

Nombre

JHON CAMILO SUAZA ST.

Fecha

dd

mes

Profesor

Materia

Institución

Curso

Nota

Silla

color
Material
Tamaño
Ocupar()
Desocupar()

Mesa

material
Altura
Color
Limpieza()
Usar()

Estudiante

nombre
edad
grado
Estudiar()
Participar()

profesor

nombre
materia
Experiencia
enseñar()
Calificar()

Puerta

material
Color
Abrir()
Cerrar()

Ventana

tipo
tamaño
material
Abrir()
Cerrar()

Computador

marca
modelo
memoria
Encender()
Apagar()

Luz

Encendido
Nivel brillo
Encender()
Apagar()

Aire

velocidad
marca
tamaño
Encender()
Apagar()

Cuaderno

color
tamaño
Cantidad hojas
Organizar información()
Planificar()

Lapiz

color
largo
tipo
Escribir()
Sacapuntas()

Borrador

color
forma
material
borrar()
Limpieza()

bolso

color
tamaño
peso
guardar()
Abrir()

Reloj

marca
hora
tipo
tamaño
color
mostrar hora()
mostrar dia()
Comprobar hora()

Peso

material
color
tamaño
Limpieza()
ensuciar()

Pared

color
material
largo
Ancho
Pintar ()
medir()

Bolsillo

color
tamaño
material
llenar ()
vaciar ()

Fuente Corriente

voltaje
color
conectar ()
desconectar ()

Sala

nombre
capacidad
función
ambiente educativo ()
lugar de aprendizaje ()

House

marca
color
tipo
mover ()
conectar ()

Teclado

marca
teclas
tamaño
escribir ()
conectar ()

CPU

marca
velocidad
memoria Ram
encender ()
procesar ()

Impresora

marca
tipo
colores
imprimir ()
escribir ()

Parlante

marca
volumen
color
reproducir ()
silenciar ()

Microfono

tipo
marca
guardar ()
silenciar ()

Camara

marca
resolución
tipo
encender ()
capturar ()

Cable

tipo
longitud
color
conectar ()
desconectar ()

Audifono

marca
tipo
volume
escuchar ()
silenciar ()

Memoria USB

capacidad
marca
color
conectar
guardar

Router

marca
velocidad
tipos conexión
conectar
desconectar

Poo

La programación orientada a objetos (poo) es una forma de crear programas organizando el código en "objetos" que son como pequeñas entidades que tienen características (atributos) y acciones (métodos). En lugar de escribir todo el código de forma desordenada.

La Poo permite agrupar lo que pertenece un mismo concepto.
Ej: Si estamos haciendo un programa una tienda, podríamos tener un objeto llamado producto con atributos como nombre, precio y cantidas y métodos como vender() o actualizar(). Así el código se vuelve más fácil de entender,

Clase

En programación Orientada a objetos una clase es como un molde o plantilla que sirve para crear objetos. Dentro de la clase se definen las propiedades (también llamadas atributos) y las acciones (métodos) que los objetos de ese tipo podrán tener. **Ej:** Si creamos una clase perro, pero podría tener atributos como nombre, raza y edad, y métodos como ladrar() o correr(). Luego apartir de esa clase, se pueden crear distintos objetos como un perro llamado "firubis" o "luna", cada uno con sus propios valores.

Objeto

Un objeto es un elemento creado apartir de una clase, es decir, una representación concreta de ese molde. Cada objeto tiene sus propios valores y puede realizar acciones definidas para la clase. **Ej:** Si tenemos la clase perro podemos crear un objeto llamado mi perro con nombre = "Rocky", raza = "Labrador" y edad = 3. Este objeto puede ejecutar acciones como ladrar() o correr().
 En resumen, una clase define como serán los objetos, y los objetos son las versiones reales que existen.

Atributo

Un atributo es una característica o propiedad que describe un objeto dentro de la programación orientada a objetos. Los atributos guardan información específica sobre cada objeto creado apartir de una clase. **Ej:** Si tenemos una clase perro, los atributos pedirán ser nombre, raza y edad. Entonces si creamos un objeto llamado mi perro podríamos asignarle nombre = "Rocky", raza = "Labrador" y edad = 3 en pocas palabras, los atributos son los datos que definen como es cada objeto.

