



Universidad
Tecmilenio®

Actividad 2

Profesional asociado en desarrollo de Software

**Fundamentos de DeVops
CC.PTTI2209FLE.201.202430.325**

**Docente
Carlos Morales Crispín**

**Alumno
Eliu Geovanni Luna Valdez**

**Fecha de entrega
3 de noviembre del 2024**

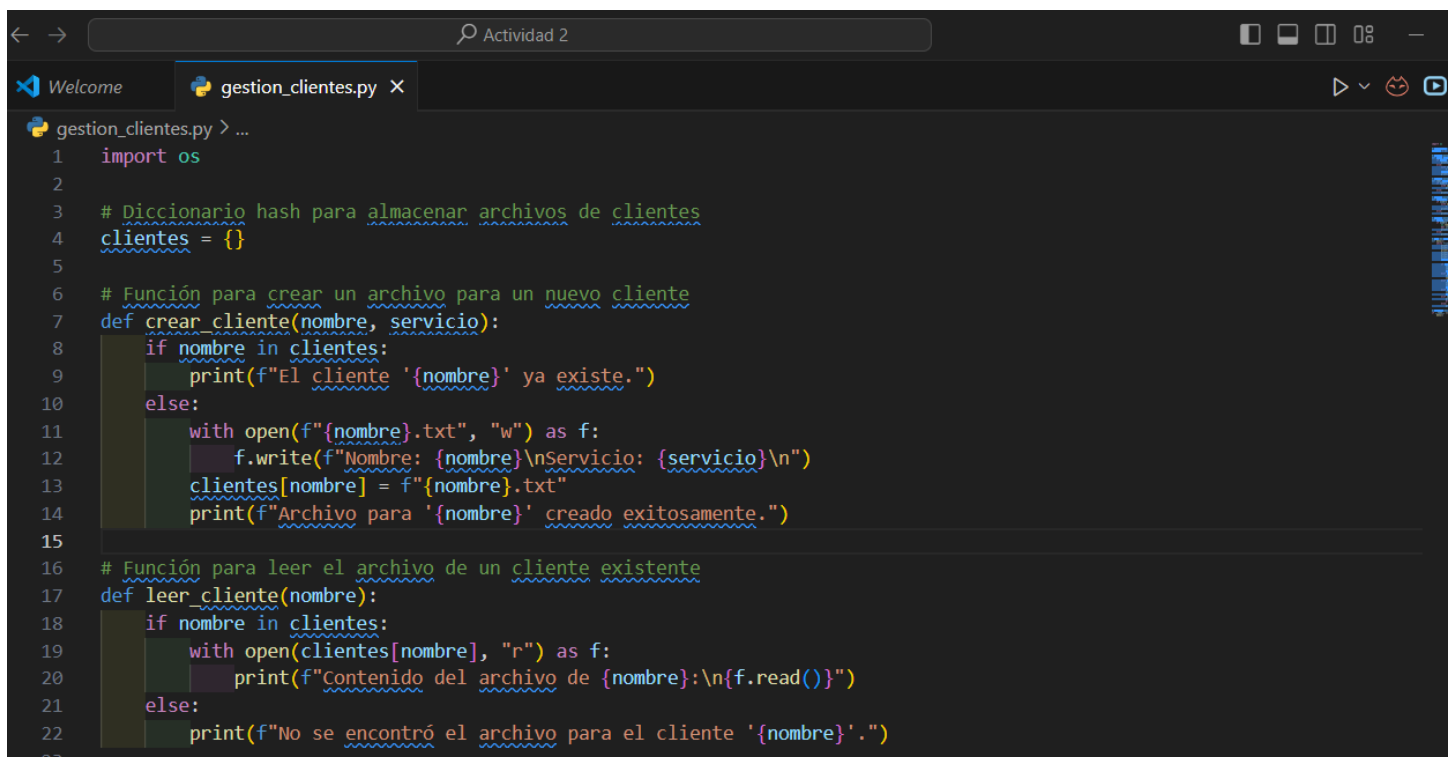
Antecedentes

La empresa Axanet guarda en un archivo los datos de cada cliente; sin embargo, ahora te pide que desarrolles una solución basada en Python para generar los archivos de los nuevos clientes y, a su vez, para revisar la información de los antiguos. Como pudiste observar, la empresa ha crecido mucho en los últimos años y, por tal motivo, requieren triplicar el número de clientes cada mes, así como mantener el servicio de los ya recurrentes.

En este sentido, recuerda que el servicio de la empresa funciona de la siguiente manera: una persona o negocio lo solicita, así que se verifica si se trata de un cliente nuevo o recurrente. En el primer caso, se genera un nuevo archivo con la información del comprador, junto con la descripción del servicio solicitado; en cambio, para el segundo escenario, se busca el archivo y únicamente se agrega la descripción de la nueva solicitud. Cualquier usuario del sistema debe ser capaz de acceder al archivo del cliente, ya sea introduciendo directamente el nombre o visualizando una lista donde se albergan los de todos.

1.- Crea un programa en Python que permita leer el archivo de un cliente existente, así como modificar su información o borrarlo; asimismo, debe ser capaz de crear un archivo para un nuevo comprador.

En este caso se realiza una pequeña modificación en el código para poder guardar los registros de los usuarios y también para poder ver, modificar y eliminar los registros.



```

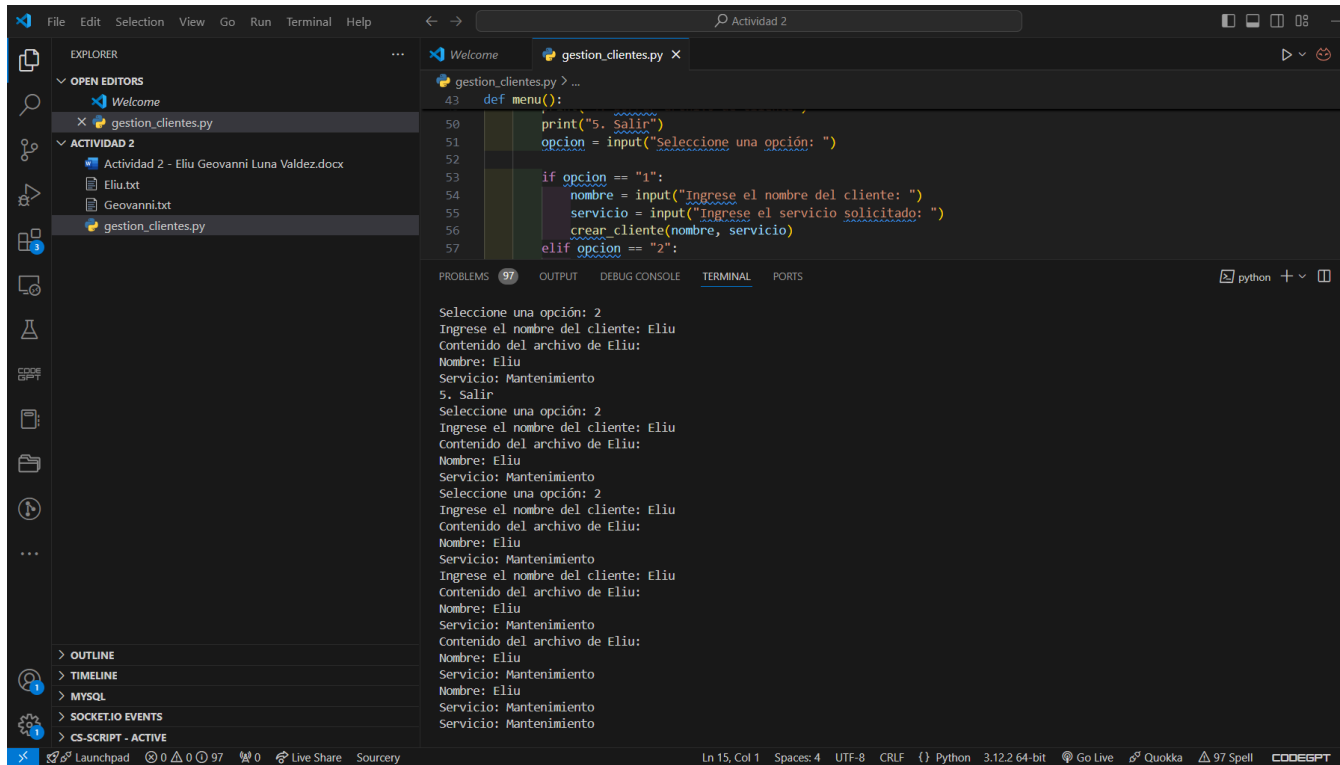
1  import os
2
3  # Diccionario hash para almacenar archivos de clientes
4  clientes = {}
5
6  # Función para crear un archivo para un nuevo cliente
7  def crear_cliente(nombre, servicio):
8      if nombre in clientes:
9          print(f"El cliente '{nombre}' ya existe.")
10     else:
11         with open(f"{nombre}.txt", "w") as f:
12             f.write(f"Nombre: {nombre}\nServicio: {servicio}\n")
13             clientes[nombre] = f"{nombre}.txt"
14             print(f"Archivo para '{nombre}' creado exitosamente.")
15
16 # Función para leer el archivo de un cliente existente
17 def leer_cliente(nombre):
18     if nombre in clientes:
19         with open(clientes[nombre], "r") as f:
20             print(f"Contenido del archivo de {nombre}: \n{f.read()}")
21     else:
22         print(f"No se encontró el archivo para el cliente '{nombre}'.")
23

