**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

| Ingeniería en Sistemas de Información | |
|---|---|
| **Cátedra**: Paradigmas y lenguajes de programación III | **Profesor**: Mgter. Ing. Agustín Encina |
| **Alumno**:Quintana Sofía | **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.

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🌐 *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)

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🚀 *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

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🎨 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.~~

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🎵 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

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🍕 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

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 📚 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

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## ⚡ 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)

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🔧 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**

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 🎨 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)

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## 📦 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)

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

**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!**

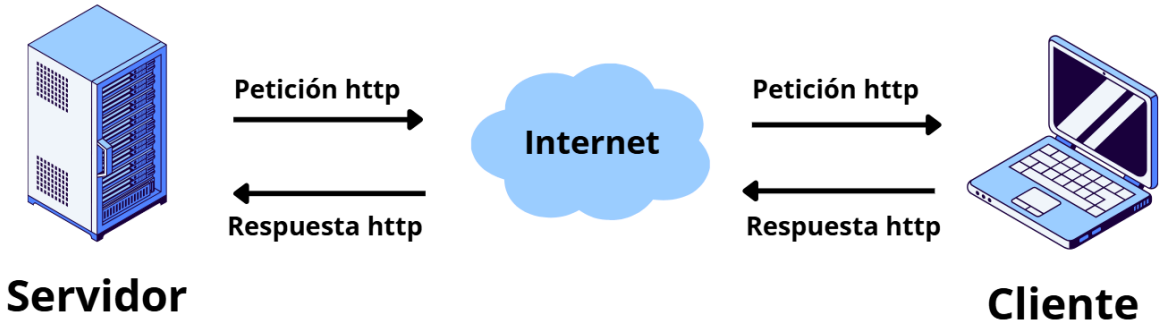
🖐️ ¡Que la fuerza del código esté contigo! 🎃

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

Desarrollo

**Parte 1: Teoría**

**Desafío 1) Arquitectura Cliente - Servidor**




En la arquitectura web cliente–servidor, se empieza en el navegador del usuario, que prepara una petición HTTP(S) cuando se escribe una URL o se hace clic en un enlace. El nombre de dominio se resuelve en DNS para obtener la dirección IP del servidor y, con eso, el navegador abre una conexión TCP y negocia TLS si es HTTPS, garantizando un canal cifrado y auténtico. La solicitud llega al servidor web (por ejemplo, Apache o Nginx), que puede devolver directamente archivos estáticos (HTML, CSS, imágenes, JS) o, si se trata de contenido dinámico, derivar la petición al runtime de la aplicación (PHP-FPM o Apache con PHP). En ese punto la lógica de negocio entra en juego: el código PHP valida datos, gestiona sesiones, aplica reglas y consulta la base de datos MySQL mediante un conector (PDO/MySQLi). La BD responde con los registros solicitados, la aplicación construye la respuesta final (HTML para renderizar o JSON para llamadas AJAX/fetch) y el servidor web la envía de vuelta por la misma conexión segura hasta el navegador, que la interpreta y muestra al usuario.

Este esquema admite componentes opcionales que mejoran rendimiento y escalabilidad, como un firewall o balanceador que filtra y reparte tráfico entre varios servidores, una capa de caché para acelerar respuestas repetidas y una CDN o almacenamiento dedicado para servir contenidos pesados más cerca del usuario. En un entorno local típico (XAMPP), la secuencia es idéntica aunque todo conviva en la misma máquina: el navegador solicita

http://localhost/sq_Parcial/sq_index.php, Apache recibe la petición, ejecuta el script PHP, éste se conecta a MySQL (por ejemplo 127.0.0.1:3306), arma el HTML con los datos obtenidos y lo devuelve al navegador. Así, cliente, servidor web, aplicación y base de datos cooperan mediante DNS, TCP, TLS y HTTP para entregar páginas y funcionalidades de forma segura y confiable.

Ejemplo con el proyecto del parcial: Un alumno accede a sq_QuizMaster desde su navegador (cliente). Primero, el dominio se resuelve por DNS y el navegador abre una conexión TCP al puerto 443; se negocia TLS para cifrado. Segundo, el cliente envía un HTTP/1.1 o HTTP/2 GET a Nginx/Apache (servidor web), que deriva la ejecución a PHP-FPM. Tercero, el backend consulta el banco de preguntas en MySQL mediante SQL y compone la respuesta HTML/CSS/JS. Cuarto, el navegador renderiza y, durante el juego, realiza peticiones AJAX (fetch) HTTP POST a /includes/sq_preguntas.php para validar respuestas, recibiendo JSON. Quinto, los recursos estáticos (JS/CSS) pueden entregarse desde caché/CDN; la sesión del jugador se identifica con cookie o token. Sexto, al finalizar, el cliente envía un POST con el puntaje; el servidor persiste el resultado en MySQL y devuelve confirmación.

