**Integrantes: Karol Yuseth Salas Correa, Juan Pablo Galviz Marulanda.**

1. **¿Qué es un computador?**

Un computador (o computadora) es una máquina electrónica diseñada para procesar, almacenar y gestionar información de manera automática y eficiente. Es una herramienta fundamental en casi todos los aspectos de la vida moderna, desde el trabajo y la educación hasta el entretenimiento y la comunicación.

Imagina que estás escribiendo un documento en un procesador de texto:

1. Entrada: Escribes en el teclado.
2. Procesamiento: La CPU procesa cada letra que escribes.
3. Almacenamiento: El documento se guarda en el disco duro.
4. Salida: Ves el texto en la pantalla y puedes imprimirlo.
5. **Qué es un programa** - Concepto básico de instrucciones para la computadora.

Un programa es una serie de instrucciones que le dicen a la computadora cómo realizar una tarea. Estas instrucciones están escritas en un lenguaje de programación y pueden incluir lógica, manejo de datos e interacción con el usuario. Los programas son la base de todo lo que hace una computadora, desde abrir un archivo hasta ejecutar un videojuego.

Ejemplo práctico: Programa simple

Imagina un programa que suma dos números:

1. Instrucción 1: Pedir al usuario que ingrese el primer número.
2. Instrucción 2: Pedir al usuario que ingrese el segundo número.
3. Instrucción 3: Sumar los dos números.
4. Instrucción 4: Mostrar el resultado en la pantalla.
5. **Lógica de programación** - Pensamiento estructurado para resolver problemas.

La lógica de programación es el proceso de pensar de manera estructurada y organizada para resolver problemas mediante la creación de programas. Es la base de la programación y consiste en descomponer un problema en pasos pequeños y lógicos que una computadora pueda entender y ejecutar.

1. **Algoritmos** - Pasos para resolver un problema (ej. pseudocódigo).

Un algoritmo es una secuencia de pasos lógicos y ordenados que se siguen para resolver un problema o realizar una tarea. Es como una receta: si sigues los pasos correctamente, obtendrás el resultado deseado. Los algoritmos son la base de la programación y se pueden expresar de muchas formas, incluyendo pseudocódigo, diagramas de flujo o directamente en un lenguaje de programación.

Características de un algoritmo

Precisión: Cada paso debe estar claramente definido y no debe haber ambigüedad.

Orden: Los pasos deben seguir una secuencia lógica.

Finitud: Debe terminar después de un número finito de pasos.

Entrada: Recibe datos iniciales para trabajar.

Salida: Produce un resultado o solución al problema.

1. **Variables - Concepto y tipos (enteros, decimales, texto).**

Variables

* Concepto: Contenedores que almacenan valores en la memoria.
* Partes:
* Nombre: Identificador (ej. edad, nombre).
* Valor: Dato almacenado (ej. 25, "Ana").
* Tipo de dato: Define qué información puede almacenar.

Tipos de datos principales

1. Enteros (int): Números sin decimales (ej. 10, -5).
2. Decimales (float): Números con decimales (ej. 3.14, -0.5).
3. Texto (string): Cadenas de caracteres (ej. "Hola", 'Juan').
4. Booleanos (bool): Valores True o False.
5. Listas (list): Colecciones ordenadas y modificables (ej. [1, 2, 3]).
6. Tuplas (tuple): Colecciones ordenadas e inmutables (ej. (1, 2, 3)).
7. Diccionarios (dict): Pares clave-valor (ej. {"nombre": "Juan"}).
8. **Tipos de datos - Números, cadenas, booleanos.**

