Refuerzo de temas y conceptos - **Juan Sebastián Uribe Lesmes**
Centro de Comercio y Turismo - Ficha **3144622**

Director de ficha - **Arle Morales**

Contexto

Refuerzo de temas y conceptos

1. Haz una investigación del concepto
2- ponga un ejemplo de uso
3- si es necesario busque una analogía para entender el tema

1. ¿Qué es un computador?
2. Qué es un programa - Concepto básico de instrucciones para la computadora.
3. Lógica de programación - Pensamiento estructurado para resolver problemas.
4. Algoritmos - Pasos para resolver un problema (ej. pseudocódigo).
5. Variables - Concepto y tipos (enteros, decimales, texto).
6. Tipos de datos - Números, cadenas, booleanos.
7. Operadores - Aritméticos (+, -, *, /), lógicos (AND, OR, NOT).
8. Estructuras de control - Condicionales (if, else).
9. Bucles - Repetición (for, while).
10. Entrada y salida - Leer datos del usuario y mostrar resultados.
11. Funciones básicas - Reutilización de código.
12. Primer lenguaje (ej. Python) - Sintaxis básica y uso.
13. Comentarios en el código - Documentar para claridad.
14. Errores comunes - Cómo identificarlos (syntax error, runtime error).
15. Depuración - Uso básico de herramientas o print statements.
16. Compiladores vs. intérpretes - Diferencia simple.
17. Cadenas de texto - Manipulación básica (concatenar, longitud).
18. Hardware básico - CPU, memoria, disco duro.
19. Software - Sistema operativo vs. aplicaciones.
20. Sistemas operativos - Windows, Linux, funciones básicas.

21. Archivos y carpetas - Organización en el sistema.
22. Terminal o consola - Comandos básicos (cd, dir, ls).
23. Fundamentos de Desarrollo de Software
24. Ciclo de vida del software - Idea básica (planificar, diseñar, codificar).
25. Requisitos - Qué quiere el usuario.
26. Prototipos - Bosquejos simples de software.
27. Interfaz de usuario - Concepto de diseño básico.
28. Pruebas - Verificar que el programa funcione.
29. Qué es una base de datos - Almacenar información organizada.
30. Internet - Cómo funciona a nivel básico.
31. Direcciones IP - Identificadores simples.
32. Navegadores - Qué hacen y cómo.
33. Cliente y servidor - Interacción básica.
34. Seguridad inicial - Contraseñas y riesgos.
35. HTML - Estructura de una página web.
36. CSS - Estilo básico (colores, fuentes).
37. JavaScript introductorio - Interactividad simple (alertas).
38. Páginas estáticas - Crear algo visible.
39. Hosting básico - Subir una página a la web.
40. Editores de código - VS Code, uso básico.
41. Control de versiones - Qué es Git (concepto inicial).
42. Repositorios - Idea de GitHub.
43. Línea de comandos - Comandos útiles para programar.
44. Entornos de desarrollo - Instalación de Python o similar.
45. Metodología ágil - Idea de iteraciones cortas.
46. Documentación - Escribir cómo funciona el código.
47. Resolución de problemas - Dividir en partes.
48. Comunicación - Explicar ideas técnicas.
49. Pensamiento crítico - Evaluar soluciones.
50. Ética en TI - Uso responsable de la tecnología.
51. Privacidad - Proteger datos básicos.

52. Persistencia - Lidiar con errores y fracasos.
53. Proyecto simple - Calculadora o lista de tareas.
54. Reutilización de código - Usar funciones ya hechas.
55. Inteligencia artificial - Qué es en términos simples.
56. Tipos de archivos: doc, pnh
57. Aplicaciones móviles - Ejemplo de su uso.
58. Videojuegos - Introducción al desarrollo básico.
59. Impacto del software - Cómo cambia el mundo.
60. Aprendizaje continuo - Importancia de seguir estudiando.

Resolución

1. ¿Qué es un computador?