**Desafío 2: Selectores de clase y selectores de ID en CSS:**

| Selector de clase | Selector de ID |
|---|---|
| Una clase(precedida por .) se usa para estilos reutilizables que pueden aplicarse a varios elementos. Tiene especificidad media, por lo que es fácil de sobreescribir y es la base del maquetado cotidiano.<br><br>*Ejemplo (con el proyecto del parcial):*<br><br>.sq_btn { padding:10px 14px; border-radius:12px; } | Un ID (precedido por #), identifica un único elemento en la página. Tiene especificidad alta, es decir, si un ID y una clase entran en conflicto, gana el ID; por eso se recomienda no usarlos para estilos generales, sino para casos puntuales.<br><br>*Ejemplo (con el proyecto del parcial):*<br><br>#sq_btn_empezar { background:var(--sq-primary); } |

| | |
|---|---|
| .sq_card { background:#fff; border:1px solid var(--sq-border); } | #sq_resultados { display:none; } |

**Desafío 3: Fundamentos de JavaScript**

Las variables en JavaScript sirven para guardar valores (números, textos, booleanos, objetos, funciones, etc.) y poder usarlos y modificarlos a lo largo del programa. Permiten nombrar datos, evitar repetir cálculos, comunicar información entre funciones y controlar el estado de una aplicación.

Hay tres formas para poder declararlas:

- **let:** declara variables mutables (pueden reasignarse) con ámbito de bloque (solo existen dentro de { ... }). Es la opción por defecto para valores que van a cambiar.

  *Ejemplo:* let contador = 0;
- **const:** declara constantes (no se pueden reasignar) también con ámbito de bloque. Si la constante es un objeto o arreglo, su referencia es constante, pero sí se pueden modificar sus propiedades.

  *Ejemplo:* const meses = ["Enero", "Febrero"];
- **var:** forma histórica. Tiene ámbito de función (o global si no hay función) y hoisting peculiar, sin embargo, facilita errores de alcance y de redeclaración.

  *Ejemplo:* var nombre = "Sofía";

Ejemplos:

1. Hoisting con var (se “eleva” solo la declaración)

```
console.log(a); // undefined (no ReferenceError)
var a = 5;      // declaración hoisted; asignación NO
```

2. Bloque con let/const (no hay hoisting utilizable; TDZ)

```
console.log(b); // ReferenceError (Temporal Dead Zone)
let b = 2;

if (true) {
  const c = 7;
}
console.log(c); // ReferenceError (scope de bloque)
```

3. var es de función (no de bloque) vs let

```
if (true) { var x = 1; }
console.log(x); // 1  (var "escapa" del bloque)

for (let i = 0; i < 2; i++) {}
console.log(i); // ReferenceError (let no escapa)
```

**Desafío 4: Introducción a PHP**

PHP es un lenguaje de programación del lado del servidor diseñado para generar contenido dinámico en la web. En el desarrollo moderno, su rol es recibir solicitudes HTTP, procesar lógica de negocio, interactuar con bases de datos (por ejemplo, MySQL mediante PDO) y devolver respuestas (HTML, JSON para APIs o fragmentos para AJAX). PHP encaja bien en arquitecturas clásicas LAMP (Linux, Apache, MySQL, PHP) y convive con frameworks (Laravel, Symfony) que aportan estructura, seguridad y escalabilidad. Aunque en la actualidad existen muchas alternativas (Node, Python, Go), PHP sigue siendo muy usado por su curva de aprendizaje accesible, enorme ecosistema y hosting económico.

**Características principales**

- Servidor/Back-end: se ejecuta en el servidor y nunca expone su código al navegador.

- Integración con HTML sencilla: permite "incrustar" PHP dentro de plantillas HTML.
- Acceso a bases de datos: soporte robusto vía PDO con consultas preparadas.
- Gestión de sesiones nativa (login, carritos, permisos).
- Amplio ecosistema: frameworks (Laravel, Symfony), CMS (WordPress, Drupal), librerías.
- Portabilidad: corre en la mayoría de hostings y sistemas operativos.

**Diferencias con lenguajes frontend (HTML/CSS/JS)**

- PHP corre en el servidor; HTML/CSS/JS corren en el navegador.
- PHP procesa datos, reglas y DB; HTML estructura, CSS estiliza, JS interactúa en el cliente.
- el código PHP no se ve desde el cliente; el HTML/CSS/JS sí.
- PHP genera el HTML/JSON que el navegador recibe y muestra.
- PHP maneja sesiones y seguridad del lado servidor; JS maneja UI y eventos del usuario.