Tipos de datos básicos

1. Números:
   * int: Enteros (ej. 25).
   * float: Decimales (ej. 19.99).
2. Cadenas (string):
   * Texto entre comillas (ej. "Hola").
3. Booleanos (bool):
   * True o False.
4. **Operadores - Aritmé7cos (+, -, \*, /), lógicos (AND, OR, NOT).**

|  |  |  |  |
| --- | --- | --- | --- |
| **OPERADOR** | **DESCRIPCIÓN** | **EJEMPLO** | **RESULTADO** |
| **+** | Suma | **5 + 1** | **6** |
| **-** | Resta | 3 - 1 | 2 |
| **\*** | Multiplicación | 2 \* 2 | 4 |
| **/** | División | 4 / 2 | 2 |
| **%** | Modulo | 10 % 3 | 1 |
| **\*\*** | Potencia | 2\*\*3 | 8 |
| **//** | División Entera | 10 // 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| OPERADOR | DESCRIPCIÓN | EJEMPLO | RESULTADO |
| AND | Verdadero si todas las condiciones son verdaderas | True AND False | FALSE |
| OR | Verdadero si al menos una condición es verdadera | True OR False | TRUE |
| NOT | Invierte el valor de la condición | NOT True | FALSE |

1. **Estructuras de control - Condicionales (if, else).**

if: Ejecuta un bloque si la condición es verdadera.

else: Ejecuta un bloque si la condición del if es falsa.

elif: Permite verificar múltiples condiciones.

1. **Bucles - Repetición (for, while).**

Los bucles, también conocidos como estructuras de repetición, son una herramienta fundamental en programación que permite ejecutar un bloque de código varias veces. Los dos tipos más comunes de bucles son for y while. A continuación, te explico cada uno:

1. Bucle for

El bucle for se utiliza cuando sabes de antemano cuántas veces quieres que se repita un bloque de código. Es especialmente útil para iterar sobre secuencias, como listas, rangos, o cualquier objeto iterable.

2. Bucle while

El bucle while se utiliza cuando quieres repetir un bloque de código mientras una condición sea verdadera. A diferencia del for, no necesitas saber de antemano cuántas veces se repetirá el bucle, ya que depende de la condición.

1. **Entrada y salida - Leer datos del usuario y mostrar resultados.**

1. Entrada (Input):

La entrada se refiere a los datos que un programa recibe, ya sea desde el usuario, un archivo, una base de datos o cualquier otra fuente externa. En el contexto de la interacción con el usuario, la entrada suele ser información que el usuario introduce a través del teclado.

2. Salida (Output):

La salida se refiere a los datos que el programa muestra o envía al usuario o a otro sistema. Esto puede ser texto en la pantalla, un archivo generado, o cualquier otra forma de comunicación.

1. **Funciones básicas - Reutilización de código.**

Las funciones son uno de los conceptos más importantes en programación. Permiten encapsular un bloque de código que realiza una tarea específica, de modo que puedas reutilizarlo en diferentes partes de tu programa sin tener que repetir el mismo código una y otra vez. Esto hace que tu código sea más organizado, legible y fácil de mantener.

¿Qué es una función?

Una función es un bloque de código que:

Recibe entradas (parámetros).

Realiza una tarea específica.

Devuelve un resultado (opcional).

Las funciones se definen una vez y se pueden llamar (usar) tantas veces como sea necesario.

1. **Primer lenguaje (ej. Python) – Sintaxis básica y uso**

Python es un lenguaje de programación interpretado, de alto nivel y orientado a objetos. Su sintaxis es sencilla y clara, lo que lo hace ideal para principiantes. Se utiliza en múltiples campos como desarrollo web, ciencia de datos, inteligencia artificial, automatización y ciberseguridad.

Algunas características clave incluyen:

Uso de indentación para estructurar el código.

Variables dinámicamente tipadas (no es necesario especificar el tipo).Amplia comunidad y gran cantidad de bibliotecas disponibles.

1. **Comentarios en el código - Documentar para claridad**

Los comentarios son fragmentos de texto dentro del código que no se ejecutan y sirven para documentar lo que hace el programa. Su propósito es mejorar la legibilidad y facilitar el mantenimiento del código.

Existen dos tipos principales:

Comentarios de una sola línea, usados para explicar partes pequeñas del código.

Comentarios de varias líneas o documentación (docstrings), empleados para describir funciones y módulos completos.

La buena documentación es esencial en equipos de trabajo y proyectos de gran escala.

1. **Errores comunes - Cómo identificarlos**

Existen tres tipos principales de errores en programación:

Errores de sintaxis: Ocurren cuando el código no sigue las reglas del lenguaje. Se detectan antes de la ejecución.