```

```
← → Actividad 2
Welcome gestion_clientes.py X
gestion_clientes.py > ...
23
24 # Función para modificar la información de un cliente existente
25 def modificar_cliente(nombre, nuevo_servicio):
26     if nombre in clientes:
27         with open(clientes[nombre], "a") as f:
28             f.write(f"Nuevo servicio: {nuevo_servicio}\n")
29             print(f"Archivo de '{nombre}' actualizado con el nuevo servicio.")
30     else:
31         print(f"No se encontró el archivo para el cliente '{nombre}'.")
32
33 # Función para borrar el archivo de un cliente
34 def borrar_cliente(nombre):
35     if nombre in clientes:
36         os.remove(clientes[nombre])
37         del clientes[nombre]
38         print(f"Archivo de '{nombre}' eliminado exitosamente.")
39     else:
40         print(f"No se encontró el archivo para el cliente '{nombre}'.")
41
42 # Menú de opciones para interactuar con el programa
43 def menu():
44     while True:
45         print("\n--- Gestión de Clientes ---")
46         print("1. Crear un nuevo cliente")
47         print("2. Leer archivo de cliente")
48         print("3. Modificar cliente existente")
49         print("4. Borrar archivo de cliente")
50         print("5. Salir")
51         opcion = input("Seleccione una opción: ")
```

```
← → Actividad 2
Welcome gestion_clientes.py X
gestion_clientes.py > ...
43 def menu():
48     print("3. Modificar cliente existente")
49     print("4. Borrar archivo de cliente")
50     print("5. Salir")
51     opcion = input("Seleccione una opción: ")
52
53     if opcion == "1":
54         nombre = input("Ingrese el nombre del cliente: ")
55         servicio = input("Ingrese el servicio solicitado: ")
56         crear_cliente(nombre, servicio)
57     elif opcion == "2":
58         nombre = input("Ingrese el nombre del cliente: ")
59         leer_cliente(nombre)
60     elif opcion == "3":
61         nombre = input("Ingrese el nombre del cliente: ")
62         nuevo_servicio = input("Ingrese el nuevo servicio solicitado: ")
63         modificar_cliente(nombre, nuevo_servicio)
64     elif opcion == "4":
65         nombre = input("Ingrese el nombre del cliente: ")
66         borrar_cliente(nombre)
67     elif opcion == "5":
68         print("Saliendo del programa.")
69         break
70     else:
71         print("Opción no válida, intente de nuevo.")
72
73 # Ejecutar el menú
74 if __name__ == "__main__":
75     menu()
76
```

Se corre en la terminal y este funciona, se crea un perfil y se guarda automáticamente, también se puede modificar y eliminar.



```
def menu():
    print("5. Salir")
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        nombre = input("Ingrese el nombre del cliente: ")
        servicio = input("Ingrese el servicio solicitado: ")
        crear_cliente(nombre, servicio)
    elif opcion == "2":
        print("Opción no válida")
```

Selecione una opción: 2
Ingrese el nombre del cliente: Eliu
Contenido del archivo de Eliu:
Nombre: Eliu
Servicio: Mantenimiento
5. Salir
Selecione una opción: 2
Ingrese el nombre del cliente: Eliu
Contenido del archivo de Eliu:
Nombre: Eliu
Servicio: Mantenimiento
Selecione una opción: 2
Ingrese el nombre del cliente: Eliu
Contenido del archivo de Eliu:
Nombre: Eliu
Servicio: Mantenimiento
Ingrese el nombre del cliente: Eliu
Contenido del archivo de Eliu:
Nombre: Eliu
Servicio: Mantenimiento
Contenido del archivo de Eliu:
Nombre: Eliu
Servicio: Mantenimiento
Nombre: Eliu
Servicio: Mantenimiento
Servicio: Mantenimiento

2.- Utiliza tablas hash para asociar el nombre del cliente con su archivo y para crear un diccionario.

En el punto 2 se usó un diccionario de Python como una tabla hash para asociar el nombre de cada cliente con su archivo correspondiente, de este modo, podremos acceder rápidamente a los archivos de clientes usando sus nombres como claves.

Esta tabla hash permitirá la búsqueda eficiente de los datos del cliente sin necesidad de recorrer manualmente todos los archivos.

Diccionario clientes: Este diccionario almacenará el nombre de cada cliente como clave y el nombre del archivo como valor, así si queremos acceder al archivo de un cliente específico, solo necesitamos buscar su nombre en el diccionario.

El uso de la tabla hash se especifica cómo funciona

- Agregar un cliente al diccionario cuando se crea un nuevo archivo.
- Consultar el diccionario para encontrar el archivo de un cliente cuando se quiere leer o modificar su información.
- Eliminar una entrada del diccionario al borrar el archivo de un cliente.

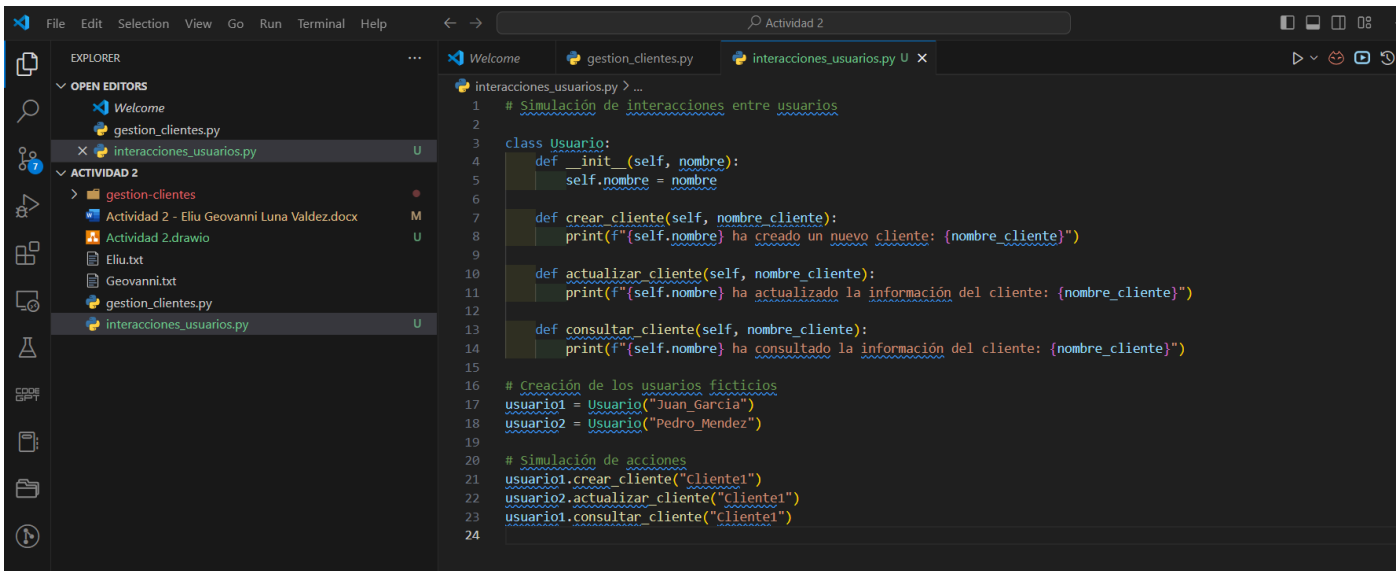
3.- Crea un grupo de trabajo con, al menos, dos usuarios (pueden ser ficticios), de tal manera que te permita simular la interacción con dichos miembros.

