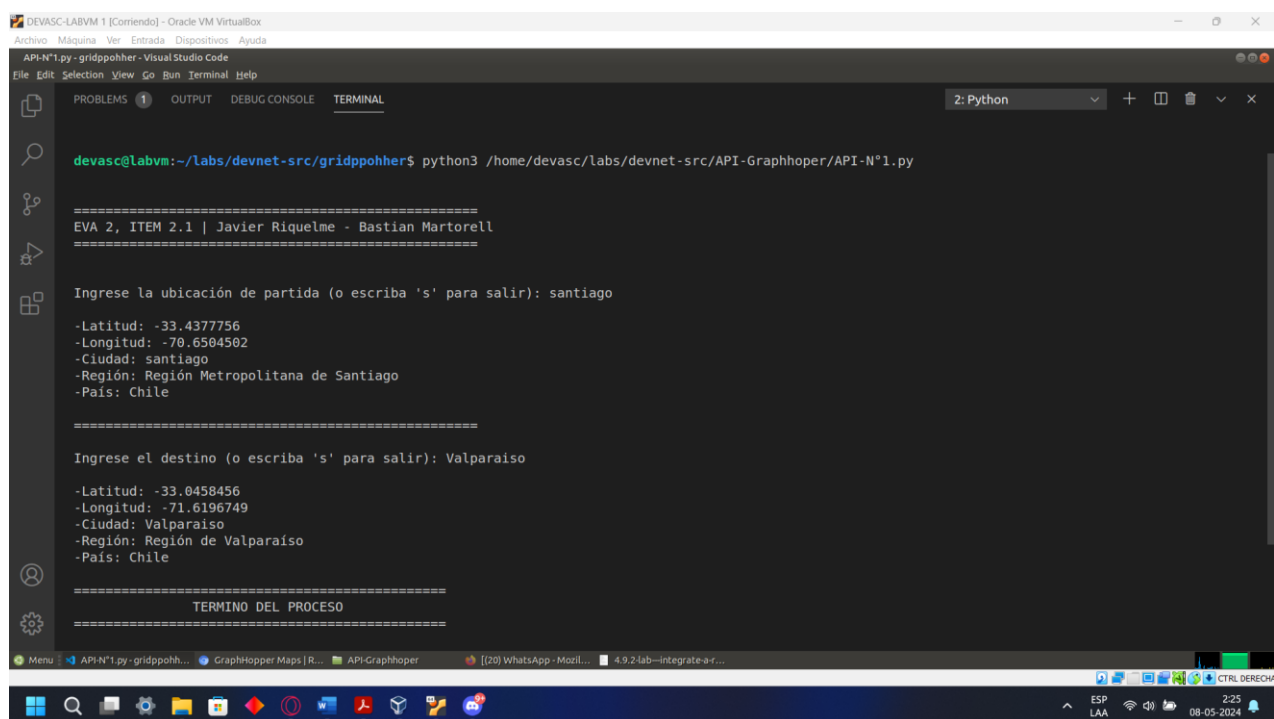
```


3.2 Ejemplo de funcionamiento

El código de la primera API desarrollada sirve para geolocalizar dos puntos geográficos y los describe de la siguiente manera:

- Latitud
- Longitud
- Ciudad
- Región
- País

A continuación, se presentará un ejemplo visual de cómo proporcionaría esta información requerida por el usuario.



```

devasc@labvm:~/labs/devnet-src/gridppohher$ python3 /home/devasc/labs/devnet-src/API-Graphhoper/API-N°1.py

=====
EVA 2, ITEM 2.1 | Javier Riquelme - Bastian Martorell
=====

Ingrese la ubicación de partida (o escriba 's' para salir): santiago

-Latitud: -33.4377756
-Longitud: -70.6504502
-Ciudad: Santiago
-Región: Región Metropolitana de Santiago
-País: Chile

=====

Ingrese el destino (o escriba 's' para salir): Valparaíso

-Latitud: -33.0458456
-Longitud: -71.6196749
-Ciudad: Valparaíso
-Región: Región de Valparaíso
-País: Chile