Errores en tiempo de ejecución: Suceden mientras el programa se está ejecutando, como intentos de dividir entre cero o acceder a una variable no definida.

Errores lógicos: Son los más difíciles de detectar, ya que el programa se ejecuta sin fallos, pero produce resultados incorrectos debido a errores en la lógica del código.

Para identificarlos, se pueden usar herramientas de depuración, pruebas y revisión manual del código.

1. **Depuración - Uso básico de herramientas o print statements.**

La depuración es el proceso de identificar y corregir errores en un programa. Algunas técnicas incluyen:

Uso de print statements: Permite imprimir valores intermedios para entender el flujo del código.

Depuradores en entornos de desarrollo (IDEs): Herramientas como el depurador de VS Code, PyCharm o GDB permiten analizar el código paso a paso.

Pruebas unitarias: Se escriben pruebas automatizadas para verificar el correcto funcionamiento del código.

Registros de errores (logs): Guardan información sobre la ejecución para ayudar a detectar fallos.

1. **Compiladores vs. intérpretes - Diferencia simple**

Un compilador y un intérprete son herramientas que permiten ejecutar código fuente, pero funcionan de manera diferente:

Compiladores: Traducen todo el código fuente a lenguaje de máquina antes de ejecutarlo. Si hay errores, el programa no se ejecutará hasta corregirlos. Ejemplos: C, C++.

Intérpretes: Ejecutan el código línea por línea sin necesidad de compilarlo antes. Esto permite probar partes del código sin compilar todo el programa. Ejemplos: Python, JavaScript.

1. **Cadenas de texto - Manipulación básica.**

Las cadenas de texto (strings) son secuencias de caracteres utilizadas para representar texto. Se pueden manipular de varias formas:

Concatenación: Unir varias cadenas en una sola.

Medición de longitud: Determinar cuántos caracteres tiene una cadena.

Conversión de mayúsculas y minúsculas: Cambiar el formato del texto.

Extracción de partes de una cadena (slicing): Obtener una porción específica de texto.

Reemplazo y búsqueda de texto: Modificar palabras dentro de una cadena.

Las cadenas son fundamentales en cualquier lenguaje de programación, ya que representan nombres, mensajes, direcciones y cualquier otro dato basado en texto.

1. **Hardware básico - CPU, memoria, disco duro.**

El hardware es la parte física de un sistema informático. Sus componentes principales incluyen:

CPU (Unidad Central de Procesamiento): Es el "cerebro" del ordenador, encargado de ejecutar instrucciones.

Memoria RAM: Almacena temporalmente datos y programas en ejecución para un acceso rápido.

Disco duro (HDD) o unidad de estado sólido (SSD): Almacenan datos de manera permanente.

Tarjeta gráfica (GPU): Maneja los gráficos y es clave en juegos y aplicaciones de alto rendimiento visual.

Cada uno de estos componentes trabaja en conjunto para garantizar el funcionamiento del sistema.

1. **Software - Sistema operativo vs. Aplicaciones.**

El software es el conjunto de programas y sistemas que permiten operar una computadora. Se divide en:

Sistema operativo: Software que gestiona el hardware y proporciona una interfaz para el usuario. Ejemplos: Windows, Linux, macOS.

Aplicaciones: Programas diseñados para realizar tareas específicas, como navegadores web, procesadores de texto o videojuegos.

Sin un sistema operativo, las aplicaciones no podrían interactuar con el hardware de la computadora.

1. **Sistemas operativos – Windows, Linux, funciones básicas.**

Un sistema operativo es el software que gestiona los recursos del sistema y permite la ejecución de aplicaciones.

Windows: Es ampliamente usado, tiene interfaz gráfica amigable y compatibilidad con muchos programas.

Linux: Es de código abierto, más seguro y flexible, ideal para servidores y desarrolladores.

macOS: Sistema de Apple, optimizado para dispositivos Mac.

Funciones básicas incluyen la gestión de archivos, control de dispositivos, administración de usuarios y ejecución de programas.

1. **Archivos y carpetas – Organización en el sistema.**

Los archivos almacenan datos y las carpetas ayudan a organizarlos dentro del sistema.

