Cuaderno de ejercicios

Tema 1

Git (trabajo individual)

Trabajo en local

1. **Inicialización de un repositorio**
    - Crea un nuevo directorio llamado mi-proyecto y conviértelo en un repositorio de Git. Dentro de este directorio, crea una subcarpeta llamada src y un archivo index.php con un mensaje de saludo.

```
mkdir mi-proyecto
cd mi-proyecto
git init
mkdir src
echo '<?php echo "Hola, mundo!"; ?>' > src/index.php
```

2. **Añadir archivos al seguimiento**
    - Crea un archivo functions.php en la carpeta src con una función PHP que retorne un saludo. Agrega este archivo al área de preparación.

```
echo '<?php function saludar() { return "Hola desde la función!"; } ?>' > src/functions.php
git add src/functions.php
```

3. **Confirmar cambios (commit)**
    - Realiza un commit con un mensaje que describa la creación del archivo functions.php.

```
git commit -m "Crea archivo functions.php con función de saludo"
```

4. **Visualización del historial de commits**
    - Consulta el historial de commits de tu repositorio y muestra sólo los mensajes de commit.

```
git log --oneline
```

5. **Ignorar archivos**
    - Crea un archivo .gitignore en el directorio raíz y configura Git para que ignore los archivos con la extensión .log.

```
echo '*.log' > .gitignore
git add .gitignore
git commit -m "Añade .gitignore para ignorar archivos .log"
```

6. **Modificar un archivo y realizar otro commit**
    - Edita el archivo index.php para que incluya y utilice la función de functions.php para mostrar el saludo. Realiza un commit describiendo los cambios.

```
echo '<?php include "functions.php"; echo saludar(); ?>' > src/index.php
git add src/index.php
git commit -m "Modifica index.php para incluir y utilizar functions.php"
```

7. **Comparar cambios**
    - Crea un fichero README.md en el directorio raíz y añádelo al repositorio. Haz un cambio en el fichero y muestra las diferencias entre el área de trabajo y el área de preparación. A continuación, añade el fichero al área de preparación y muestra las diferencias entre el área de preparación y el último commit.

```
echo 'Ejercicios T1' > README.md
git add README.md
git commit -m "Añadido fichero README.md"
echo 'Ejercicios T1 mi-proyecto' > README.md
git diff
git add README.md
git diff --staged
```

8. **Eliminar archivos**
    - Elimina el archivo index.php y realiza un commit para reflejar el cambio.

```
rm src/index.php
git rm src/index.php
git commit -m "Elimina archivo index.php"
```

Trabajo en local y ramas

9. **Crear y cambiar de ramas**
    - Crea una nueva rama llamada funcionalidad-1 y cambia a esa rama. En esta rama, añade una nueva función en functions.php que retorne un mensaje personalizado.

```
git branch funcionalidad-1
git checkout funcionalidad-1
echo '<?php function saludarPersonalizado($nombre) { return "Hola, $nombre!"; } ?>' >> src/functions.php
```

10. **Hacer commits en la nueva rama**
    - Realiza un commit en la rama funcionalidad-1 describiendo los cambios realizados en functions.php.

```
git add src/functions.php
git commit -m "Añade función saludarPersonalizado en functions.php"
```

11. **Fusionar ramas (merge)**
    - Vuelve a la rama principal (main o master) y fusiona los cambios de la rama funcionalidad-1.

```
git checkout main
git merge funcionalidad-1
```