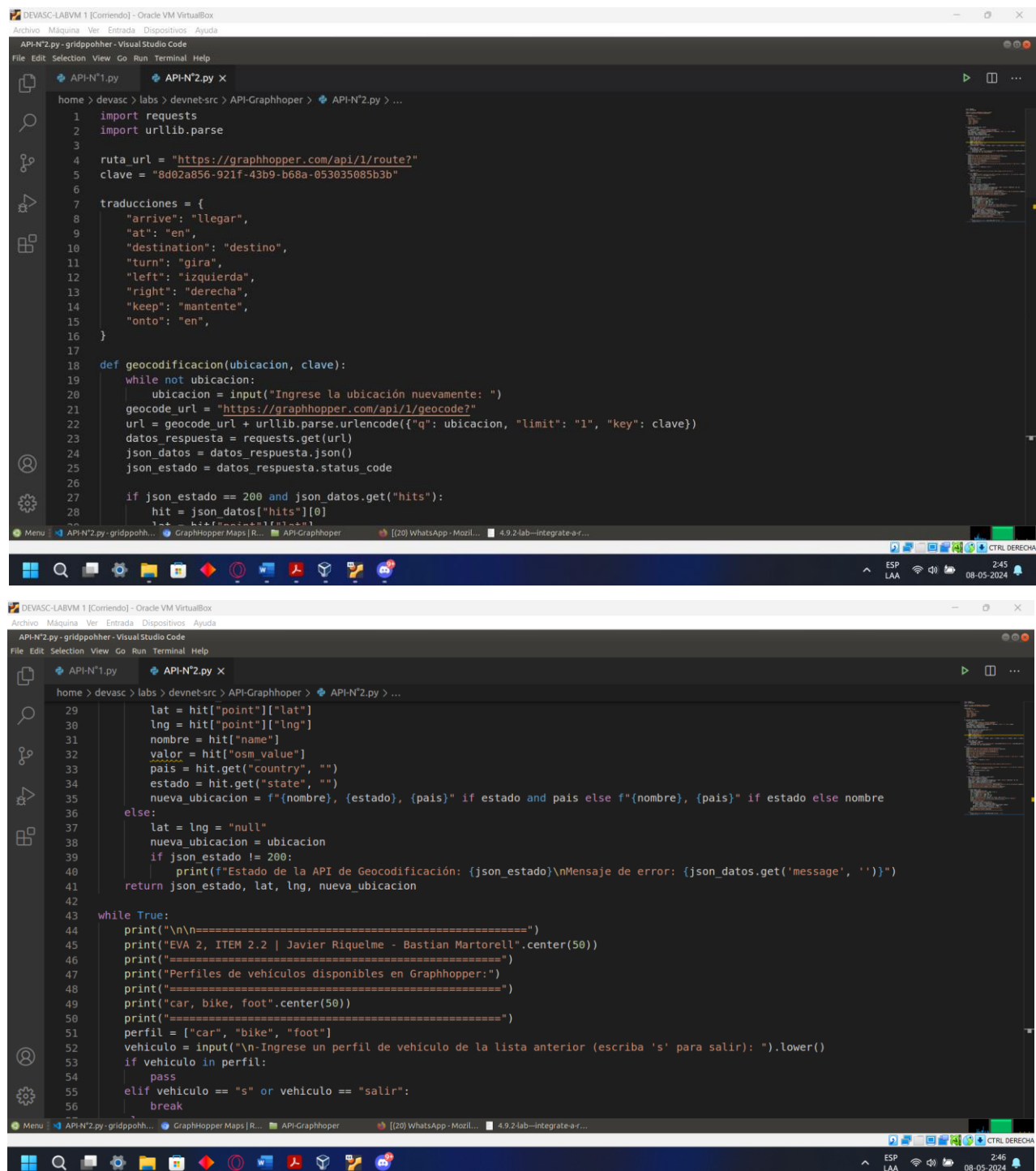
=====

TERMINO DEL PROCESO
=====
  
```

En el ejemplo anterior se muestran las geolocalizaciones de las ciudades de Santiago y Valparaíso, donde se pueden obtener los datos previamente explicados.

3.3 Desarrollo de segunda API

El siguiente código en Python está desarrollado para crear una aplicación que utiliza la API de GraphHopper para calcular la distancia entre ciudades de América Latina. Esta aplicación interactúa con el usuario, permitiéndole ingresar ciudades de origen y destino, elegir el medio de transporte y proporcionar resultados en kilómetros y millas, así como la duración estimada del viaje en horas, minutos y segundos. El código se estructura de manera organizada y eficiente para cumplir con estos requisitos, como se detalla a continuación.