Archivos: Pueden ser documentos, imágenes, videos o programas ejecutables.

Carpetas: Permiten clasificar archivos de manera estructurada.

El buen manejo de archivos y carpetas mejora la eficiencia y facilita la búsqueda de información.

1. **Terminal o consola - Comandos básicos.**

La terminal o consola es una interfaz basada en texto que permite ejecutar comandos en el sistema. Algunos comandos básicos incluyen:

cd (Cambiar directorio)

ls / dir (Listar archivos y carpetas)

mkdir (Crear una carpeta)

rm (Eliminar archivos)

El uso de la terminal es clave en administración de sistemas y desarrollo de software.

1. **Fundamentos de Desarrollo de Software.**

El desarrollo de software es el proceso de diseñar, programar, probar y mantener aplicaciones informáticas. Requiere conocimientos en algoritmos, estructuras de datos, lenguajes de programación y bases de datos.

1. **Ciclo de vida del software – Idea básica.**

El desarrollo de software sigue varias etapas:

Planificación: Identificar necesidades y objetivos.

Diseño: Crear esquemas y definir la estructura del programa.

Implementación: Escribir el código.

Pruebas: Verificar el funcionamiento y corregir errores.

Mantenimiento: Actualizar y mejorar el software con el tiempo.

1. **Requisitos - Qué quiere el usuario.**

Los requisitos del software definen lo que el usuario necesita y espera del sistema. Son fundamentales en la planificación y diseño, ya que aseguran que el producto final cumpla con las expectativas. Se dividen en:

1. Requisitos funcionales:
   * Especifican qué debe hacer el software.
   * Ejemplo: "El sistema debe permitir a los usuarios registrarse e iniciar sesión".
2. Requisitos no funcionales:
   * Describen cómo debe comportarse el sistema en términos de rendimiento, seguridad y escalabilidad.
   * Ejemplo: "La aplicación debe cargar en menos de 3 segundos".
3. Requisitos del negocio:
   * Relacionados con los objetivos comerciales que el software debe ayudar a alcanzar.
   * Ejemplo: "El sistema debe permitir pagos en línea para aumentar las ventas".
4. Requisitos del usuario:
   * Necesidades y expectativas de la experiencia de usuario (UX).
   * Ejemplo: "La interfaz debe ser intuitiva y accesible para personas con discapacidad visual".

La recopilación de requisitos se hace mediante entrevistas con clientes, encuestas, análisis de mercado y documentación de procesos existentes.

1. **Prototipos - Bosquejos simples de software.**

Un prototipo es una representación visual o funcional preliminar de una aplicación que permite probar ideas antes del desarrollo completo.

Tipos de prototipos

1. Prototipo en papel:
   * Esquemas dibujados a mano de las pantallas o flujos del software.
   * Útil en las primeras fases de diseño.
2. Prototipo de baja fidelidad:
   * Representación digital básica sin funcionalidad real.
   * Se usa para definir la estructura y disposición de los elementos.
3. Prototipo de alta fidelidad:
   * Más detallado e interactivo, con apariencia similar a la versión final.
   * Se crea con herramientas como Figma, Adobe XD o Sketch.
4. Prototipo funcional:
   * Versión preliminar del software con funciones básicas implementadas.
   * Se utiliza para validar la viabilidad del proyecto.
5. **Interfaz de usuario - Concepto de diseño básico.**

La interfaz de usuario (UI, por sus siglas en inglés) es la parte del software con la que interactúan los usuarios. Debe ser intuitiva, atractiva y eficiente para garantizar una buena experiencia.

Principios básicos del diseño de interfaces

1. Simplicidad: Menos es más. Evitar elementos innecesarios.
2. Consistencia: Usar los mismos colores, tipografías y estilos en toda la aplicación.
3. Accesibilidad: Incluir opciones para personas con discapacidades (ej., texto alternativo en imágenes).
4. Visibilidad: Resaltar elementos importantes como botones de acción.
5. Retroalimentación: Informar al usuario sobre acciones realizadas, como mensajes de éxito o error.