Se crea usuarios ficticios para simular interacción. Ejemplo:

```
usuarios = {
```

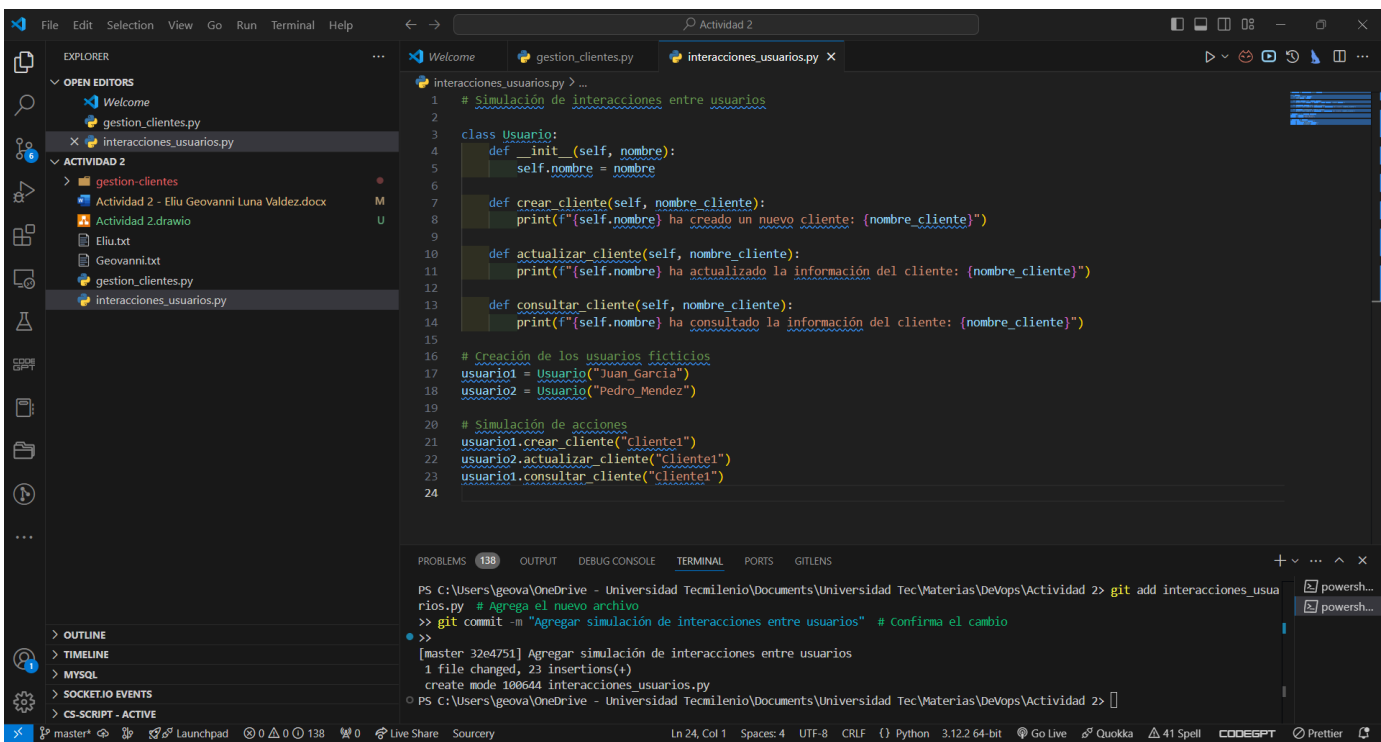
```
    "Usuario1": "Juan_Garcia",
```

```
    "Usuario2": "Pedro_Mendez"
```



```
1 # Simulación de interacciones entre usuarios
2
3 class Usuario:
4     def __init__(self, nombre):
5         self.nombre = nombre
6
7     def crear_cliente(self, nombre_cliente):
8         print(f"{self.nombre} ha creado un nuevo cliente: {nombre_cliente}")
9
10    def actualizar_cliente(self, nombre_cliente):
11        print(f"{self.nombre} ha actualizado la información del cliente: {nombre_cliente}")
12
13    def consultar_cliente(self, nombre_cliente):
14        print(f"{self.nombre} ha consultado la información del cliente: {nombre_cliente}")
15
16 # Creación de los usuarios ficticios
17 usuario1 = Usuario("Juan_Garcia")
18 usuario2 = Usuario("Pedro_Mendez")
19
20 # Simulación de acciones
21 usuario1.crear_cliente("cliente1")
22 usuario2.actualizar_cliente("cliente1")
23 usuario1.consultar_cliente("cliente1")
24
```

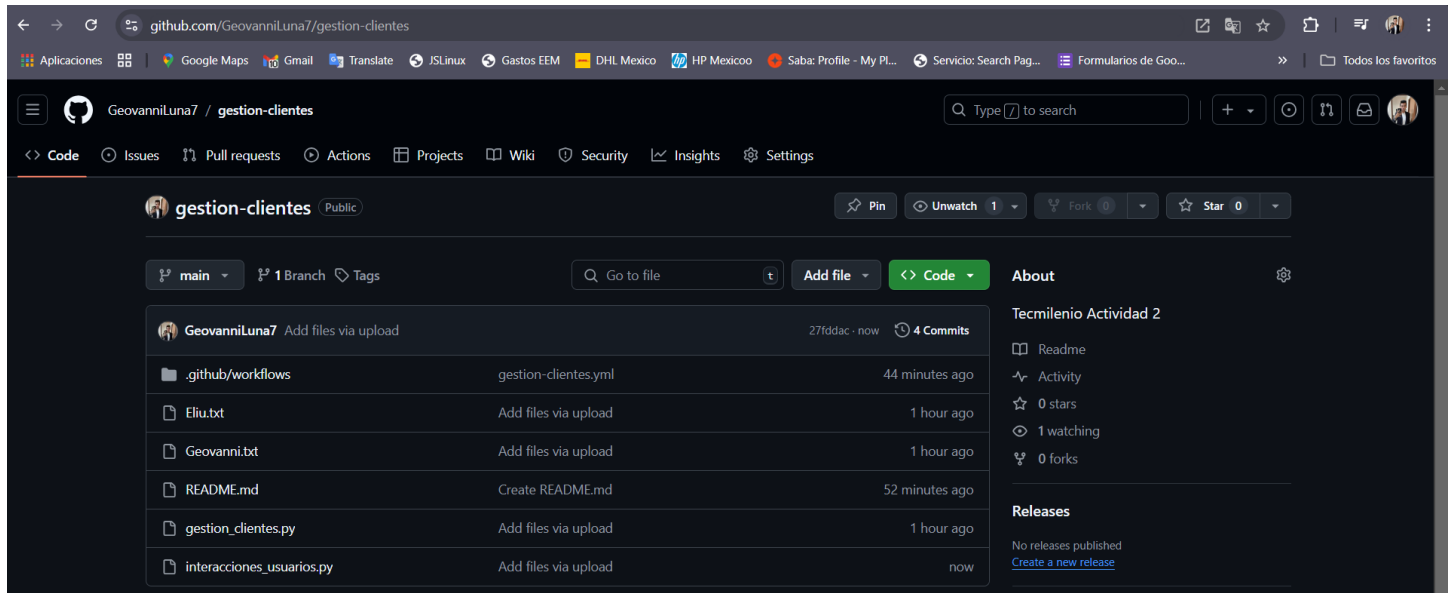
Se añade los cambios en Git.



```
1 # Simulación de interacciones entre usuarios
2
3 class Usuario:
4     def __init__(self, nombre):
5         self.nombre = nombre
6
7     def crear_cliente(self, nombre_cliente):
8         print(f"{self.nombre} ha creado un nuevo cliente: {nombre_cliente}")
9
10    def actualizar_cliente(self, nombre_cliente):
11        print(f"{self.nombre} ha actualizado la información del cliente: {nombre_cliente}")
12
13    def consultar_cliente(self, nombre_cliente):
14        print(f"{self.nombre} ha consultado la información del cliente: {nombre_cliente}")
15
16 # Creación de los usuarios ficticios
17 usuario1 = Usuario("Juan_Garcia")
18 usuario2 = Usuario("Pedro_Mendez")
19
20 # Simulación de acciones
21 usuario1.crear_cliente("cliente1")
22 usuario2.actualizar_cliente("cliente1")
23 usuario1.consultar_cliente("cliente1")
24
```

```
PS C:\Users\geova\OneDrive - Universidad Tecnológico de Costa Rica\Documents\Universidad Tec\Materias\DeVops\Actividad 2> git add interacciones_usua
rios.py # Agrega el nuevo archivo
>> git commit -m "Agregar simulación de interacciones entre usuarios" # Confirma el cambio
[master 32e4751] Agregar simulación de interacciones entre usuarios
1 file changed, 23 insertions(+)
create mode 100644 interacciones_usuarios.py
PS C:\Users\geova\OneDrive - Universidad Tecnológico de Costa Rica\Documents\Universidad Tec\Materias\DeVops\Actividad 2>
```