12. **Resolver conflictos de fusión**
    - Crea un conflicto de fusión editando functions.php en la rama main y funcionalidad-1, luego fusiónalas y resuelve el conflicto.

```
# En la rama main
git checkout main
echo '<?php function saludarPersonalizado($nombre) { return "Hola, $nombre!"; } ?>' >> src/functions.php
git commit -am "Añade función saludarPersonalizado en main"

# En la rama funcionalidad-1
git checkout funcionalidad-1
echo '<?php function saludarPersonalizado($nombre) { return "¡Hola, $nombre!"; } ?>' >> src/functions.php
git commit -am "Añade función saludarPersonalizado en funcionalidad-1"
```

```
# Volver a main y fusionar
git checkout main
git merge funcionalidad-1
# Resolver conflictos manualmente en src/functions.php
# Resultado esperado:
# <?php
# function saludar() { return "Hola desde la función!"; }
# function saludarPersonalizado($nombre) { return "Hola, $nombre!"; }
# ?>
git add src/functions.php
git commit -m "Resuelve conflictos de fusión entre main y funcionalidad-1"
```

13. **Rebase**

- Realiza un rebase de la rama funcionalidad-1 sobre main y explica las diferencias con merge.

git rebase permite reaplicar commits de una rama sobre otra, de manera que parezca que los commits se construyeron directamente sobre una base común. Facilita la integración y simplifica el historial de commits.

```
git checkout funcionalidad-1
git rebase main
```

Estas instrucciones hacen que se pase de una situación como esta de ejemplo:

```
A---B---C (main)
     \
      D---E (funcionalidad-1)
```

A una como esta:

```
A---B---C---D'---E' (funcionalidad-1)
```

Es decir, la rama “funcionalidad-1” ha sido modificada haciendo que sus commits D y E se reescriban como D’ y E’.

Hay que tener en cuenta que la rama “main” sigue con su historial de commits exactamente igual.

```
A---B---C (main)
```

Con respecto a marge, en rebase la rama “main” no se ve afectada, pero en merge se realizaría una unión entre ambas ramas (“main” y “funcionalidad-1”).

Con git merge:

```
A---B---C---M (main)
     \     /
      D---E (funcionalidad-1)
```

Con git rebase:

```
A---B---C---D'---E' (main, funcionalidad-1)
```

Más información:

https://hakdogan26.medium.com/understanding-git-merge-vs-rebase-b1d6e4bccb1d

Trabajo en remoto con GitHub

14. **Crear un repositorio en GitHub**
    - Crea un nuevo repositorio en GitHub llamado mi-proyecto y conéctalo a tu repositorio local. Realiza un push de tus cambios locales al repositorio remoto.

```
git remote add origin https://github.com/tu-usuario/mi-proyecto.git
git push -u origin main
```

15. **Clonar un repositorio**
    - Clona un repositorio existente de GitHub a tu máquina local. Añade un nuevo archivo README.md con la descripción del proyecto y realiza un push de los cambios al repositorio remoto.

```
git clone https://github.com/tu-usuario/mi-proyecto.git
cd mi-proyecto
echo '# Mi Proyecto\nEste es un proyecto de ejemplo.' > README.md
git add README.md
git commit -m "Añade README.md con la descripción del proyecto"
git push origin main
```

16. **Trabajar con ramas en remoto**
    - Crea una nueva rama en tu repositorio local llamada funcionalidad-2 y publícala en GitHub. Realiza cambios en functions.php en la nueva rama y realiza un push de los cambios.

```
git checkout -b funcionalidad-2
echo '<?php function despedida() { return "Adiós!"; } ?>' >>
src/functions.php
git add src/functions.php
git commit -m "Añade función despedida en functions.php"
git push -u origin funcionalidad-2
```

17. **Realizar pull de cambios**

    - Modifica desde GitHub el fichero README.md. Obtén los últimos cambios desde el repositorio remoto y actualiza tu repositorio local.

```
git pull origin main
```

18. **Fork y pull request**

    - Haz un fork de un repositorio en GitHub de algún compañero (o de otra cuenta de GitHub que tengas), clona tu fork localmente, realiza cambios en index.php, y crea un pull request.