Componentes clave de una interfaz

* Botones: Permiten ejecutar acciones.
* Formularios: Para ingresar datos (nombre, correo, contraseña, etc.).
* Menús de navegación: Ayudan a moverse dentro del sistema.
* Notificaciones: Alertan sobre eventos importantes. Las herramientas más usadas para diseñar interfaces son Figma, Adobe XD y Sketch.

1. **Pruebas - Verificar que el programa funcione.**

Las pruebas de software son un proceso clave para garantizar que el programa funcione correctamente antes de ser lanzado.

Tipos de pruebas

1. Pruebas unitarias:
   * Verifican que cada parte del código funcione de manera independiente.
   * Generalmente automatizadas con herramientas como JUnit (Java) o PyTest (Python).
2. Pruebas de integración:
   * Aseguran que diferentes módulos del software funcionen juntos correctamente.
3. Pruebas funcionales:
   * Comprueban que el software cumple con los requisitos definidos.
   * Se basan en escenarios de usuario reales.
4. Pruebas de rendimiento:
   * Evalúan la velocidad, estabilidad y escalabilidad del sistema.
   * Ejemplo: Simular miles de usuarios conectándose al mismo tiempo.
5. Pruebas de seguridad:
   * Buscan vulnerabilidades que podrían comprometer los datos del usuario.
6. Pruebas de aceptación del usuario (UAT):
   * Se realizan con usuarios finales para validar si el software es útil y cumple con las expectativas.

Beneficios de las pruebas de software

* Reducen costos al encontrar errores antes del lanzamiento.
* Mejoran la seguridad y confiabilidad del sistema.
* Garantizan una mejor experiencia para los usuarios.

Las pruebas pueden ser manuales o automatizadas con herramientas como Selenium, JUnit o Cypress.

1. **Qué es una base de datos - Almacenar información organizada.**

Una base de datos es un sistema que permite almacenar, organizar y gestionar información de manera estructurada.

Tipos de bases de datos:

1. Bases de datos relacionales (SQL):
   * Almacenan datos en tablas organizadas con filas y columnas.
   * Utilizan el lenguaje SQL (Structured Query Language) para acceder a los datos.
   * Ejemplo: MySQL, PostgreSQL, SQL Server.
2. Bases de datos NoSQL:
   * No utilizan tablas; pueden almacenar datos en documentos, grafos o pares clave-valor.
   * Son más flexibles y escalables para grandes volúmenes de datos.
   * Ejemplo: MongoDB, Firebase, Cassandra.

Conceptos clave en bases de datos relacionales

* Tabla: Conjunto de datos organizados en filas y columnas.
* Registro (fila): Cada entrada en una tabla representa una entidad única.
* Campo (columna): Representa un atributo de los datos.
* Clave primaria: Identificador único para cada registro.
* Relaciones: Conexión entre tablas para evitar duplicación de datos.

Ejemplo de aplicación de bases de datos

* Comercio electrónico: Almacena información de clientes, productos y pedidos.
* Redes sociales: Guarda perfiles de usuarios, publicaciones y comentarios.
* Bancos: Maneja transacciones y cuentas de clientes de manera segura.

Herramientas para manejar bases de datos

* Gestores de bases de datos (DBMS): MySQL, PostgreSQL, Oracle, SQL Server.
* Herramientas de consulta: SQL, MongoDB Query Language.
* Plataformas en la nube: Firebase, Amazon RDS, Google Cloud Firestore.

Las bases de datos son esenciales para casi todas las aplicaciones modernas, desde pequeñas páginas web hasta sistemas bancarios y redes sociales.

1. **Internet - Cómo funciona a nivel básico**

Internet es una red global de computadoras que se comunican entre sí utilizando protocolos estándar como TCP/IP. Permite la transferencia de datos y la conexión entre dispositivos en todo el mundo.

**Ejemplo de uso:**

Al visitar una página web, el navegador solicita información al servidor de la página y recibe los datos para mostrarlos en la pantalla. Analogía: Internet es como un sistema postal gigante donde las computadoras envían y reciben

cartas (datos) a través de diferentes oficinas de correos (servidores).