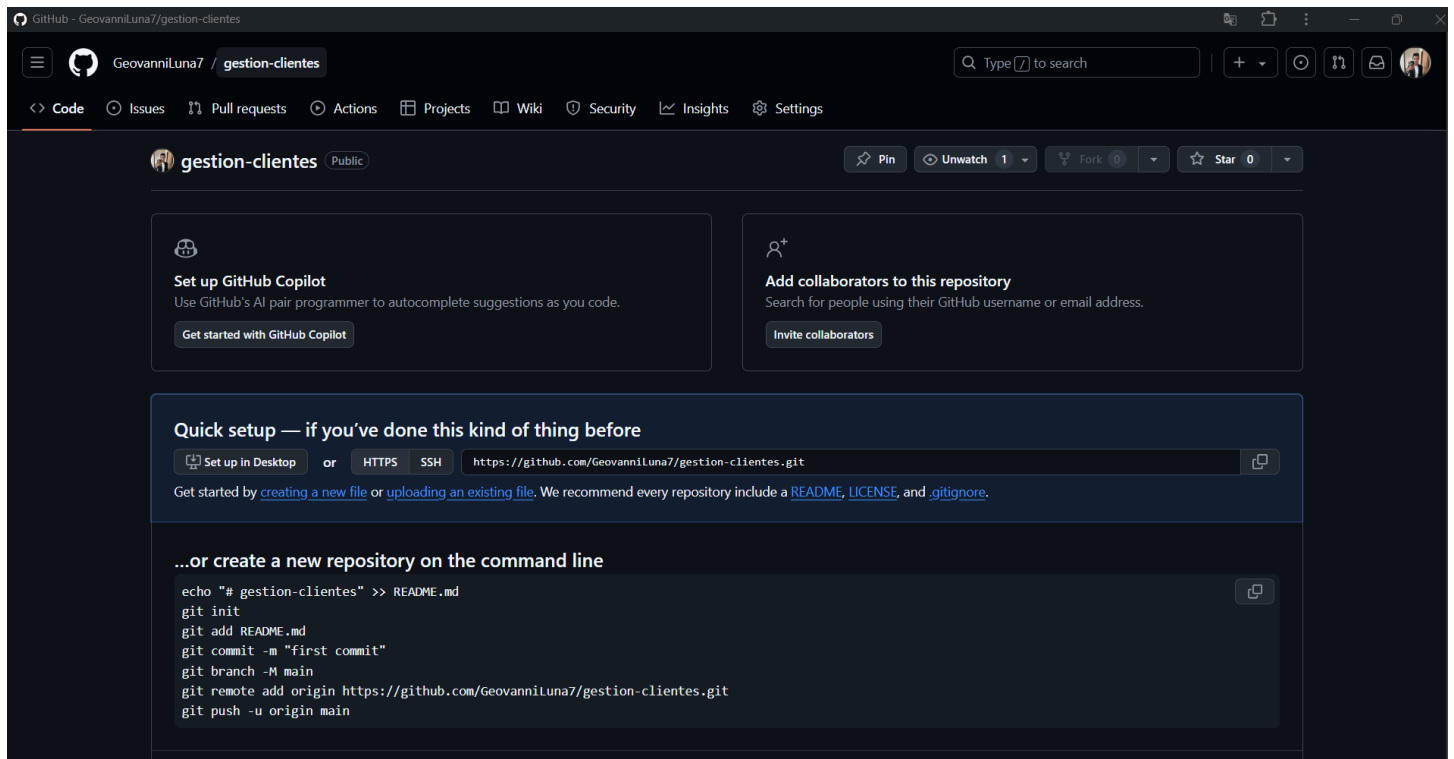
Se verifica y el archivo ya está en Git



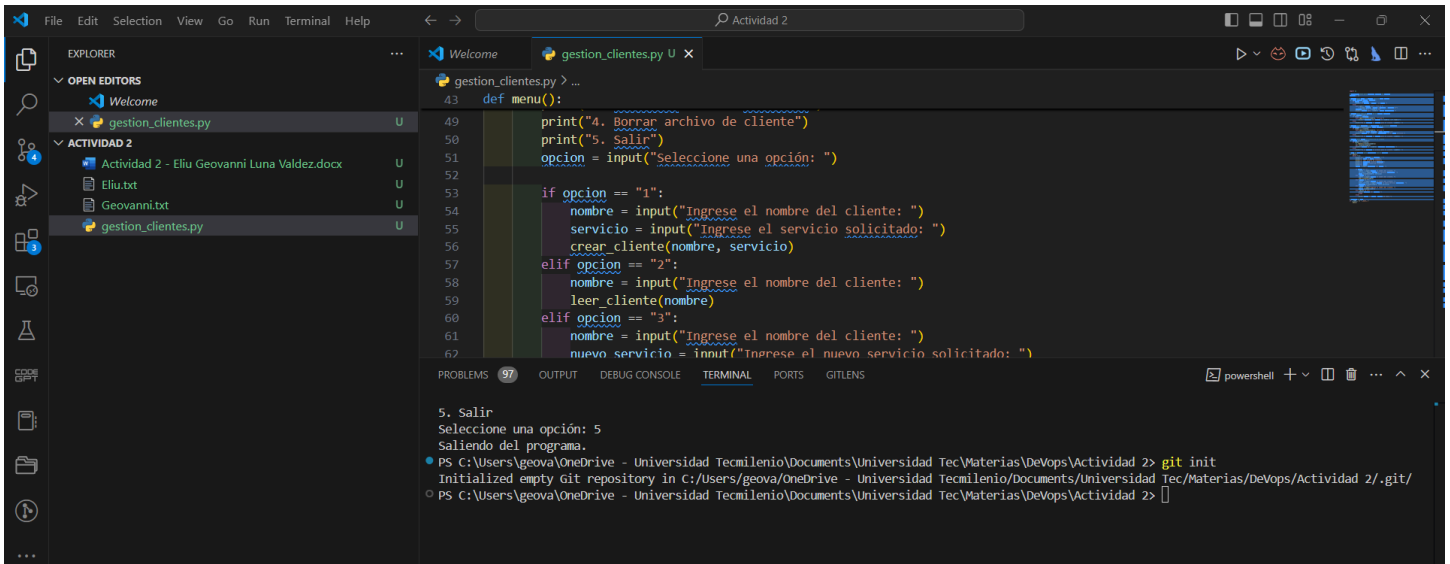
Estos usuarios pueden simular consultas o modificaciones en los archivos de clientes y/o con GitHub Actions, podemos ver registros de sus actividades.

4.- Crea un proyecto en GitHub donde colaboren los usuarios del punto anterior.

Se crea un nuevo repositorio.



Se abre Visual Studio Code, se ejecuta código y se abre la terminal, se inicializa Git en la terminal con git init.