Un fork es una copia de un repositorio (repo_original) de GitHub a otro repositorio (repo_copia). El dueño del repo_copia realiza entonces los cambios que considere en su repo_copia (repo_copia_modificado) y cuando haya terminado, envía una pull request al dueño del repo_original. Entonces el dueño del repo_original revisa los cambios (hechos en repo_copia_modificado) y los integra (todos, algunos o ninguno) en el repo_original.

```
# En GitHub: Hacer fork del repositorio de un compañero
git clone https://github.com/tu-usuario/mi-proyecto-fork.git
cd mi-proyecto-fork
echo '<?php include "functions.php"; echo saludar(); ?>' > src/index.php
git add src/index.php
git commit -m "Modifica index.php para incluir y utilizar functions.php"
git push origin main
# En GitHub: Crear pull request
```

Más información: https://aprendegit.com/fork-de-repositorios-para-que-sirve/
https://aprendegit.com/que-es-un-pull-request/

19. **Revisar pull requests**

    - Revisa y comenta el pull request en GitHub que haya creado tu compañero (o tú mismo, si utilizas dos cuentas de GitHub). Aprueba o solicita cambios según sea necesario.

```
En GitHub: Revisa, comenta, aprueba o solicita cambios en el pull request.
```

Más información: https://aprendegit.com/que-es-un-pull-request/

Git (trabajo en equipo)

En este bloque de ejercicios hay que trabajar por parejas.

Vais a desarrollar una sencilla aplicación web de tipo calculadora. Debe ser lo más básica posible, pero debe contar con una interfaz web que permita introducir 2 números y realizar operaciones de suma, resta, multiplicación o división y mostrar el resultado. El proyecto deberá tener 3 ramas, una rama para la interfaz, otra para realizar la suma y la resta, y otra para realizar la multiplicación y la división. Una vez estén todas las ramas finalizadas, se deberá hacer un merge a la rama main.

Deberéis crear un repositorio de GitHub para realizar los ejercicios.

1. **Inicializar el repositorio y configurar el entorno**
    - Uno de los integrantes del equipo (A) crea un nuevo repositorio en GitHub llamado calculadora-web y lo inicializa con un archivo README.md. Luego, clona el repositorio en su máquina local e invita al otro integrante (B) como colaborador.
    - Clonar el repositorio en la máquina local de B.

```
# Persona A
git init
echo "# Calculadora Web" > README.md
git add README.md
git commit -m "Inicializa el repositorio con README"
git remote add origin https://github.com/tu-usuario/calculadora-web.git
git push -u origin main

# Persona B
git clone https://github.com/tu-usuario/calculadora-web.git
```

2. **Estructurar el proyecto**
    - "A" crea la estructura inicial del proyecto en la rama main con los archivos y carpetas necesarios para una aplicación web simple (HTML, CSS, y PHP).
        - index.html para la interfaz de usuario.
        - styles.css para el estilo.
        - calculadora.php para manejar las operaciones de la calculadora.

```
# Persona A
mkdir src
echo '<!DOCTYPE html>
<html>
<head>
    <title>Calculadora</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
</body>
</html>' > src/index.html

echo 'body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 50px;
}
form {
    display: inline-block;
    margin: auto;
}
input, button {
    display: block;
    margin: 5px;
}' > src/styles.css

echo '<?php
$num1 = $_POST["num1"];
$num2 = $_POST["num2"];
$operation = $_POST["operation"];
switch ($operation) {
}
?>' > src/calculator.php

git add src/
git commit -m "Añade estructura inicial del proyecto"
git push origin main
```

3. **Crear las ramas de trabajo**

- "A" crea tres ramas adicionales desde main:
  - funcionalidad/interfaz
  - funcionalidad/suma-resta
  - funcionalidad/multiplicacion-division

```
# Persona A
git checkout -b feature/interfaz
git push -u origin feature/interfaz

git checkout main
git checkout -b feature/suma-resta
git push -u origin feature/suma-resta

git checkout main
git checkout -b feature/multiplicacion-division
git push -u origin feature/multiplicacion-division
```