1. **Direcciones IP - Identificadores simples**

Una dirección IP (Internet Protocol) es un número único que identifica a un dispositivo dentro de una red, permitiendo la comunicación entre ellos.

**Ejemplo de uso:**

Cuando ingresas una dirección web, tu computadora usa la dirección IP del servidor para localizarlo y conectarse a él. Analogía: Una dirección IP es como la dirección de una casa en una ciudad, permitiendo que las cartas (datos) lleguen al lugar correcto.

1. **Navegadores - Qué hacen y cómo**

Los navegadores son programas que permiten acceder y visualizar páginas web en Internet, interpretando lenguajes como HTML, CSS y JavaScript.

**Ejemplo de uso:**

Chrome, Firefox y Edge son navegadores populares utilizados para visitar sitios web. Analogía: Un navegador es como una ventana a Internet, permitiendo ver y explorar el contenido disponible en la web.

1. **Cliente y servidor - Interacción básica**

El modelo cliente-servidor describe cómo los dispositivos (clientes) solicitan y reciben servicios de servidores en una red.

**Ejemplo de uso:**

Al ver un video en YouTube, tu navegador (cliente) solicita el video a los servidores de YouTube, que lo envían para que puedas verlo. Analogía: Es como un restaurante: el cliente pide comida (datos) y el mesero (servidor) la entrega desde la cocina (base de datos).

1. **Seguridad inicial - Contraseñas y riesgos**

La seguridad en Internet implica proteger la información personal mediante contraseñas seguras y evitando riesgos como virus o fraudes.

**Ejemplo de uso:**

Usar contraseñas largas y únicas para cada cuenta en línea y activar la verificación en dos pasos. Analogía: La seguridad en Internet es como cerrar con llave la puerta de tu casa para evitar intrusos.

1. **HTML - Estructura de una página web**

HTML (HyperText Markup Language) es el lenguaje de marcado utilizado para crear la estructura básica de una página web.

**Ejemplo de uso:**

Crear una página con títulos, párrafos e imágenes utilizando etiquetas HTML. Analogía: HTML es como los cimientos y estructura de una casa; define dónde estarán las paredes, puertas y ventanas de una página web.

1. **CSS - Estilo básico (colores, fuentes)**

CSS (Cascading Style Sheets) es un lenguaje que permite diseñar y dar estilo a las páginas web, controlando colores, fuentes y distribución.

**Ejemplo de uso:**

Cambiar el color de fondo y el tamaño del texto de una página web. Analogía: CSS es como la decoración de una casa; permite pintar las paredes, elegir muebles y darle un aspecto atractivo a la página.

1. **JavaScript introductorio - Interactividad simple (alertas)**

JavaScript es un lenguaje de programación que permite agregar interactividad a las páginas web.

**Ejemplo de uso:**

Crear un botón en una página web que muestre un mensaje emergente cuando se presiona. Analogía: JavaScript es como los interruptores de luz en

una casa, activando acciones cuando se presionan.

1. **Páginas estáticas - Crear algo visible**

Una página estática es una página web que muestra la misma información a todos los usuarios y no cambia dinámicamente.

**Ejemplo de uso:**

Un sitio web de presentación personal con información fija sobre una persona o empresa.

1. **Hosting básico - Subir una página a la web**

El hosting es el servicio que permite almacenar y publicar páginas web en Internet.

**Ejemplo de uso:**

Usar plataformas como GitHub Pages o Netlify para subir un sitio web creado en HTML y CSS.

1. **Editores de código - VS Code, uso básico**

Los editores de código son programas diseñados para escribir y editar código de manera eficiente.

**Ejemplo de uso:**

Usar Visual Studio Code para escribir y ejecutar un programa en Python o JavaScript.

1. **Control de versiones - Qué es Git (concepto inicial)**

Git es un sistema que permite llevar un historial de cambios en el código, facilitando la colaboración y el desarrollo.

**Ejemplo de uso:**

Guardar diferentes versiones de un proyecto y restaurar una versión anterior si es necesario.

1. **Repositorios - Idea de GitHub**

Un repositorio en GitHub es un espacio en línea donde se almacena código y permite colaborar en proyectos de software.

