Ingeniería en Sistemas de Información	
Cátedra: Paradigmas y lenguajes de programación III	**Profesor**: Mgter. Ing. Agustín Encina
Alumno: Stupniki Hernan	**Fecha**:29/10/2025

Duración máxima: 2.30 horas 💻

Instrucciones Generales:

- Este examen es interactivo y se compone de varias decisiones que tomarás a lo largo del camino.
- Siga las instrucciones cuidadosamente en cada punto de decisión.
- La puntuación total se basará en las decisiones tomadas y en la implementación de las tareas relacionadas con cada opción.
- No se permiten consultas en línea ni colaboración con otros **estudiantes ni con un transformador generativo preentrenado**.

📜 NARRATIVA DE LA AVENTURA

Has sido contratado por una startup tecnológica que necesita urgentemente un proyecto web funcional. Tu misión es demostrar tus habilidades como desarrollador full-stack navegando por diferentes desafíos. Cada decisión que tomes definirá tu camino y las tecnologías que dominarás.

¡Tu reputación como desarrollador está en juego! 🎯

⚔️ PARTE 1: DESAFÍOS TEÓRICOS (20 puntos)

🎲 ELECCIÓN DE MISIÓN INICIAL

Antes de comenzar tu proyecto, el equipo técnico necesita evaluar tus conocimientos fundamentales. Elige tu ruta de especialización:

Elige tu Proyecto *(tildar la opción que vas a desarrollar)*:

- ☐ Ruta A: desarrolla el grupo A de preguntas.
- ☑ ~~Ruta B: desarrolla el grupo B de preguntas.~~

🌐 *RUTA A: "El Arquitecto Web" (Fundamentos y Estructura)*

Desafío 1 - Arquitectura de la Web (5 puntos)

> *El CTO te pregunta durante la reunión inicial...*

Dibuja y explica detalladamente la arquitectura Cliente-Servidor. Incluye:

- Componentes principales
- Flujo de comunicación
- Protocolos involucrados
- Ejemplo práctico con un caso real

Desafío 2 - Maestría en CSS (5 puntos)

> *El diseñador UX necesita claridad en la nomenclatura...*

Explica la diferencia entre selectores de clase y selectores de ID en CSS:

- ¿Cuándo usar cada uno?
- Nivel de especificidad
- Proporciona 2 ejemplos prácticos de cada uno aplicados a una interfaz real

Desafío 3 - Fundamentos de JavaScript (5 puntos)

> *El líder técnico evalúa tu comprensión de JS...*

Explica el concepto de variables en JavaScript:

- Propósito y utilidad
- Diferencias entre *var, let y const*
- Proporciona 3 ejemplos mostrando scope y hoisting

Desafío 4 - Introducción a PHP (5 puntos)

> *El backend developer senior te hace una pregunta clave...*

¿Qué es PHP y cuál es su rol en el desarrollo web moderno?

- Características principales
- Diferencias con lenguajes frontend
- Ejemplo de código PHP integrado en HTML (procesamiento de formulario)

🚀 *RUTA B: "El Innovador Técnico" (Evolución y Arquitectura)*

Desafío 1 - Evolución del HTML (5 puntos)

El product manager pregunta sobre tecnologías modernas...

Explica las diferencias clave entre HTML y HTML5:

- Nuevas etiquetas semánticas
- Mejoras en accesibilidad y SEO
- ¿Cómo HTML5 revolucionó el desarrollo web?

Desafío 2 - Arquitectura CSS Avanzada (5 puntos)

El tech lead quiere saber si conoces buenas prácticas...

Explica la diferencia entre arquitectura y metodología en CSS:

- Menciona al menos UNA arquitectura (ej: ITCSS, SMACSS, Atomic)
- Menciona al menos UNA metodología (ej: BEM, OOCSS, SUIT)
- ¿Por qué son importantes en proyectos grandes?

Desafío 3 - JavaScript vs PHP (5 puntos)

El arquitecto de software evalúa tu visión técnica...

Compara y contrasta JavaScript y PHP:

- Diferencias fundamentales (ejecución, tipado, uso)
- 3 escenarios donde JavaScript es más apropiado
- 3 escenarios donde PHP es más apropiado
- Ejemplo de código de cada uno

Desafío 4 - Conexión a Bases de Datos (5 puntos)

El DBA necesita confirmar tus conocimientos de persistencia...

