Documentación ULTRA-DETALLADA: index.php

Examen Parcial – Nivel A

1 <?php
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

2 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

3 * Examen Parcial – Nivel A (Regular)
• Rol: comentario; sin efecto en tiempo de ejecución.

4 * Mini■sistema: Registro de pedidos con totales y filtro
• Rol: comentario; sin efecto en tiempo de ejecución.

5 * Versión 100% en español (identificadores y comentarios).
• Rol: comentario; sin efecto en tiempo de ejecución.

6 *
• Rol: comentario; sin efecto en tiempo de ejecución.

7 * Flujo general:
• Rol: comentario; sin efecto en tiempo de ejecución.

8 * ■■■
• Rol: comentario; sin efecto en tiempo de ejecución.

9 * ■ 1) Al cargar la página, se inicia sesión y se prepara el estado. ■
• Rol: comentario; sin efecto en tiempo de ejecución.

10 * ■ 2) Si llega POST, se validan entradas, se calculan totales y ■
• Rol: comentario; sin efecto en tiempo de ejecución.

11 * ■ se inserta el pedido en la sesión (máx. 10). ■
• Rol: comentario; sin efecto en tiempo de ejecución.

12 * ■ 3) Se lee el filtro GET (total mínimo) y se renderiza la tabla. ■
• Rol: comentario; sin efecto en tiempo de ejecución.

13 * ■■■
• Rol: comentario; sin efecto en tiempo de ejecución.

14 */
• Rol: fin de DocBlock.

15

16 declare(strict_types=1); // Activa tipado estricto en PHP 7+
• Rol: directiva strict_types.
• Efecto: activa tipado estricto en este archivo; prohíbe coerciones implícitas en firmas.

17 session_start(); // Crea/continúa una sesión para usar $_SESSION
• Rol: sesión HTTP.
• Efecto: habilita $_SESSION. Debe llamarse antes de enviar salida para evitar error de cabeceras.

18

19 /* ■■■

20 Funciones de saneamiento y validación

21 ■■■ */

22

23 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

24 * depurar_texto
• Rol: comentario; sin efecto en tiempo de ejecución.

25 * - Recorta espacios al inicio/fin y elimina caracteres de control no imprimibles.
• Rol: comentario; sin efecto en tiempo de ejecución.

26 * @param string $texto Cadena de entrada cruda del usuario.
• Rol: comentario; sin efecto en tiempo de ejecución.

27 * @return string Cadena saneada.
• Rol: comentario; sin efecto en tiempo de ejecución.

28 */
• Rol: fin de DocBlock.

29 function depurar_texto(string $texto): string {
• Rol: declaración de función.
• Nombre: depurar_texto
• Parámetro $texto: tipo declarado string.
• Retorno: string.

30 $texto = trim($texto);
• Función: trim(cadena). Elimina espacios/blancos en extremos.

31 // Regex: rangos de caracteres de control ASCII, excepto \n y \t.
• Rol: comentario; sin efecto en tiempo de ejecución.

32 // Reemplaza por vacío cualquier carácter no deseado.
• Rol: comentario; sin efecto en tiempo de ejecución.

33 return preg_replace('/[\x00-\x08\x0B\x0C\x0E-\x1F\x7F]/u', '', $texto) ?? '';
• Rol: retorno; finaliza la función devolviendo el valor indicado.

34 }

35

36 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

37 * a_flotante
• Rol: comentario; sin efecto en tiempo de ejecución.

38 * - Convierte una cadena a número flotante permitiendo coma decimal.
• Rol: comentario; sin efecto en tiempo de ejecución.

39 * @param string $texto_numerico Cadena que representa un número.
• Rol: comentario; sin efecto en tiempo de ejecución.

40 * @return array{ok:bool,val?:float,msg?:string}
• Rol: comentario; sin efecto en tiempo de ejecución.

41 * ok=true → conversión exitosa, val contiene el número.
• Rol: comentario; sin efecto en tiempo de ejecución.

42 * ok=false → msg explica la causa.
• Rol: comentario; sin efecto en tiempo de ejecución.

43 */
• Rol: fin de DocBlock.

