

PRACTICA DE CAMPO 2

TÉCNICAS DE PROGRAM.ORIE. OBJ

Profesor:

- Martin Eduardo Torres Rodríguez

Alumno:

- Miguel Jair Roa Navarro - N00459166

2025

Practica de Campo Semana 2

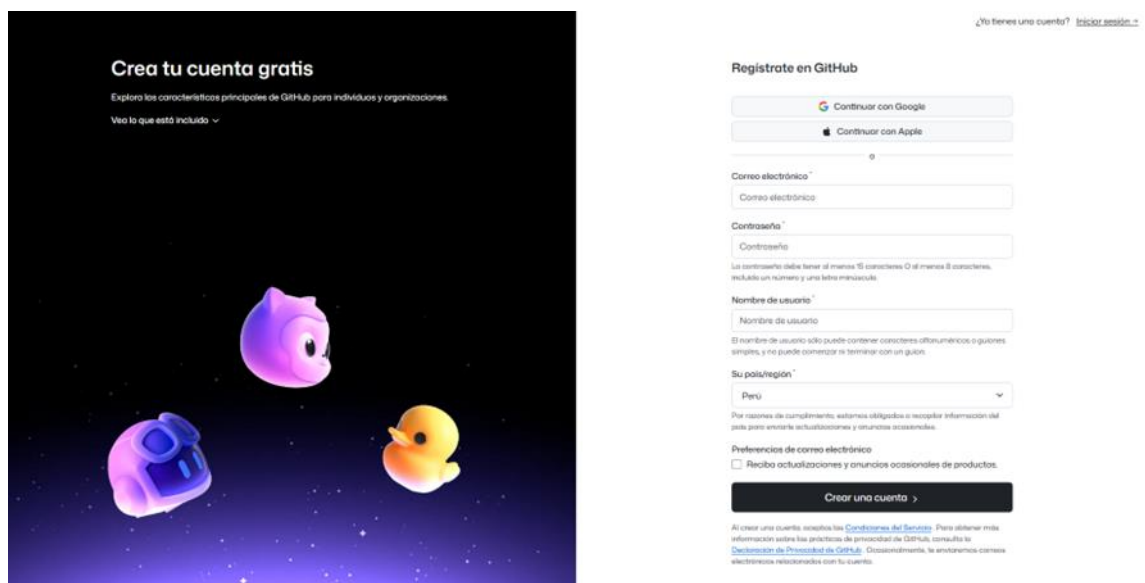
Repositorio GitHub: <https://github.com/MiguelRoa-UPN/POO-Git-Casos.git>

1. Creación de cuenta en GIT HUB

Nos dirigimos a la web oficial de GitHub y seleccionamos INICIAR SESION



Procedemos a completar el formulario de registro ingresando nuestros datos personales y credenciales requeridas, con el fin de crear y habilitar una nueva cuenta en GitHub.



The screenshot shows the GitHub registration form. On the left, there's a section titled "Crea tu cuenta gratis" with a description and a link to "Ver lo que está incluido". On the right, the "Regístrate en GitHub" form is displayed. It includes options to "Continuar con Google" or "Continuar con Apple", followed by fields for "Correo electrónico", "Contraseña", "Nombre de usuario", and "Su país/region". There's a checkbox for "Recibo actualizaciones y anuncios ocasionales de productos." and a "Crear una cuenta" button. At the bottom, there's a link to "Ver las condiciones de servicio de GitHub".

Se enviará un correo de verificación a la dirección de correo electrónico registrada durante la creación de la cuenta en GitHub, el cual deberá ser confirmado para validar el registro.



Here's your GitHub launch code!



Continue signing up for GitHub by entering the code below:

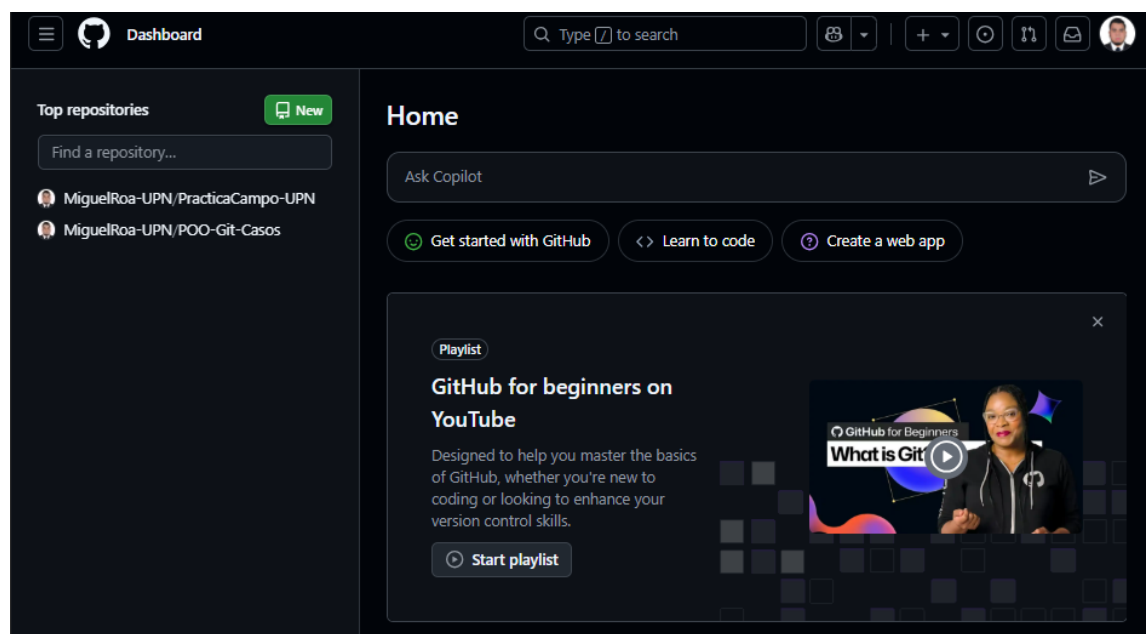
85860065

Open GitHub

Not able to enter the code? Paste the following link into your browser:

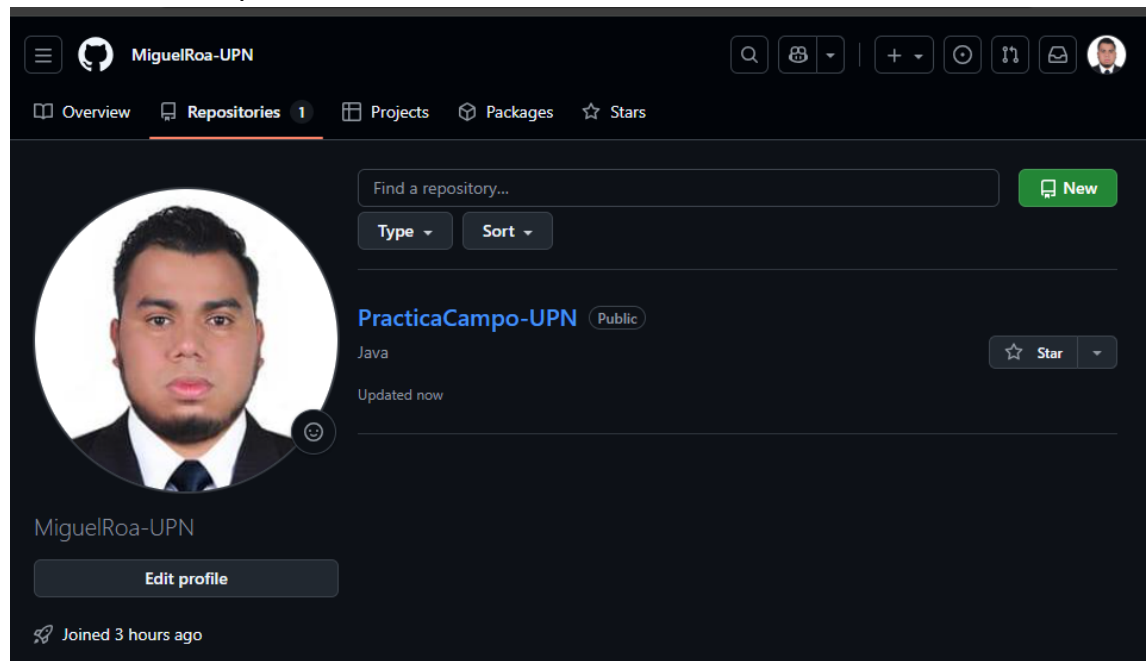
https://github.com/account_verifications/confirm/050bd611-73b6-4f2d-a5e4-75607f6f12bd/85860065

Una vez completado el proceso de verificación, el usuario estará habilitado para acceder a la cuenta en GitHub y utilizar sus funcionalidades.

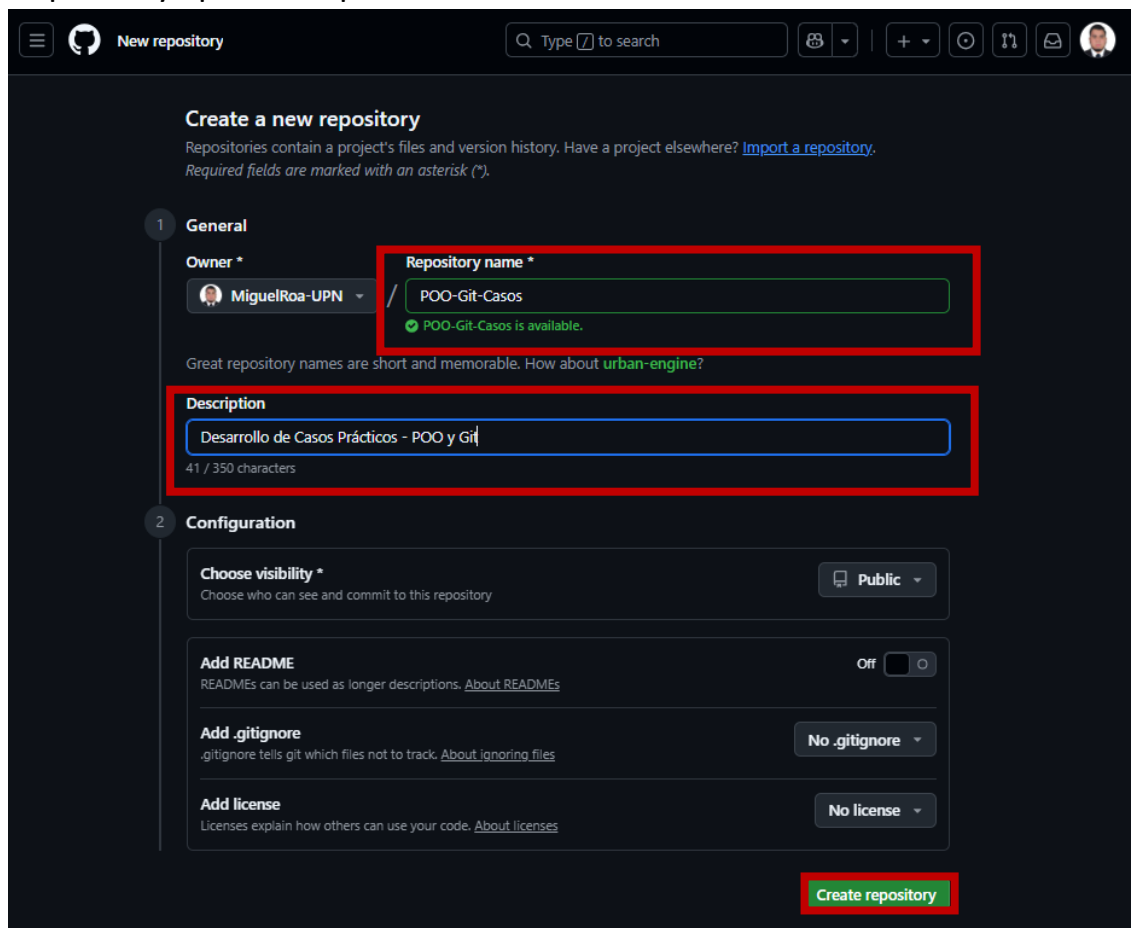


2. Creación repositorio en GitHub

Seleccionamos la opción “NEW” para iniciar el proceso de creación de un nuevo repositorio en GitHub.