4. **Implementar la interfaz**

- "B" cambia a la rama funcionalidad/interfaz y desarrolla la interfaz de usuario en index.html, incluyendo un formulario con dos campos numéricos y botones para las operaciones de suma, resta, multiplicación y división, así como un campo para el resultado.

```
# Persona B
git pull
git checkout feature/interfaz
# Realiza las modificaciones necesarias en src/index.html
<body>
    <form action="calculator.php" method="post">
        <input type="number" name="num1" placeholder="Número 1" required>
        <input type="number" name="num2" placeholder="Número 2" required>
        <button type="submit" name="operation" value="suma">Sumar</button>
        <button type="submit" name="operation" value="resta">Restar</button>
        <button type="submit" name="operation"
value="multiplicacion">Multiplicar</button>
        <button type="submit" name="operation"
value="division">Dividir</button>
    </form>
```

```
</body>

git add src/index.html
git commit -m "Desarrolla la interfaz de usuario"
git push origin feature/interfaz
```

5. **Implementar suma y resta**

   - "A" cambia a la rama funcionalidad/suma-resta y desarrolla las funcionalidades de suma y resta en calculadora.php, asegurándose de que se manejen las solicitudes del formulario de index.html.

```
# Persona A
git checkout feature/suma-resta
# Realiza las modificaciones necesarias en src/calculator.php
switch ($operation) {
    case "suma":
        echo "Resultado: " . ($num1 + $num2);
        break;
    case "resta":
        echo "Resultado: " . ($num1 - $num2);
        break;
}

git add src/calculator.php
git commit -m "Implementa funcionalidades de suma y resta"
git push origin feature/suma-resta
```

6. **Implementar multiplicación y división**

   - "B" cambia a la rama funcionalidad/multiplicacion-division y desarrolla las funcionalidades de multiplicación y división en calculadora.php, asegurándose de que se manejen las solicitudes del formulario de index.html.

```
# Persona B
git checkout feature/multiplicacion-division
# Realiza las modificaciones necesarias en src/calculator.php
switch ($operation) {
    case "multiplicacion":
        echo "Resultado: " . ($num1 * $num2);
```

```
        break;
    case "division":
        echo "Resultado: " . ($num1 / $num2);
        break;
}
```

```
git add src/calculator.php
git commit -m "Implementa funcionalidades de multiplicación y división"
git push origin feature/multiplicacion-division
```

7. **Revisión y merge de la interfaz**
    - "A" revisa los cambios en la rama funcionalidad/interfaz, realiza un merge a main y resuelve cualquier conflicto.

```
# Persona A
git checkout main
git pull origin main
git checkout feature/interfaz
git pull origin feature/interfaz
git checkout main
git merge feature/interfaz
git push origin main
```

8. **Revisión y merge de suma y resta**
    - "B" revisa los cambios en la rama funcionalidad/suma-resta, realiza un merge a main y resuelve cualquier conflicto.

```
git checkout main
git pull origin main
git checkout feature/suma-resta
git pull origin feature/suma-resta
git checkout main
git merge feature/suma-resta
git push origin main
```

9. **Revisión y merge de multiplicación y división**

- "A" revisa los cambios en la rama funcionalidad/multiplicacion-division, realiza un merge a main y resuelve cualquier conflicto.

```
# Persona A
git checkout main
git pull origin main
git checkout feature/multiplicacion-division
git pull origin feature/multiplicacion-division
git checkout main
git merge feature/multiplicacion-division
git push origin main
```

10. **Merge final y prueba de la aplicación**

- Ambos integrantes se aseguran de que la rama main tiene la última versión del código de todas las ramas, realizan pruebas finales de la aplicación, y actualizan el archivo README.md con instrucciones de uso.