44 function a_flotante(string $texto_numerico): array {
• Rol: declaración de función.
• Nombre: a_flotante
• Parámetro $texto_numerico: tipo declarado string.

• Retorno: array.

45 $s = str_replace(',', '.', trim($texto_numerico)); // admite "12,5"
• Función: str_replace(busca, reemplaza, sujeto). Reemplazo literal; no usa regex.
• Función: trim(cadena). Elimina espacios/blancos en extremos.

46 if ($s === '') return ['ok'=>false,'msg'=>'Vacío'];
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• Comparación con cadena vacía para validar presencia de datos.

47 if (!is_numeric($s)) return ['ok'=>false,'msg'=>'No numérico'];
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• Función: is_numeric(x). True si x es número o string numérico válido (incluye notación científica).

48 return ['ok'=>true,'val'=>(float)$s];
• Rol: retorno; finaliza la función devolviendo el valor indicado.

49 }

50

51 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

52 * a_entero
• Rol: comentario; sin efecto en tiempo de ejecución.

53 * - Convierte una cadena a entero con validación estricta.
• Rol: comentario; sin efecto en tiempo de ejecución.

54 * @param string $texto_numerico
• Rol: comentario; sin efecto en tiempo de ejecución.

55 * @return array{ok:bool,val?:int,msg?:string}
• Rol: comentario; sin efecto en tiempo de ejecución.

56 */
• Rol: fin de DocBlock.

57 function a_entero(string $texto_numerico): array {
• Rol: declaración de función.
• Nombre: a_entero
• Parámetro $texto_numerico: tipo declarado string.
• Retorno: array.

58 $s = trim($texto_numerico);
• Función: trim(cadena). Elimina espacios/blancos en extremos.

59 // Acepta solo dígitos opcionalmente con signo negativo. Aquí requerimos no negativo en validación de negocio.
• Rol: comentario; sin efecto en tiempo de ejecución.

60 if ($s === '' || !preg_match('/^-?\d+$/', $s)) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• Comparación con cadena vacía para validar presencia de datos.
• Función: preg_match(patrón, sujeto, [coincidencias], [flags], [offset]). Devuelve 1/0 o false.
• Patrón: /^-?\d+$/ → Patrón específico aplicado al sujeto; revisar anclas (^, $), clases (\d) y cuantificadores (+, *, ?).

61 return ['ok'=>false,'msg'=>'No entero'];
• Rol: retorno; finaliza la función devolviendo el valor indicado.

62 }

63 return ['ok'=>true,'val'=>(int)$s];
• Rol: retorno; finaliza la función devolviendo el valor indicado.

64 }

65

66 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

67 * calcular_totales
• Rol: comentario; sin efecto en tiempo de ejecución.

68 * - Aplica fórmulas de negocio del pedido.
• Rol: comentario; sin efecto en tiempo de ejecución.

69 * @param float $precio_unitario Precio unitario (>0)
• Rol: comentario; sin efecto en tiempo de ejecución.

70 * @param int $cantidad Cantidad (>0)
• Rol: comentario; sin efecto en tiempo de ejecución.

71 * @param float $descuento_porcentaje Descuento en % (0..100)
• Rol: comentario; sin efecto en tiempo de ejecución.

72 * @param bool $aplica_iva Si aplica 16% de IVA
• Rol: comentario; sin efecto en tiempo de ejecución.

73 * @return array{subtotal:float,descuento_aplicado:float,base:float,iva_monto:float,total:float}
• Rol: comentario; sin efecto en tiempo de ejecución.

74 */
• Rol: fin de DocBlock.

75 function calcular_totales(float $precio_unitario, int $cantidad, float $descuento_porcentaje, bool $aplica_iva): array {
• Rol: declaración de función.
• Nombre: calcular_totales
• Parámetro $precio_unitario: tipo declarado float.
• Parámetro $cantidad: tipo declarado int.
• Parámetro $descuento_porcentaje: tipo declarado float.
• Parámetro $aplica_iva: tipo declarado bool.
• Retorno: array.