- **Investigación:** Un computador es un dispositivo electrónico que procesa datos mediante instrucciones predefinidas.
- **Ejemplo:** Un teléfono inteligente es un tipo de computador que permite navegar por internet y ejecutar aplicaciones.
- **Analogía:** Un computador es como una fábrica que toma materias primas (datos) y produce productos terminados (resultados).

2. ¿Qué es un programa?

- **Investigación:** Un programa es un conjunto de instrucciones que un computador ejecuta para realizar una tarea.
- **Ejemplo:** Microsoft Word es un programa que permite crear y editar documentos de texto.
- **Analogía:** Un programa es como una receta de cocina, donde cada instrucción debe seguirse en orden para obtener el plato final.

3. Lógica de programación

- **Investigación:** Es el pensamiento estructurado que permite resolver problemas a través de instrucciones lógicas.
- **Ejemplo:** Diseñar un algoritmo para organizar una lista de números de menor a mayor.
- **Analogía:** La lógica de programación es como planificar una ruta en un mapa para llegar más rápido a un destino.

4. Algoritmos

- **Investigación:** Son pasos secuenciales para resolver un problema.
- **Ejemplo:** Un algoritmo para hacer un café: hervir agua, agregar café, servir en taza.
- **Analogía:** Es como seguir las instrucciones de un manual para armar un mueble.

5. Variables

- **Investigación:** Son espacios en la memoria donde se almacenan valores.
- **Ejemplo:** En un programa de calculadora, una variable puede almacenar el número ingresado por el usuario.
- **Analogía:** Una variable es como una caja etiquetada donde guardamos objetos específicos.

6. Tipos de datos

- **Investigación:** Son las diferentes categorías de información que una variable puede almacenar.
- **Ejemplo:** Números enteros (5), decimales (3.14), cadenas de texto ("Hola"), booleanos (true/false).
- **Analogía:** Los tipos de datos son como los diferentes tipos de contenedores en una cocina: frascos para líquidos, cajas para sólidos.

7. Operadores

- **Investigación:** Son símbolos que permiten realizar operaciones matemáticas y lógicas.
- **Ejemplo:** `5 + 3` da como resultado `8`.
- **Analogía:** Son como herramientas en un taller: un martillo para clavar, una llave para ajustar.

8. Estructuras de control

- **Investigación:** Son instrucciones que permiten tomar decisiones en un programa.
- **Ejemplo:** Si una persona tiene más de 18 años, puede votar; si no, no puede.
- **Analogía:** Es como un semáforo que decide si los coches deben avanzar o detenerse.

9. Bucles

- **Investigación:** Son estructuras que permiten repetir una acción varias veces.
- **Ejemplo:** Un bucle que imprime "Hola" cinco veces en pantalla.
- **Analogía:** Es como una lavadora que repite ciclos de lavado hasta completar el programa.

10. Entrada y salida

- **Investigación:** Es la interacción entre el usuario y el programa mediante datos ingresados y resultados mostrados.
- **Ejemplo:** Un formulario web donde se ingresa un nombre y el sistema responde con "Hola, [nombre]".
- **Analogía:** Como una conversación donde una persona pregunta y otra responde.

11. Funciones básicas

- **Investigación:** Son bloques de código reutilizables que ejecutan una tarea específica.
- **Ejemplo:** Una función en Python que suma dos números.
- **Analogía:** Una función es como una máquina expendedora: ingresas un código y obtienes un producto.

12. Primer lenguaje de programación (Python)

- **Investigación:** Python es un lenguaje fácil de aprender con una sintaxis clara y concisa.
- **Ejemplo:** `print("Hola, mundo")`.
- **Analogía:** Python es como un lenguaje sencillo para escribir instrucciones sin complicaciones.

13. Comentarios en el código

- **Investigación:** Son notas dentro del código que explican su propósito.
- **Ejemplo:** `# Esto es un comentario en Python`.
- **Analogía:** Son como notas adhesivas en un libro para recordar puntos clave.