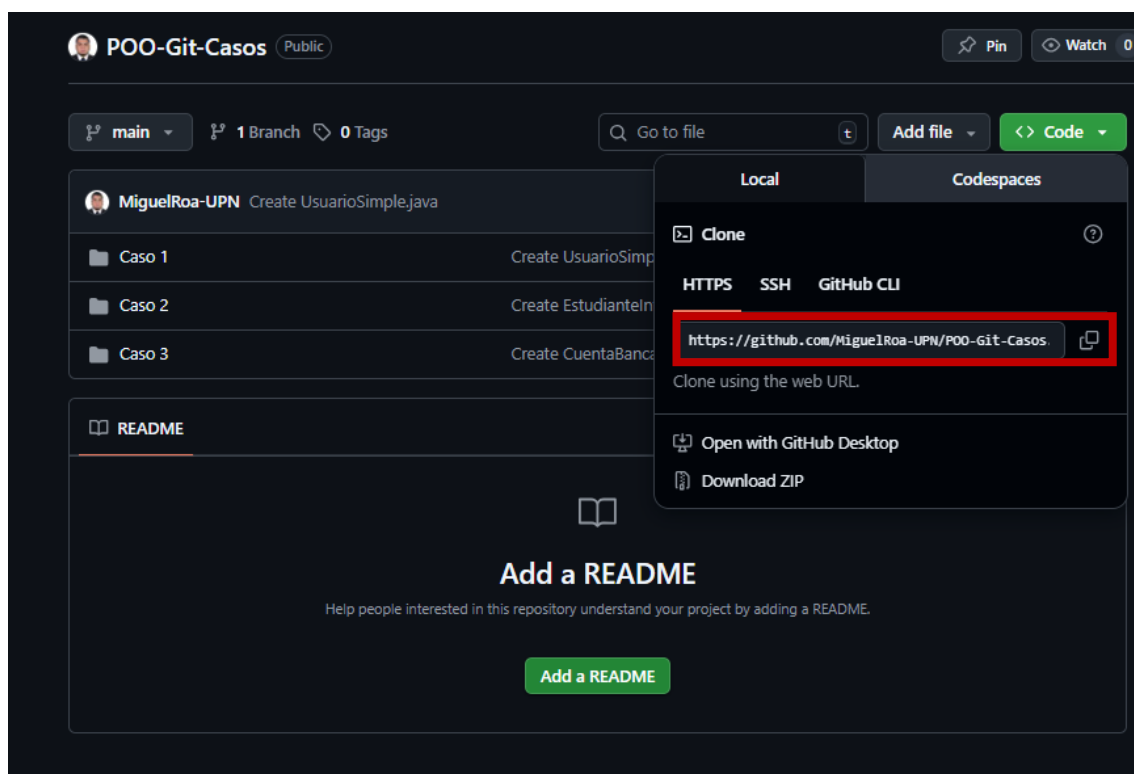


Se asigna un nombre al repositorio, se incorpora una descripción opcional y posteriormente se hace clic en el botón “Create Repository” para completar su creación.



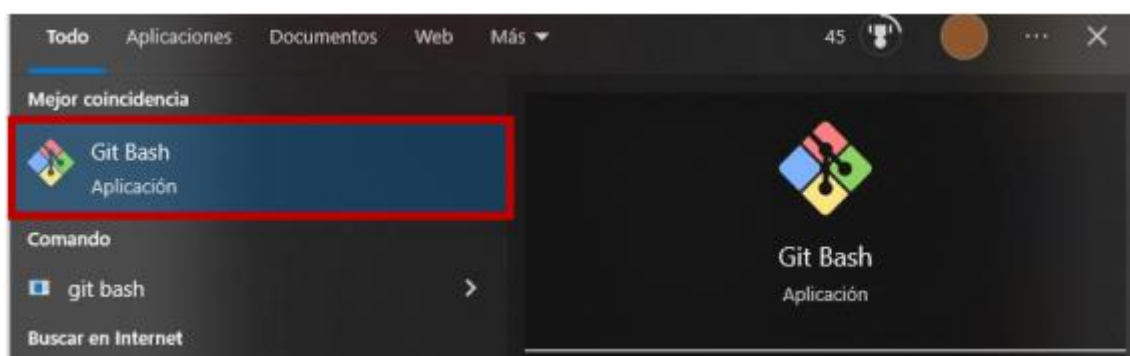
3. Clonar repositorio GitHub en maquina local

Se copia la URL del repositorio generada en GitHub para su posterior utilización.



Se inicia el programa Git Bash.

Nota: En caso de no contar con la aplicación, se debe descargar e instalar Git Bash en el equipo antes de continuar con el proceso.



Se debe configurar la sesión de GitHub previamente, con el fin de que Git reconozca el origen y las credenciales asociadas al repositorio remoto

```

MINGW64:/C/Users/Miguel Roa/Casos Practicos - POO
Miguel Roa@DESKTOP-S6IPCVI MINGW64 ~ (main)
$ cd "/C/Users/Miguel Roa/Casos Practicos - POO"

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO (main)
$ git config user.name
MiguelRoa-UPN

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO (main)
$ git config user.email
n00459166@upn.pe

```

Se ejecuta el comando **git clone "URL_del_repositorio_GitHub"** para clonar el repositorio remoto en el entorno local.

```

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO (main)
$ git clone https://github.com/MiguelRoa-UPN/POO-Git-Casos.git
Cloning into 'POO-Git-Casos'...
warning: You appear to have cloned an empty repository.

```

Se verifica que el repositorio haya sido clonado correctamente en el entorno local.

