Glosario CH31

# Indice

[Frontend](#tfjhq7lczxfl)

[Backend](#9eid2irnas88)

[API](#fospf747wgii)

[Interfaz](#7dfdvuzbocd7)

[Lenguajes de programación](#du3jzi8lol5m)

[Internet](#hwfvpyge417z)

[Cliente/Servidor](#rseg9qwwfvs)

[Solicitudes HTTP](#m6qsk8smom9t)

[CRUD](#np1qszae3p85)

[Respuestas del servidor](#pn0xsubbx9e1)

[TCP/IP](#dot0i5wrub38)

**Frontend** la parte visual

**Ux / UI:** User Experiencie / User Interface

**HTML:** Hypertext Markup Language

**CSS:** Cascade Style Sheet

**JS:** JavaScript

**Bootstrap:** framework de diseño y desarrollo web

**Validaciones**

**Consumo de APIs**

**Lógica en el cliente**

**Interface web**

**Backend** lo que no vemos, el cerebro, lógica

**Java:** Lenguaje de programación en la parte del servidor

**SQL:** Structured Query Language, lenguaje para gestionar y manipular bases de datos

**Spring Boot:** framework de desarrollo que facilita la creación de aplicaciones Java. Reduce configuraciones complejas y permite centrarse en la lógica de negocio.

**Bases de datos**

**Rendimiento**

**Lógica en el servidor**

**Seguridad**

**Microservicios**

**API** Interfaz de programación de aplicaciones



**Interfaz** Intermediario, puente, conexión



**Sintaxis**: Forma de “escribir”

**Lenguajes de alto nivel:** se asemeja al lenguaje humano y utiliza muchas palabras (Java, Python)

**Lenguajes de bajo nivel:** lenguaje binario (0 y 1)

**Tipado:** cómo se declaran las variables

**Declarar las variables:** especificar nombre y tipo de variable

**Variable:** donde se guardan los datos de un programa

**Internet:** redes interconectadas

**WWW:** World Wide Web / Red Mundial

Red:

**Cliente:** quien solicita y recibe servicios o recursos

**Servidor:** proporciona servicios, recursos o información

**Protocolo:** Forma de hacer las cosas

**Servicios:** Información, recursos, imágenes, videos, etc.

**Base de datos:** Información categorizada y organizada

**Solicitudes GET - POST - PUT - DELETE**

**Get:** obtener

**Post:** agregar

**Put:** modificar

**Delete:** borrar

**CRUD:** **C**reate, **R**ead, **U**pdate, **D**elete, representa operaciones básicas para gestionar información en bases de datos

Códigos de respuesta para conexiones del Servidor

**200:** Aquí está la hamburguesa, ok

**301:** La hamburguesa la venden en otro restaurante, Movido permanentemente, Redirección

**302:** Movido temporalmente

**401:** No tiene permitido comprar hamburguesas, Unauthorized

**404:** Hamburguesa no encontrada, Not found

**500:** La cocina se prendió fuego, Internal server error

**El servidor** (quien prepara nuestra hamburguesa) toma todos los datos de nuestra petición (carne, pan, jitomate, lechuga, etc), la manda **al internet** (el encargado del mostrador) quien es quien entrega **nuestra petición** tal cual la pedimos (el cliente).

**TCP/IP**: protocolo de control de transmisión / protocolo de internet

**HTTP:** Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol)

**Algoritmo:** conjunto de instrucciones o reglas que guían la realización de una tarea o resolución de un problema. Es como una receta que te dice exactamente qué hacer en cada paso para alcanzar un resultado deseado

**lista de comandos terminal**

**pwd:** print word directory te da una mejor explicaciónde donde estas trabajando (la ruta de dirección)

**ls:** te enlista los elementos que se encuentran en la ruta

-**parametros:** usas el - después del comando

- l : formato long

-a: all muestra todos

-R: hacer el comando recursivo

**clear:**  limpiar la ventana de comando

**history:** muestra un historial de los comando usados

**cd:** change directory/ cambiar de directorio **sintaxis: cd directorio**

parametro: .. (espacio punto punto) regresar un ruta del directorio

**mkdir:** make directory

**rmdir:** remove directory

el signo $ indica que esta listo para llevar a cabo un comando si no aparece esta llevando a cabo un proceso

**cp:** copiar sintaxis: cp Cama.txt ../CuartoMemo/

**rm:** remove

la estructura de un comando es:

que quiero hacer / como lo quiero hacer / a quien lo voy a hacer



**GIT:** sistema de control de versionado

Git hub permite en un solo archivo tener versiones de tal manera que si quieres puedes volver a algún punto en específico cada versión viene con un comentario y continuar el flujo hasta el punto deseado y volver a a cualquier versión posible que tenga un comentario

**Repositorio:** Es un proyecto que se encuentra en la nube. esto permite que varias personas trabajen en un mismo código, algo similar como documentos de drive

Comandos Git

Usamos el prefijo **git** para especificar

- **git init** haces que el directorio actual se convierta en un repositorio

- **git status** te dice el estado actual del repositorio

Ejemplo: dice que el archivo de glosario de terminos tecnicos no está trackeado

**- git add** agrega el archivo para poder trabajarlo

add . (“add espacio punto“) es para agregar a la mesa de trabajo todos los documentos de directorio

**git commit** se usa para agregar el comentario, pero nos abre el editor de texto que en este caso es vim

-m nos permite agregar el comentario seguido de el comando ej: git commit **-m** “se hicieron cambio a este archivo ”

**git log:** Te muestra los commits realizados a los documentos

**git push:** envía los cambios a la red para que quede de manera global

**git remote add origin:** es para poder decir cual es el repositorio de github con el que mi carpeta de proyecto estará sincronizada

ejemplo: git remote add origin <https://github.com/Angelhdz98/Glosario_Generation.git>

nota:

Parametros git

-- version: no dice la versión de git o programa del prefijo