14. Errores comunes

- **Investigación:** Son fallos en el código que impiden su correcta ejecución.
- **Ejemplo:** `SyntaxError: Unexpected EOF while parsing`.
- **Analogía:** Un error en el código es como una señal de tránsito mal colocada que confunde a los conductores.

15. Depuración

- **Investigación:** Es el proceso de encontrar y corregir errores en un programa.
- **Ejemplo:** Usar `print()` en Python para rastrear valores.
- **Analogía:** Es como buscar una fuga en una tubería usando agua teñida para detectar el problema.

16. Compiladores vs. intérpretes

- **Investigación:** Un compilador traduce todo el código de una vez antes de ejecutarlo, mientras que un intérprete lo ejecuta línea por línea.
- **Ejemplo:** C usa un compilador; JavaScript usa un intérprete.
- **Analogía:** Un compilador es como traducir todo un libro antes de leerlo, mientras que un intérprete es como leer y traducir al mismo tiempo.

17. Cadenas de texto

- **Investigación:** Son secuencias de caracteres utilizadas para representar palabras y frases.
- **Ejemplo:** `"Hola, mundo"`.

- **Analogía:** Una cadena de texto es como una frase escrita en una nota.

18. Hardware básico

- **Investigación:** Componentes físicos de un computador, como CPU, RAM, disco duro.
- **Ejemplo:** La CPU es el cerebro del computador, la RAM es su memoria de corto plazo.
- **Analogía:** La CPU es como un chef, la RAM es su memoria de recetas temporales.

19. Software

- **Investigación:** Programas y sistemas que permiten el funcionamiento del hardware.
- **Ejemplo:** Un sistema operativo como Windows o Linux.
- **Analogía:** El software es como las instrucciones de un electrodoméstico.

20. Sistemas operativos

- **Investigación:** Programas que gestionan hardware y software.
- **Ejemplo:** Windows, macOS, Linux.
- **Analogía:** Es como el gerente de una tienda que organiza todo.

21. Archivos y carpetas

- **Investigación:** Formas de organizar información en un sistema.
- **Ejemplo:** Un archivo de texto, una carpeta con fotos.
- **Analogía:** Son como cajones y carpetas en una oficina.

22. Terminal o consola

- **Investigación:** Herramienta para ejecutar comandos en texto.
- **Ejemplo:** `cd` para cambiar de directorio, `ls` para listar archivos.
- **Analogía:** Es como hablar directamente con el sistema en su propio idioma.

23. Ciclo de vida del software

- **Investigación:** Etapas de desarrollo de un software (planificación, diseño, implementación, pruebas, mantenimiento).
- **Ejemplo:** Crear una app desde la idea hasta su lanzamiento.
- **Analogía:** Es como construir una casa paso a paso.

24. Requisitos

- **Investigación:** Definir qué necesita el usuario antes de programar.
- **Ejemplo:** Un cliente quiere una app para agendar citas.
- **Analogía:** Es como hacer una lista de compras antes de cocinar.

25. Prototipos

- **Investigación:** Bocetos o modelos preliminares de software.
- **Ejemplo:** Un diseño de interfaz en papel antes de programarlo.
- **Analogía:** Es como un borrador de un dibujo antes del final.

26. Interfaz de usuario

- **Investigación:** La parte del software con la que interactúa el usuario.
- **Ejemplo:** La pantalla de inicio de una app.
- **Analogía:** Es como el tablero de un auto con botones y controles.

27. Pruebas

- **Investigación:** Verificación de que el software funcione correctamente.
- **Ejemplo:** Probar si un botón realmente guarda la información.
- **Analogía:** Es como revisar una bicicleta antes de usarla.

28. Qué es una base de datos

- **Investigación:** Un sistema para almacenar información de forma organizada.
- **Ejemplo:** Una base de datos de clientes con nombres y correos.
- **Analogía:** Es como un archivador con fichas organizadas.

