

CLASE NRO. 01: CONCEPTOS PREVIOS + INTRODUCCIÓN A LA PROGRAMACIÓN

Introducción a la Programación

La programación es el proceso de diseñar y construir un conjunto de instrucciones que le dicen a una computadora cómo realizar una tarea específica. Esto implica escribir código en un lenguaje que la computadora pueda entender.

Qué es la Programación

La programación es la actividad de escribir, depurar y mantener el código fuente de programas de computadora. El objetivo es crear un software que realice tareas útiles.

Lenguajes de Programación

Los lenguajes de programación son herramientas que los programadores utilizan para comunicarse con la computadora. Existen diferentes lenguajes para distintos tipos de tareas.

Tipos de Lenguajes de Programación

1. **Lenguajes de Bajo Nivel:** Cercanos al lenguaje de máquina, como ensamblador.
2. **Lenguajes de Alto Nivel:** Más cercanos al lenguaje humano, como Python, Java, C++.
3. **Lenguajes de Script:** Especialmente diseñados para automatizar tareas, como JavaScript y PHP.
4. **Lenguajes de Programación Funcional:** Se centran en la aplicación de funciones, como Haskell y Scala.

Componentes Básicos de un Programa

- **Variables:** Espacios de almacenamiento que contienen datos que pueden cambiar durante la ejecución.
- **Tipos de Datos:** Clasificación de datos (números, texto, booleanos, etc.).

Estructuras de Control

- **Estructuras Condicionales:** Permiten tomar decisiones en el código. Ejemplo: `if`, `else`.
- **Estructuras Repetitivas:** Permiten ejecutar un bloque de código varias veces. Ejemplo: `for`, `while`.

Depuración y Manejo de Errores

La depuración es el proceso de identificar y corregir errores en el código. Manejar errores implica anticipar problemas y diseñar el programa para que los maneje de manera efectiva.

Introducción al Protocolo HTTP y HTTPS

- **HTTP** (Protocolo de Transferencia de Hipertexto): Protocolo para la transferencia de datos en la web.
- **HTTPS**: Versión segura de HTTP, que utiliza cifrado para proteger los datos.

Esquema de Petición / Respuesta

El modelo de comunicación en la web. Un cliente (navegador) realiza una solicitud a un servidor, que responde con los datos solicitados.

Diferencias entre Conceptos

- **Página Web:** Un documento que se muestra en un navegador.
- **Sitio Web:** Un conjunto de páginas web relacionadas.
- **Aplicación Web:** Un programa que se ejecuta en un servidor y se accede a través de un navegador.

Arquitectura de una Aplicación Web

Estructura que define cómo se organizan y comunican los componentes de una aplicación web, que generalmente incluye un cliente (frontend), un servidor (backend) y una base de datos.

Patrón de Diseño Modelo Vista Controlador (MVC)

Un patrón arquitectónico que separa la aplicación en tres componentes:

- **Modelo:** Maneja los datos y la lógica de negocio.
- **Vista:** Presenta los datos al usuario.
- **Controlador:** Interactúa entre el modelo y la vista, procesando las entradas del usuario.

CLASE NRO. 02: PATRÓN MODELO VISTA CONTROLADOR

Creación de Primer Proyecto con JavaScript

1. Documento HTML: Crear un archivo HTML básico con estructura.
2. Hoja de Estilos: Incorporar un archivo CSS para el estilo visual.
3. Controlador: Agregar código JavaScript para manejar la lógica.
4. Sentencias Básicas:
 - `alert()` : Muestra un mensaje emergente.
 - `console.log()` : Imprime mensajes en la consola para depuración.
 - `prompt()` : Solicita entrada del usuario.
 - `confirm()` : Muestra un cuadro de diálogo de confirmación.
↓
 - `console.error()` : Imprime errores en la consola.
 - `console.warn()` : Imprime advertencias en la consola.
 - Comentarios: Se pueden realizar en el código con `//` para comentarios de una línea o `/*...*/` para comentarios de múltiples líneas.

CLASE NRO. 03: TIPOS DE DATOS – OPERACIONES CON VARIABLES

Qué es una Variable

Una variable es un contenedor que almacena datos. Su valor puede cambiar durante la ejecución del programa.