76 $subtotal = $precio_unitario * $cantidad;

77 $descuento_aplicado = $subtotal * ($descuento_porcentaje / 100.0);

78 $base = $subtotal - $descuento_aplicado;

79 $iva_monto = $aplica_iva ? ($base * 0.16) : 0.0;

80 $total = $base + $iva_monto;

81 return compact('subtotal','descuento_aplicado','base','iva_monto','total');
• Rol: retorno; finaliza la función devolviendo el valor indicado.

82 }

83

84 /* ■■■

85 Estado en sesión

86 ■■■ */

87

88 if (!isset($_SESSION['pedidos']) || !is_array($_SESSION['pedidos'])) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• isset(x): true si existe y no es null.

89 $_SESSION['pedidos'] = []; // Arreglo de pedidos recientes (máx. 10)

90 }

91

92 /**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

93 * insertar_pedido
• Rol: comentario; sin efecto en tiempo de ejecución.

94 * - Inserta al inicio del arreglo de sesión y limita a 10 elementos.
• Rol: comentario; sin efecto en tiempo de ejecución.

95 * @param array $pedido Estructura con datos del pedido + totales.
• Rol: comentario; sin efecto en tiempo de ejecución.

96 * @return void
• Rol: comentario; sin efecto en tiempo de ejecución.

97 */
• Rol: fin de DocBlock.

98 function insertar_pedido(array $pedido): void {
• Rol: declaración de función.
• Nombre: insertar_pedido
• Parámetro $pedido: tipo declarado array.
• Retorno: void.

99 array_unshift($_SESSION['pedidos'], $pedido); // Inserta al frente
• Función: array_unshift(&arr;, valor). Inserta al inicio y reindexa índices numéricos.

100 if (count($_SESSION['pedidos']) > 10) { // Mantener 10 recientes
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

101 array_pop($_SESSION['pedidos']); // Elimina el más viejo

102 }

103 }

104

105 /* ■■ ■■■■■■■■■

106 Proceso de formulario (POST)

107 ■■ ■■■■■■■ */

108

109 $errores = []; // Mapa campo → mensaje
• Rol: inicialización de arreglo vacío [].

110 $anteriores = [// Valores previos para re■mostrar el formulario
• Rol: inicialización de arreglo asociativo con claves y valores.

111 'producto' => '',

112 'precio' => '',

113 'cantidad' => '',

114 'descuento' => '0',

115 'iva' => '' // 'on' si checkbox marcado

116];

117

118 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• Condición: $_SERVER['REQUEST_METHOD'] === 'POST' (método HTTP).

119 // 1) Extraer entradas crudas
• Rol: comentario; sin efecto en tiempo de ejecución.

120 $anteriores['producto'] = depurar_texto($_POST['producto'] ?? '');

121 $anteriores['precio'] = trim((string)($_POST['precio'] ?? ''));
• Función: trim(cadena). Elimina espacios/blancos en extremos.

122 $anteriores['cantidad'] = trim((string)($_POST['cantidad'] ?? ''));
• Función: trim(cadena). Elimina espacios/blancos en extremos.

123 $anteriores['descuento'] = trim((string)($_POST['descuento'] ?? '0'));
• Función: trim(cadena). Elimina espacios/blancos en extremos.

124 $anteriores['iva'] = isset($_POST['iva']) ? 'on' : '';

125

126 // 2) Validar cada campo
• Rol: comentario; sin efecto en tiempo de ejecución.

127 if ($anteriores['producto'] === '') {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• Comparación con cadena vacía para validar presencia de datos.

128 $errores['producto'] = 'Producto obligatorio.';

129 } elseif (mb_strlen($anteriores['producto']) > 100) {
• Rol: rama elseif; evalúa si el if anterior fue false.

130 $errores['producto'] = 'Máximo 100 caracteres.';

131 }

132

133 $v_precio = a_flotante($anteriores['precio']);

134 if (!$v_precio['ok']) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

135 $errores['precio'] = 'Precio inválido.';

136 } elseif ($v_precio['val'] <= 0) {
• Rol: rama elseif; evalúa si el if anterior fue false.