```

home > devasc > labs > devnetsrc > API-Graphhopper > API-Nº2.py > ...
1 import requests
2 import urllib.parse
3
4 ruta_url = "https://graphhopper.com/api/1/route?"
5 clave = "8d02a856-921f-43b9-b68a-053035085b3b"
6
7 traducciones = {
8     "arrive": "llegar",
9     "at": "en",
10    "destination": "destino",
11    "turn": "gira",
12    "left": "izquierda",
13    "right": "derecha",
14    "keep": "mantente",
15    "onto": "en",
16 }
17
18 def geocodificacion(ubicacion, clave):
19     while not ubicacion:
20         ubicacion = input("Ingrese la ubicación nuevamente: ")
21         geocode_url = "https://graphhopper.com/api/1/geocode?"
22         url = geocode_url + urllib.parse.urlencode({"q": ubicacion, "limit": "1", "key": clave})
23         datos_respuesta = requests.get(url)
24         json_datos = datos_respuesta.json()
25         json_estado = datos_respuesta.status_code
26
27         if json_estado == 200 and json_datos.get("hits"):
28             hit = json_datos["hits"][0]
29             lat = hit["point"]["lat"]
30             lng = hit["point"]["lng"]
31             nombre = hit["name"]
32             valor = hit["osm value"]
33             pais = hit.get("country", "")
34             estado = hit.get("state", "")
35             nueva_ubicacion = f"{nombre}, {estado}, {pais}" if estado and pais else f"{nombre}, {pais}" if estado else nombre
36         else:
37             lat = lng = "null"
38             nueva_ubicacion = ubicacion
39             if json_estado != 200:
40                 print(f"Estado de la API de Geocodificación: {json_estado}\nMensaje de error: {json_datos.get('message', '')}")
41             return json_estado, lat, lng, nueva_ubicacion
42
43 while True:
44     print("\n\n=====")
45     print("EVA 2, ITEM 2.2 | Javier Riquelme - Bastian Martorell".center(50))
46     print("=====")
47     print("Perfiles de vehiculos disponibles en Graphhopper:")
48     print("=====")
49     print("car, bike, foot".center(50))
50     print("=====")
51     perfil = ["car", "bike", "foot"]
52     vehiculo = input("\n-Ingrese un perfil de vehiculo de la lista anterior (escriba 's' para salir): ").lower()
53     if vehiculo in perfil:
54         pass
55     elif vehiculo == "s" or vehiculo == "salir":
56         break

```

```

DEVASC-LABVM 1 [Comando] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
API-Nº2.py - gridppohher - VisualStudio Code
File Edit Selection View Go Run Terminal Help
API-Nº1.py API-Nº2.py X
home > devasc > labs > devnet-src > API-Graphhopper > API-Nº2.py > ...
57 else:
58     vehiculo = "car"
59     print("-No se ingresó un perfil de vehículo válido. Usando el perfil de carro.")
60
61     for i in range(2):
62         loc = input("-Ingrese la ubicación de partida (o escriba 's' para salir): " if i == 0 else "-Ingrese el destino (o escriba 's' para salir): ")
63         if loc.lower() == "s" or loc.lower() == "salir":
64             break
65         resultado = geocodificacion(loc, clave)
66         if i == 0:
67             orig = resultado
68         else:
69             dest = resultado
70
71     if all(result[0] == 200 for result in [orig, dest]):
72         op = f"&point={orig[1]}%2C{orig[2]}"
73         dp = f"&point={dest[1]}%2C{dest[2]}"
74         paths_url = ruta_url + urllib.parse.urlencode({"key": clave, "vehicle": vehiculo}) + op + dp
75         paths_estado = requests.get(paths_url).status_code
76         paths_datos = requests.get(paths_url).json()
77         print("\n=====")
78         print(f"\n-Estado de la API de Enrutamiento: {paths_estado}\n-URL de la API de Enrutamiento: {paths_url}")
79         print("\n=====")
80         print(f"\n-Direcciones desde {orig[3]} hasta {dest[3]} en {vehiculo}")
81         print("\n=====")
82
83     if paths_estado == 200:
84         path = paths_datos["paths"][0]

```