Este equipo > Disco local (C:) > Usuarios > Miguel Roa > Casos Practicos - POO >

Nombre	Fecha de modificación	Tipo
POO-Git-Casos	23/10/2025 19:51	Carpeta de archivos

4. Primer Commit

Se crean las carpetas **Caso1**, **Caso2** y **Caso3**, conforme a la estructura requerida en el proyecto, con el propósito de realizar las primeras modificaciones y organizar adecuadamente los contenidos.

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00 (main)
$ cd P00-Git-Casos

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ mkdir Caso1

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ mkdir Caso2

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ mkdir Caso3
```

Se valida la correcta creación de los directorios en el entorno local.

Disco local (C:) > Usuarios > Miguel Roa > Casos Practicos - P00 > P00-Git-Casos

Nombre	Fecha de modificación	Tipo
Caso1	23/10/2025 19:57	Carpeta de archivos
Caso2	23/10/2025 19:57	Carpeta de archivos
Caso3	23/10/2025 19:57	Carpeta de archivos

Se procede a la creación de los archivos .java conforme a la estructura establecida

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ cd "/C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos"

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso1 (main)
$ cd Caso1
$ touch UsuarioSimple.java

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso2 (main)
$ cd "/C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos"

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso2 (main)
$ touch EstudianteInteractivo.java

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso3 (main)
$ cd "/C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos"

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso3 (main)
$ touch CuentaBancaria.java
```

Se efectúa la navegación dentro del repositorio local y se ejecuta el primer **commit**, registrando los cambios iniciales.

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos/Caso3 (main)
$ cd "/C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos"

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ ls
Caso1/ Caso2/ Caso3/

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Caso1/
    Caso2/
    Caso3/

nothing added to commit but untracked files present (use "git add" to track)

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ git add .

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ git commit -m "Semana 2 - Java"
[main (root-commit) 107d663] Semana 2 - Java
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Caso1/UsuarioSimple.java
 create mode 100644 Caso2/EstudianteInteractivo.java
 create mode 100644 Caso3/CuentaBancaria.java

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 441 bytes | 441.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MiguelRoa-UPN/P00-Git-Casos.git
 * [new branch]      main -> main

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git-Casos (main)
$ git log
commit 107d663f55fcfff0676d980f09f6452726377cd4 (HEAD -> main, origin/main)
Author: MiguelRoa-UPN <n00459166@upn.pe>
Date:   Fri Oct 24 18:30:46 2025 -0500

    Semana 2 - Java
```


Se crea una nueva rama mediante **branch** y se realiza el cambio entre ramas con el comando **checkout**, permitiendo desarrollar funcionalidades de forma aislada. Adicionalmente, se integran los cambios mediante la operación **merge** para unificar el trabajo en el repositorio principal.

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git branch RamaSemana2

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git checkout
Your branch is up to date with 'origin/main'.

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git checkout RamaSemana2
Switched to branch 'RamaSemana2'

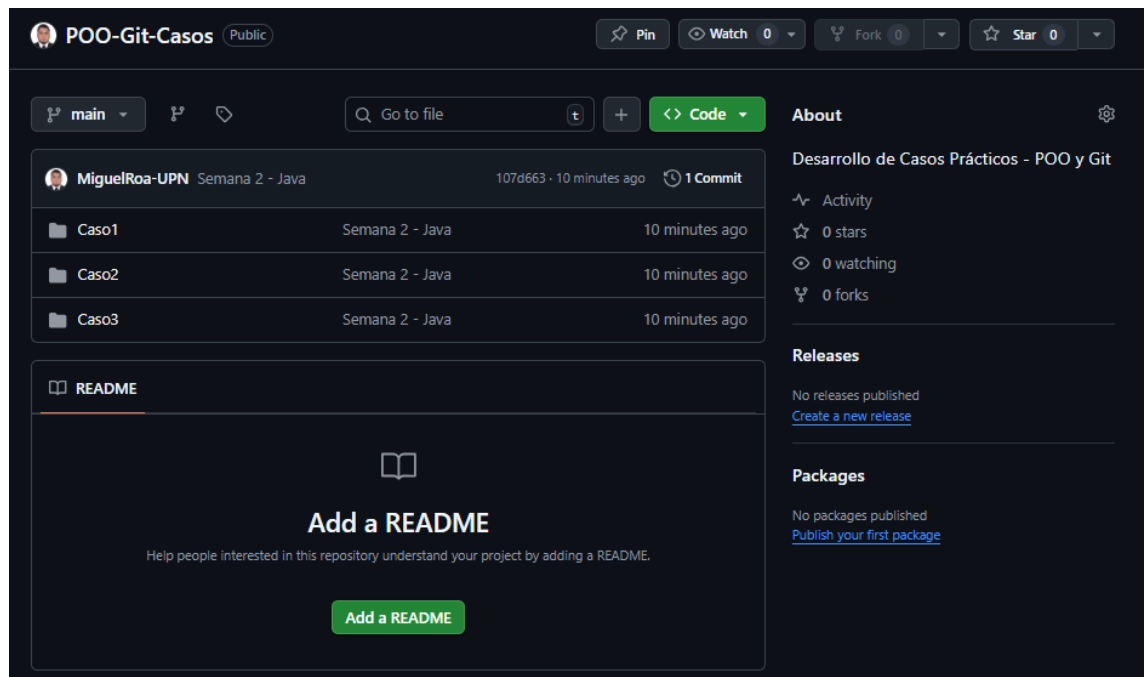
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (RamaSemana2)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git pull
Already up to date.

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git merge RamaSemana2
Already up to date.

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - P00/P00-Git
-Casos (main)
$ git push
Everything up-to-date
```

Se verifica la correcta sincronización y actualización de los cambios en el repositorio remoto de GitHub



5. Desarrollo de casos

Se procede a desarrollar la codificación en lenguaje Java conforme a los casos solicitados, siguiendo la estructura definida