Formas de Declarar una Variable

- Usando `var`, `let`, o `const`:
 - `var` : Declaración tradicional, puede ser re-declarada y su alcance es global o funcional.
 - `let` : Alcance de bloque, no se puede re-declarar en el mismo bloque.
 - `const` : Constante, su valor no puede cambiar.

Particularidades al Asignar un Nombre

- Deben ser descriptivos y no pueden empezar con un número.
- No pueden contener espacios y no deben usar palabras reservadas.

Diferentes Tipos de Variables

1. **Tipo Numérico:** Almacena números (enteros, decimales).
2. **Tipo String:** Almacena texto (se encierra entre comillas).
3. **Tipo Boolean:** Solo puede ser `true` o `false`.
4. **Valores Nulos:** Representa la ausencia intencionada de valor.
5. **Valores Indefinidos:** Se usa cuando no se ha asignado valor a una variable.

Bloques de Código

Se definen con llaves `{}` y ayudan a agrupar instrucciones.

JavaScript: Lenguaje Débilmente Tipado

No requiere declarar el tipo de variable, lo que permite flexibilidad pero también puede generar errores.

Operaciones entre Variables

- Suma: `a + b`
 - Resta: `a - b`
 - Multiplicación: `a * b`
 - División: `a / b`
 - Suma de Cadenas: `cadena1 + cadena2`
 - Interpolación de Cadenas: Usar template literals, `const mensaje = Hola ${nombre} ;`
-

CLASE NRO. 04: ESTRUCTURAS CONDICIONALES Y COMPARACIONES LÓGICAS

Creación del Documento HTML

Se inicia creando un archivo HTML básico que sirva como base para la interacción.

Incorporación de Código JavaScript

El código se agrega para manejar la lógica del programa y las interacciones del usuario.

Declaración de Variables Desde el Teclado

Utilizando `prompt()` para recibir datos del usuario.

- - - - -

Estructuras Condicionales

Operaciones de Comparación Lógica Básicas

- Mayor que (`>`): `A > B`
- Menor que (`<`): `A < B`
- Igual que (`=`): `A == B`
- Mayor o igual que (`>=`): `A >= B`
- Menor o igual que (`<=`): `A <= B`

Condicionales (Verdadero/Falso)

Uso de `if` y `else` para ejecutar diferentes bloques de código según las condiciones.

Inicialización de Variables

Asignar valores a las variables ingresadas por el usuario.

Operaciones Numéricas Simples

Realizar cálculos básicos y mostrar los resultados usando `console.log()`.

Visualización de Problemas

- División por Cero: Mostrar un mensaje de advertencia al intentar dividir entre cero.
- Variable No Inicializada: Demostrar el resultado de operar con una variable que no tiene valor asignado.

Estructuras Condicionales

Operaciones de Comparación Lógica Básicas

- Mayor que (`>`): `A > B`
- Menor que (`<`): `A < B`
- Igual que (`=`): `A == B`
- Mayor o igual que (`>=`): `A >= B`
- Menor o igual que (`<=`): `A <= B`

Condicionales (Verdadero/Falso)

Uso de `if` y `else` para ejecutar diferentes bloques de código según las condiciones.

CLASE NRO. 05: ERRORES TÍPICOS Y TRY/CATCH

Errores Comunes

- No Inicializar Variables: Provoca errores al intentar usar variables sin asignarles un valor.
- Comparaciones Erróneas: Usar `==` en lugar de `===` puede causar errores de lógica.
- Olvidar Paréntesis o Llaves: Necesarios para la correcta estructura del código.
- Indentación Incorrecta: La falta de organización en el código puede generar confusiones.

Introducción a Try/Catch

La sentencia `try/catch` se utiliza para manejar errores en tiempo de ejecución. Permite intentar ejecutar un bloque de código y capturar errores si ocurren, evitando que el programa se detenga.

CLASE NRO. 06: ESTRUCTURAS REPETITIVAS

Estructuras Repetitivas

- Ciclo For: Permite ejecutar un bloque de código un número determinado de veces.
- Ciclo While: Ejecuta el bloque mientras la condición sea verdadera.
- Ciclo Do While: Similar al `while`, pero garantiza que el bloque se ejecute al menos una vez.

Declaración e Inicialización de Variables

Se inicializan contadores y otras variables dentro del ciclo.

