UNIVERSIDAD DE LAS FUERZAS ARMADAS “ESPE”

PROGRAMACIÓN INTEGRATIVA DE COMPONENTES

Hecho por: **NRC:** 16496

- Torres Chávez Marlon Pavel **Fecha:** 2026-06-18

INTRODUCCION

La programación orientada a objetos (OOP) es un paradigma de programación que utiliza "objetos" y sus interacciones para diseñar aplicaciones y programas informáticos. Este enfoque permite modelar entidades del mundo real mediante clases y objetos, facilitando la organización, mantenimiento y reutilización del código. En el contexto de desarrollo web, JavaScript es un lenguaje fundamental que, aunque inicialmente orientado a scripts de cliente, ha evolucionado para soportar plenamente OOP.

Este proyecto tiene como objetivo aplicar los fundamentos y principios del paradigma OOP en JavaScript para desarrollar una aplicación web. La aplicación se centrará en la gestión de una biblioteca, incorporando características como la administración de libros y usuarios, utilizando conceptos como herencia, asociación, agregación y composición.

DESARROLLO

El objetivo es desarrollar una aplicación web funcional que gestione una biblioteca. La aplicación permitirá agregar, eliminar y visualizar libros y usuarios, reflejando la estructura y relaciones típicas de una biblioteca real. Este ejercicio está orientado a fortalecer la comprensión y habilidades en OOP con JavaScript, integrando aspectos teóricos y prácticos de este paradigma de programación.

Funcionalidades de la Aplicación

1. Gestión de Libros:
    a. Agregar nuevos libros con detalles como título, autor e ISBN.
    b. Eliminar libros existentes.
    c. Visualizar la lista de libros disponibles en la biblioteca.
2. Gestión de Usuarios:
    a. Agregar nuevos usuarios con detalles como nombre, edad e ID de usuario.
    b. Asignar y registrar libros prestados a los usuarios.
    c. Visualizar los detalles de los usuarios y los libros que tienen prestados.

Estructura del Proyecto

El proyecto se estructurará utilizando clases para representar las diferentes entidades de la biblioteca:

- Clase Persona: Representa una persona genérica con atributos comunes como nombre y edad.
- Clase Bibliotecario: Hereda de Persona, añadiendo un identificador único de empleado.
- Clase Usuario: Hereda de Persona, añadiendo un identificador único de usuario y una lista de libros prestados.
- Clase Libro: Define los atributos y métodos relacionados con los libros.
- Clase Biblioteca: Maneja la colección de libros y usuarios, proporcionando métodos para agregar, eliminar y obtener detalles de los mismos.

HTML:

En index.html, estructura el HTML con Bootstrap para una apariencia básica.

Incluye enlaces a Bootstrap y los archivos CSS y JavaScript personalizados.

CSS:

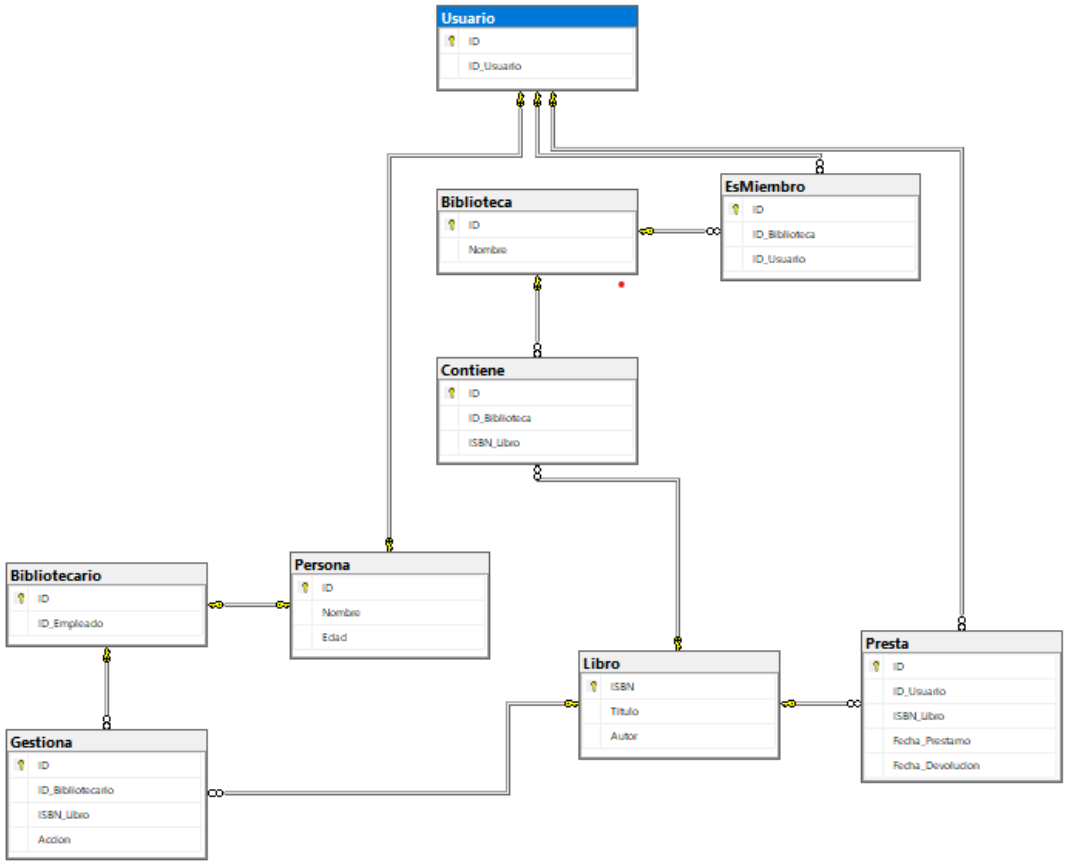
En css/styles.css, agrega estilos para la apariencia personalizada de la aplicación.