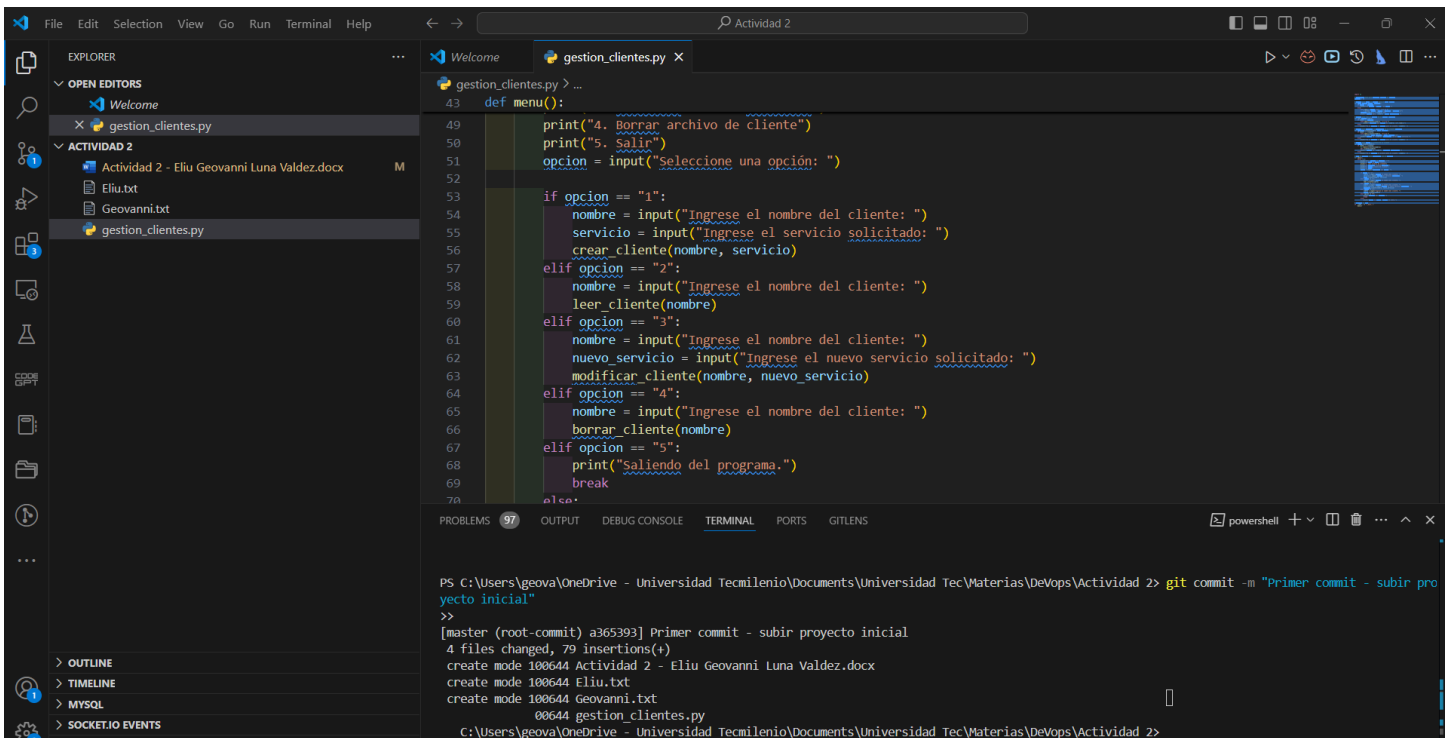


```
def menu():
    print("4. Borrar archivo de cliente")
    print("5. Salir")
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        nombre = input("Ingrese el nombre del cliente: ")
        servicio = input("Ingrese el servicio solicitado: ")
        crear_cliente(nombre, servicio)
    elif opcion == "2":
        nombre = input("Ingrese el nombre del cliente: ")
        leer_cliente(nombre)
    elif opcion == "3":
        nombre = input("Ingrese el nombre del cliente: ")
        nuevo_servicio = input("Ingrese el nuevo servicio solicitado: ")

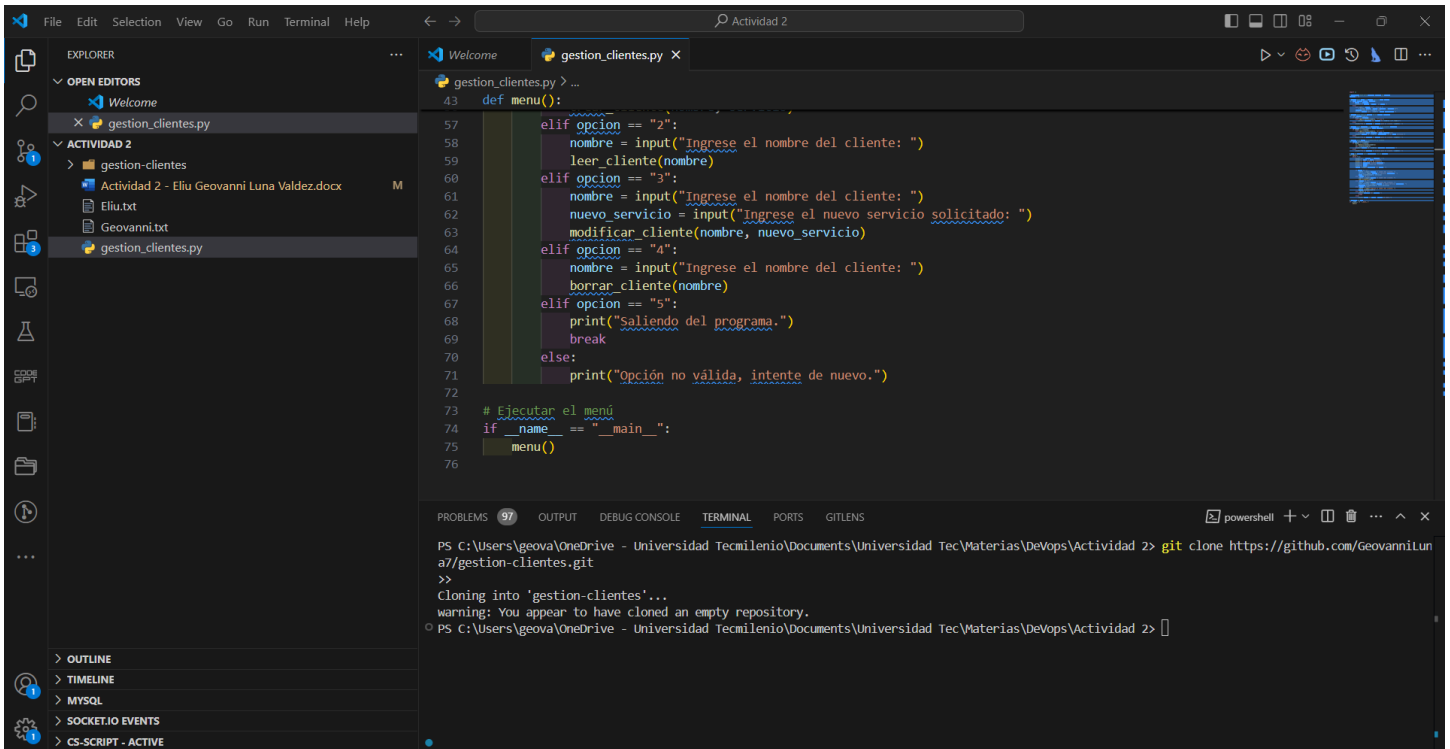
5. Salir
Seleccione una opción: 5
Saliendo del programa.
PS C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad TecMaterias\DeVops\Actividad 2> git init
Initialized empty Git repository in C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad TecMaterias\DeVops\Actividad 2\.git\
PS C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad TecMaterias\DeVops\Actividad 2>
```

Crea un commit



```
PS C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad TecMaterias\DeVops\Actividad 2> git commit -m "Primer commit - subir proyecto inicial"
>>
[master (root-commit) a365393] Primer commit - subir proyecto inicial
4 files changed, 79 insertions(+)
create mode 100644 Actividad 2 - Eliu Geovanni Luna Valdez.docx
create mode 100644 Eliu.txt
create mode 100644 Geovanni.txt
00644 gestion_clientes.py
C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad TecMaterias\DeVops\Actividad 2>
```