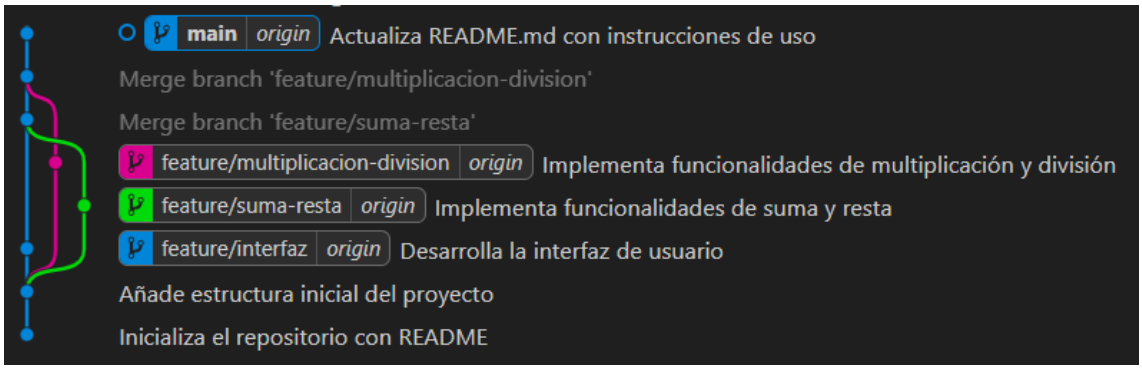
```
# Ambos integrantes
git checkout main
git pull origin main
# Probar la aplicación y actualizar README.md
echo "# Calculadora Web
Esta es una aplicación web básica de calculadora que permite realizar
operaciones de suma, resta, multiplicación y división.

## Instrucciones
1. Introduce dos números en los campos proporcionados.
2. Selecciona la operación que deseas realizar.
3. Haz clic en el botón correspondiente para ver el resultado.

## Instalación
Clona el repositorio y abre el archivo `index.html` en tu navegador." >
README.md

git add README.md
git commit -m "Actualiza README.md con instrucciones de uso"
git push origin main
```

Finalmente, puedes utilizar el plugin Git Graph de VS Code para verel trabajo que has ido realizando:



Pruebas

1. Prueba la aplicación en localhost. Hazlo de dos formas, con el servidor web Apache y con el servidor web incorporado (built-in) de PHP.

Servidor web Apache: copia el contenido de la carpeta src en un subdirectorio (p.ej. calculadora-web) de la carpeta htdocs (o equivalente) de Apache. Arranca Apache y desde el navegador accede a la ruta http://localhost/calculadora-web

Servidor web incorporado (built-in) de PHP: navega hasta la carpeta src del proyecto y ejecuta la instrucción "php -S localhost:8000". Desde el navegador accede a la ruta http://localhost:8000

2. Prueba la aplicación en la máquina virtual Ubuntu (accede primero desde el navegador de Ubuntu y luego desde el navegador de la máquina host). Hazlo también de dos formas, con el servidor web Apache y con el servidor web incorporado (built-in) de PHP.

Puedes copiar el contenido de la carpeta src vía SFTP a la MV Ubuntu, al directorio correspondiente de Apache (p.ej. /var/www/html/calculadora-web) y acceder luego desde dentro de la máquina virtual (http://localhost/calculadora-web) o desde la máquina host (http://IP-MV-Ubuntu/calculadora-web).

Para ejecutar el servidor incorporado (built-in) de PHP necesitas tener instalado PHP y repetir los mismos pasos que en el ejercicio anterior.

3. Crea un Docker personalizado con tu aplicación y luego ejecútalo en localhost (hazlo de manera similar a cómo lo has hecho en el ejemplo práctico del Tema 1).

Crea un fichero Dockerfile en el directorio del proyecto con el siguiente contenido:

```
# Usa una imagen base oficial de PHP con Apache

FROM php:8.2-apache

# Copia el contenido de la carpeta "src" en el contenedor

COPY src/ /var/www/html/
```

Crea la imagen:

```
docker build -t calculadora-web .
```

Ejecuta la imagen:

```
docker run -d -p 8080:80 calculadora-web
```

Accede a la aplicación en el navegador: http://localhost:8080

Ten en cuenta que hay que utilizar la imagen base php:8.2-apache, que incorpora Apache y PHP. En el ejemplo visto en el tema 1 se utilizaba solo la de Apache (httpd) porque se trataba de una web estática (sin PHP).