CENTRO DE ESTUDIOS TECNOLÓGICOS

INDUSTRIAL Y DE SERVICIOS (CETis 84)

NO.84

Desarrolla aplicaciones con conexión a base de datos

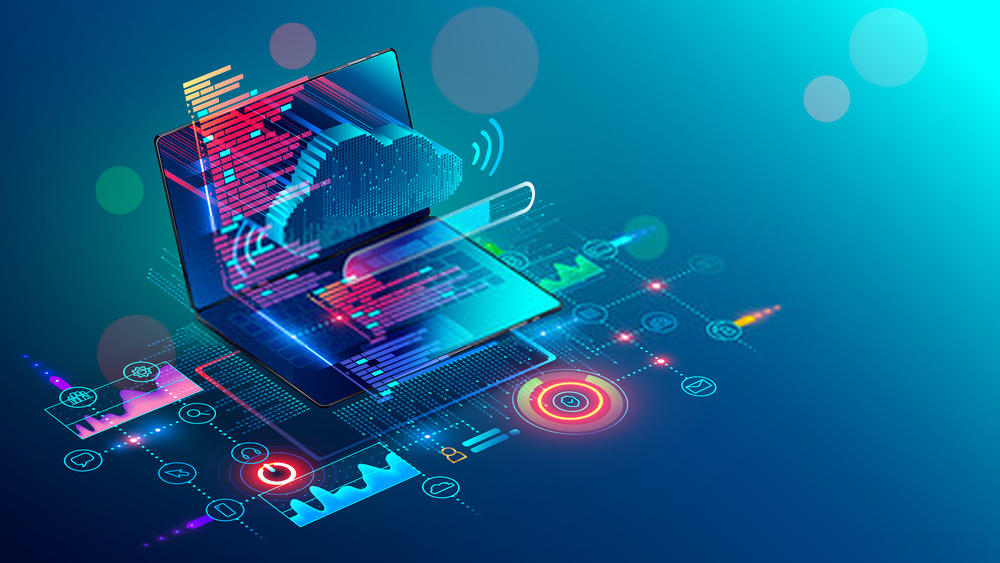
ALUMN@:

* Sánchez Basurto Alexa

DOCENTE:

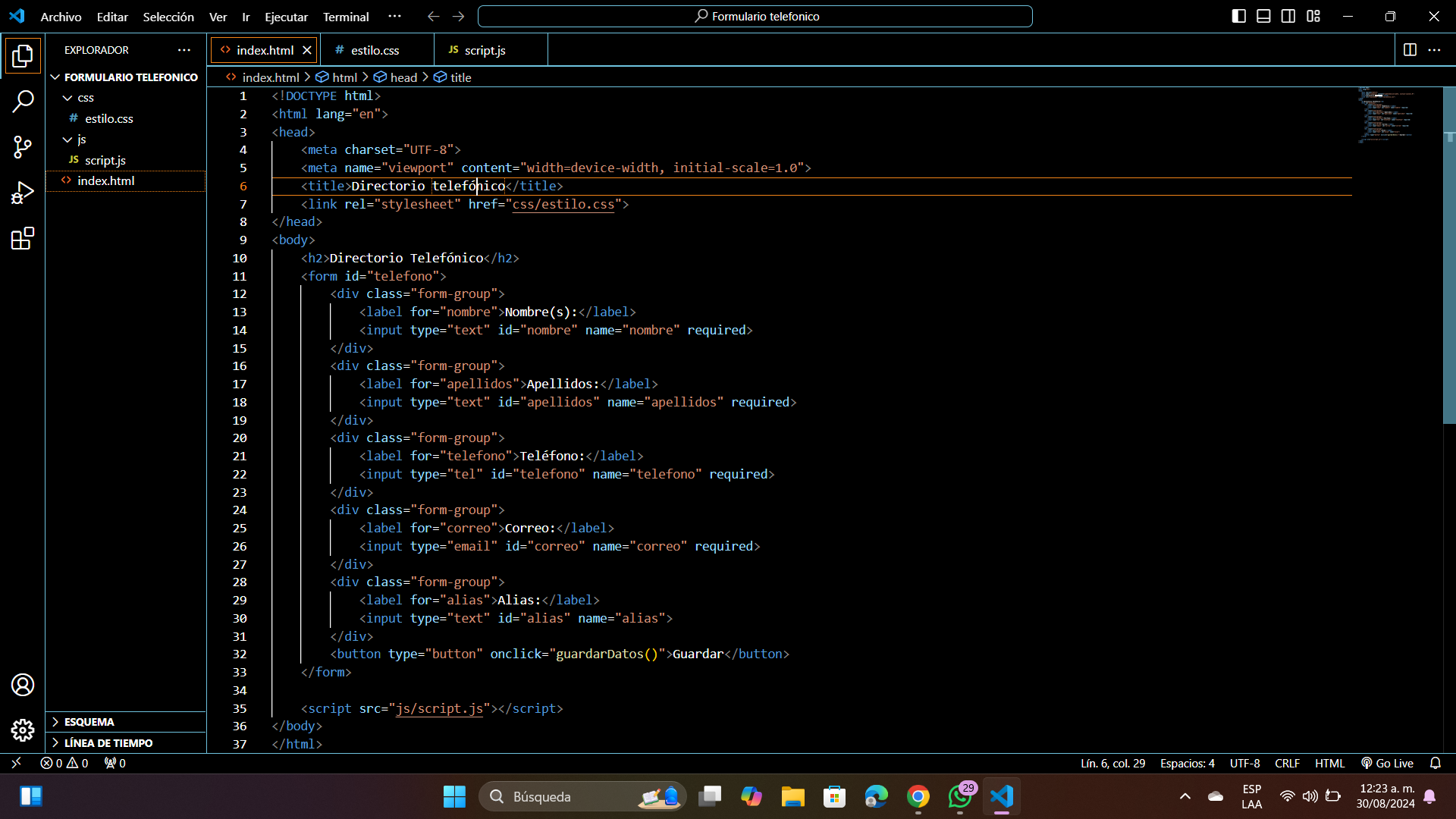
* Gabriel Ignacio China Cortez