CASO 1: Lectura de datos simples con Scanner

<pre> 1- import java.util.Scanner; 2 3- public class Main { 4- public static void main(String[] args) { 5 6 Scanner sc = new Scanner(System.in); 7 8 // Solicita y guarda el nombre 9 System.out.print("Ingrese su Nombre: "); 10 String nombre = sc.nextLine(); 11 12 // Solicita y guarda el apellido 13 System.out.print("Ingrese su Apellido: "); 14 String apellido = sc.nextLine(); 15 16 // Muestra el resultado 17 System.out.println("Hola, " + nombre + " " + 18 apellido); 19 sc.close(); 20 } 21 } 22 23 </pre>	<pre> Ingrese su Nombre: Miguel Ingrese su Apellido: Roa Hola, Miguel Roa === Code Execution Successful === </pre>
---	---

CASO 2: Clase Estudiante con atributos privados

```

1- import java.util.Scanner;
2
3- class Estudiante {
4     private String nombre;
5     private String apellido;
6     private int edad;
7     // Constructor para inicializar atributos
8-     public Estudiante(String nombre, String apellido, int edad) {
9         this.nombre = nombre;
10        this.apellido = apellido;
11        this.edad = edad;
12    }
13    // Métodos para obtener los datos
14    public String getNombre() { return nombre; }
15    public String getApellido() { return apellido; }
16    public int getEdad() { return edad; }
17 }
18
19- public class Main {
20-     public static void main(String[] args) {
21         Scanner sc = new Scanner(System.in);
22
23         System.out.print("Nombre: ");
24         String nombre = sc.nextLine(); // Leer nombre
25
26         System.out.print("Apellido: ");
27         String apellido = sc.nextLine(); // Leer nombre
28
29         System.out.print("Edad: ");
30         int edad = sc.nextInt(); // Leer edad
31
32         Estudiante est = new Estudiante(nombre, apellido, edad);
33
34         // Mostrar datos del estudiante
35         System.out.println("Estudiante: " + est.getNombre() + " " + est.getApellido() + ",
36                             Edad: " + est.getEdad());
37
38         sc.close();
39     }
}

```

Nombre: Miguel
Apellido: Roa
Edad: 16
Estudiante: Miguel Roa, Edad: 16

=== Code Execution Successful ===

CASO 3: Clase CuentaBancaria con validación

```

1- import java.util.Scanner;
2
3- class CuentaBancaria {
4     private double saldo;
5     // Método para depositar dinero
6-     public void depositar(double monto) {
7         saldo += monto;
8     }
9     // Método para retirar dinero
10-    public void retirar(double monto) {
11-        if (monto <= saldo) {
12-            saldo -= monto;
13-        } else {
14-            System.out.println("Fondos insuficientes."); // Mensaje si no hay
15-                               saldo suficiente
16-        }
17    }
18-    // Método para consultar el saldo
19-    public double getSaldo() {
20-        return saldo;
21-    }
22 }
23- public class Main {
24-     public static void main(String[] args) {
25         Scanner sc = new Scanner(System.in);
26         CuentaBancaria cuenta = new CuentaBancaria();
27         System.out.print("Monto a depositar: ");
28         cuenta.depositar(sc.nextDouble()); // Guarda y deposita el monto
29         System.out.print("Monto a retirar: ");
30         cuenta.retirar(sc.nextDouble()); // Guarda y retira el monto
31         System.out.println("Saldo final: " + cuenta.getSaldo());
32         sc.close();
33     }
34 }

```

Monto a depositar: 600
Monto a retirar: 500
Saldo final: 100.0

=== Code Execution Successful ===

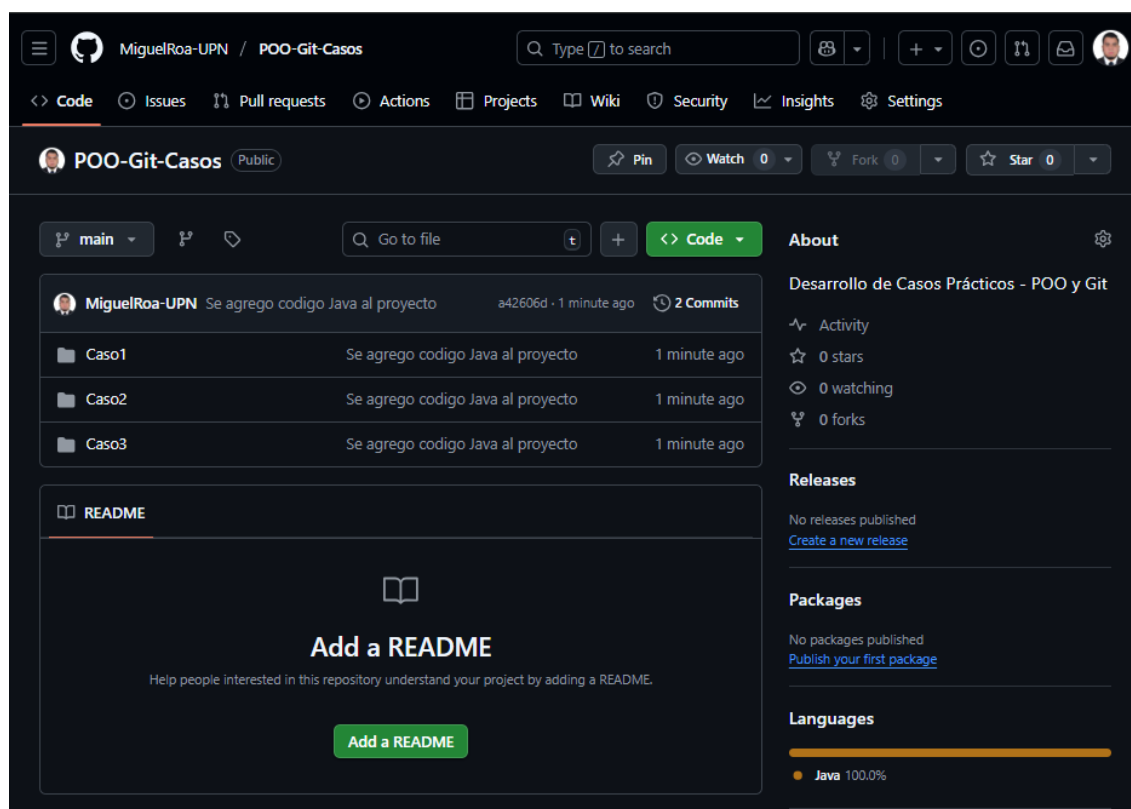
Una vez finalizada la codificación y ubicados los archivos en la ruta correspondiente del entorno local, se procede a actualizar el repositorio remoto en GitHub mediante los comandos de control de versiones.