Método

Un método es una acción o comportamiento que puede realizar un objeto en la programación orientada a objetos. Los métodos se definen dentro de una clase y permite que los objetos hagan cosas o modifiquen sus atributos. **Ej:** Si tenemos una clase perro podria tener métodos como ladriar() o comer(). Entonces si creamos un objeto llamado mi_perro podríamos hacer ejecutar mis perro. ladriga() para simular que el perro ladra. En resumen, los métodos son las funciones que permiten a los objetos actuar o responder.

Abstracción

La abstracción en la programación orientada a objetos es el proceso de simplificar la realidad mostrando solo los detalles importantes y ocultando lo que no es necesario. permite enfocarse por como funciona internamente. **Ej:** Cuando usamos un objeto Auto solo necesitamos saber que tiene métodos como encender() o acelerar(), pero no es necesario conocer todo el código interno que hace que el motor funcione. En resumen, la abstracción ayuda a manejar la complejidad mostrando solo lo que realmente importa.

Encapsulamiento

El encapsulamiento en la programación orientada a objetos consiste en proteger los datos de un objeto para que no puedan ser modificados directamente desde fuera de la clase. Esto se logra guardando los atributos como privados y usando métodos para acceder o cambiar su valor llamados getters y setters. **Ej:** Si tenemos una clase VentaBarataria el saldo debería ser un atributo privado, y solo se podría modificar usando métodos como depositar() o retirar(). En resumen, el encapsulamiento mantiene la información segura y evita que se altere de forma incorrecta.

Polimorfismo.

El polimorfismo en la programación orientada a objetos es la capacidad que tienen los objetos de diferentes clases para responder de distintas maneras a un mismo mensaje o método. En otras palabras, permite usar el mismo nombre de método, pero con comportamientos diferentes según el objeto que lo use. **Ej:** Si tenemos una clase Animal con un método hacerSonido(), las clases hijas perro y gato pueden tener su propia versión del método: El perro ladra y el gato miaula. Así, cuando llamamos a hacerSonido(), cada uno actua según su tipo. En resumen, el polimorfismo permite utilizar el mismo método adaptándolo a cada caso.

Herencia

La Herencia es como cuando un hijo recibe cosas de sus padres pero en la programación. En la programación orientada a objetos y métodos de otra lo que significa que puede usar lo que ya existe sin tener que volver a escribirlo.

Ej.: Si tenemos una clase animal que tiene cosas como nombre y comer(). podemos crear una clase perro que herede de Animal. Así perro ya sabrá como comer y tener un nombre pero también podrá tener algo propio como ladra(). En pocas palabras, la herencia permite aprovechar el trabajo hecho y crear nuevas clases más específicas sin empezar desde cero.

Clase padre

Una clase padre (también llamada superclase) es la clase principal de la que otras clases heredan sus atributos y métodos. Es como el modelo general del que se basan los demás. Ej.: Si tenemos una clase Animal con atributos como nombre y edad, y métodos como comer() o dormir(), esa sería la clase padre. Luego, podemos crear clases hijas como perro o gato que hereden todo eso de Animal y pueden agregar sus propias características. En resumen la clase padre es la que comparte sus propiedades con otras clases.

Clase hija

Una clase hija (también llamada subclase) es una clase que hereda los atributos y métodos de otra clase, llamada clase padre. Esto significa que puede usar todo lo que tiene clase padre y además, agregar o cambiar cosas según necesite. Ej.: Si la clase Animal es la clase padre y tiene métodos como comer() y dormir(), una clase hija llamada perro puede heredar esas funciones y añadir una nueva, como ladra(). En pocas palabras, la clase hija aprovecha lo que ya existe en la clase padre y lo adopta o amplía para crear algo más específico.

N capas

La arquitectura en N capas es una forma de organizar un programa dividiéndolo en varias partes ("capas") donde cada una tiene una función específica. Esto ayuda a que el código sea más ordenado, fácil de mantener y modificar. Ej.: Una aplicación puede tener:

Capa de presentación, que es la interfaz que ve el usuario.

Capa de lógica o negocio, donde se procesan las reglas y funciones del programa.

Capa de datos, que se encarga de guardar y obtener información de la base de datos.

En resumen, la arquitectura en **N** capas separa las responsabilidades del programa para hacerlo más claro.