137 $errores['precio'] = 'Precio debe ser > 0.';

138 }

139

140 $v_cantidad = a_entero($anteriores['cantidad']);

141 if (!$v_cantidad['ok']) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

142 $errores['cantidad'] = 'Cantidad inválida.';

143 } elseif ($v_cantidad['val'] <= 0) {
• Rol: rama elseif; evalúa si el if anterior fue false.

144 $errores['cantidad'] = 'Cantidad debe ser > 0.';

145 }

146

147 $v_desc = a_flotante($anteriores['descuento']);

148 if (!$v_desc['ok']) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

149 $errores['descuento'] = 'Descuento inválido.';

150 } elseif ($v_desc['val'] < 0 || $v_desc['val'] > 100) {
• Rol: rama elseif; evalúa si el if anterior fue false.

151 $errores['descuento'] = 'Descuento debe estar entre 0 y 100.';

152 }

153

154 $aplica_iva = ($anteriores['iva'] === 'on');

155

156 // 3) Si no hay errores → calcular totales, guardar y aplicar PRG
• Rol: comentario; sin efecto en tiempo de ejecución.

157 if (empty($errores)) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

158 $totales = calcular_totales((float)$v_precio['val'], (int)$v_cantidad['val'], (float)$v_desc['val'], $aplica_iva);

159 $pedido = [
• Rol: inicialización de arreglo asociativo con claves y valores.

160 'fecha_hora' => date('Y-m-d H:i:s'),

161 'producto' => $anteriores['producto'],

162 'precio' => (float)$v_precio['val'],

163 'cantidad' => (int)$v_cantidad['val'],

164 'descuento' => (float)$v_desc['val'],

165 'iva' => $aplica_iva,

166 'totales' => $totales

167];

168 insertar_pedido($pedido);

169

170 // Patrón PRG (Post/Redirect/Get): evita reenvíos al refrescar
• Rol: comentario; sin efecto en tiempo de ejecución.

171 $mensaje = urlencode('Pedido agregado correctamente.');

172 header('Location: ' . strtok($_SERVER['REQUEST_URI'], '?') . '?mensaje=' . $mensaje);
• Función: header('Location: URL'). Redirección 302 por defecto. Requiere no haber enviado salida.

173 exit;

174 }

175 }

176

177 /* ■■■ ■■■■■■■■■■

178 Filtro de listado (GET)

179 ■■■ ■■■■■■■■ */

180

$minimo_total = 0.0;

if (isset($_GET['minimo_total']) && $_GET['minimo_total'] !== '') {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.
• isset(x): true si existe y no es null.
• Comparación con cadena vacía para validar presencia de datos.

$v_min = a_flotante((string)$_GET['minimo_total']);

if ($v_min['ok'] && $v_min['val'] >= 0) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

$minimo_total = (float)$v_min['val'];

}

}

/* ■■■

Preparación de datos para la vista

■■■ */

$pedidos = $_SESSION['pedidos']; // copia local para iterar

$pedidos_filtrados = [];
• Rol: inicialización de arreglo vacío [].

$suma_totales = 0.0;

foreach ($pedidos as $p) {
• Rol: bucle foreach sobre arreglo/iterable.

$total_pedido = (float)($p['totales']['total'] ?? 0.0);

if ($total_pedido >= $minimo_total) {
• Rol: estructura condicional. Ejecuta el bloque si la condición es verdadera.

$pedidos_filtrados[] = $p;

$suma_totales += $total_pedido;

}

}

$conteo_pedidos = count($pedidos_filtrados);

$promedio_tickets = $conteo_pedidos > 0 ? $suma_totales / $conteo_pedidos : 0.0;

/**
• Rol: inicio de DocBlock; documentación para personas y herramientas.

*** formatear_numero**
• Rol: comentario; sin efecto en tiempo de ejecución.

*** - Imprime un flotante con 2 decimales, punto decimal y coma de miles.**
• Rol: comentario; sin efecto en tiempo de ejecución.

***/**
• Rol: fin de DocBlock.

function formatear_numero(float $n): string { return number_format($n, 2, '.', ','); }

• Rol: declaración de función.
• Nombre: formatear_numero
• Parámetro $n: tipo declarado float.
• Retorno: string.
• Función: number_format(número, decimales, separador_decimal, separador_miles). Retorna string formateado.