**Ejemplo de uso:**

Subir un proyecto a GitHub para compartirlo con otros desarrolladores y recibir mejoras.

1. **Línea de comandos - Comandos útiles para programar**

La línea de comandos es una interfaz que permite ejecutar instrucciones directamente en el sistema operativo.

**Ejemplo de uso:**

Usar el comando 'mkdir' para crear una nueva carpeta desde la terminal.

1. **Entornos de desarrollo - Instalación de Python o similar**

Un entorno de desarrollo es el conjunto de herramientas necesarias para programar en un lenguaje específico.

**Ejemplo de uso:**

Instalar Python y configurar un editor de código para escribir programas en ese lenguaje.

1. **Metodología ágil - Idea de iteraciones cortas**

La metodología ágil es una forma de gestionar proyectos que se basa en ciclos cortos de desarrollo y mejora continua.

**Ejemplo de uso:**

Un equipo de desarrollo que trabaja en sprints de dos semanas para lanzar nuevas funciones de una aplicación.

1. **Documentación - Escribir cómo funciona el código**

La documentación es la escritura de instrucciones y explicaciones sobre cómo funciona un código o sistema.

**Ejemplo de uso:**

Crear un archivo README.md en GitHub para explicar cómo instalar y usar un software.

1. **Resolución de problemas - Dividir en partes**

La resolución de problemas en programación implica descomponer un problema complejo en partes más manejables.

**Ejemplo de uso:**

Dividir el desarrollo de una aplicación en módulos como interfaz, base de datos y lógica de negocio.

1. **Comunicación - Explicar ideas técnicas**

La comunicación en tecnología es la habilidad de explicar conceptos técnicos de manera clara y comprensible.

**Ejemplo de uso:**

Un desarrollador que presenta una solución técnica a su equipo usando diagramas y ejemplos.

1. **Pensamiento crítico - Evaluar soluciones**

El pensamiento crítico es la habilidad de analizar y evaluar información de manera objetiva para formar juicios fundamentados. Implica cuestionar la veracidad de las afirmaciones, identificar sesgos y considerar múltiples perspectivas antes de llegar a una conclusión.

**Ejemplo de uso:**

Al enfrentarse a una noticia en redes sociales, una persona con pensamiento crítico investigará la fuente, contrastará la información con otros medios y evaluará la evidencia antes de aceptarla como verdadera. Analogía: El pensamiento crítico es como un filtro de agua que elimina las impurezas; de manera similar, esta habilidad filtra la información errónea o engañosa, permitiendo que solo lo veraz y relevante llegue a nuestra comprensión.

1. **Ética en TI - Uso responsable de la tecnología**

La ética en Tecnologías de la Información (TI) se refiere a los principios y normas que guían el uso adecuado y responsable de la tecnología. Incluye aspectos como la privacidad, la seguridad, la propiedad intelectual y el impacto social de las tecnologías.

**Ejemplo de uso:**

Una empresa que desarrolla software debe garantizar que sus productos no infrinjan la privacidad de los usuarios y que no sean utilizados para actividades ilícitas o dañinas. Analogía: La ética en TI es como las reglas de tránsito que aseguran que todos los conductores utilicen las vías de manera segura y ordenada; de igual forma, la ética guía a los profesionales de TI para que la tecnología beneficie a la sociedad sin causar perjuicios.

1. **Privacidad - Proteger datos básicos**

La privacidad es el derecho de las personas a controlar la información que comparten sobre sí mismas y a proteger sus datos personales de accesos no autorizados.

**Ejemplo de uso:**

Configurar las opciones de privacidad en una red social para que solo amigos cercanos puedan ver las publicaciones personales. Analogía: La privacidad es como las cortinas de una casa que permiten a los habitantes decidir cuándo y cuánto quieren que los demás vean del interior; de manera similar, las personas controlan qué información personal comparten y con quién.

1. **Persistencia - Lidiar con errores y fracasos**

La persistencia es la capacidad de mantener el esfuerzo y la determinación frente a obstáculos, errores o fracasos, con el objetivo de alcanzar una meta.