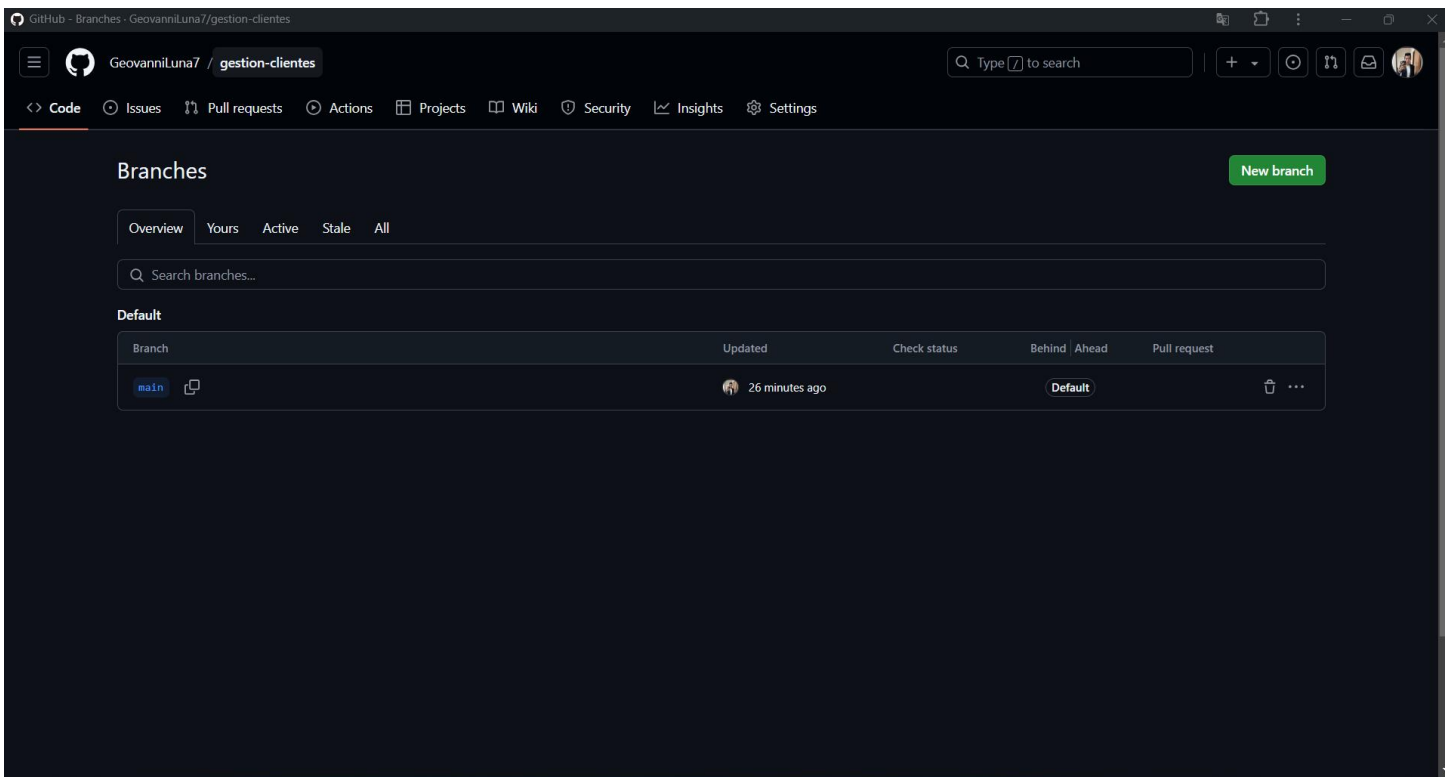
Se clona el repositorio desde Hub con `git clone https://github.com/GeovanniLuna7/gestion-clientes.git`, esto descargará el repositorio en una nueva carpeta llamada `gestion-clientes`



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project structure with a folder named 'ACTIVIDAD 2' containing files like 'Actividad 2 - Eliu Geovanni Luna Valdez.docx', 'Eliu.txt', 'Geovanni.txt', and 'gestion_clientes.py'. The main editor displays the code for 'gestion_clientes.py', which includes a menu function with options for adding, modifying, and deleting clients, and a main execution block. The terminal at the bottom shows the command to clone the repository from GitHub, which was executed successfully.

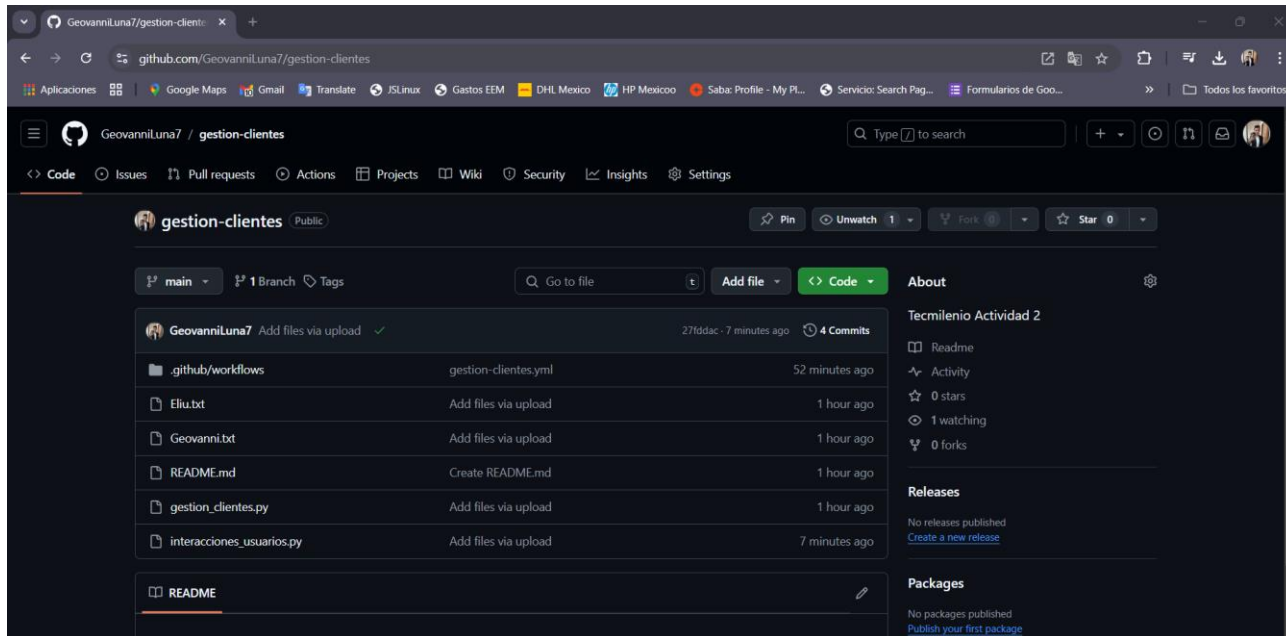
```
def menu():
    43
    57     elif opcion == "2":
    58         nombre = input("Ingrese el nombre del cliente: ")
    59         leer_cliente(nombre)
    60     elif opcion == "3":
    61         nombre = input("Ingrese el nombre del cliente: ")
    62         nuevo_servicio = input("Ingrese el nuevo servicio solicitado: ")
    63         modificar_cliente(nombre, nuevo_servicio)
    64     elif opcion == "4":
    65         nombre = input("Ingrese el nombre del cliente: ")
    66         borrar_cliente(nombre)
    67     elif opcion == "5":
    68         print("Saliendo del programa.")
    69         break
    70     else:
    71         print("Opción no válida, intente de nuevo.")
    72
    73 # Ejecutar el menú
    74 if __name__ == "__main__":
    75     menu()
    76
```

```
PS C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad Tec\Materias\DevOps\Actividad 2> git clone https://github.com/GeovanniLuna7/gestion-clientes.git
Cloning into 'gestion-clientes'...
warning: You appear to have cloned an empty repository.
PS C:\Users\geova\OneDrive - Universidad Tecmilenio\Documents\Universidad Tec\Materias\DevOps\Actividad 2>
```