```

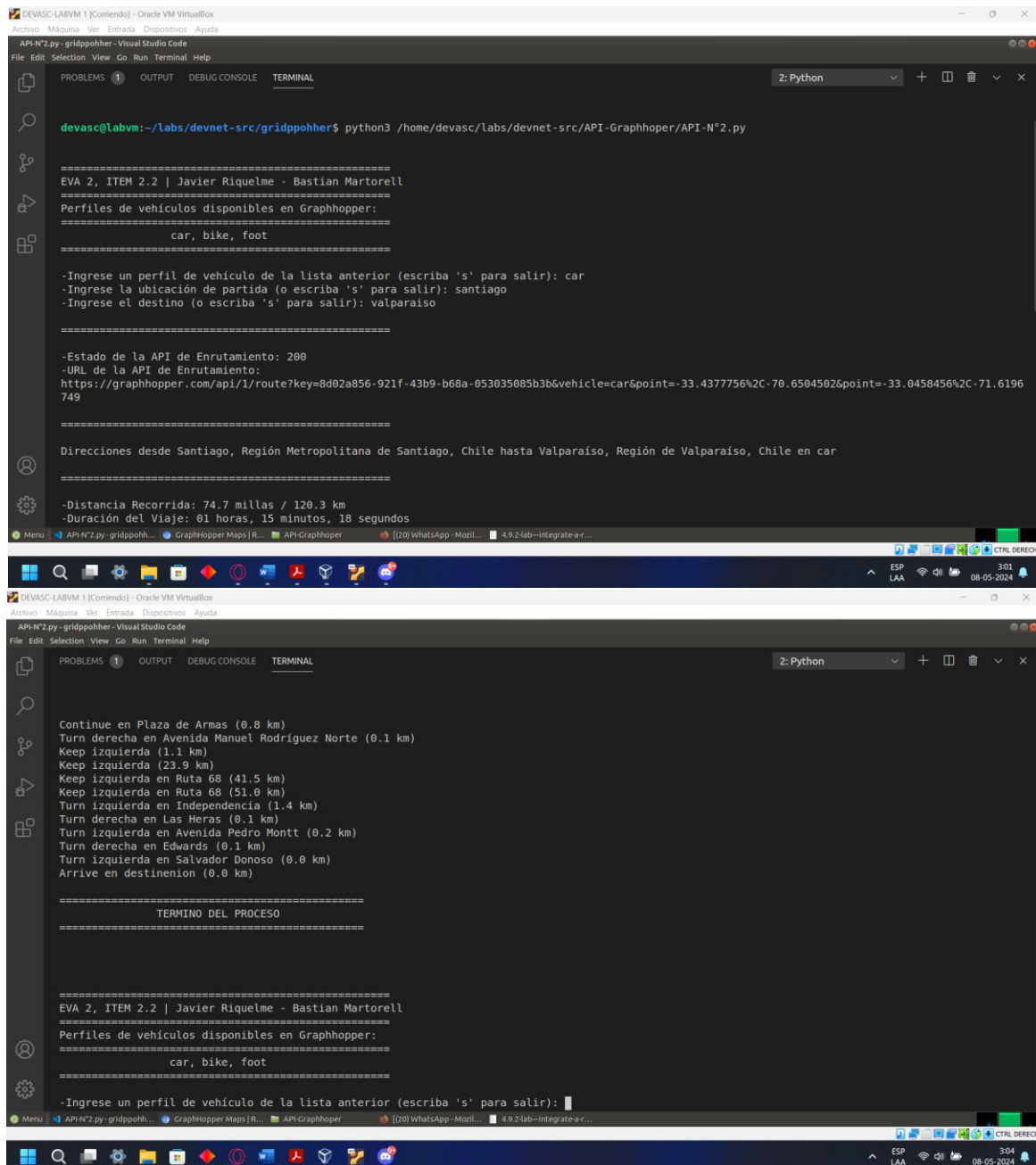
DEVASC-LABVM 1 [Comando] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
API-Nº2.py - gridppohher - VisualStudio Code
File Edit Selection View Go Run Terminal Help
API-Nº1.py API-Nº2.py X
home > devasc > labs > devnet-src > API-Graphhopper > API-Nº2.py > ...
85 millas = round(path["distance"] / 1000 / 1.61, 1)
86 km = round(path["distance"] / 1000, 1)
87 sec = int(path["time"] / 1000 % 60)
88 min = int(path["time"] / 1000 / 60 % 60)
89 hr = int(path["time"] / 1000 / 60 / 60)
90 print(f"-Distancia Recorrida: {millas:.1f} millas / {km:.1f} km")
91 print(f"-Duración del Viaje: {hr:02d} horas, {min:02d} minutos, {sec:02d} segundos")
92 print("\n=====")
93 print("Ruta a seguir:\n")
94 for instruccion in path["instructions"]:
95     instruccion_texto = instruccion["text"]
96     for ingles, espanol in traducciones.items():
97         instruccion_texto = instruccion_texto.replace(ingles, espanol)
98         distancia = round(instruccion["distance"] / 1000, 1)
99         print(f"{instruccion_texto} ({distancia:.1f} km)")
100     print("\n=====")
101     print("TERMINO DEL PROCESO".center(50))
102     print("=====\n\n")
103
104 else:
105     print(f"Mensaje de error: {paths_datos.get('message', '')}")
106     print("\n=====")

```

3.4 Ejemplo de funcionamiento

El código mostrado anteriormente en el desarrollo del segundo programa proporcionará información detallada de un trazado de ruta con 2 ubicaciones. Estas ubicaciones son las mismas que las geolocalizaciones vistas en el código anterior. A diferencia del anterior y como fue explicado en el desarrollo de este código, este mostrará los siguientes datos que el usuario requiera:

- Distancia recorrida
- Duración del viaje
- Ruta a seguir



