UNIVERSIDAD DE LA FRONTERA

Mini-Taller Semana 2

Convenciones Git

Importante:

- El presente mini taller es de carácter individual.
- Queda estrictamente prohibido copiar. Utilizar las respuestas de otra persona y presentarlas como propias invalidará la entrega de su mini taller.
- El mini taller no posee nota, siendo su situación final cada semana determinada por uno de los siguientes tres escenarios:

  a) No presenta el mini taller, o su respuesta se encuentra en blanco.

  b) Entrega pero no cumple con lo solicitado (presenta al menos un 50% de lo que se solicita, pero no cumple con la totalidad de lo solicitado).

  c) Cumple (el mini taller fue respondido en su totalidad).

- El plazo máximo de entrega corresponde al día lunes 26 de agosto de 2024 a las 13:00 hrs.
- No se aceptarán entregas fuera de plazo.
- Para entregar este mini taller usted debe tener un repositorio llamado de acuerdo al siguiente formato:
  - **mini_talleres_nombre_apellido.**
- El mini taller debe ser entregado a través del link de github que se especifica en las instrucciones.
- No se aceptarán otros formatos de entrega además de los especificados en el punto anterior.

Instrucciones:

a) Realice un listado resumen de las principales convenciones git utilizadas actualmente en el desarrollo de software.

b) Simule 5 commits, escribiendo su título y descripción en base a las convenciones estudiadas.

c) Cree una rama en su repositorio de mini talleres en github, la cual se llame “**mini_taller_semana_2**”, y suba a aquella rama un archivo pdf que contenga lo solicitado.

UNIVERSIDAD DE LA FRONTERA

Desarrollo Actividades:

1. Listado de las principales convenciones git:
    1. **Estructura del mensaje en un commit:** Los mensajes de un commit deben seguir una estructura o modelo determinado, esto debido a que se debe proporcionar la información con claridad y consistencia. Ejemplo de estructura a seguir:
        - <type> [optional scope]: <description>

          [optional body]

          [optional footer(s)]

    2. **Tipos de commit:** Existen distintos tipos de commit, cada uno correlacionado con una acción en específica, los principales y más usados son:
        - **[fix]**: Corrección de un error en la base del código.
        - **[feat]**: Introduce nuevas características en la base del código.
        - **[BREAKING CHANGE]**: Introduce un cambio importante que rompe la compatibilidad en el uso de la API.
    3. **Otros tipos de commit:** Existen otros tipos de commits permitidos según las convención de Angular que incluyen:
        - **[chore]**: Cambios menores no relacionados con la funcionalidad o corrección de errores. (Ej: Actualización de dependencias).
        - **[docs]**: Actualización de documentación.
        - **[style]**: Cambios orientados al formato del código y no afectan la lógica. (Ej: espaciados, puntos, comas, etc.).
        - **[refactor]**: Cambios en el código que no agregan nuevas características.
        - **[perf]**: Mejoras de rendimiento.
        - **[test]**: Añadir o corregir pruebas automáticas.

4. **Estructuras opcionales:** Estas estructuras son utilizadas para proporcionar un mensaje aún más claro y conciso, pero, a diferencia del <type> y <description>, no son de carácter obligatorio en un mensaje commit, estos son:
    - **[scope]** (ámbito): Proporciona información de donde se realizó el cambio.
    - **[body]** (cuerpo): Proporciona información adicional o necesaria acerca del cambio realizado.
    - **[footer]** (nota de pie): Proporciona referencias acerca de problemas o cambios importantes para el cambio realizado.
5. **Commits frecuentes:** Se debe realizar commits de manera frecuente, y de un solo tipo. No puede existir un commit que pertenezca a más de un tipo, si es el caso, se deben realizar commits pequeños hasta alcanzar el estado deseado, con el fin de mantener un orden acerca de los cambios realizados en la base del código.

2. Simulación de commits:

1. style: aplicar formato a los archivos CSS

   Se aplicó un formato consistente a los archivos CSS, corrigiendo la indentación y eliminando espacios en blanco.

2. chore: actualizar dependencias del proyecto

   Actualización de las dependencias del proyecto a sus versiones más recientes.

3. feat(usuario): agregar funcionalidad de registro de usuarios

   Se implementó la funcionalidad de registro de nuevos usuarios con validación de email y contraseña.

4. test(usuario): agregar pruebas unitarias para el módulo de autenticación

Se añadieron pruebas unitarias para verificar la autenticación de usuarios.

5. fix(frontend): corregir error de diseño en dispositivos móviles

   Se ajustaron estilos CSS para corregir problemas que afectan la visualización de imágenes.