5.- Aloja tu aplicación en GitHub

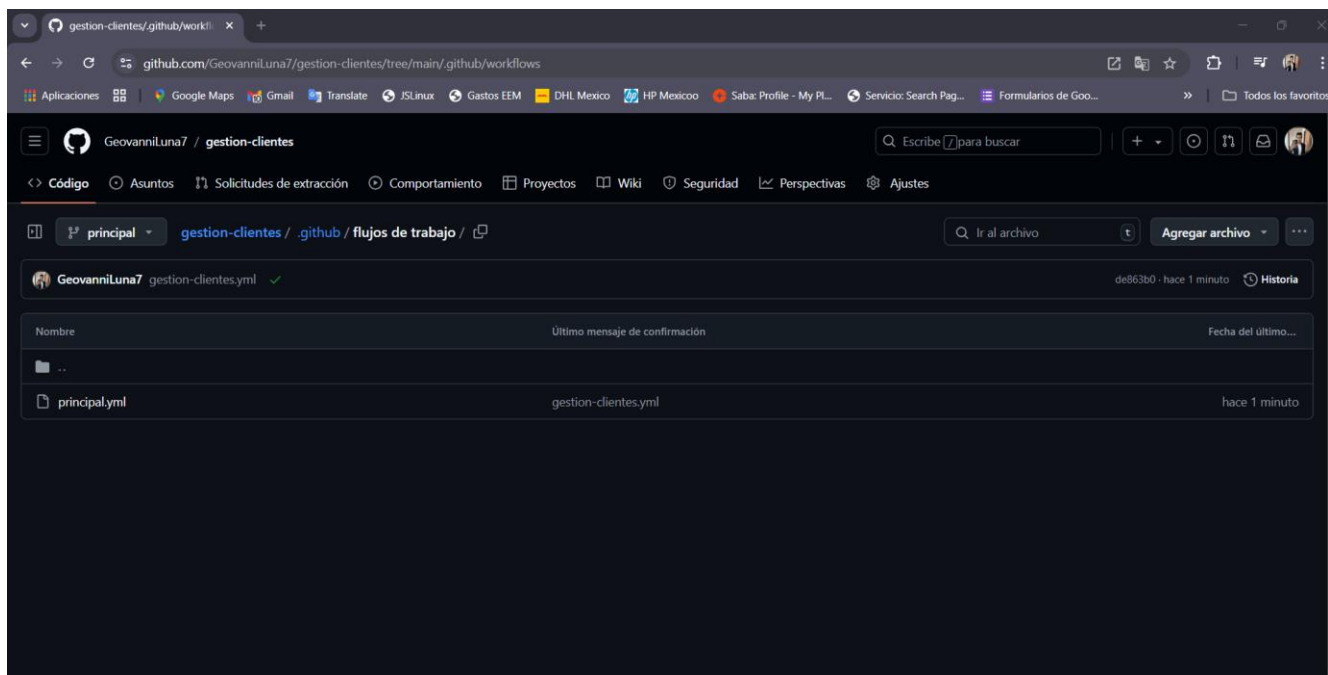
Se sube la aplicación en Git en publico, se anexa link <https://github.com/GeovanniLuna7/gestion-clientes.git>



6.- Genera, al menos, tres flujos en GitHub Actions para gestionar lo siguiente:

Creación de un nuevo cliente: debes mostrar un mensaje que indique dicha acción a cada miembro del equipo.
Actualización de un cliente recurrente: debes mostrar un mensaje que indique dicha acción a cada miembro del equipo.
Consulta de un cliente: debes mostrar un mensaje que indique dicha acción a cada miembro del equipo.

Se integra Action y se crea un archivo gestion-clientes.yml



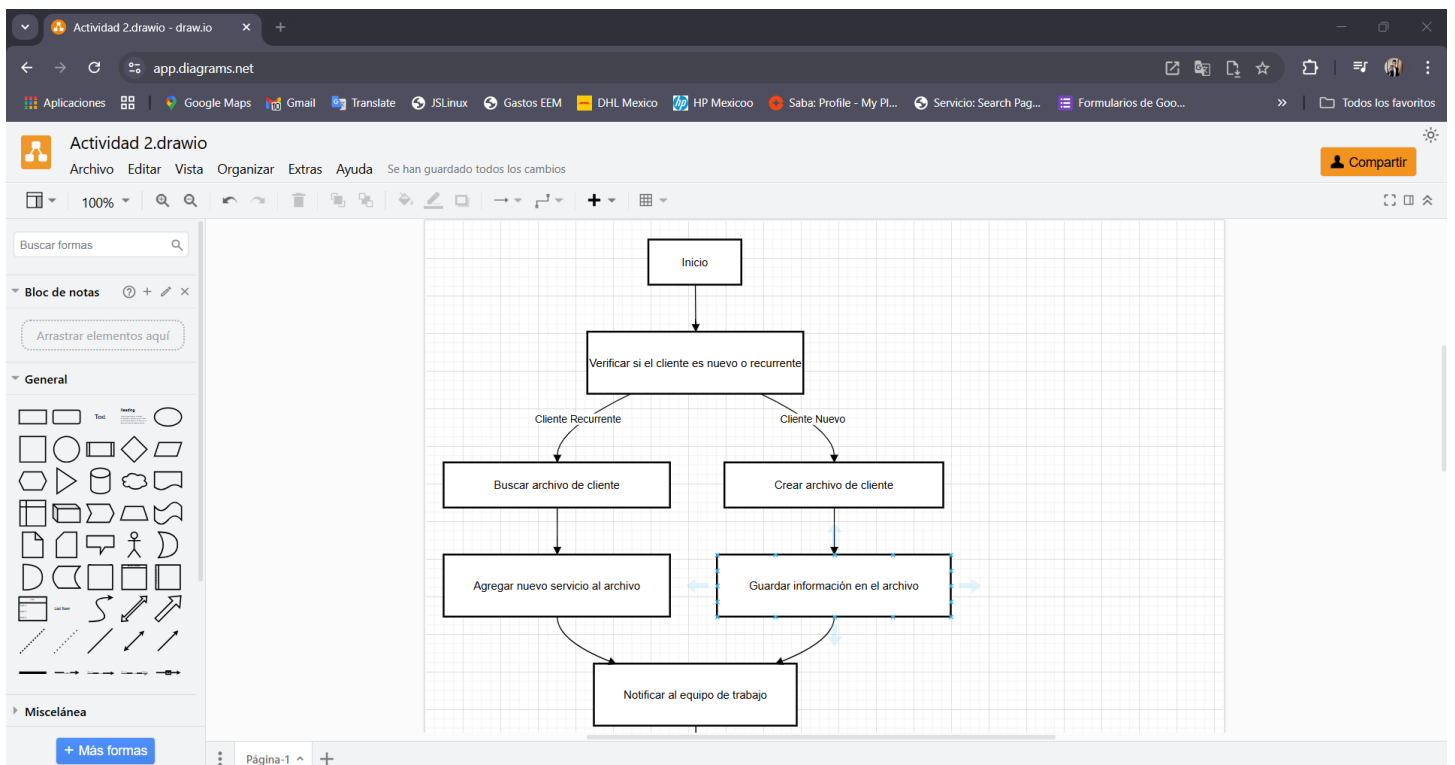
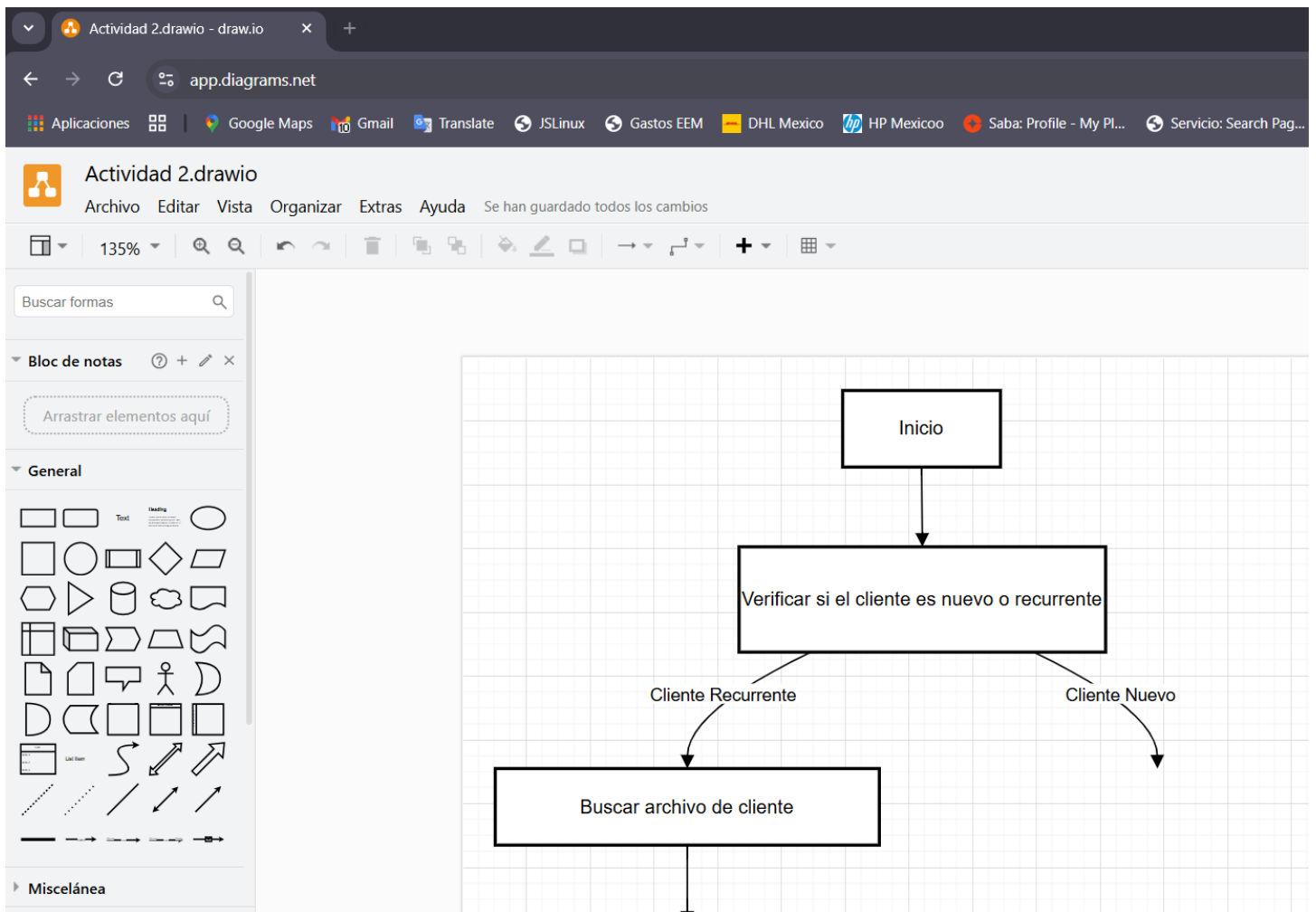
The screenshot shows a web browser displaying a GitHub repository page for 'gestion-clientes'. The URL is 'github.com/GeovanniLuna7/gestion-clientes/blob/main/.github/workflows/principal.yml'. The page shows the workflow file 'principal.yml' with the following content:

```
1 nombre:Gestión de clientes
2
3 en:
4   #Estos flujos se ejecutarán en eventos push y pull_request
5   empujar:
6     sucursales:
7       -principal
8   solicitud_de_extracción:
9     sucursales:
10      -principal
11
12 trabajos:
13   crear_cliente:
14     sigue_corriendo:Ubuntu más reciente
15     si:github.nombre_del_evento == 'push'
16     pasos:
17       -nombre:Código de pago
18         usos:acciones/checkout@v2
19
20     -nombre:Crear nuevo cliente
```