29. Internet

- **Investigación:** Red global de computadoras interconectadas.
- **Ejemplo:** Ver una página web desde un navegador.
- **Analogía:** Es como un sistema de carreteras que conecta ciudades.

30. Direcciones IP

- **Investigación:** Identificadores numéricos de dispositivos en la red.
- **Ejemplo:** `192.168.1.1`.
- **Analogía:** Es como una dirección postal para ubicar casas.

31. Navegadores

- **Investigación:** Programas que permiten acceder a sitios web.
- **Ejemplo:** Chrome, Firefox.
- **Analogía:** Son como taxis que te llevan a diferentes direcciones web.

32. Cliente y servidor

- **Investigación:** Modelo de comunicación en el que un cliente solicita y un servidor responde.
- **Ejemplo:** Cuando accedes a una página web.
- **Analogía:** Es como un mesero que lleva un pedido de un cliente a la cocina.

33. Seguridad inicial

- **Investigación:** Medidas básicas para proteger datos y sistemas.
- **Ejemplo:** Usar contraseñas seguras.
- **Analogía:** Es como ponerle llave a tu casa.

34. HTML

- **Investigación:** Lenguaje para estructurar páginas web.
- **Ejemplo:** `<h1>Hola</h1>`
- **Analogía:** Es como los ladrillos de una casa web.

35. CSS

- **Investigación:** Lenguaje para diseñar páginas web.
- **Ejemplo:** `color: red;`
- **Analogía:** Es como la pintura y decoración de una casa.

36. JavaScript introductorio

- **Investigación:** Lenguaje de programación para interactividad en la web.
- **Ejemplo:** `alert("Hola");`
- **Analogía:** Es como darle vida a una página estática.

37. Páginas estáticas

- **Investigación:** Páginas web sin cambios dinámicos.
- **Ejemplo:** Una página HTML sin interacción.
- **Analogía:** Es como un cartel impreso.

38. Hosting básico

- **Investigación:** Subir una página web a internet.
- **Ejemplo:** Usar GitHub Pages o Netlify.
- **Analogía:** Es como alquilar un local en internet.

39. Editores de código

- **Investigación:** Programas para escribir código.
- **Ejemplo:** VS Code, Sublime Text.
- **Analogía:** Son como cuadernos digitales para programadores.

40. Control de versiones (Git)

- **Investigación:** Sistema para rastrear cambios en el código.
- **Ejemplo:** `git commit -m "mensaje"`
- **Analogía:** Es como guardar versiones de un documento.

41. Repositorios (GitHub)

- **Investigación:** Lugares donde se almacenan proyectos con Git.
- **Ejemplo:** Un repositorio de código en GitHub.
- **Analogía:** Es como una biblioteca de proyectos.

42. Línea de comandos

- **Investigación:** Herramienta para interactuar con la computadora escribiendo comandos.
- **Ejemplo:** `mkdir nueva_carpeta`
- **Analogía:** Es como un control remoto para el sistema.

43. Entornos de desarrollo

- **Investigación:** Espacios con herramientas para programar.
- **Ejemplo:** VS Code con extensiones de Python.
- **Analogía:** Es como un taller de herramientas para programadores.

44. Inteligencia artificial

- **Investigación:** Programas que imitan la inteligencia humana.
- **Ejemplo:** ChatGPT.
- **Analogía:** Es como un asistente que aprende con el tiempo.

45. Impacto del software

- **Investigación:** Cómo la tecnología cambia el mundo.
- **Ejemplo:** Aplicaciones médicas que salvan vidas.
- **Analogía:** Es como la electricidad, invisible pero vital.

46. Aprendizaje continuo

- **Investigación:** La programación evoluciona constantemente.
- **Ejemplo:** Aprender nuevas tecnologías como React o Python.
- **Analogía:** Es como entrenar en un deporte: nunca dejas de mejorar.

