Tarea 1: Preguntas Teóricas

Daniel Urbina Siezar*, Esteban Morera Ulate*

*Escuela de Ingeniería Mecatrónica,
Instituto Tecnológico de Costa Rica (ITCR), 30101 Cartago, Costa Rica,
{marurbina, emorera}@estudiantec.cr

1. **¿Explique la principal utilidad de Git como herramienta de desarrollo de código?**
   La principal utilidad de Git para el desarrollo de código son las capacidades que tiene para el manejo de versiones de forma colaborativa, permitiendo tener acceso al historial completo de las versiones del proyecto. Además, permite que múltiples personas trabajen en un mismo proyecto sin sobrescribir el trabajo realizado de manera simultánea. Git también permite crear *branches* para desarrollar nuevas funciones de forma controlada, corregir errores y realizar prototipado. Adicionalmente, Git puede identificar cuando ciertos cambios en el código entran en conflicto y ayuda a proceder de la mejor forma, reduciendo el riesgo de pérdida de avances en el código.
2. **Explique la diferencia entre Git y GitHub**
   Git es el sistema de control de versiones como tal, este puede funcionar de forma local en una computadora y sirve para gestionar versiones del código sin necesidad de conexión a internet. Por otro lado, GitHub es una plataforma que utiliza Git en la nube, permitiendo alojar repositorios en internet, trabajar de forma colaborativa y automatizar pruebas. GitHub requiere conexión a internet y funciona como un servicio.
3. **¿Qué es un branch?**
   Un branch funciona para realizar cambios al código sin necesidad de afectar el código principal (*main*), esto permite desarrollar nuevas funcionalidades, corregir errores presentes en el *main* y también para experimentar, además de permitir trabajar en equipo sin interferencia con las otras personas que colaboran. Una vez se esté satisfecho con los cambios realizados en un branch, se puede hacer un merge con el *main*, lo cual implementa los cambios deseados.
4. **En el contexto de GitHub, ¿qué es un Pull Request?**
   En GitHub, un Pull Request es una solicitud para implementar los cambios realizados en un branch, pasando por un proceso de revisión y discusión del código antes de realizar el *merge* con el *main*. Este proceso evita que se integren códigos con errores al código principal y deja un registro de los cambios realizados y la razón de dichos cambios.
5. **¿Qué es un commit?**
   Un commit permite guardar todos los cambios realizados hasta un momento específico junto con una descripción. Este contiene información como el autor, la fecha y un identificador único. Un commit representa un punto específico en el historial del proyecto y es inmutable, lo que significa que Git no modifica commits anteriores, sino que crea nuevos. Esto facilita el entendimiento de los cambios realizados y la detección del momento en que se introdujo un error en el código.
6. **Describa lo que sucede al ejecutar la siguiente operación: git rebase main**
   Al ejecutar git rebase main, Git toma los commits de la rama actual y los vuelve a aplicar uno por uno sobre la rama *main* hasta llegar a su versión más reciente. Esto facilita un historial lineal y limpio, asignandole nuevos identificadores a los commits. En caso de que Git detecte conflictos, el proceso se detiene hasta que el usuario resuelva los conflictos.
7. **Explique qué es un merge conflict y cómo lo resolvería**
   Un *merge conflict* ocurre cuando Git no puede combinar automáticamente los cambios realizados en dos ramas porque una misma parte del código fue modificada en ambas. Para resolverlo, Git pide al usuario decidir cuál versión debe mantenerse o cómo combinar los cambios. También pueden ocurrir conflictos cuando una rama renombra o mueve un archivo y otra lo edita. Para reducir la probabilidad de que sucedan estos conflictos, se recomienda realizar commits pequeños y frecuentes, y sincronizar la rama con regularidad.
8. **¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?**
   Una prueba unitaria es una prueba automatizada que verifica que una unidad pequeña del código funcione correctamente de forma aislada. Esta unidad puede ser una función, un método, una clase o un módulo pequeño. Su objetivo es detectar errores de forma temprana, evitar que cambios realizados en el futuro dañen funcionalidades existentes y fortalecer la documentación del código. Una prueba unitaria debe ser repetible, rápida, aislada y determinista, implicando que los resultados para una prueba siempre deben ser los mismos para poder tener seguridad en esta.
9. **Bajo el contexto de pytest, ¿cuál es la utilidad de un assert?**
   En pytest, un assert permite comprobar si los resultados obtenidos coinciden con los esperados, determinando si una prueba es aprobada o fallida. Su uso permite detectar errores de forma automática y expresar claramente el

comportamiento esperado del código. Es común utilizarlo con comparaciones de valores, condiciones lógicas y comprobación de excepciones.

10. **Mencione y explique tres errores de formato detectables con Flake8**
    Flake8 es una herramienta de análisis para Python que permite detectar diversos errores de formato y estilo, entre ellos:
    - **Espacios en blanco incorrectos (E225/E231):** Detecta el uso incorrecto de espacios alrededor de operadores o después de comas, lo cual reduce la claridad y rompe la consistencia del estilo del código.
    - **Indentación incorrecta (E111/E114):** Detecta una indentación inconsistente o incorrecta, lo cual es problemático debido a que la indentación define la lógica de ejecución en Python.
    - **Exceso de longitud de línea (E501):** Detecta líneas de código que superan el límite recomendado de caracteres establecido por PEP 8, lo cual afecta la legibilidad y el mantenimiento del código.