Estructuras Condicionales Dentro de Ciclos

Uso de `if/else` dentro de los ciclos para controlar el flujo.

Comprobaciones de Salida de Ciclos

Se puede usar `break` para salir de un ciclo o `continue` para omitir una iteración.

CLASE NRO. 07: OBJETOS Y ARREGLOS

Qué es un Objeto

Un objeto es una colección de propiedades, que pueden ser variables o funciones, agrupadas bajo un mismo nombre.

Declaración de Objetos

Se puede crear un objeto utilizando la sintaxis de llaves `{}`.

```
const persona = {
    nombre: "Juan",
    edad: 25,
    saludar: function() {
        console.log(`Hola, mi nombre es ${this.nombre}`);
    }
};
```

Arreglos

Un arreglo es una colección de elementos que pueden ser accedidos por su índice.

Declaración de Arreglos

Se crean utilizando corchetes `[]`.

```
const numeros = [1, 2, 3, 4, 5];
```

Acceso a Elementos

Los elementos se acceden usando el índice, comenzando desde 0.

Métodos Comunes de Arreglos

- `push()` : Agrega un elemento al final.
- `pop()` : Elimina el último elemento.
- `shift()` : Elimina el primer elemento.
- `unshift()` : Agrega un elemento al principio.

Métodos de Objetos

Los métodos son funciones que pertenecen a un objeto.

CLASE NRO. 08: CREANDO UN PROYECTO SIMPLE

Estructura del Proyecto

Definir una carpeta que contenga todos los archivos necesarios (HTML, CSS, JavaScript).

Creación de un HTML Básico

Configurar la estructura básica con los elementos `<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`.

Incorporación de CSS

Crear un archivo CSS para el estilo del proyecto y enlazarlo en el HTML.

Uso de JavaScript para Interacciones

Agregar un archivo JavaScript para manejar las interacciones y lógica de la aplicación.

Implementación de Funcionalidades

Desarrollar las características que debe tener la aplicación, utilizando las herramientas aprendidas.

Pruebas y Depuración

Ejecutar la aplicación y verificar que todas las funcionalidades trabajen como se espera. Depurar cualquier error que surja.

CLASE NRO. 09: DOM (DOCUMENT OBJECT MODEL)

Qué es el DOM

El Document Object Model (DOM) es una representación estructurada de un documento HTML. Permite a los lenguajes de programación interactuar con el contenido, la estructura y el estilo de la página web.

Cómo Funciona el DOM

El DOM convierte el documento HTML en un árbol de nodos. Cada nodo representa una parte del documento (elementos, atributos, texto, etc.). Esto permite manipular la página web de manera dinámica.

Acceso a Elementos del DOM

Se pueden utilizar diferentes métodos para acceder a los elementos del DOM:

- `getElementById(id)` : Selecciona un elemento por su ID.
- `getElementsByClassName(className)` : Selecciona todos los elementos que tienen una clase específica.
- `getElementsByTagName(tagName)` : Selecciona todos los elementos de un tipo específico.
- `querySelector(selector)` : Selecciona el primer elemento que coincide con un selector CSS.
- `querySelectorAll(selector)` : Selecciona todos los elementos que coinciden con un selector CSS.

Manipulación del DOM

- **Modificar Contenido:** Se puede cambiar el contenido de un elemento con `innerHTML` o `textContent`.
- **Modificar Atributos:** Se puede acceder y cambiar atributos usando `setAttribute()` o accediendo directamente al atributo.
- **Estilos:** Se puede modificar el estilo CSS de un elemento usando la propiedad `style`.

```
// Seleccionar un elemento
const parrafo = document.getElementById("miParrafo");
// Cambiar su contenido
parrafo.textContent = "Nuevo texto para el párrafo.";
// Cambiar su estilo
parrafo.style.color = "blue";
```

CLASE NRO. 10: EVENTOS EN JAVASCRIPT

Qué son los Eventos

Los eventos son acciones que ocurren en la página web, como clics de mouse, presiones de teclas, movimientos del mouse, etc. JavaScript permite escuchar y responder a estos eventos.

Tipos de Eventos Comunes

- `click` : Ocurre cuando un elemento es clickeado.
- `mouseover` : Ocurre cuando el mouse se mueve sobre un elemento.
- `keydown` : Ocurre cuando una tecla es presionada.
- `submit` : Ocurre cuando un formulario es enviado.