JavaScript:

En js/scripts.js, añade funcionalidad para cambiar el texto del párrafo cuando se hace clic en el botón.

EJERCICIO

Modelo Entidad Relación en SQL Server



CODIGO: Index.html

```html
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gestión de Biblioteca</title>
  <script defer src="../css/script.js"></script>
  <link href="css/estilo.css" rel="stylesheet">
</head>
<body>
  <h1>Gestión de Biblioteca</h1>

  <h2>Agregar Libro</h2>
  <form id="agregarLibroForm">
    <label for="titulo">Título:</label>
    <input type="text" id="titulo" required>
    <label for="autor">Autor:</label>
    <input type="text" id="autor" required>
    <label for="isbn">ISBN:</label>
    <input type="text" id="isbn" required>
    <button type="submit">Agregar Libro</button>
  </form>

  <h2>Biblioteca</h2>
  <div id="detallesBiblioteca"></div>

  <h2>Usuarios</h2>
  <form id="agregarUsuarioForm">
    <label for="nombreUsuario">Nombre:</label>
    <input type="text" id="nombreUsuario" required>
    <label for="edadUsuario">Edad:</label>
    <input type="number" id="edadUsuario" required>
    <label for="idUsuario">ID Usuario:</label>
    <input type="text" id="idUsuario" required>
    <button type="submit">Agregar Usuario</button>
  </form>
  <div id="detallesUsuarios"></div>
</body>
</html>
```