SEMESTRE Y ÁREA: 5°E Programación



MANZANILLO, COLIMA 29/08/2024

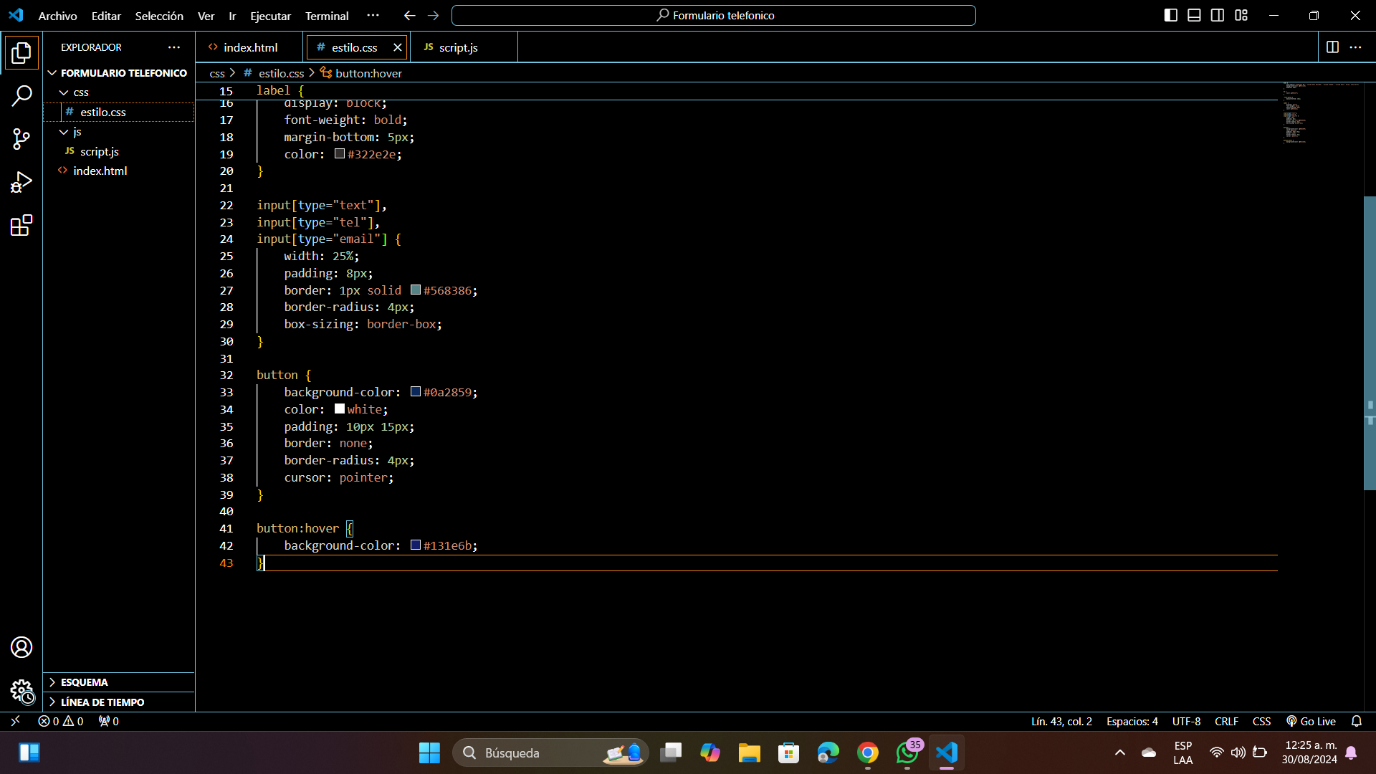
