SEGUIMIENTO EN GIT DEL SEGUNDO ENTREGABLE

VICTORIA GARCÍA Y GUSTAVO GARCÍA

LA URL DE NUESTRO REPOSITORIO EN GITHUB ES LA SIGUIENTE:

https://github.com/victoriagarciah-um-es/pcd_entregable2_victoria_gustavo.git

1. CREACIÓN DEL DIRECTORIO DE TRABAJO Y REPOSITORIO GIT

Primero, creamos un directorio llamado PCD_ENTREGABLE2_VICTORIA_GUSTAVO en nuestra máquina local que corresponderá al directorio de trabajo en este boletín. A continuación, vamos a asociar un repositorio git a él ejecutando git init. Esto generará un directorio .git dentro de nuestro directorio de trabajo. Ahora registramos nuestras credenciales de GitHub en nuestro ordenador. Para ello ejecutamos los siguientes dos comandos:

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git init
Initialized empty Git repository in C:/Users/fghel/Documents/VICTORIA/datos/segundo/segundo_cuatri/programacion/pcd_entr
egable2_victoria_gustavo/.git/

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git config
--global user.email victoria.garciah@um.es

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git config
--global user.name victoriagarciah-um-es
```

En Github hemos creado un nuevo repositorio remoto, con el mismo nombre que nuestro directorio de trabajo local, al que subiremos los cambios de nuestro directorio.

A continuación, vamos a subir nuestro repositorio local a dicho repositorio remoto. Para ello, debemos ejecutar los siguientes comandos git.

Con el primer comando (git remote add origin) asociamos a nuestro repositorio remoto en github el nombre origin. A continuación, con git branch renombramos la única rama de nuestro proyecto hasta ahora como development, a la que añadiremos las actualizaciones del trabajo.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git remote
add origin https://github.com/victoriagarciah-um-es/pcd_entregable2_victoria_gustavo.git

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git branch
-M development

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git remote
set-url origin https://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@github.com/victoriagarciah-um-es/pcd_entregable2_victori
a_gustavo
```

En este punto ya estamos en disposición de añadir cualquier fichero a nuestro repositorio git mediante el comando git add. Si después ejecutamos el comando git status para ver el estado actual de nuestro repositorio veríamos que nuestro fichero .py está en el área preparatoria (staging area) antes del commit.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git status
On branch development
Your branch is up to date with 'origin/development'.
```

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        diagrama_casos_uso_def.uxf
        diagrama_clases_def.uxf
        diagrama_secuencia_def.uxf
```

Vemos en un primer status que tenemos que añadir al git los tres diagramas que hemos añadido a nuestro repositorio local.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git add dia
grama_casos_uso_def.uxf

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git add dia
grama_clases_def.uxf

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git add dia
grama_secuencia_def.uxf

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git status
On branch development
Your branch is up to date with 'origin/development'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   diagrama_casos_uso_def.uxf
        new file:   diagrama_clases_def.uxf
        new file:   diagrama_secuencia_def.uxf
```

Ahora, con un segundo status vemos que los archivos están preparados para ser subidos.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git commit
-m "Añadimos todos los diagramas"
[development d0406b1] Añadimos todos los diagramas
 3 files changed, 881 insertions(+)
 create mode 100644 diagrama_casos_uso_def.uxf
 create mode 100644 diagrama_clases_def.uxf
 create mode 100644 diagrama_secuencia_def.uxf

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git push -u
 origin development
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.53 KiB | 862.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/victoriagarciah-um-es/pcd_entregable2_victoria_gustavo
   001b87b..d0406b1  development -> development
branch 'development' set up to track 'origin/development'.
```

Además, hemos subido nuestra primera versión del código, y con todos estos archivos ya estamos preparados para hacer un primer tag de la evolución de nuestro trabajo.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git tag -a
tag1 -m "Codigo sin pruebas y diagramas"

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable2_victoria_gustavo>git push or
igin --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 179 bytes | 179.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/victoriagarciah-um-es/pcd_entregable2_victoria_gustavo
 * [new tag]         tag1 -> tag1
```

Por último, aunque no lo vayamos a utilizar ahora, añadimos la otra rama de nuestro trabajo, main, que será en la que subiremos el proyecto final.

```
C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable1_victoria_gustavo>git branch -M main

C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\pcd_entregable1_victoria_gustavo>git push -u origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'main' on GitHub by visiting:
remote:      https://github.com/victoriagarciah-um-es/pcd_entregable1_victoria_gustavo/pull/new/main
remote:
To https://github.com/victoriagarciah-um-es/pcd_entregable1_victoria_gustavo
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

2. PYTEST

Vemos que los 6 test que hemos creado pasan todos.

```
PS C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\practicas\ENTREGABLE2> pytest tests.py
============================= test session starts =============================
platform win32 -- Python 3.9.12, pytest-7.1.1, pluggy-1.0.0
rootdir: C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\practicas\ENTREGABLE2
plugins: anyio-3.5.0
collected 6 items

tests.py ......                                                          [100%]

============================== 6 passed in 0.24s ==============================
PS C:\Users\fghel\Documents\VICTORIA\datos\segundo\segundo_cuatri\programacion\practicas\ENTREGABLE2>
```

3. ELECCIÓN DE LOS PATRONES DE DISEÑO

Los cuatro patrones de diseño que hemos elegido para el trabajo son SINGLETON, OBSERVER, CHAIN OF RESPONSABILITY Y STRATEGY.

1. SINGLETON (ÚNICO)

El objetivo del patrón singleton es solo tener una instancia de una clase y proporcionar un punto de acceso global a ella, acceso controlado a la única instancia. Permite el refinamiento de operaciones y la representación.

2. OBSERVER

El propósito de observer es definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda a objeto observado. El Observer permite que los sujetos y los observadores estén desacoplados. Esto significa que los cambios en el sujeto no requieren cambios en los observadores y viceversa.

3. CHAIN OF RESPONSABILITY

Utilizamos este patrón para evitar acoplar el emisor de una petición a su receptor, dando a más de un objeto la posibilidad de responder a una petición. Encadena los objetos receptores y pasa la petición a través de la cadena hasta que es procesada por algún objeto.

4. STRATEGY

Utilizamos este último patrón para elegir una de las distintas formas que tenemos para ejecutar la primera parte de la chain of responsability. Este patrón define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.