Ejemplo:

```php
<?php
$enviado = false;
$errores = [];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $nombre  = trim($_POST['nombre']  ?? '');
  $email   = trim($_POST['email']   ?? '');
  $mensaje = trim($_POST['mensaje'] ?? '');

  // Validar
  if ($nombre === '') { $errores[] = 'El nombre es obligatorio.'; }
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) { $errores[] = 'Email inválido.'; }
  if ($mensaje === '') { $errores[] = 'El mensaje no puede estar vacío.'; }

  //  Resultado
```

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

```
  if (!$errores) { $enviado = true; }
}

function e($str){ return htmlspecialchars($str ?? '', ENT_QUOTES,
'UTF-8'); }
?>
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Contacto</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>

body{font-family:system-ui,Arial,sans-serif;max-width:720px;margin:32px
auto;padding:0 16px;}
    form{display:grid;gap:12px;background:#fff;border:1px solid
#e5e7eb;border-radius:12px;padding:16px;}
    label{font-weight:600}
    input,textarea{width:100%;padding:10px;border:1px solid
#cbd5e1;border-radius:8px}
    .ok{background:#ecfdf5;border:1px solid
#10b981;color:#065f46;padding:12px;border-radius:8px}
    .err{background:#fef2f2;border:1px solid
#ef4444;color:#7f1d1d;padding:12px;border-radius:8px}
    button{padding:10px
14px;border:0;border-radius:8px;background:#2563eb;color:#fff;cursor:po
inter}
    button:hover{filter:brightness(0.95)}
  </style>
</head>
<body>

<h1>Contacto</h1>
```

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

```
<?php if ($enviado): ?>
  <p class="ok">¡Gracias, <?= e($nombre) ?>! Recibimos tu mensaje.</p>
<?php else: ?>
  <?php if ($errores): ?>
    <div class="err">
      <strong>Revisá estos puntos:</strong>
      <ul>
        <?php foreach ($errores as $e) echo '<li>'.e($e).'</li>'; ?>
      </ul>
    </div>
  <?php endif; ?>

  <form method="post" action="">
    <div>
      <label for="nombre">Nombre</label>
      <input id="nombre" name="nombre" value="<?= e($_POST['nombre'] ??
'') ?>">
    </div>
    <div>
      <label for="email">Email</label>
      <input id="email" name="email" type="email" value="<?=
e($_POST['email'] ?? '') ?>">
    </div>
    <div>
      <label for="mensaje">Mensaje</label>
      <textarea id="mensaje" name="mensaje" rows="5"><?=
e($_POST['mensaje'] ?? '') ?></textarea>
    </div>
    <button type="submit">Enviar</button>
  </form>
<?php endif; ?>
</body>
</html>
```

**Examen Interactivo de PLP III:** La Aventura del Desarrollador Web 🎮

## Parte 2: "QuizMaster" - Plataforma de Trivia

### Index

La página inicial presenta el flujo del juego sq_QuizMaster: permite ingresar el nombre del jugador y elegir categoría y dificultad. Desde allí se inicia la partida y se navega entre Inicio - Juego - Resultados con un router por hash. La interfaz usa componentes coherentes (header, HUD de puntaje/tiempo, tarjeta de pregunta, ranking) y está adaptada a móvil con un menú hamburguesa.

### Funcionalidad JavaScript

El script controla todo el ciclo del quiz: carga 15 preguntas vía fetch (GET), mezcla opciones, gestiona temporizador por pregunta, valida en tiempo real la respuesta con fetch (POST) y calcula el puntaje progresivo (bonus por rapidez y penalización por error/tiempo). También administra el estado del juego (posición, racha, correctas), muestra feedback visual inmediato y al finalizar arma el resumen y solicita el Top 10 del ranking.

### PHP

El backend expone APIs simples:

- sq_preguntas.php entrega preguntas y opciones (GET) y valida respuestas (POST) sin revelar la correcta.
- sq_ranking.php persiste la partida (jugador, puntaje, aciertos, duración) y devuelve el ranking.

La conexión se realiza con PDO y prepared statements para seguridad y compatibilidad, aislando credenciales en sq_conexion.php.

### SQL

La base sq_parcial_plp3 incluye tablas normalizadas: sq_categoria, sq_pregunta, sq_opcion (1 correcta por pregunta), sq_partida y sq_partida_respuesta. Hay claves primarias/foráneas con cascada donde corresponde, índices para consultas frecuentes y restricciones de unicidad para evitar duplicados. Se proveen scripts separados de estructura y datos para instalación limpia y verificación de integridad.

## CSS

La hoja sq_estilos.css define una paleta rosada con variables CSS, tipografía del sistema y componentes reutilizables (botones, tarjetas, alertas). Se cuida la accesibilidad con :focus-visible, buen contraste y tamaños táctiles. El layout usa Grid/Flex, sombras suaves y transiciones; el menú es adaptativo y la UI mantiene consistencia en desktop y mobile.