Describe los conceptos fundamentales para conectar PHP con una Base de Datos:

- Métodos de conexión (MySQLi vs PDO)
- Pasos para establecer conexión
- Manejo de errores
- Ejemplo de código con consulta preparada

🎨 PARTE 2: PROYECTO PRÁCTICO (80 puntos - *distribuidos en 4 niveles*)

🗺️ Nivel 1 : ELECCIÓN DE PROYECTO BASE (20 puntos)

⚠ **REGLA CRÍTICA DE NOMENCLATURA:** Todos los archivos, carpetas, clases, funciones, tablas, etc., deben usar como prefijo tus iniciales.

Ejemplo: Si eres María González López (MGL):

- 📁 Carpeta: mgl_assets/
- 📄 CSS: mgl_estilos.css
- 🗄️ Base de datos: mgl_parcial_plp3
- 📊 Tabla: mgl_usuarios
- ⚙️ Función: function mgl_validar()
- 🖼️ Imagen: mgl_logo.png
- 🎯 Clase CSS: .mgl-header

🎯 **MISIÓN PRINCIPAL - Elige tu Proyecto *(tildar la opción que vas a desarrollar)*:**

- ☐ Opción A: "MusicStream" - Plataforma de Música Online
- ☑ ~~Opción B: "FoodExpress" - Sistema de Pedidos Online.~~
- ☐ Opción C: "QuizMaster" - Plataforma de Trivia.

🎵 PROYECTO A: "MusicStream" - Plataforma de Música Online

Una discográfica indie quiere su propia plataforma de streaming

Requisitos Funcionales:

- Catálogo de álbumes/canciones con reproductor básico (mínimo 8 items)
- Sistema de búsqueda por artista, género o álbum
- Formulario de suscripción con validación
- Listas de reproducción o favoritos (almacenadas en BD)
- Panel para agregar/editar canciones (CRUD)

Requisitos No Funcionales:

- Mínimo 3 secciones distintas (header, galería, formulario)
- Diseño responsive (3 breakpoints)
- Navegación intuitiva y accesible
- Código comentado y estructura modular

🍕 PROYECTO B: "FoodExpress" - Sistema de Pedidos Online

Un restaurante local necesita digitalizar sus pedidos

Requisitos Funcionales:

- Menú de productos con categorías (mínimo 10 productos)
- Carrito de compras dinámico con subtotales
- Formulario de pedido que guarda en BD
- Sistema de filtrado por categoría
- Panel administrativo para gestionar productos

Requisitos No Funcionales:

- Mínimo 3 secciones (menú, carrito, checkout)
- Responsive design con mobile-first
- Feedback visual en todas las interacciones
- Tiempo de carga optimizado

📚 PROYECTO C: "QuizMaster" - Plataforma de Trivia

Una institución educativa quiere gamificar el aprendizaje

Requisitos Funcionales:

- Sistema de preguntas con múltiple opción (mínimo 15 preguntas)
- Validación de respuestas en tiempo real
- Sistema de puntuación y temporizador
- Tabla de mejores puntajes (stored en BD)
- Categorías temáticas con dificultad variable

Requisitos No Funcionales:

- Mínimo 3 secciones (inicio, juego, resultados)
- Animaciones fluidas y feedback inmediato
- Diseño responsive
- Interfaz intuitiva sin instrucciones complejas

⚡ NIVEL 2: Interactividad con JavaScript (20 puntos)

📝 **Documenta:** Comenta la funcionalidad al inicio del archivo JS (3-5 líneas).

Para PROYECTO A (MusicStream):

Implementar: Reproductor Interactivo con Playlist

- Play/Pause/Skip con controles visuales
- Barra de progreso funcional
- Lista de reproducción dinámica
- Almacenar última canción reproducida

Para PROYECTO B (FoodExpress):

Implementar: Carrito de Compras Dinámico

- Agregar/eliminar productos sin recargar
- Cálculo automático de subtotales
- Validación de cantidades
- Mostrar contador de items en el carrito

Para PROYECTO C (QuizMaster):

Implementar: Lógica de Juego Completa

- Algoritmo de validación de respuestas
- Sistema de puntuación progresiva
- Temporizador con penalización
- Feedback visual inmediato (correcto/incorrecto)

🔧 NIVEL 3: Backend con PHP (20 puntos)

⚠️ **OBLIGATORIO:** Conexión e interacción con Base de Datos MySQL

📝 **Documenta:** Comenta la funcionalidad PHP implementada.

