PARTE 1 – DESAFÍOS TEÓRICOS

Ruta B – “El Innovador Técnico”

Alumno: Tarnowski, Tobías Ian

Carrera: Ingeniería en Sistemas de Información – UCP

Cátedra: Paradigmas y Lenguajes de Programación III

Profesor: Mgter. Ing. Agustín Encina

Fecha: 29/10/2025

Desafío 1 – Evolución del HTML

Diferencias entre HTML tradicional y HTML5, con foco en semántica, accesibilidad y evolución tecnológica.

HTML5 significó una gran actualización respecto al HTML clásico.

Mientras que las versiones anteriores servían solo para estructurar texto e imágenes, HTML5 incorporó etiquetas semánticas como <header>, <section>, <article>, <aside> y <footer>, que aportan significado y jerarquía al contenido, ayudando al SEO y a la accesibilidad.

También integró soporte nativo para audio y video mediante las etiquetas <audio> y <video>, eliminando la dependencia de complementos externos como Flash.

En cuanto a accesibilidad, introdujo nuevos atributos ARIA y una mejor semántica para que los lectores de pantalla interpreten correctamente la información.

En resumen, HTML5 transformó el desarrollo web moderno, permitiendo crear sitios más organizados, accesibles, interactivos y adaptables a dispositivos móviles sin depender de tecnologías externas.

Desafío 2 – Arquitectura CSS Avanzada

Diferencia entre arquitectura y metodología en CSS, con ejemplos prácticos.

En CSS, una arquitectura determina cómo se estructura y ordena el código a nivel global, mientras que una metodología define la forma de nombrar y relacionar los selectores para mantener un código claro y fácil de mantener.

Arquitectura (ejemplo: ITCSS): organiza los estilos por niveles, desde las reglas más generales hasta las más específicas (configuración, utilidades, componentes, etc.), para evitar conflictos entre hojas y garantizar escalabilidad.

Metodología (ejemplo: BEM): siglas de Block, Element, Modifier. Define una convención de nombres predecible, como .menu__item--activo, que facilita entender la relación entre componentes.

En proyectos grandes, aplicar ambas es fundamental para mantener consistencia visual, mejorar la colaboración entre desarrolladores y reducir la duplicación de código.

Desafío 3 – JavaScript vs PHP

Comparación entre ambos lenguajes y sus contextos de uso.

Aspecto	JavaScript	PHP
Dónde se ejecuta	En el navegador o entorno cliente	En el servidor web
Tipado	Dinámico y flexible	Dinámico con tipado fuerte opcional
Principal objetivo	Interactividad y manipulación del DOM	Procesamiento del lado servidor y bases de datos
Entorno común	Navegador, Node.js	Apache, Nginx o XAMPP

Casos en los que conviene usar JavaScript:

Crear interactividad o animaciones en el navegador.

Desarrollar aplicaciones SPA o PWA.

Validar formularios y manipular contenido dinámico.

Casos en los que conviene usar PHP:

Gestionar usuarios, sesiones o autenticaciones.

Procesar formularios y guardar datos en una base de datos.

Construir páginas dinámicas o paneles administrativos.

Ejemplo JavaScript:

```
document.querySelector("#boton").addEventListener("click", () => {
  alert("Tu pedido fue enviado con éxito.");
});
```

Ejemplo PHP:

```
<?php
$pdo = new PDO('mysql:host=localhost;dbname=tienda','root','');
$query = $pdo->prepare("SELECT * FROM productos WHERE id=?");
$query->execute([$_GET['id']]);
$producto = $query->fetch();
echo $producto['nombre'];
?>
```

🎯 Desafío 4 – Conexión a Bases de Datos

Conceptos esenciales para conectar PHP con una base de datos y uso de consultas preparadas.

En PHP, las formas más comunes de conectarse a una base de datos son MySQLi y PDO (PHP Data Objects).

El método PDO es más recomendable porque brinda soporte para múltiples motores de bases de datos (MySQL, PostgreSQL, SQLite) y protege contra inyecciones SQL mediante consultas preparadas.

Pasos básicos de conexión con PDO:

Crear el objeto PDO con el DSN y las credenciales.

Configurar el modo de errores.

Preparar la consulta con prepare().

Ejecutar con execute() pasando los valores.

Obtener los datos con fetch() o fetchAll().

Ejemplo:

```
<?php
try {
  $pdo = new
PDO('mysql:host=localhost;dbname=foodexpress;charset=utf8mb4','root','');
  $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

  $stmt = $pdo->prepare("SELECT * FROM productos WHERE categoria_id =
?");
  $stmt->execute([$_GET['cat']]);
  $resultados = $stmt->fetchAll();

  foreach ($resultados as $r) {
    echo "<li>{$r['nombre']} - $ {$r['precio']}</li>";
  }
} catch (PDOException $e) {
  echo "Error: " . $e->getMessage();
```

```
}
?>
```

Ventajas principales:

Previene ataques SQL Injection.

Código más ordenado y reutilizable.

Compatible con distintas bases sin modificar la lógica principal.