Práctica en clases

Carrera: Software	**Docente:** John Cevallos	**Nivel:** Cuarto
Periodo lectivo: 2025-2026(1)	**Asignatura:** Aplicaciones para el Cliente Web	**Paralelo(s):** "B"
Número de Práctica/taller: 3	**Fecha:** 30 de mayo de 2025	

Nombre de la Unidad: Programación del cliente con JavaScript.	
Tema: Aplicación de tipos, interfaces, clases y funciones con base en las entidades del proyecto autónomo	**Número horas: 2**

OBJETIVO DE LA PRÁCTICA COMPLEMENTARIA

Aplicar los fundamentos principales de TypeScript mediante la creación de modelos, estructuras de datos, y lógica de negocio simulada, tomando como base las entidades definidas en el módulo del proyecto autónomo de cada estudiante.

MATERIALES

- Visual Studio Code
- Node.js + TypeScript
- GitHub
- Repositorio del proyecto autónomo

Contenidos relacionados

- Tipos de datos en TypeScript
- Interfaces
- Clases y objetos
- Arreglos
- Funciones básicas
- Tipado de parámetros y retorno

Instrucciones Generales

Desarrolla un módulo funcional en TypeScript que incluya lo siguiente:

1. **Definición de Tipos Básicos:**

   Declara al menos 5 variables de tipos primitivos (string, number, boolean) relacionadas a tu proyecto (por ejemplo, nombre del producto, stock disponible, etc.).

2. **Creación de Interfaces:**

   Define al menos 3 interfaces correspondientes a tus entidades del proyecto (ej. Cliente, Producto, Reserva, etc.).

3. **Definición de Clases:**

   Crea al menos 3 clases basadas en las interfaces anteriores, implementando constructor y al menos un método en cada clase que aplique alguna lógica de validación que usted considere conveniente para su dominio.

4. **Creación de Arreglos Tipados:**

   Crea un arreglo por cada entidad y llénelos con al menos 10 elementos simulados.

5. **Funciones Tipadas:**

   Implementa 3 funciones que manipule los arreglos del punto anterior:

   - Para mostrar todos los elementos de los arreglos por consola.
   - Que permita filtrar o contar elementos con base en alguna condición propia a su dominio (ej. productos disponibles, usuarios activos).
   - Que inserte y elimine elementos de uno de los arreglos.

6. **Uso de tipos especiales:**

   Utiliza readonly, optional (?) y union types (string | number) dentro de tus interfaces o funciones anteriores y ubícala en este punto.

7. Uso de map() para transformar datos

- Recorre un arreglo de una entidad (por ejemplo, productos) y genera un nuevo arreglo con otro formato (por ejemplo, sólo los nombres en mayúsculas, o un resumen con nombre y precio).
- Ejemplo: transformar clientes en un arreglo de correos electrónicos.

8. Uso de filter() para seleccionar datos

- Crea filtros para obtener subconjuntos de los datos (ej. productos disponibles, usuarios activos, reservas pendientes).
- Mostrar en consola el arreglo filtrado.

9. Uso de reduce() para cálculos acumulativos

- Calcular totales: suma de precios, cantidad de reservas, promedio de calificaciones, etc. en alguno de tus arreglos.
- Mostrar el resultado de la operación en consola.

10. Simular relaciones entre entidades como objetos conectados

Para recordar un ejemplo: una clase Pedido debe tener un atributo cliente (que será una instancia de la clase Cliente) y un arreglo de Producto.

- Crear instancias simuladas respetando estas relaciones y crear un objeto padre que las contenga como si fuese una base de datos.

11. Simular una operación de negocio simple entre entidades

Algunos ejemplos que le podría ayudar a resolver este punto:

- Por ejemplo: crear una función que genere un resumen de un pedido, listando el nombre del cliente y los productos incluidos.
- Otra idea: contar cuántos productos ha comprado un cliente.

12. Imprimir estructuras anidadas

- Mostrar en consola la información de un objeto complejo con relaciones anidadas.

  Por ejemplo: mostrar todos los pedidos con sus productos y cliente responsable.

Entregable

- Proyecto en TypeScript estructurado en un solo archivo o en carpetas
- Repositorio subido a GitHub con nombre: ts-fundamentos-[apellido]
- Captura de ejecución en consola del archivo principal (tsc + node)

CONSIDERACIONES

- No es necesario implementar HTML ni manipulación del DOM en esta práctica.
- Se valorará claridad, comentarios en el código y aplicación correcta de la tipificación.
- Esta práctica servirá como base para futuras manipulaciones del DOM.

ENLACES DE PÁGINAS OFICIALES

- TypeScript - Tipado
- MDN - Funciones
- GitHub Docs