CODIGO: script.js

```
// Definición de la clase base Persona
class Persona {
    // Constructor de la clase Persona que inicializa nombre y edad
    constructor(nombre, edad) {
      this.nombre = nombre;
      this.edad = edad;
    }

    // Método para obtener los detalles de la persona
    obtenerDetalles() {
      return `Nombre: ${this.nombre}, Edad: ${this.edad}`;
    }
  }

  // Definición de la clase Bibliotecario que hereda de Persona
  class Bibliotecario extends Persona {
    // Constructor de la clase Bibliotecario que inicializa nombre, edad e idEmpleado
    constructor(nombre, edad, idEmpleado) {
      super(nombre, edad); // Llama al constructor de la clase base (Persona)
      this.idEmpleado = idEmpleado;
    }

    // Método para gestionar libros (placeholder para lógica futura)
    gestionarLibro(libro, accion) {
      // Lógica para gestionar libros
    }
  }

  // Definición de la clase Usuario que hereda de Persona
  class Usuario extends Persona {
    // Constructor de la clase Usuario que inicializa nombre, edad e idUsuario
    constructor(nombre, edad, idUsuario) {
      super(nombre, edad); // Llama al constructor de la clase base (Persona)
      this.idUsuario = idUsuario;
      this.librosPrestados = []; // Inicializa la lista de libros prestados
    }
```

```
    // Método para prestar un libro y añadirlo a la lista de libros prestados
    prestarLibro(libro) {
      this.librosPrestados.push(libro);
    }

    // Método para devolver un libro y eliminarlo de la lista de libros prestados
    devolverLibro(libro) {
      const indice = this.librosPrestados.indexOf(libro);
      if (indice > -1) {
        this.librosPrestados.splice(indice, 1);
      }
    }

    // Método para obtener la lista de libros prestados
    obtenerLibrosPrestados() {
      return this.librosPrestados;
    }
  }

  // Definición de la clase Libro
  class Libro {
    // Constructor de la clase Libro que inicializa título, autor e ISBN
    constructor(titulo, autor, isbn) {
      this.titulo = titulo;
      this.autor = autor;
      this.isbn = isbn;
    }

    // Método para obtener los detalles del libro
    obtenerDetalles() {
      return `Título: ${this.titulo}, Autor: ${this.autor}, ISBN: ${this.isbn}`;
    }
  }

  // Definición de la clase Biblioteca
  class Biblioteca {
    // Constructor de la clase Biblioteca que inicializa nombre, lista de libros y lista de
usuarios
    constructor(nombre) {
      this.nombre = nombre;
      this.libros = [];
      this.usuarios = [];
    }
```

```
  // Método para agregar un libro a la biblioteca
    agregarLibro(libro) {
      this.libros.push(libro);
    }

    // Método para eliminar un libro de la biblioteca
    eliminarLibro(libro) {
      const indice = this.libros.indexOf(libro);
      if (indice > -1) {
        this.libros.splice(indice, 1);
      }
    }

    // Método para agregar un usuario a la biblioteca
    agregarUsuario(usuario) {
      this.usuarios.push(usuario);
    }

    // Método para obtener los detalles de la biblioteca
    obtenerDetalles() {
      return `Biblioteca: ${this.nombre}, Libros: ${this.libros.length}, Usuarios:
${this.usuarios.length}`;
    }

    // Método para obtener la lista de libros de la biblioteca
    obtenerLibros() {
      return this.libros;
    }

    // Método para obtener la lista de usuarios de la biblioteca
    obtenerUsuarios() {
      return this.usuarios;
    }
  }

  // Espera a que el documento esté completamente cargado
  document.addEventListener('DOMContentLoaded', () => {
    // Crea una instancia de la clase Biblioteca
    const biblioteca = new Biblioteca("Biblioteca Central");
```

```
    // Maneja el evento de envío del formulario para agregar un libro

    const agregarLibroForm = document.getElementById('agregarLibroForm');
    agregarLibroForm.addEventListener('submit', (event) => {
      event.preventDefault(); // Previene el comportamiento por defecto del formulario
      const titulo = document.getElementById('titulo').value;
      const autor = document.getElementById('autor').value;
      const isbn = document.getElementById('isbn').value;
      const nuevoLibro = new Libro(titulo, autor, isbn); // Crea una instancia de la
clase Libro
      biblioteca.agregarLibro(nuevoLibro); // Agrega el libro a la biblioteca
      actualizarDetallesBiblioteca(); // Actualiza los detalles mostrados de la
biblioteca
    });

    // Maneja el evento de envío del formulario para agregar un usuario
    const agregarUsuarioForm = document.getElementById('agregarUsuarioForm');
    agregarUsuarioForm.addEventListener('submit', (event) => {
      event.preventDefault(); // Previene el comportamiento por defecto del formulario
      const nombre = document.getElementById('nombreUsuario').value;
      const edad = document.getElementById('edadUsuario').value;
      const idUsuario = document.getElementById('idUsuario').value;
      const nuevoUsuario = new Usuario(nombre, edad, idUsuario); // Crea una instancia de
la clase Usuario
      biblioteca.agregarUsuario(nuevoUsuario); // Agrega el usuario a la biblioteca
      actualizarDetallesUsuarios(); // Actualiza los detalles mostrados de los usuarios
    });

    // Función para actualizar los detalles de la biblioteca mostrados en la página
    function actualizarDetallesBiblioteca() {
      const detallesBiblioteca = document.getElementById('detallesBiblioteca');
      const libros = biblioteca.obtenerLibros().map(libro =>
libro.obtenerDetalles()).join('<br>');
      detallesBiblioteca.innerHTML = `Biblioteca:
${biblioteca.nombre}<br>Libros:<br>${libros}`;
    }

    // Función para actualizar los detalles de los usuarios mostrados en la página
    function actualizarDetallesUsuarios() {
      const detallesUsuarios = document.getElementById('detallesUsuarios');
      const usuarios = biblioteca.obtenerUsuarios().map(usuario => {
        const librosPrestados = usuario.obtenerLibrosPrestados().map(libro =>
libro.titulo).join(', ');
        return `Usuario: ${usuario.nombre}, Libros Prestados: ${librosPrestados}`;
      }).join('<br>');
      detallesUsuarios.innerHTML = `Usuarios:<br>${usuarios}`;
    }
  });
```