```

devasc@labvm:~/labs/devnet-src/gridppohher$ python3 /home/devasc/labs/devnet-src/API-Graphhopper/API-N°2.py

=====
EVA 2, ITEM 2.2 | Javier Riquelme - Bastian Martorell
=====
Perfiles de vehículos disponibles en Graphhopper:
=====
car, bike, foot
=====

-Ingrese un perfil de vehículo de la lista anterior (escriba 's' para salir): car
-Ingrese la ubicación de partida (o escriba 's' para salir): santiago
-Ingrese el destino (o escriba 's' para salir): valparaíso
=====

-Estado de la API de Enrutamiento: 200
-URL de la API de Enrutamiento:
https://graphhopper.com/api/1/route?key=8d02a856-921f-43b9-b68a-053035085b3b&vehicle=car&point=-33.4377756%2C-70.6504502&point=-33.0458456%2C-71.6196749
=====

Direcciones desde Santiago, Región Metropolitana de Santiago, Chile hasta Valparaíso, Región de Valparaíso, Chile en car
=====

-Distancia Recorrida: 74.7 millas / 120.3 km
-Duración del Viaje: 01 horas, 15 minutos, 18 segundos

Continue en Plaza de Armas (0.8 km)
Turn derecha en Avenida Manuel Rodríguez Norte (0.1 km)
Keep izquierda (1.1 km)
Keep izquierda (23.9 km)
Keep izquierda en Ruta 68 (41.5 km)
Keep izquierda en Ruta 68 (51.0 km)
Turn izquierda en Independencia (1.4 km)
Turn derecha en Las Heras (0.1 km)
Turn izquierda en Avenida Pedro Montt (0.2 km)
Turn derecha en Edwards (0.1 km)
Turn izquierda en Salvador Donoso (0.0 km)
Arrive en destinenion (0.0 km)

=====
TERMINO DEL PROCESO
=====

EVA 2, ITEM 2.2 | Javier Riquelme - Bastian Martorell
=====
Perfiles de vehículos disponibles en Graphhopper:
=====
car, bike, foot
=====

-Ingrese un perfil de vehículo de la lista anterior (escriba 's' para salir):

```

4 Conclusión

En este informe, se ha explorado el proceso de creación y desarrollo de un entorno DevNet, abarcando tanto la configuración del repositorio Git como la implementación de dos aplicaciones prácticas en Python. La primera aplicación permite obtener la geolocalización precisa de ubicaciones ingresadas por el usuario, brindando detalles como latitud, longitud, ciudad, región y país. La segunda aplicación, por su parte, utiliza la API de GraphHopper para calcular la distancia entre ciudades de América Latina, ofreciendo resultados en kilómetros y millas, así como la duración estimada del viaje y una narrativa detallada de la ruta.

A lo largo del proceso, se han aplicado técnicas y herramientas esenciales para el desarrollo de software, como el control de versiones con Git, la creación de ramas, la fusión de cambios y la interacción con APIs externas. Estas prácticas permiten un flujo de trabajo más eficiente, una mayor colaboración en equipo y la capacidad de integrar funcionalidades adicionales de manera fluida.

En resumen, este informe demuestra la importancia de contar con un entorno de desarrollo sólido y versátil, así como la capacidad de crear aplicaciones prácticas y útiles que resuelvan problemas específicos. Las soluciones presentadas brindan una experiencia mejorada al usuario, al tiempo que demuestran la versatilidad y el potencial del lenguaje de programación Python en combinación con APIs externas. Esta combinación de habilidades técnicas y enfoque en la experiencia del usuario sienta las bases para el desarrollo de proyectos más complejos y escalables en el futuro.