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git add .

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git commit -m "Se agrego codigo Java al proyecto"
[main a42606d] Se agrego codigo Java al proyecto
3 files changed, 95 insertions(+)

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git push
Enumerating objects: 15, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.55 KiB | 1.55 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MiguelRoa-UPN/POO-Git-Casos.git
107d663..a42606d main -> main
```

Se verifica la correcta sincronización y actualización de los cambios en el repositorio remoto de GitHub



The screenshot shows the GitHub repository page for 'POO-Git-Casos' by 'MiguelRoa-UPN'. The repository is public and has 0 stars, 0 forks, and 0 watchers. The main branch is 'main'. The commit history shows a recent commit 'Se agrego codigo Java al proyecto' by 'MiguelRoa-UPN' with a message 'Se agrego codigo Java al proyecto' and a commit hash 'a42606d'. The repository contains three files: 'Caso1', 'Caso2', and 'Caso3'. The README section is currently empty, with a prompt to 'Add a README'. The right sidebar shows the repository's activity, including a section for 'Releases' and 'Packages'.

The screenshot shows a GitHub repository named 'POO-Git-Casos' by user 'MiguelRoa-UPN'. The 'Code' tab is selected, displaying the file 'UsuarioSimple.java' under the 'Caso1' directory. The code is a Java program that uses the Scanner class to read user input for a name and a last name, and then prints a greeting. The commit message is 'Se agrego codigo Java al proyecto' and it was committed 1 minute ago. The file explorer on the left shows the repository structure with folders 'Caso1', 'Caso2', and 'Caso3', and the file 'UsuarioSimple.java' selected.

```

1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args) {
5
6          Scanner sc = new Scanner(System.in);
7
8          // Solicita y guarda el nombre
9          System.out.print("Ingrese su Nombre: ");
10         String nombre = sc.nextLine();
11
12         // Solicita y guarda el apellido
13         System.out.print("Ingrese su Apellido: ");
14         String apellido = sc.nextLine();
15
16         // Muestra el resultado
17         System.out.println("Hola, " + nombre + " " + apellido);
18
19         sc.close();
20     }
21 }

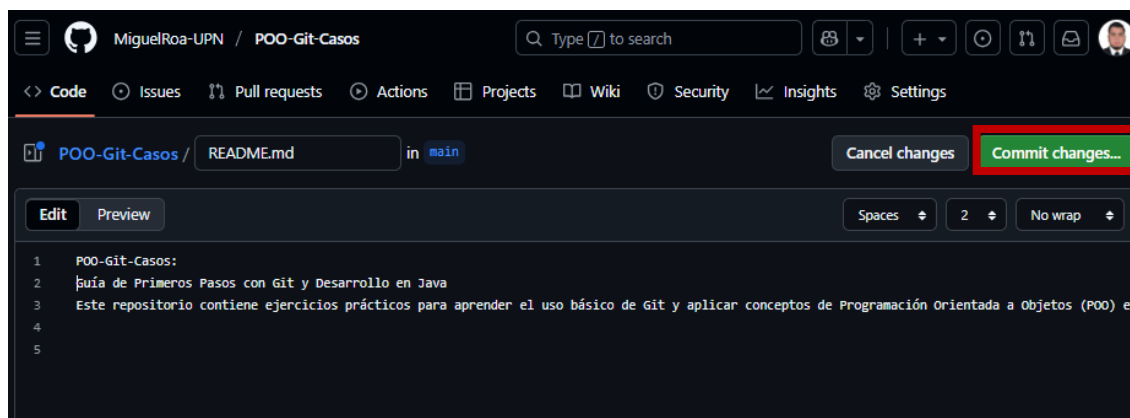
```

Se incorpora el archivo **README** al repositorio y se registra un comentario descriptivo

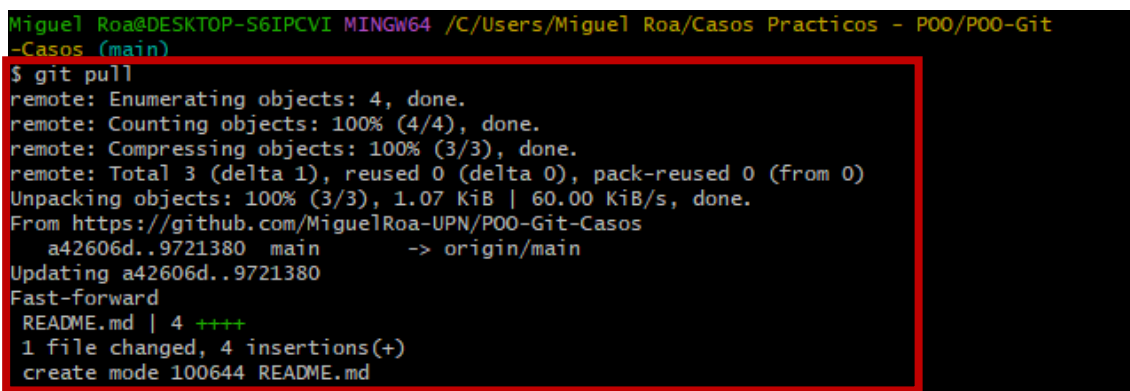
The screenshot shows the main page of the 'POO-Git-Casos' repository. The 'About' section is visible, describing the project as 'Desarrollo de Casos Prácticos - POO y Git'. The 'Releases' section shows no published releases. The 'Packages' section shows no published packages. The 'Languages' section shows that the repository is 100% Java. The 'README' section is highlighted with a red box, showing a prompt to 'Add a README' to help people understand the project.

Add a README

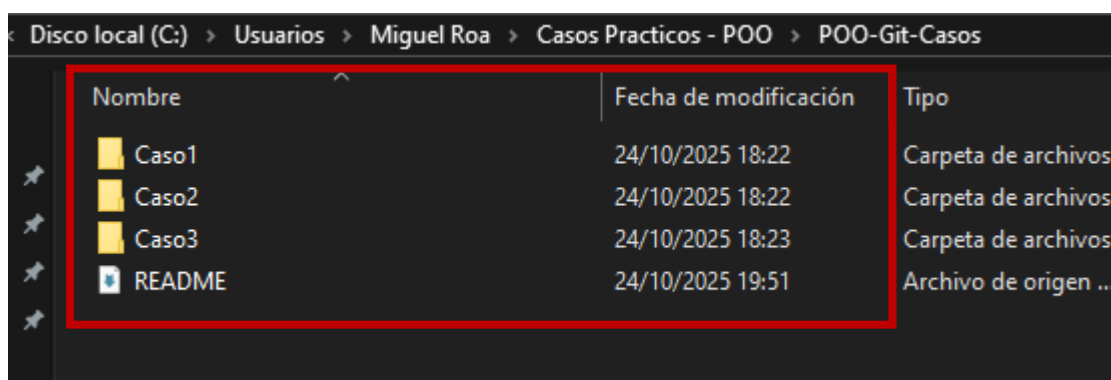
Help people interested in this repository understand your project by adding a README.



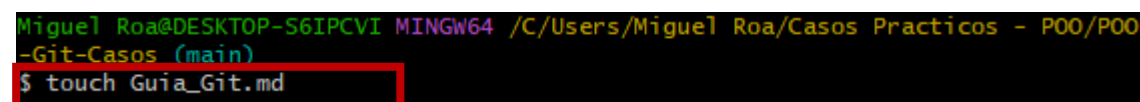
Se accede a Git Bash y se ejecuta el comando **git pull** con el propósito de sincronizar el repositorio local con los cambios realizados en GitHub, asegurando la actualización de los archivos en el equipo.



Se verifica la creación exitosa del archivo **README** en el entorno local.



Se crea el archivo **Guia_Git.md** en el entorno local y posteriormente se documentan los comandos utilizados durante el desarrollo del proyecto.