HTML



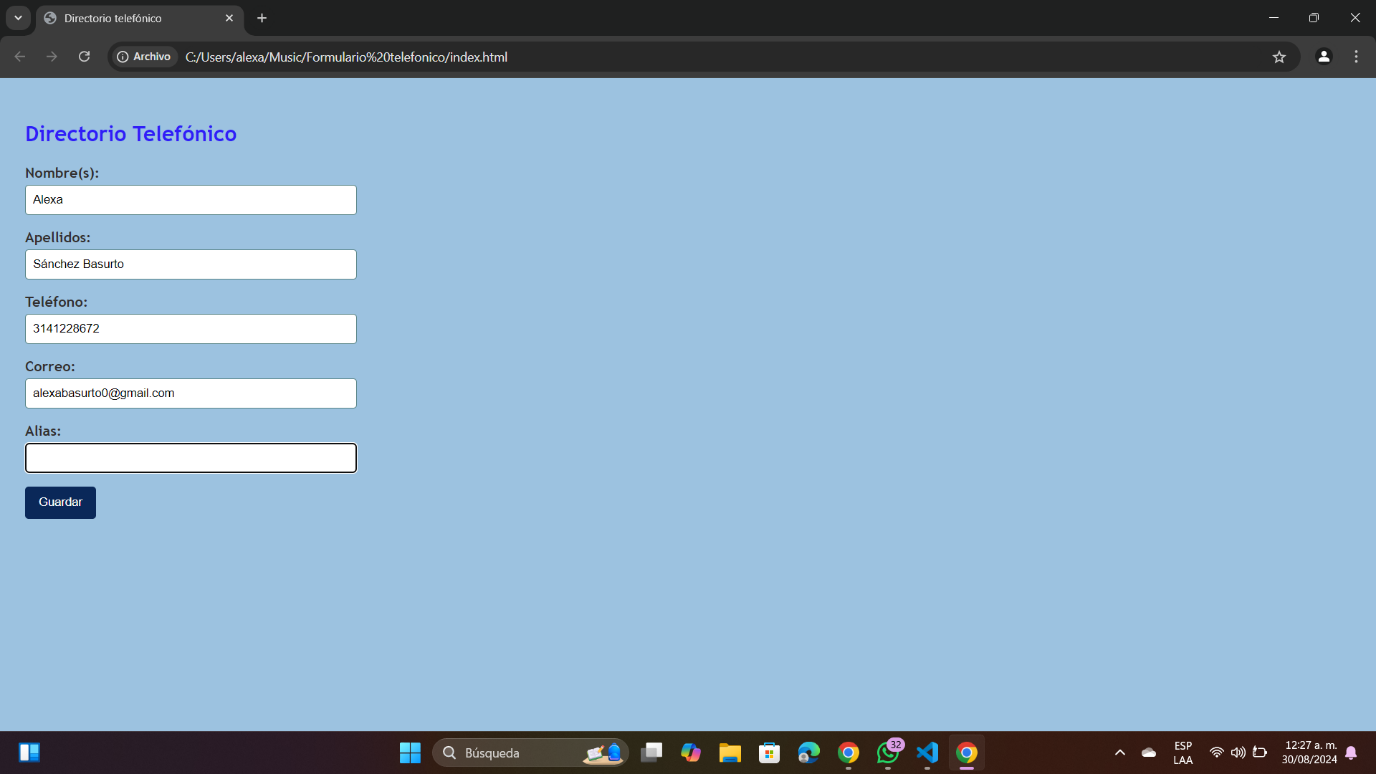
JS(script)