Manejo de Eventos

Se pueden manejar eventos de varias maneras:

- A través de HTML: Usando atributos como `onclick`.
- Usando `addEventListener`: Método más moderno y flexible.

```
const boton = document.getElementById("miBoton");
boton.addEventListener("click", function() {
    alert("¡Botón clickeado!");
});
```

Prevención del Comportamiento por Defecto

En algunos casos, como en los formularios, se puede prevenir el comportamiento por defecto (por ejemplo, enviar el formulario) usando `event.preventDefault()`.

```
form.addEventListener("submit", function(event) {
  event.preventDefault();
  // Lógica para manejar el envío del formulario
});
```

CLASE NRO. 11: AJAX (ASYNCHRONOUS JAVASCRIPT AND XML)

Qué es AJAX

AJAX es una técnica que permite enviar y recibir datos de un servidor de forma asíncrona sin tener que recargar la página. Esto permite crear aplicaciones web más dinámicas y rápidas.

Cómo Funciona AJAX

AJAX utiliza el objeto `XMLHttpRequest` o la API `fetch()` para realizar solicitudes al servidor. La respuesta puede ser en formato JSON, XML o HTML.

Ejemplo de Uso de AJAX

Usando `fetch()` para obtener datos de una API:

```
fetch('https://api.ejemplo.com/datos')
  .then(response => response.json())
  .then(data => {
    console.log(data);
  })
  .catch(error => console.error('Error:', error));
```

Manejo de Respuestas

La respuesta del servidor puede ser procesada y utilizada para actualizar el DOM, permitiendo que la interfaz de usuario responda dinámicamente a los cambios.

Ejemplo de Uso de Local Storage

CLASE NRO. 12: ALMACENAMIENTO LOCAL (LOCAL STORAGE Y SESSION STORAGE)

Qué es Local Storage

Local Storage permite almacenar datos en el navegador de manera persistente, incluso cuando el usuario cierra la pestaña o el navegador. Los datos se almacenan en pares clave-valor.

Uso de Local Storage

- Guardar datos: `localStorage.setItem('clave', 'valor')`
- Obtener datos: `localStorage.getItem('clave')`
- Eliminar datos: `localStorage.removeItem('clave')`
- Limpiar todos los datos: `localStorage.clear()`

Ejemplo de Uso de Local Storage

```
localStorage.setItem('nombre', 'Juan');
const nombre = localStorage.getItem('nombre');
console.log(nombre); // Juan
```

Qué es Session Storage

Session Storage es similar a Local Storage, pero los datos se almacenan solo durante la sesión de la página. Se eliminan cuando se cierra la pestaña del navegador.

Uso de Session Storage

- Funciona de manera similar a Local Storage:

```
sessionStorage.setItem('clave', 'valor');
```

CLASE NRO. 13: INTRODUCCIÓN A FRAMEWORKS Y LIBRERÍAS DE JAVASCRIPT

Qué son Frameworks y Librerías

- **Librerías:** Conjuntos de funciones y herramientas que puedes utilizar para facilitar tareas específicas (ej. jQuery).
- **Frameworks:** Estructuras más completas que proporcionan un conjunto de herramientas y convenciones para desarrollar aplicaciones (ej. React, Angular, Vue.js).

Ventajas de Usar Frameworks y Librerías

- **Ahorro de tiempo:** Permiten realizar tareas comunes más rápidamente.
 - **Estructura:** Ayudan a mantener el código organizado y modular.
 - **Comunidad:** Cuentan con soporte de comunidades grandes, lo que facilita encontrar soluciones y recursos.
-

CLASE NRO. 14: INTRODUCCIÓN A PROGRAMACIÓN ASINCRÓNICA

Qué es la Programación Asincrónica

La programación asincrónica permite que una aplicación realice tareas en segundo plano sin bloquear la ejecución de otras tareas. Esto es fundamental para mantener la interfaz de usuario responsive.

Callbacks

Funciones que se pasan como argumentos a otras funciones y que se ejecutan una vez que se completa una tarea.