**Ejemplo de uso:**

Un programador que enfrenta errores constantes en su código continúa depurándolo y buscando soluciones hasta lograr que funcione correctamente. Analogía: La persistencia es como el agua que, gota a gota, puede erosionar una roca; aunque el progreso pueda ser lento, la constancia lleva al éxito.

1. **Proyecto simple - Calculadora o lista de tareas**

Un proyecto simple en programación es una tarea de desarrollo que tiene un alcance limitado y es adecuada para principiantes. Ejemplos comunes incluyen una calculadora básica o una aplicación de lista de tareas.

**Ejemplo de uso:**

Desarrollar una aplicación que permita al usuario agregar, eliminar y marcar como completadas tareas diarias. Analogía: Un proyecto simple es como armar un rompecabezas de pocas piezas; es una actividad manejable que ayuda a desarrollar habilidades básicas antes de abordar desafíos más

complejos.

1. **Reutilización de código - Usar funciones ya hechas**

La reutilización de código consiste en emplear fragmentos de código existentes en nuevos proyectos o funciones, con el fin de ahorrar tiempo y reducir errores.

**Ejemplo de uso:**

Utilizar una función previamente desarrollada para validar direcciones de correo electrónico en diferentes partes de una aplicación. Analogía: Reutilizar código es como usar piezas de Lego ya ensambladas para construir una nueva estructura; en lugar de crear cada componente desde cero, se aprovechan partes existentes para agilizar el proceso.

1. **Inteligencia artificial - Qué es en términos simples**

La inteligencia artificial (IA) es la rama de la informática que busca crear sistemas capaces de realizar tareas que requieren inteligencia humana, como el aprendizaje, el razonamiento y la adaptación.

**Ejemplo de uso:**

Asistentes virtuales como Siri o Alexa utilizan IA para comprender y responder a las consultas de los usuarios. Analogía: La IA es como un aprendiz que, con el tiempo y la experiencia, mejora su capacidad para realizar tareas específicas sin intervención humana constante.

1. **Tipos de archivos: doc, png**

Diferentes tipos de archivos digitales tienen distintos propósitos y formatos. - \*\*.doc:\*\* Es la extensión de archivos de documentos de Microsoft Word, utilizados para procesar texto y que pueden contener imágenes, tablas y formato avanzado. - \*\*.png:\*\* Es un formato de imagen de alta calidad sin pérdida de datos.

**Ejemplo de uso:**

- \*\*.doc:\*\* Un informe escrito en Microsoft Word guardado como 'informe.doc'. - \*\*.png:\*\* Una imagen de alta resolución guardada como 'imagen.png'. Analogía: \*\*.doc\*\* es como una hoja de papel digital en la que se puede escribir, dibujar y dar formato al texto, mientras que \*\*.png\*\* es como una fotografía impresa que mantiene todos los detalles y colores originales sin pérdida de calidad.

1. **Aplicaciones móviles - Ejemplo de su uso**

Las aplicaciones móviles son programas diseñados para ejecutarse en dispositivos móviles, como smartphones o tabletas, y que permiten a los usuarios realizar diversas tareas.

**Ejemplo de uso:**

Una aplicación de banca móvil que permite a los usuarios consultar saldos, realizar transferencias y pagar servicios desde su teléfono.

1. **Videojuegos - Introducción al desarrollo básico**

El desarrollo de videojuegos consiste en la creación de software interactivo con fines de entretenimiento, educación o simulación.

**Ejemplo de uso:**

Crear un juego sencillo en Unity donde el jugador deba esquivar obstáculos y alcanzar una meta.

1. **Impacto del software - Cómo cambia el mundo**

El software ha transformado múltiples aspectos de la vida moderna, desde la comunicación y la educación hasta la medicina y la economía.

**Ejemplo de uso:**

Aplicaciones de telemedicina que permiten a los médicos atender pacientes a distancia.

1. **Aprendizaje continuo - Importancia de seguir estudiando**

El aprendizaje continuo es la práctica de seguir adquiriendo conocimientos y habilidades a lo largo de la vida para adaptarse a un mundo en constante cambio.

**Ejemplo de uso:**

Un programador que estudia nuevas tecnologías como la inteligencia artificial para mejorar sus habilidades laborales.