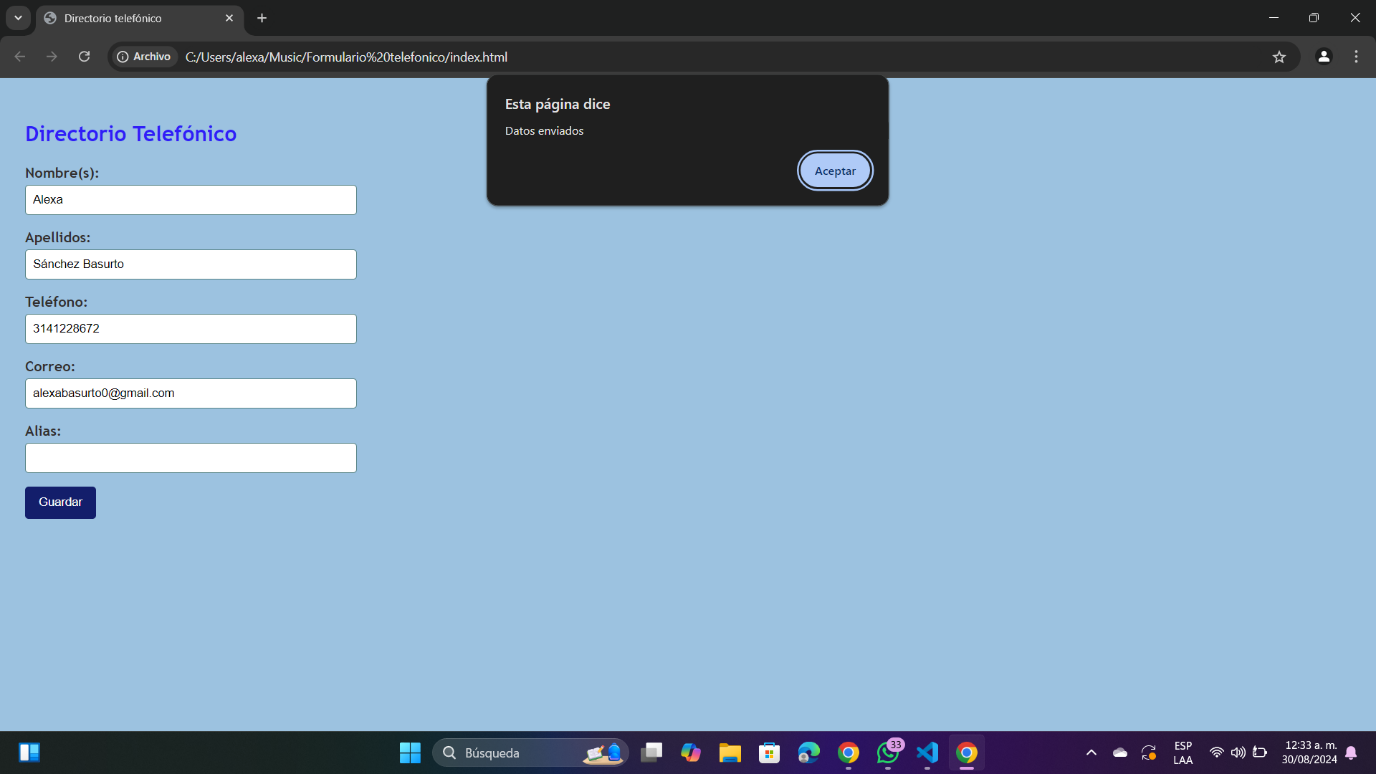


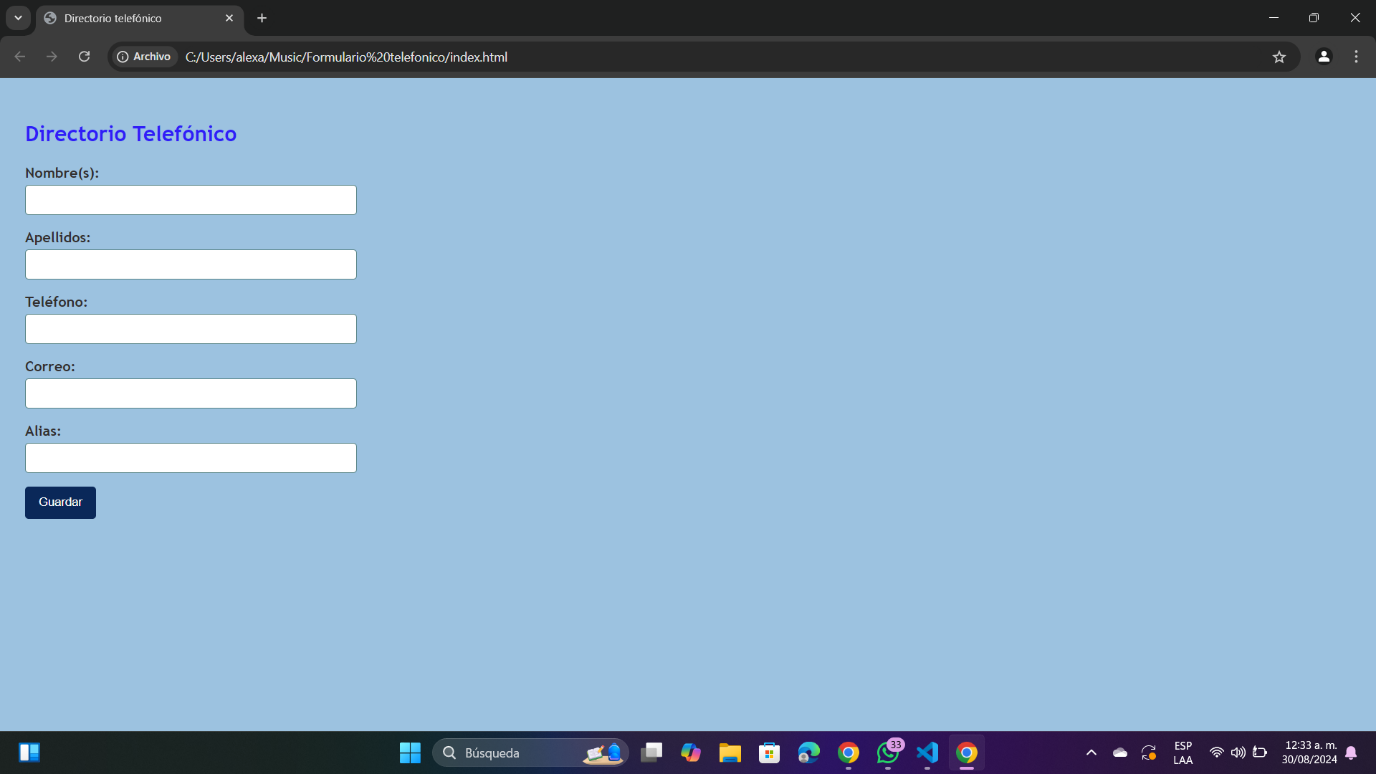
CSS



Funcionamiento







5 buenas prácticas de HTML, css, Java, php

### **HTML:**

1. **Uso de etiquetas semánticas:** Emplea etiquetas como <header>, <footer>, <article>, <section>, y <nav> para darle significado al contenido y mejorar la accesibilidad.
2. **Estructura lógica del documento:** Organiza el contenido de forma jerárquica usando encabezados (<h1>, <h2>, etc.) para que la estructura sea clara tanto para los usuarios como para los motores de búsqueda.
3. **Atributos alt en imágenes:** Siempre incluye descripciones en el atributo alt de las imágenes para mejorar la accesibilidad y SEO.
4. **Validación de HTML:** Asegúrate de que tu código HTML sea válido utilizando herramientas como el W3C Validator para evitar errores que puedan afectar el rendimiento y la accesibilidad.
5. **Evitar el uso de etiquetas obsoletas:** No utilices etiquetas que están desactualizadas o en desuso, como <font> o <center>. En su lugar, usa CSS para la presentación.

### **CSS:**

1. **Organización modular:** Divide el CSS en secciones o módulos según la funcionalidad (por ejemplo, estilos para la cabecera, cuerpo, y pie de página) para mantener un código limpio y manejable.
2. **Uso de clases en lugar de IDs para el estilo:** Las clases (.class) son más flexibles y reutilizables que los IDs (#id), lo que facilita el mantenimiento del CSS.
3. **Minimización y compresión del CSS:** Reduce el tamaño de los archivos CSS eliminando espacios en blanco y comentarios antes de subirlos a producción para mejorar los tiempos de carga.
4. **Uso de preprocesadores CSS:** Herramientas como SASS o LESS pueden hacer que tu CSS sea más mantenible y eficiente al permitir el uso de variables, funciones, y anidamiento.