Promesas

Un objeto que representa el eventual resultado (o fallo) de una operación asíncrona.

```
const miPromesa = new Promise((resolve, reject) => {
    // Lógica asíncrona
    if (todoBien) {
        resolve('Éxito');
    } else {
        reject('Error');
    }
});
miPromesa
    .then(resultado => console.log(resultado))
    .catch(error => console.error(error));
```

Async/Await

Sintaxis que simplifica el manejo de promesas, permitiendo escribir código asíncrono de manera más legible.

```
async function miFuncion() {
    try {
        const resultado = await miPromesa;
        console.log(resultado);
    } catch (error) {
        console.error(error);
    }
}
```

CLASE NRO. 15: PRÁCTICA FINAL

Desarrollo de un Proyecto Completo

En esta clase, los estudiantes aplicarán todos los conceptos aprendidos para desarrollar un proyecto completo que incluya:

- **HTML:** Estructura del sitio web.
- **CSS:** Estilo y diseño.
- **JavaScript:** Lógica de la aplicación, manipulación del DOM, manejo de eventos, y comunicación con APIs.
- **AJAX:** Para realizar solicitudes y cargar datos dinámicamente.
- **Almacenamiento:** Uso de Local Storage o Session Storage para mantener datos en la aplicación.

Ejemplo de Proyecto

- Crear una aplicación de lista de tareas donde los usuarios puedan agregar, eliminar y marcar tareas como completadas. Los datos de las tareas se pueden almacenar en Local Storage para mantenerlos incluso después de recargar la página.

Presentación de Proyectos

Los estudiantes presentarán sus proyectos finales, explicando las decisiones tomadas y los desafíos superados.

CLASE NRO. 16: CONTROL DE VERSIONES CON GIT Y GITHUB

Introducción a Git

Git es un sistema de control de versiones distribuido que permite rastrear los cambios en el código a lo largo del tiempo, facilitando la colaboración y el manejo de versiones en proyectos de software.

Conceptos Básicos de Git

- **Repositorio:** Un espacio donde se almacena el proyecto y su historial de cambios.
- **Commit:** Una captura de los cambios realizados en el proyecto. Cada commit tiene un mensaje asociado que describe los cambios.
- **Branch:** Una rama del proyecto que permite trabajar en nuevas funcionalidades sin afectar la rama principal.
- **Merge:** Fusionar los cambios de una rama en otra.

Comandos Básicos de Git

- `git init`: Inicializa un repositorio en un directorio.
- `git clone <url>`: Clona un repositorio remoto en tu máquina local.
- `git add <archivo>`: Agrega cambios de un archivo al área de staging.
- `git commit -m "mensaje"`: Crea un commit con los cambios agregados.
- `git push`: Envía los commits locales al repositorio remoto.
- `git pull`: Trae los cambios del repositorio remoto a tu máquina local.

Uso de GitHub

GitHub es una plataforma para alojar proyectos de Git de manera remota, permitiendo la colaboración y facilitando la gestión de repositorios.

Flujo de Trabajo en GitHub

1. Crear un repositorio en GitHub.
 2. Clonar el repositorio en tu máquina local.
 3. Realizar cambios en tu código y hacer commits.
 4. Enviar los cambios al repositorio remoto con `git push`.
 5. Realizar pull requests para colaborar en proyectos de otras personas.
-

CLASE NRO. 17: INTRODUCCIÓN A NODE.JS Y EL ENTORNO BACKEND

¿Qué es Node.js?

Node.js es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript del lado del servidor. Está construido sobre el motor V8 de Google Chrome y permite crear aplicaciones escalables y rápidas.

Características de Node.js

- **Basado en Eventos:** Utiliza un modelo basado en eventos y es no bloqueante, lo que lo hace ideal para aplicaciones de tiempo real.
- **Servidor:** Node.js permite crear servidores web con JavaScript.
- **Gestión de Dependencias:** A través de npm (Node Package Manager), es fácil instalar y gestionar librerías y módulos.

Creación de un Servidor Básico con Node.js

```
const http = require('http');

const servidor = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('¡Hola Mundo!');
});

servidor.listen(3000, () => {
    console.log('El servidor está corriendo en http://localhost:3000');
});
```

CLASE NRO. 18: EXPRESS.JS Y EL DESARROLLO DE APLICACIONES WEB

Introducción a Express.js

Express.js es un framework web para Node.js que facilita la creación de aplicaciones web y APIs. Proporciona una estructura robusta y herramientas útiles para manejar rutas, solicitudes HTTP y middleware.

