Trabajo Práctico Integrador

Tema: “Sistemas de Gestión de la Configuración”

Integrantes:

Arrúa, Ángel Alejandro

Blanche, Mateo Gabriel

Cammisi, José Carlos

Malatesta, Giovanni Exequiel

Yucci, Franco

FECHA DE ENTREGA: 26 de Septiembre de 2025

Consigna:

Para afrontar los altos estándares de calidad que demanda la implementación del sistema planteado en el Caso de Estudio, se deberá definir un mecanismo de control de versiones.

Para esto, utilizaremos Github como herramienta y "Gitflow" como flujo de ramificaciones de GIT (git branching workflow).

Consignas:

1. **Crear un repositorio GIT en Github. Otorgar permisos de lectura a los docentes de la cátedra.**

Repositorio creado en: https://github.com/Cammisi/TpIngCalidad/tree/main

2. **Subir al repositorio un software de su elección (preferentemente en Java pero pueden utilizar el lenguaje y framework que deseen). Sobre este software:**
    a. **Realizar una modificación y publicarla (Push)**

Para realizar esta modificación realizamos cambios en clases de modelo (Cliente.java, Producto.java, etc) y luego mediante el uso de los siguientes comandos subimos los cambios al repositorio remoto:

git add .
git commit -m "Agregamos atributos en Cliente, Pedido y Producto"
git push

    b. **¿Cómo hacemos para no subir cambios de configuraciones locales? Configurar el repositorio para ignorar estos archivos.**

Para no subir cambios de configuraciones locales al repositorio remoto lo que hicimos fue configurar un archivo llamado .gitignore donde se configuran todas aquellas cosas que se deben omitir (no subir) al hacer un *git add .* y un posterior *git push*

    c. **Crear un nuevo branch para preparar el Release 1**

Para crear una nueva branch para desarrollar el Primer Release de nuestro proyecto ejecutamos el siguiente comando en la terminal:

git checkout -b release-1

    d. **Realizar modificaciones en Release 1**

Se realizaron las modificaciones que consisten en agregar atributos en la clase Cliente.java

    e. **Llevar al entorno productivo Release 1. ¿Cómo lo hace siguiendo Gitflow?**

Para llevar los cambios realizados en Release 1 al entorno productivo siguiendo GitFlow ejecutamos los siguientes comandos:

git add .
git commit -m "Agregar atributos en Cliente.java"
git push origin release-1
git checkout main
git merge release-1
git push origin main

    f. **Se encontró un error en la versión productiva, ¿Cómo lo corregimos? realizar una nueva rama para corregir este problema siguiendo GitFlow.**

Para corregir el error detectado, creamos una rama mediante el uso del comando *git checkout -b hotfix-1* . Luego, realizamos los cambios correspondientes y luego ejecutamos los comandos:

git add .

git commit -m "arreglar problema en main Cliente"

git push origin hotfix-1

g. Llevar al entorno productivo los cambios que corrigen el problema.

Para llevar al entorno productivo los cambios que corrigen el problema lo que debemos hacer es cambiarnos de rama (a la rama main) y mergear los cambios realizados en la branch *hotfix-1.* Esto lo podemos realizar con la siguiente secuencia de mensajes:

git checkout main

git merge hotfix-1

git push origin main

h. Crear otra rama para incorporar una nueva funcionalidad.|

Para crear una nueva rama para el desarrollo de una nueva funcionalidad ejecutamos por consola el siguiente comando:

git checkout -b feature-nueva-funcionalidad

i. Sobre esta nueva rama, creada en el punto anterior, realizar una modificación A y publicarla (push). Luego realizar una modificación B y publicarla (push). Deshacer la modificación B volviendo al estado en A.

Para realizar esto se siguieron los siguientes pasos ejecutando los siguientes comandos:

**** se realizó la modificación A****

git add .

git commit -m "Agregar nuevos atributos en la Clase Coordenada"

git push origin feature-nueva-funcionalidad

**** se realizó la modificación B****

git add .

git commit -m "Se agrego el metodo toString a la Clase Coordenada"

git push origin feature-nueva-funcionalidad

Luego, para eliminar los cambios realizados en la modificación B ejecutamos los siguientes comandos:

git reset --hard HEAD~1

git push origin feature-nueva-funcionalidad --force

j. Llevar los cambios de la rama creada en el punto anterior a producción. ¿Cómo lo hace siguiendo Gitflow

Para llevar los cambios realizados anteriormente a producción ejecutamos los siguientes comandos por consola:

git checkout main

git merge feature-nueva-funcionalidad

git push origin main

3. ¿Cómo podemos documentar con git?

a. Realizar un readme para el código subido. ¿Qué documentarían allí? Versionar el README en el repositorio.

Se modificó el README creado por defecto por GitHub añadiendo información necesaria sobre el repositorio y se subieron dichos cambios mediante la ejecución de los siguientes comandos:

git add README.md

git commit -m "Añadido README"

git push origin main

b. Si un externo realiza una modificación, necesitamos entender el cambio realizado (PR o Pull Request). ¿Qué datos le pediría a esa persona que complete? ¿Qué nos ofrece GitHub para ayudarnos con esto?

Cuando un externo realiza una modificación y la somete a través de un Pull Request, es esencial que proporcione un título claro, una descripción detallada, instrucciones para probar los cambios, el impacto esperado, y cualquier referencia relevante. GitHub nos facilita la colaboración mediante plantillas de PR, revisiones colaborativas, y herramientas de verificación automática, permitiendo un flujo de trabajo eficiente y una fácil comprensión de las contribuciones.