**Markdown**

es un lenguaje de marcado ligero diseñado para que sea fácil de leer y escribir, tanto en su forma original como ya procesada (formateada). Se utiliza ampliamente en entornos de desarrollo, documentación técnica y plataformas colaborativas como GitHub, Reddit, foros, blogs, wikis, entre otros.

Git

1 ¿Qué es un repositorio en Git y como se diferencia de un proyecto “normal”?

Es un directorio especial que contiene todos los archivos y el historial de versiones de un proyecto, un proyecto normal es una carpeta con archivos. Un repositorio Git es un proyecto que usa Git para registrar cambios, gestionar versiones y facilitar la colaboración.

2 ¿Cuáles son las tres áreas principales de Git?

* Working Directory (directorio de trabajo): Es la carpeta donde tú editas los archivos.
* Staging Area (área de preparación): Es como una lista donde dices qué cambios quieres guardar.
* Repository (repositorio): Es donde Git guarda todo el historial de cambios.

3 ¿Cómo representa Git los cambios internamente?

Git guarda los archivos como objetos:

* Blob: Guarda el contenido del archivo.
* Tree: Guarda las carpetas y cómo se organizan.
* Commit: Guarda la historia de los cambios.
* Tag: Es como una etiqueta que marca una versión importante.

4 ¿Cómo se crea un commit y qué guarda?

Un commit se crea con Git commit. Guarda:

* Los cambios que hiciste.
* Quién los hizo.
* Cuando se hicieron.
* Un mensaje explicando el cambio.

5. ¿Cuál es la diferencia entre git pull y git fetch?

* git fetch: solo trae los cambios del servidor, pero no los aplica.
* git pull: trae y aplica los cambios automáticamente.

6. ¿Qué es un branch (rama)?

Es una línea de trabajo separada. Git usa ramas para que puedas trabajar en nuevas ideas sin dañar el proyecto principal.

7. ¿Cómo se hace un merge y qué conflictos hay?

* Merge es juntar dos ramas.
* A veces, si dos personas cambian el mismo texto, hay un conflicto.
* Git te pide que elijas qué parte conservar.

8. ¿Cómo funciona el área de staging (git add)?

git add manda los cambios al área de preparación (staging).  
Si no haces este paso, Git no guarda ese cambio cuando haces el commit.

9. ¿Qué es? gitignore?

Es un archivo donde tú le dices a Git qué cosas no debe guardar, como carpetas temporales o archivos privados.

10. ¿Qué diferencia hay entre --amend y un nuevo commit?

* --amend: cambia el último commit (corrige algo).
* Nuevo commit: agrega uno nuevo al final.

11. ¿Para qué sirve git stash?

Guarda cambios temporalmente si necesitas cambiar de tarea sin perder lo que llevabas.

12. ¿Cómo deshacer cambios en Git?

Git tiene comandos para deshacer errores:

* git reset: borra cambios en el historial.
* git revert: crea un nuevo commit que revierte otro.
* git checkout: regresa archivos a un estado anterior.

13. ¿Cómo se configuran los remotos (origin, upstream)?

* Origin: es el repositorio principal (donde subes tu código).
* Upstream: es otro repositorio, como el original de donde hiciste una copia (fork).

Se usan comandos como:

git remote add origin <url>

git remote add upstream <url>

14. ¿Cómo ver el historial de cambios?

Puedes ver qué pasó con estos comandos:

* git log: muestra la historia de todos los commits.
* git diff: muestra exactamente qué cambió.
* git show: muestra un commit completo (contenido, autor, fecha...).

PROGRAMACIÓN

15. ¿Cuáles son los tipos de datos primitivos en Java?

Son los datos más básicos que Java puede usar:

* int: números enteros (como 5)
* double: números con decimales (como 3.14)
* char: una sola letra (como 'A')
* boolean: verdadero o falso (true o false)
* byte, short, long, float: otros tipos de números, más grandes o pequeños

16. ¿Cómo funcionan las estructuras de control de flujo como if, else, switch y bucles en Java?

* if / else: El programa toma decisiones. Si pasa algo, hace una cosa. Si no, hace otra.
* switch: Elige entre varias opciones dependiendo del valor.
* while / for: Son repeticiones. Hacen algo muchas veces.

17. ¿Por qué es importante usar nombres significativos para variables y métodos?

Porque así es más fácil entender el código.

18. ¿Qué es la Programación Orientada a Objetos (POO)?

Es una forma de programar donde todo se basa en objetos, que tienen datos (atributos) y acciones (métodos).

19. ¿Cuáles son los cuatro pilares de la POO?

1. Abstracción: Mostrar lo importante, esconder lo complicado.
2. Encapsulamiento: Proteger los datos dentro del objeto.
3. Herencia: Reutilizar código de otros objetos.
4. Polimorfismo: Usar una misma acción, pero que se comporte diferente según el caso.

20. ¿Qué es la herencia en POO y cómo se utiliza en Java?

La herencia permite que una clase (hijo) herede cosas de otra (padre).

21. ¿Qué son los modificadores de acceso y cuáles son los más comunes en Java?

Controlan quién puede ver o usar algo en tu programa:

* public: cualquiera puede usarlo.
* private: solo se usa dentro de la misma clase.
* protected: se usa dentro de la clase o sus hijos.

22. ¿Qué es una variable de entorno y por qué son importantes para Java o la programación?

Es una configuración del sistema que el programa usa para funcionar bien (como saber dónde está instalado Java).