212

213 ?>
• Rol: cierre del bloque PHP; desde aquí se emite HTML sin evaluar.

214 <!doctype html>
• Rol: declaración HTML5 (modo estándar).

215 <html lang="es">
• Atributos: lang="es": idioma del documento.

216 <head>

217 <meta charset="utf-8" />
• Atributos: charset="utf-8": codificación de caracteres.

218 <meta name="viewport" content="width=device-width,initial-scale=1" />
• Atributos: name="viewport": clave enviada en la petición (GET).

219 <title>Nivel A – Registro de pedidos (versión español)</title>

220 <style>

221 /* Estilos mínimos embebidos para no depender de archivo externo */

222 *{box-sizing:border-box}
• Rol: comentario; sin efecto en tiempo de ejecución.

223 body{font-family:system-ui,-apple-system,Segoe UI,Roboto,Ubuntu,Arial,sans-serif;margin:24px;line-height:1.35}
• CSS: font-family: pila de fuentes del sistema para legibilidad y rendimiento. margin: 24px; margen externo del elemento.

224 h1,h2{margin:0 0 10px 0}
• CSS: margin: 0 0 10px 0; margen externo del elemento.

225 fieldset{border:1px solid #ccc;padding:12px;border-radius:8px;margin-bottom:16px}
• CSS: border: 1px solid #ccc; estilo y color del borde. padding: 12px; relleno interno del elemento. border-radius: 8px; radios de las esquinas.

226 label{display:block;margin:6px 0 4px;font-weight:600}
• CSS: display:block: ocupa toda la línea disponible. margin: 6px 0 4px; margen externo del elemento.

227 input[type=text],input[type=number]{width:100%;padding:8px;border:1px solid #bbb;border-radius:6px}
• CSS: width: 100%; ancho del control. padding: 8px; relleno interno del elemento. border: 1px solid #bbb; estilo y color del borde. border-radius: 6px; radios de las esquinas.

228 button{padding:8px 14px;border:0;border-radius:8px;cursor:pointer}
• CSS: padding: 8px 14px; relleno interno del elemento. border: 0; estilo y color del borde. border-radius: 8px; radios de las esquinas. cursor:pointer: apariencia de mano al pasar el mouse.

229 button.primario{background:#0a7;color:white}
• CSS: background: #0a7; color de fondo. color: white; color del texto.

230 button.secundario{background:#07a;color:white}
• CSS: background: #07a; color de fondo. color: white; color del texto.

231 table{border-collapse:collapse;width:100%;margin-top:12px}
• CSS: border-collapse:collapse; fusiona bordes adyacentes de la tabla. width: 100%; ancho del control.

232 th,td{border:1px solid #ddd;padding:8px;text-align:left}

• CSS: border: 1px solid #ddd; estilo y color del borde. padding: 8px; relleno interno del elemento. text-align: left; alineación horizontal del contenido.

233 th{background:#f4f4f4}
• CSS: background: #f4f4f4; color de fondo.

234 tr:nth-child(even){background:#fafafa}
• CSS: background: #fafafa; color de fondo.

235 .nota{font-size:0.92rem;color:#333;background:#f9f9ff;border:1px dashed #99f;padding:8px;border-radius:6px;margin:8px 0}
• CSS: color: #333; color del texto. background: #f9f9ff; color de fondo. border: 1px dashed #99f; estilo y color del borde. padding: 8px; relleno interno del elemento. border-radius: 6px; radios de las esquinas. margin: 8px 0; margen externo del elemento.

236 .error{color:#b00020;font-weight:700}
• CSS: color: #b00020; color del texto.

237 .exito{color:#0a7;font-weight:700}
• CSS: color: #0a7; color del texto.

238 small.mono{font-family:ui-monospace, SFMono-Regular, Menlo, Consolas, "Liberation Mono", monospace;color:#555}
• CSS: font-family: pila de fuentes del sistema para legibilidad y rendimiento. color: #555; color del texto.

239 .resumen{margin-top:8px;padding:8px;border-left:4px solid #07a;background:#eef7ff}
• CSS: padding: 8px; relleno interno del elemento. background: #eef7ff; color de fondo.