```
C: > Users > Miguel Roa > Casos Practicos - POO > POO-Git-Casos > Guia_Git.md > # Guia Git: Flujo de Trabajo para el Proyecto POO-Git-Casos

1 # Guia Git: Flujo de Trabajo para el Proyecto POO-Git-Casos
2
3 Este documento resume los comandos utilizados durante el desarrollo del proyecto:
4
5 1. **Verificar configuracion de usuario local**
6 | - Ver nombre de usuario: `git config user.name`
7 | - Ver correo electronico: `git config user.email`
8
9 2. **Clonar repositorio desde GitHub**
10 | - `git clone https://github.com/MiguelRoa-UPN/POO-Git-Casos.git`
11
12 3. **Consultar estado del repositorio**
13 | - `git status`
14
15 4. **Visualizar historial de commits**
16 | - `git log`
17
18 5. **Guardar cambios en el repositorio local**
19 | - Agregar archivos: `git add .`
20 | - Registrar commit: `git commit -m "Descripción del cambio"`
21
22 6. **Crear y cambiar de rama**
23 | - Crear rama: `git branch RamaSemana2`
24 | - Cambiar de rama: `git checkout RamaSemana2`
25
26 7. **Fusionar ramas**
27 | - Cambiar a main: `git checkout main`
28 | - Fusionar cambios: `git merge RamaSemana2`
29
30 8. **Listar archivos y carpetas del directorio actual**
31 | - `ls`
32
33 9. **Subir cambios al repositorio remoto**
34 | - `git push`
35
36 10. **Actualizar repositorio local**
37 | - `git pull`
38
```

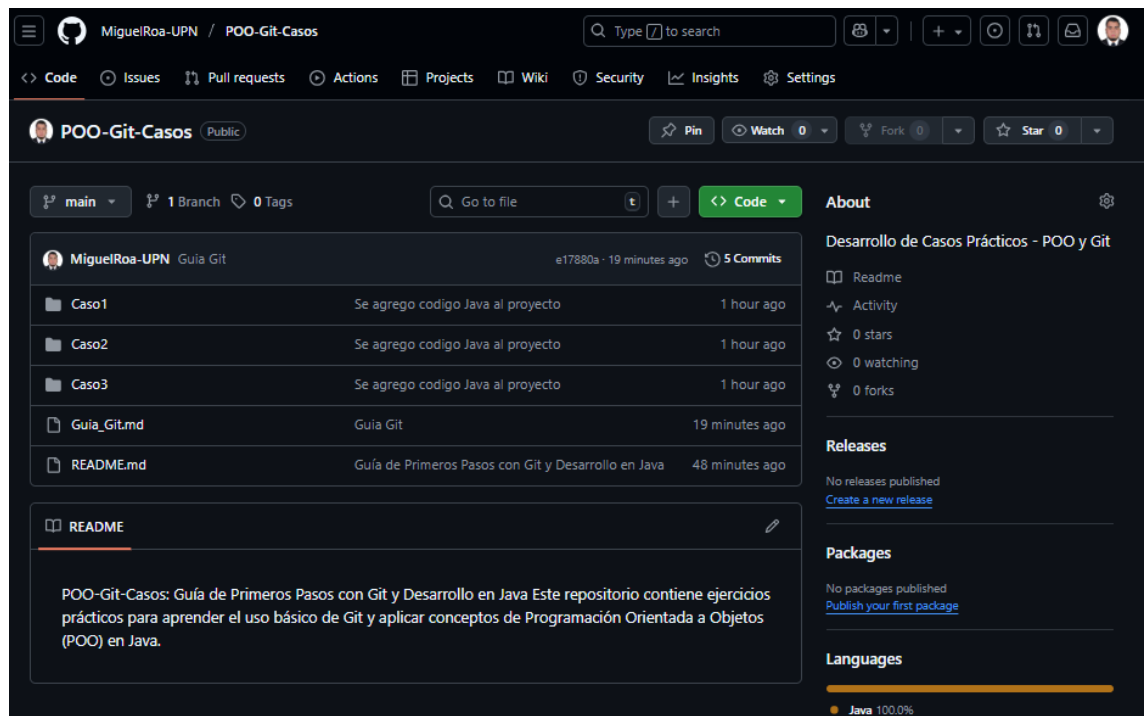
Se utilizan los comandos **git add**, **git commit** y **git push** para registrar los cambios realizados en el entorno local y sincronizarlos con el repositorio remoto en GitHub, garantizando la actualización del historial del proyecto.