Implementación Requerida:

Para MusicStream:

- CRUD de canciones/álbumes
- Sistema de favoritos persistente
- Búsqueda con queries SQL

Para FoodExpress:

- CRUD de productos
- Registro de pedidos en BD
- Cálculo de totales en servidor

Para QuizMaster:

- Banco de preguntas desde BD
- Sistema de ranking persistente
- Registro de partidas jugadas

📊 **Base de Datos:**

Crear con mínimo 2 tablas relacionadas:

- Claves primarias y foráneas
- Datos de prueba (mínimo 10 registros)

Exportar:

- **[iniciales]_estructura.sql**
- **[iniciales]_datos.sql**

🎨 NIVEL 4: Diseño y Experiencia Visual (20 puntos)

📝 **Documenta:** Explica tus decisiones de diseño (paleta, tipografía, layout).

Requisitos Funcionales:

- Paleta de colores coherente (4-5 colores)
- Tipografía consistente (jerarquía clara)
- Responsive: 3 breakpoints mínimo
- Menú adaptativo (hamburguesa en mobile)

Requisitos No Funcionales:

- Transiciones suaves (hover, focus)
- Loading states visibles
- Contraste adecuado (accesibilidad)
- Espaciado uniforme (grid/flexbox)

📦 ENTREGA FINAL

Estructura del Proyecto:

```
[INICIALES]_Parcial_PLP3/
├── index.html (o index.php)
├── css/
│   └── [iniciales]_estilos.css
├── js/
│   └── [iniciales]_script.js
├── includes/
│   └──[iniciales]_conexion.php
│
├── [iniciales]_assets/
│   └── images/
│
├── database/
│   ├── [iniciales]_estructura.sql
│   └── [iniciales]_datos.sql
│
├── docs/
│   └── [APELLIDO]_[NOMBRE]_Parcial.pdf
└── README.md
```

📤 Método de Entrega:

1. Archivo ZIP: [APELLIDO]_[NOMBRE]_PLP3.zip
2. Repositorio GIT con commits descriptivos
3. Subir a aula virtual dentro del tiempo del examen

🏆 EVALUACIÓN

Puntos Bonus (+10 máximo):

- ✨ Creatividad excepcional (+3)
- 🔒 Seguridad (prepared statements) (+2)
- ♿ Accesibilidad (ARIA, semántica) (+2)
- 📱 Features avanzadas (+3)

Penalizaciones:

- ❌ Sin nomenclatura de prefijos: -5 pts
- ❌ Código sin comentarios: -3 pts

- ❌ No funciona: -10 pts
- ❌ Plagio: 0 en el parcial

🎯 CHECKLIST FINAL

- ☑ ~~Teoría completa~~
- ☑ ~~Nomenclatura con prefijo~~
- ☑ ~~Proyecto funcional~~
- ☑ ~~BD exportada~~
- ☑ ~~CSS responsive~~
- ☑ ~~JavaScript comentado~~
- ☑ ~~PHP con BD funcionando~~
- ☑ ~~README.md claro~~
- ☑ ~~GIT con commits~~
- ☑ ~~ZIP correctamente nombrado~~

🚀 ¡ÉXITO, DESARROLLADOR!

🖖 ¡Que la fuerza del código esté contigo! 🎃

Desarrollo

Desafío 1 – Evolución del HTML
HTML5 trajo un cambio enorme frente al HTML clásico. Sumó etiquetas semánticas como header, nav, section, article o footer que hacen el código más claro y fácil de interpretar, tanto para buscadores como para lectores de pantalla. También incorporó video, audio y validaciones sin depender de plugins. En general, volvió la web más ordenada, accesible y moderna.

Desafío 2 – Arquitectura CSS Avanzada
La arquitectura en CSS se refiere a cómo se organiza todo el código y las capas de estilo, mientras que la metodología define cómo se nombran las clases y se reutilizan los componentes. Un ejemplo de arquitectura es ITCSS y de metodología BEM. Usarlas ayuda a mantener el proyecto limpio, escalable y fácil de entender cuando hay muchos desarrolladores.