240 hr{border:0;border-top:1px solid #e5e5e5;margin:16px 0}
• CSS: border: 0; estilo y color del borde. margin: 16px 0; margen externo del elemento.

241 </style>

242 </head>

243 <body>

244 <h1>Nivel A – Registro de pedidos con totales y filtro (100% español)</h1>

**245 **

246 Este mini■sistema usa PHP puro y sesiones como almacén temporal.

247 La tabla refleja como máximo los 10 pedidos más recientes.

**248 **

249

250 <!-- Formulario de captura -->

251 <form method="post" novalidate>
• Atributos: novalidate: desactiva validación HTML5; validación queda del lado servidor.

252 <fieldset>

253 <legend>Captura de pedido</legend>

254

255 <label for="producto">Producto</label>
• Atributos: for="producto": asocia la etiqueta con el control id="producto".

256 <input type="text" id="producto" name="producto" maxlength="100"
• Atributos: type="text": tipo de control. id="producto": identificador único en el DOM. name="producto": clave enviada en la petición (GET). maxlength="100": longitud máxima en caracteres.

257 value="<?php echo htmlspecialchars($anteriores['producto']); ?>" />
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

258 <?php if(isset($errores['producto'])): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**259 <?php echo $errores['producto']; ?> **

260 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

261

**262 **

**263 **

264 <label for="precio">Precio unitario</label>
• Atributos: for="precio": asocia la etiqueta con el control id="precio".

265 <input type="number" id="precio" name="precio" step="0.01" min="0.01"
• Atributos: type="number": tipo de control. id="precio": identificador único en el DOM. name="precio": clave enviada en la petición (GET). step="0.01": incremento permitido (decimales de 0.01). min="0.01": valor mínimo permitido por el navegador.

266 value="<?php echo htmlspecialchars($anteriores['precio']); ?>" />
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

267 <?php if(isset($errores['precio'])): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**268 <?php echo $errores['precio']; ?> **

269 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**270 **

**271 **

272 <label for="cantidad">Cantidad</label>
• Atributos: for="cantidad": asocia la etiqueta con el control id="cantidad".

273 <input type="number" id="cantidad" name="cantidad" step="1" min="1"
• Atributos: type="number": tipo de control. id="cantidad": identificador único en el DOM. name="cantidad": clave enviada en la petición (GET). step="1": incremento permitido (enteros). min="1": valor mínimo permitido por el navegador.

274 value="<?php echo htmlspecialchars($anteriores['cantidad']); ?>" />
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

275 <?php if(isset($errores['cantidad'])): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**276 <?php echo $errores['cantidad']; ?> **

277 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**278 **

**279 **

280 <label for="descuento">Descuento (%)</label>
• Atributos: for="descuento": asocia la etiqueta con el control id="descuento".

281 <input type="number" id="descuento" name="descuento" step="0.01" min="0" max="100"
• Atributos: type="number": tipo de control. id="descuento": identificador único en el DOM. name="descuento": clave enviada en la petición (GET). step="0.01": incremento permitido (decimales de 0.01). min="0": valor mínimo permitido por el navegador. max="100": valor máximo permitido.

282 value="<?php echo htmlspecialchars($anteriores['descuento']); ?>" />

• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

283 <?php if(isset($errores['descuento'])): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**284 <?php echo $errores['descuento']; ?> **

285 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**286 **

**287 **

288

289 <label style="margin-top:8px">

290 <input type="checkbox" name="iva" <?php echo $anteriores['iva']==='on' ? 'checked' : ''; ?> />
• Atributos: type="checkbox": tipo de control. name="iva": clave enviada en la petición (GET).

291 Aplicar IVA 16%

292 </label>

293

**294 **

295 <button class="primario" type="submit">Agregar pedido</button>
• Atributos: type="submit": tipo de control.