5. **Compatibilidad entre navegadores:** Asegúrate de que tu CSS funcione correctamente en todos los navegadores principales y versiones recientes, utilizando herramientas como Autoprefixer para gestionar los prefijos específicos de cada navegador.

### **Java:**

1. **Nombrado consistente y descriptivo:** Utiliza nombres claros y descriptivos para variables, métodos, y clases. Sigue las convenciones de Java (camelCase para variables y métodos, PascalCase para clases).
2. **Documentación adecuada:** Comenta tu código usando Javadoc para que otros desarrolladores puedan entender fácilmente la funcionalidad del código.
3. **Modularidad:** Divide el código en métodos y clases modulares y reutilizables para mantener un código limpio y fácil de mantener.
4. **Manejo adecuado de excepciones:** Captura y maneja las excepciones de manera adecuada, utilizando bloques try-catch y lanzando excepciones personalizadas cuando sea necesario.
5. **Pruebas unitarias:** Escribe pruebas unitarias para asegurarte de que cada componente funcione como se espera. Usa frameworks como JUnit para automatizar las pruebas.

### **PHP:**

1. **Validación y saneamiento de datos:** Siempre valida y sanea los datos recibidos de los usuarios para prevenir vulnerabilidades de seguridad como SQL Injection y Cross-Site Scripting (XSS).
2. **Separación de lógica y presentación:** Sigue el patrón MVC (Modelo-Vista-Controlador) o utiliza plantillas para separar la lógica de negocio de la presentación, facilitando el mantenimiento y la escalabilidad.
3. **Uso de prepared statements:** Al interactuar con bases de datos, usa prepared statements para evitar inyecciones SQL y mejorar la seguridad.
4. **Manejo de errores:** Configura un manejo adecuado de errores y excepciones, y evita mostrar mensajes de error detallados en entornos de producción.
5. **Uso de funciones y clases:** Organiza tu código en funciones y clases para mejorar la reutilización y el mantenimiento. Usa espacios de nombres (namespaces) para evitar conflictos de nombre en proyectos grandes.