Desafío 3 – JavaScript vs PHP
JavaScript se ejecuta en el navegador (y también en el servidor con Node.js), es asíncrono y se usa para interfaces interactivas y tiempo real. PHP corre en el servidor y genera el contenido que llega al usuario, siendo ideal para sitios tradicionales y sistemas como WordPress.

Ejemplo Javascript:

ejemplojs.js
```
1   console.log("Hola desde JavaScript");
```

Ejemplo php:

ejemplophp.php
```
1   <?php echo "Hola desde PHP"; ?>
2
```

Desafío 4 – Conexión a Bases de Datos
En PHP se puede conectar una base de datos con MySQLi o PDO, siendo PDO más seguro y compatible con varios motores. Para hacerlo se crea una conexión, se prepara la consulta, se pasan los datos y se ejecuta.

conexion_db.php
```
1   $pdo = new PDO("mysql:host=localhost;dbname=app", "root", "");
2   $stmt = $pdo->prepare("INSERT INTO users (name) VALUES (:name)");
3   $stmt->execute([':name' => 'Tobías']);
4
```

Documentación de la funcionalidad PHP implementada

El proyecto FoodExpress está hecho con PHP y MySQL, y la idea fue mantener una estructura simple pero bien organizada.
La conexión a la base de datos se maneja desde un solo archivo (`hs_conexion.php`) usando PDO, lo que permite trabajar con consultas preparadas y manejar errores de forma más segura.

El carrito de compras (`hs_carrito.php`) funciona con sesiones, guardando los productos que el usuario va agregando.
Desde este archivo también se manejan las acciones del carrito con AJAX, como agregar, cambiar cantidades o eliminar productos, sin tener que recargar la página.
Los precios, subtotales y totales siempre se recalculan en el servidor, para evitar que alguien los modifique desde el navegador.

En el checkout, el archivo `hs_guardar_pedido.php` recibe los datos del formulario, valida los campos y guarda el pedido en la base de datos.
Para eso se usa una transacción que inserta primero el pedido y después el detalle con los productos. Si algo falla, se cancela todo para que la información quede consistente.

En la parte de administración (`hs_admin`) se pueden agregar, editar o desactivar productos, sin borrarlos definitivamente (soft delete).
Esto permite mantener la información y tener más control sobre el catálogo.
Todas las consultas usan sentencias preparadas, lo que evita problemas de seguridad como la inyección SQL.

En general, la parte PHP del proyecto busca que todo funcione de forma segura, ordenada y clara, combinando la lógica del servidor con la parte visual para que la experiencia sea fluida, sin recargar las páginas y manteniendo los datos siempre correctos.

Documentación de diseño (paleta, tipografía y layout)

El diseño de FoodExpress se pensó para que sea moderno, claro y fácil de usar, tanto en computadora como en celular.
Se eligió una paleta de colores cálida para transmitir cercanía y energía:

Rojo como color principal (`#ff6347`) y un tono más oscuro para los botones en hover (`#e5533d`).

Blanco y gris claro para los fondos, y texto en gris oscuro (#222) para asegurar buena legibilidad.

También se usan colores de estado: verde para éxito, amarillo para aviso y rojo para error.

En cuanto a la tipografía, se usa el estilo del sistema (por ejemplo Roboto, Segoe UI o Arial) para que cargue rápido y se vea bien en todos los dispositivos.
Los títulos son grandes y marcados (h1 de 32px, h2 de 24px, h3 de 18px) y el texto base de 16px, con una jerarquía que ayuda a leer fácilmente.

El layout está hecho con grid y flexbox, usando espacios uniformes entre los elementos para mantener el orden.
El sitio es totalmente responsive, con tres tamaños principales:

En celular, todo se muestra en una columna y el menú se vuelve tipo hamburguesa.

En tablet, se muestran tres columnas y el menú ya aparece desplegado.

En pantallas grandes, hay hasta cuatro columnas y más margen en los costados.

También se agregaron transiciones suaves en botones y enlaces para dar una sensación más fluida, y un loader con spinner cuando se envían formularios o se cargan datos.
Los botones e inputs muestran un borde o sombra al hacer foco, para mejorar la accesibilidad y que se pueda navegar con teclado.