296 <small class="mono">PRG activo: tras agregar, redirige para evitar reenvíos.</small>

**297 **

298 <?php if(isset($_GET['mensaje'])): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

**299 <?php echo htmlspecialchars($_GET['mensaje']); ?> **
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

300 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

301 </fieldset>

302 </form>

303

304 <!-- Filtro por total mínimo -->

305 <form method="get">

306 <fieldset>

307 <legend>Filtro por total mínimo</legend>

308 <label for="minimo_total">Mostrar pedidos con total ≥</label>
• Atributos: for="minimo_total": asocia la etiqueta con el control id="minimo_total".

309 <input type="number" id="minimo_total" name="minimo_total" step="0.01" min="0"
• Atributos: type="number": tipo de control. id="minimo_total": identificador único en el DOM. name="minimo_total": clave enviada en la petición (GET). step="0.01": incremento permitido (decimales de 0.01). min="0": valor mínimo permitido por el navegador.

310 value="<?php echo htmlspecialchars((string)$minimo_total); ?>" />
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

**311 **

312 <button class="secundario" type="submit">Aplicar filtro</button>
• Atributos: type="submit": tipo de control.

313 <a href="<?php echo strtok($_SERVER['REQUEST_URI'],'?'); ?>">
• Atributos: href="<?php echo strtok($_SERVER[": destino del enlace; aquí se usa PHP_SELF escapado.

314 <button type="button">Limpiar filtro</button>
• Atributos: type="button": tipo de control.

**315 **

**316 **

317 </fieldset>

318 </form>

319

320 <!-- Tabla de pedidos -->

321 <h2>Pedidos recientes</h2>

322 <table>

323 <thead>

324 <tr>

325 <th>Fecha</th>

326 <th>Producto</th>

327 <th>Precio</th>

328 <th>Cantidad</th>

329 <th>Desc. %</th>

330 <th>IVA</th>

331 <th>Subtotal</th>

332 <th>Base</th>

333 <th>IVA $</th>

334 <th>Total</th>

335 </tr>

336 </thead>

337 <tbody>

338 <?php if (empty($pedidos_filtrados)): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

339 <tr><td colspan="10">Sin pedidos para mostrar.</td></tr>

340 <?php else: ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

341 <?php foreach ($pedidos_filtrados as $p): ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

342 <tr>

343 <td><?php echo htmlspecialchars($p['fecha_hora']); ?></td>
• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

344 <td><?php echo htmlspecialchars($p['producto']); ?></td>

• Función: htmlspecialchars(texto[, ENT_QUOTES, 'UTF-8']). Escapa &, <, > y comillas.
• Seguridad: usar ENT_QUOTES y 'UTF-8' evita XSS por comillas simples y dobles.

345 <td>$<?php echo formatear_numero((float)$p['precio']); ?></td>

346 <td><?php echo (int)$p['cantidad']; ?></td>

347 <td><?php echo formatear_numero((float)$p['descuento']); ?>%</td>

348 <td><?php echo $p['iva'] ? 'Sí' : 'No'; ?></td>

349 <td>$<?php echo formatear_numero((float)$p['totales']['subtotal']); ?></td>

350 <td>$<?php echo formatear_numero((float)$p['totales']['base']); ?></td>

351 <td>$<?php echo formatear_numero((float)$p['totales']['iva_monto']); ?></td>

352 <td>$<?php echo formatear_numero((float)$p['totales']['total']); ?></td>

353 </tr>

354 <?php endforeach; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

355 <?php endif; ?>
• Rol: apertura del intérprete PHP; el servidor evalúa el bloque.

356 </tbody>

357 </table>

358

**359 **

360 Resumen del listado:

**361 Pedidos mostrados: <?php echo $conteo_pedidos; ?> **

**362 Suma de totales: $<?php echo formatear_numero($suma_totales); ?> **

**363 Ticket promedio: $<?php echo formatear_numero($promedio_tickets); ?> **

**364 **

365

366 <hr />

**367 **

368 Pruebas rápidas sugeridas:

**369 **

**370 P1: precio 120.50, cantidad 3, desc 10, IVA ✓ → verifica total.
**

**371 P2: precio 99.99, cantidad 1, desc 0, IVA ✗ → total=99.99.
**

**372 Filtro total ≥ 300 → debería ocultar P2 si queda por debajo.
**

**373 **

**374 **

375 </body>

376 </html>