Instalación de Express

```
npm install express
```

Servidor Básico con Express.js

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('¡Bienvenido a mi aplicación Express!');
});

app.listen(3000, () => {
    console.log('Servidor Express corriendo en http://localhost:3000');
});
```

Rutas y Métodos HTTP

Express permite manejar diferentes rutas y métodos HTTP (GET , POST , PUT , DELETE).

```
app.post('/registro', (req, res) => {
    res.send('Formulario de registro enviado.');
});
```

CLASE NRO. 19: BASES DE DATOS Y NODE.JS (MySQL)

Conexión de Node.js con una Base de Datos

Node.js se puede conectar a bases de datos relacionales como MySQL para almacenar y recuperar datos.

Instalación de MySQL para Node.js

```
npm install mysql
```

Conexión a la Base de Datos

```
const mysql = require('mysql');
const conexion = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'mi_base_de_datos'
});

conexion.connect((error) => {
    if (error) throw error;
    console.log('Conexión exitosa a la base de datos.');
});
```

Realización de Consultas

```
conexion.query('SELECT * FROM usuarios', (error, resultados) => {
  if (error) throw error;
  console.log(resultados);
});
```

CLASE NRO. 20: AUTENTICACIÓN Y SEGURIDAD EN APLICACIONES WEB

Introducción a la Autenticación

La autenticación es el proceso de verificar la identidad de un usuario. En aplicaciones web, se suele utilizar sistemas de autenticación basados en sesiones o tokens.

Manejo de Sesiones

Express permite gestionar sesiones usando paquetes como `express-session`.

```
npm install express-session
```

Uso de Middleware de Sesiones

```
const session = require('express-session');

app.use(session({
  secret: 'miSecreto',
  resave: false,
  saveUninitialized: true
}));
```

Autenticación con Passport.js

Passport.js es una librería para manejar la autenticación de usuarios en Node.js.

```
npm install passport passport-local
```

Ejemplo de Autenticación con Passport

```
const passport = require('passport');
const LocalStrategy = require('passport-local').strategy;

passport.use(new LocalStrategy((username, password, done) => {
    // Lógica de autenticación aquí
}));
```

CLASE NRO. 21: DESPLIEGUE DE APLICACIONES WEB

Opciones de Despliegue

Las aplicaciones web creadas con Node.js pueden ser desplegadas en diferentes plataformas:

- Heroku
- Vercel
- AWS (Amazon Web Services)
- DigitalOcean

Despliegue en Heroku

1. Instalar Heroku CLI.
 2. Inicializar el repositorio Git.
 3. Iniciar sesión en Heroku con `heroku login`.
 4. Crear una nueva aplicación con `heroku create`.
 5. Desplegar la aplicación con `git push heroku main`.
-

CLASE NRO. 22: TESTING Y DEPURACIÓN EN JAVASCRIPT

Testing Unitario

El testing unitario es la práctica de escribir pruebas para componentes individuales del código. Herramientas como **Jest** o **Mocha** son populares en el ecosistema de JavaScript.

Ejemplo de Prueba Unitarias con Jest

```
npm install --save-dev jest
```

```
test('Suma dos números', () => {
  expect(2 + 2).toBe(4);
});
```

CLASE NRO. 23: WEB SOCKETS Y APLICACIONES EN TIEMPO REAL

Qué son los WebSockets

Los WebSockets permiten una comunicación bidireccional en tiempo real entre un cliente y un servidor. Esto es ideal para aplicaciones como chats, juegos multijugador, etc.

Uso de WebSockets con Socket.io

```
npm install socket.io
```

Ejemplo de Aplicación en Tiempo Real

```
const io = require('socket.io')(servidor);

io.on('connection', (socket) => {
    console.log('Usuario conectado');
    socket.on('mensaje', (data) => {
        io.emit('mensaje', data);
    });
});
```

CLASE NRO. 24: PROYECTO FINAL Y PRESENTACIÓN

Desarrollo de Proyecto Completo

Los estudiantes aplicarán todos los conocimientos adquiridos para crear un proyecto final completo, integrando:

- **Frontend** con HTML, CSS, y JavaScript.
- **Backend** con Node.js y Express.
- **Base de datos** utilizando MySQL.
- Autenticación y seguridad.