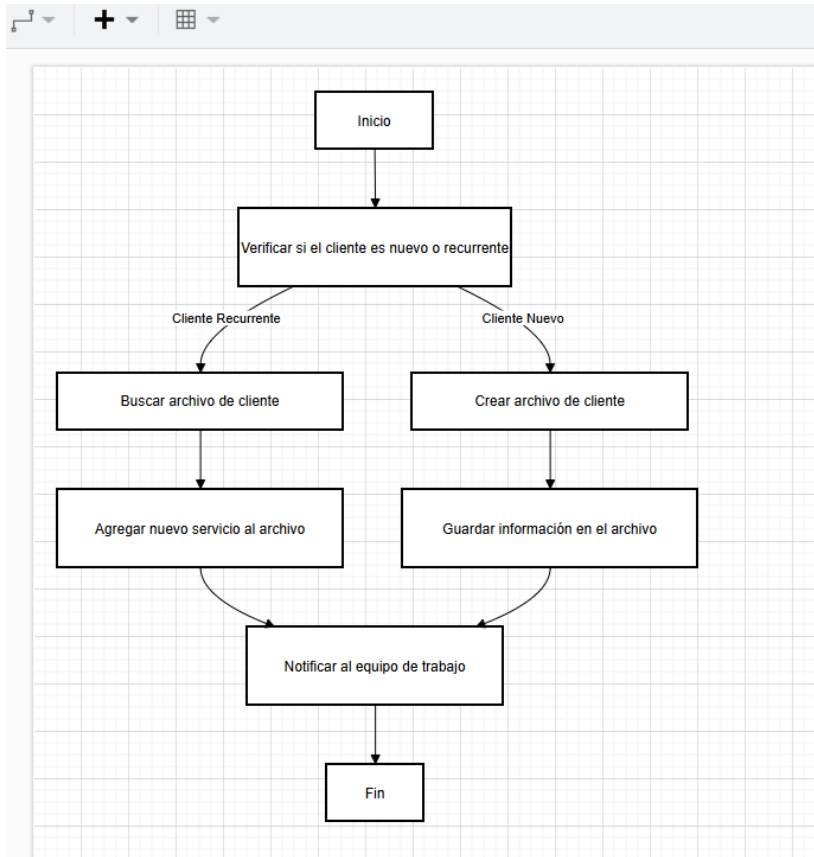
En este caso el flujo de trabajo

- 1.- Nombre del flujo: Se llama Gestion de Clientes.
- 2.- Eventos que lo activan: Se activa en los eventos push y pull_request en la rama main.
- 3.- Jobs (trabajos):
 - crear_cliente: Este trabajo se ejecuta cuando se hace un push y simula la creación de un nuevo cliente. Se envía un mensaje de notificación a los miembros del equipo.
 - actualizar_cliente: Este trabajo simula la actualización de un cliente recurrente y envía una notificación.
 - consultar_cliente: Este trabajo simula la consulta de un cliente y notifica al equipo.
- 7.- Genera un documento en Word con el diagrama de flujo y el pseudocódigo de tu aplicación; además, explica de forma detallada qué fue exactamente lo que realizaste en cada uno de los puntos anteriores. No olvides compartir en el documento la liga de tu proyecto.

A lo largo del documento se comparte el código y la explicación de cada uno en este paso se genera el diagrama de flujo con la herramienta draw.io



Resultado final del diagrama, Se comparte link <https://drive.google.com/file/d/1wajYAYekcrpT3-jXewdrfXUPkO2w4xZS/view?usp=sharing>



Trabajos consultados

ellibrodepython. 2024. Hash en Python
<https://ellibrodepython.com/hash-python>

YouTube (Enero 2024) Fundamentos de operaciones y desarrollo DevOps semana 3
<https://www.youtube.com/watch?v=PdUrjVYwewQ>

YouTube - tecmilenio.mx (ene 2024) Fundamentos de operaciones y desarrollo DevOps semana 1
https://www.youtube.com/watch?v=DTTIF6_BqL0

learn.microsoft.com (mar 2024) Creación de un grupo de administración con Python
<https://learn.microsoft.com/es-es/azure/governance/management-groups/create-management-group-python>

docs.github.com (2024) Inicio rápido para compilar aplicaciones de GitHub
<https://docs.github.com/es/apps/creating-github-apps/writing-code-for-a-github-app/quickstart>

docs.github.com (2024) Flujos de trabajo de Git
<https://docs.github.com/es/get-started/getting-started-with-git/git-workflows>

atlassian.com (2022). Comparar flujos de trabajo de Git: lo que debes saber
<https://www.atlassian.com/es/git/tutorials/comparing-workflows>