No SQL

NoSQL es un tipo de base de datos que guarda la información de una manera más libre y flexible que los bases de datos normales (como las de tipo SQL). En lugar de usar tablas con filas y columnas, NoSQL puede guardar los datos como documentos, listas o pares de "clave y valor". Esto hace que sea más fácil trabajar con mucha información o con datos que cambian seguido. **Ej:** MongoDB; los datos se guardan como si fueran archivos de texto con nombre y contenido. NoSQL sirve para guardar datos sin tantas reglas, de una forma más rápida y adaptable.

JSON

JSON es una forma muy simple de guardar y compartir información entre programas. Se parece a como escribirías datos en una lista o un cuaderno, con nombres y valores **Ej:**

```
{  
  "nombre": "Ana",  
  "edad": 20,  
  "ciudad": "Madrid"  
}
```

Esto quiere decir que el nombre es Ana, la edad es 20 y la ciudad es Madrid. JSON se usa mucho porque es fácil de entender tanto para las personas como para las computadoras.

XML

XML es una forma de guardar y organizar información usando etiquetas parecida a como se escriben las páginas web en HTML. Cada dato se encierra entre una etiqueta de apertura y una cierre. por **Ej:**

```
<persona>  
<nombre>Ana</nombre>  
<edad>20</edad>  
<ciudad>Madrid</ciudad>  
</persona>
```

Así el programa puede entender qué significa cada dato. XML se usa mucho para intercambiar información entre diferentes sistemas o aplicaciones. En pocas palabras, es una manera ordenada y clara de guardar datos usando etiquetas que describen qué es cada cosa.

Nombre

JHON CAMILO SUAZO S.

Fecha

dia / mes / año

Profesor

Materia

Institución

Curso

Nota

Grafo

Un grafo es una forma de mostrar cómo se conectan unas cosas con otras. Se dibuja con puntos (que representan cosas como personas o lugares) y líneas (que muestran las conexiones entre ellos). Por (Ej): Si piensas en una red social, cada persona sería un punto, y las líneas servirían los amigos que las unen. En pocas palabras, un grafo sirve para entender relaciones o conexiones entre diferentes cosas, como amigos, calles o páginas en internet.

Escala V

↓ La escala V es la linea que va de abajo hacia arriba en un dibujo o gráfico. Sirve para mostrar o medir cosas de forma vertical. Por (Ej): en una gráfica de ventas la linea vertical (escala V) muestra los números o cantidades que representan cuánto se vendió. En pocas palabras, la escala V es la parte del costado que te ayuda a ver cuánto vale o mide algo en un gráfico.

↑ Escala H

La escala H es la linea que va de izquierda a derecha en un dibujo o gráfico. Sirve para mostrar o medir o gráfico) Sirve para mostrar o medir cosas de forma horizontal. Por (Ej): en una gráfica donde se ven las ventas por meses, los meses van escritos abajo en la linea horizontal: esa es la escala H. En palabras simples, es la parte de abajo que te ayuda a leer o comparar los datos de un gráfico.

firebase

firebase es una herramienta de google que ayuda a crear aplicaciones sin tener que programar todo desde cero. Ofrece muchas funciones ya listas como guardar datos, registrar usuarios, enviar notificaciones o analizar cómo usan la app las personas. Por (Ej): con firebase puedes hacer que una app guarde información en tiempo real o que los usuarios inicien sesión con su cuenta de google. En palabras simples, firebase es como una caja de herramientas, que facilita y acelera la creación de aplicaciones web y móviles.

MongoDB

Es una base de datos que guarda la información de una forma más flexible que las bases de datos tradicionales, en lugar de usar tablas con filas y columnas. guarda los datos en documentos parecidos a los archivos JSON que son fáciles de leer. Por **Ej:** un documento en MongoDB puede verse así:

```
{  
  "nombre": "Ana",  
  "edad": 20,  
  "ciudad": "Madrid"  
}
```

ESTO hace que sea muy práctico para guardar datos que cambian mucho o que no siguen siempre la misma estructura. En pocas palabras, MongoDB es una base de datos modelada, rápida y fácil de usar, muy común en aplicaciones web y móviles.