```
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git add Guia_Git.md

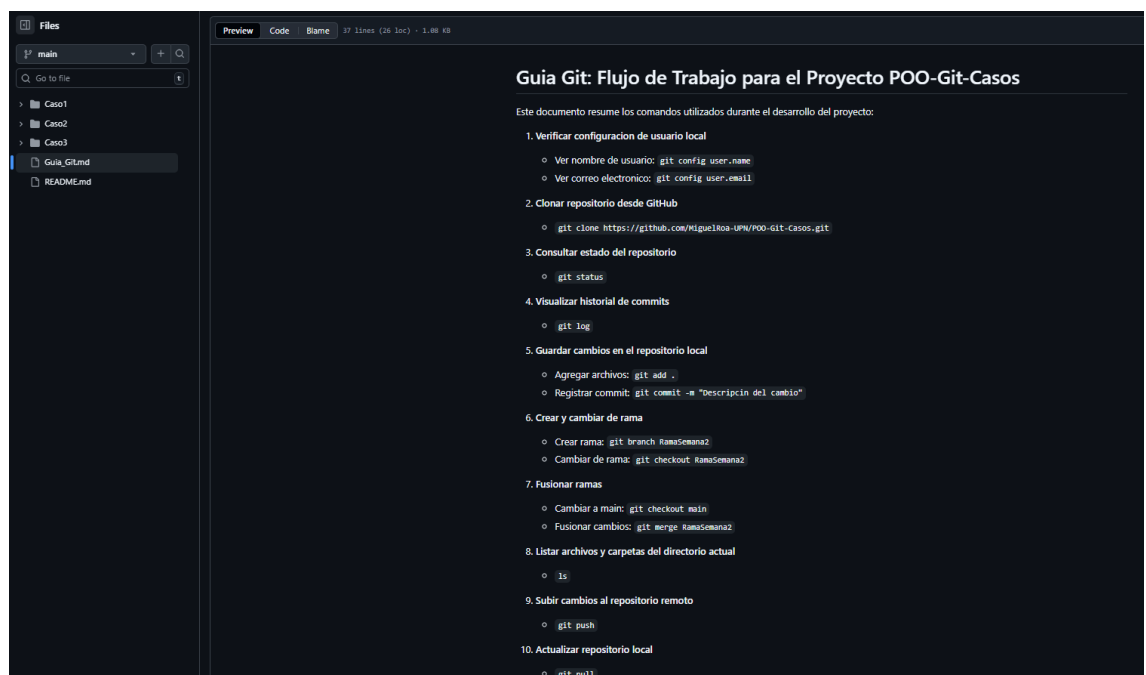
Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git commit -m "Guia Git"
[main e1c99e9] Guia Git
1 file changed, 37 insertions(+)
create mode 100644 Guia_Git.md

Miguel Roa@DESKTOP-S6IPCVI MINGW64 /C/Users/Miguel Roa/Casos Practicos - POO/POO-Git-Casos (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 794 bytes | 794.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MiguelRoa-UPN/POO-Git-Casos.git
bbb388c..e1c99e9 main -> main
```

Se verifica la correcta sincronización y actualización de los cambios en el repositorio remoto de GitHub.



The screenshot shows the GitHub interface for the repository 'MiguelRoa-UPN / POO-Git-Casos'. The repository is public and has 5 commits. The file list includes 'Caso1', 'Caso2', 'Caso3', 'Guia_Git.md', and 'README.md'. The 'README.md' file is selected, showing its content: 'POO-Git-Casos: Guía de Primeros Pasos con Git y Desarrollo en Java Este repositorio contiene ejercicios prácticos para aprender el uso básico de Git y aplicar conceptos de Programación Orientada a Objetos (POO) en Java.' The right sidebar shows the repository's activity, including 0 stars, 0 watching, and 0 forks. The 'Languages' section shows 'Java' at 100.0%.



The screenshot shows the 'Guia Git: Flujo de Trabajo para el Proyecto POO-Git-Casos' document. The document is a guide for using Git and is organized into 10 numbered steps:

1. Verificar configuración de usuario local
 - Ver nombre de usuario: `git config user.name`
 - Ver correo electrónico: `git config user.email`
2. Clonar repositorio desde GitHub
 - `git clone https://github.com/MiguelRoa-UPN/POO-Git-Casos.git`
3. Consultar estado del repositorio
 - `git status`
4. Visualizar historial de commits
 - `git log`
5. Guardar cambios en el repositorio local
 - Agregar archivos: `git add .`
 - Registrar commit: `git commit -m "Descripción del cambio"`
6. Crear y cambiar de rama
 - Crear rama: `git branch Ramasemana2`
 - Cambiar de rama: `git checkout Ramasemana2`
7. Fusionar ramas
 - Cambiar a main: `git checkout main`
 - Fusionar cambios: `git merge Ramasemana2`
8. Listar archivos y carpetas del directorio actual
 - `ls`
9. Subir cambios al repositorio remoto
 - `git push`
10. Actualizar repositorio local
 - `git pull`