47. API (Interfaz de Programación de Aplicaciones)

- **Investigación:** Es un conjunto de reglas que permite a diferentes programas comunicarse entre sí.
- **Ejemplo:** La API de Google Maps que se usa para mostrar mapas en una app.
- **Analogía:** Es como un mesero que toma tu pedido y lo lleva a la cocina sin que necesites entrar a ella.

48. JSON y XML

- **Investigación:** Son formatos para estructurar y compartir datos entre sistemas.
- **Ejemplo:** `{"nombre": "Juan", "edad": 25}` en JSON.
- **Analogía:** Son como etiquetas en una caja para indicar su contenido.

49. Frameworks y librerías

- **Investigación:** Son herramientas que facilitan el desarrollo de software.
- **Ejemplo:** React para crear interfaces web, TensorFlow para IA.
- **Analogía:** Un framework es como un set de LEGO con piezas prediseñadas para construir rápido.

50. Desarrollo backend y frontend

- **Investigación:** Backend maneja la lógica y datos; frontend, la interfaz del usuario.
- **Ejemplo:** Backend con Node.js, frontend con React.
- **Analogía:** Backend es la cocina de un restaurante; frontend es la mesa del cliente.

51. Bases de datos relacionales vs. NoSQL

- **Investigación:** Relacionales almacenan datos en tablas, NoSQL en estructuras flexibles.
- **Ejemplo:** MySQL (relacional) vs. MongoDB (NoSQL).
- **Analogía:** Relacional es como una hoja de cálculo ordenada; NoSQL es como un diario con notas libres.

52. Cloud computing

- **Investigación:** Uso de servidores en la nube en lugar de hardware local.
- **Ejemplo:** Google Drive, AWS.
- **Analogía:** Es como alquilar un almacén en lugar de tenerlo en casa.

53. Virtualización y contenedores

- **Investigación:** Tecnologías que permiten ejecutar sistemas en entornos aislados.
- **Ejemplo:** Docker para crear contenedores.
- **Analogía:** Un contenedor es como una mochila con todas tus herramientas listas para usar en cualquier computadora.

54. Seguridad en desarrollo

- **Investigación:** Prácticas para evitar vulnerabilidades en software.
- **Ejemplo:** Encriptación de contraseñas.
- **Analogía:** Es como poner cerrojos a las puertas de una casa.

55. Blockchain

- **Investigación:** Tecnología de registro distribuido que almacena datos de forma segura.
- **Ejemplo:** Bitcoin y otras criptomonedas.
- **Analogía:** Es como un libro de cuentas inmutable que todos pueden verificar.

56. Programación orientada a objetos (POO)

- **Investigación:** Paradigma basado en la creación de objetos con atributos y métodos.
- **Ejemplo:** Un objeto "Coche" con atributos (color, marca) y métodos (acelerar, frenar).
- **Analogía:** Es como diseñar moldes para hacer figuras de arcilla.

57. Metodologías ágiles

- **Investigación:** Enfoques flexibles para desarrollo de software, como Scrum o Kanban.
- **Ejemplo:** Dividir un proyecto en sprints de dos semanas.
- **Analogía:** Es como planificar una obra de teatro en pequeños ensayos en lugar de un único gran ensayo final.

58. Inteligencia artificial aplicada

- **Investigación:** Uso de IA en diversas industrias.
- **Ejemplo:** Chatbots de atención al cliente.
- **Analogía:** Es como tener un asistente virtual que aprende con el tiempo.

59. Ética en tecnología

- **Investigación:** Reflexión sobre el impacto social del software.
- **Ejemplo:** Privacidad en redes sociales.
- **Analogía:** Es como decidir las reglas de un juego justo para todos.

60. Tendencias futuras en informática

- **Investigación:** Innovaciones tecnológicas en desarrollo.
- **Ejemplo:** Computación cuántica, metaverso.
- **Analogía:** Es como predecir qué dispositivos usarán las próximas generaciones.