Docker

Es una herramienta que sirve para crear, mover y ejecutar aplicaciones de manera fácil y rápida. Funciona usando algo llamado contenedores, que son como cajas donde se guarda todo lo que una aplicación necesita para funcionar (como archivos, configuraciones y dependencias). Así, no importa en qué computadora o servidor la ejecutes, siempre funcionará igual.

Por **Ej:** Si haces un app en tu laptop y la metes en un contenedor Docker, luego puedes llevártela a otro equipo y se ejecutará sin problemas. En palabras simples, Docker ayuda a que las aplicaciones sean más portátiles y no den errores por diferencias entre equipos.

Imagen

Una imagen en Docker es como una receta que tiene todo lo que una aplicación necesita para funcionar; el sistema los archivos y las instrucciones. A partir de esa receta, Docker puede crear "contenedores" que son como platos listos hechos con esa receta. Por **Ej:** una imagen puede tener una app lista para usarse. Y cada vez que la ejecutas, Docker crea una copia lista para funcionar. En palabras simples, una imagen en Docker es el modelo que se usa para crear la aplicación que vas a correr dentro de los contenedores.

Contenedor

Un contenedor en Docker es como una caja donde se ejecuta una aplicación con todo lo que necesita para funcionar. Dentro del contenedor están los archivos de configuración y programa que la app usará y todo estará separado del resto del sistema para que no haya errores ni conflictos.

Por ej.: Puedes tener un contenedor con una página web y otro con una base de datos, y ambos funcionan sin molestar a uno al otro. En palabras simples un contenedor es un espacio aparte donde vive y corre una aplicación, lista para usar en cualquier computadora.

Puerto

Un puerto es como una puerta que usa una computadora para comunicarse con el exterior o con otros programas. Cada puerto tiene un número, que lo identifica.

y sirve para que la información llegue al lugar correcto.

Por ej.: Cuandoieres una página web tu navegador se conecta al puerto 80 o 443 que son los usados.

Por los sitios web. En Docker los puertos sirven para conectar un contenedor con el mundo fuera de él. Por ej.: para que una aplicación dentro del contenedor pueda ser vista desde tu computadora.

En pocas palabras un puerto es el camino por donde entra o sale la información.

Hub

Un hub es como un punto central de conexión donde se juntan o se componen cosas. En el caso de Docker, el Docker Hub es una página o servicio donde se guardan y comparten imágenes de Docker.

Por ej.: Si necesitas una imagen de una base de datos como MySQL o de un app como WordPress, puedes ir a Docker Hub y descargarla lista para usar. También puedes subir tus propias imágenes para que otras las uses.

En palabras simples el hub es como una tienda o biblioteca en línea donde encuentras y comparte imágenes de Docker.

Dockerfile

Un Dockerfile es un archivo donde se escriben las instrucciones que Docker usa para crear una imagen. Es como una receta paso a paso que dice qué cosas se deben instalar, copiar o configurar dentro de la imagen.

Por ej.: En un Dockerfile puedes poner cosas como:

Que sistema base usar (por ej.: Ubuntu).

Que archivos copiar.

Que comandos ejecutar.

Cuando Docker lee ese archivo, sigue las instrucciones y crea una imagen lista para usarse. En palabras simples un Dockerfile es el que te dice a Docker como construir una aplicación dentro de una imagen.

Docker Compose

Docker Compose

Docker Compose es una herramienta que te ayuda a manejar varios contenedores de Docker al mismo tiempo sin complicarte. En lugar de escribir muchos comandos solo usas un archivo (llamado Docker Compose YML) donde dices qué contenedor quiere usar y como deben trabajar juntos.

por EJ: Si tiene una aplicación que necesita una base de datos y un servidor con Docker Compose podes encenderlos los dos con un solo comando. En pocas palabras Docker Compose te permite levantar varios contenedores de una forma rápida.

Diagrama de clases

Un diagrama de clases es un dibujo que muestra como se organizan las clases en un programa y como se relacionan entre ellas. Se usa mucho en la programación orientada a objetos para planear y entender mejor el código.

Cada clase se dibuja como un recuadro con tres partes:

El nombre de la clase

sus atributos (las características).

Sus métodos (las acciones que puede hacer.)

por EJ: Una clase perro podría tener los atributos nombre y edad y los métodos ladear y correr. En palabras simples, un diagrama de clases es una forma visual de ver como están conectadas las partes de un programa.