Instituto Gubernamental Técnico

“Juan Alberto Melgar Castro”

Diseño Web

Manual Técnico de Pagina Hamburguesas

Lic. Redin Méndez

Esdras Mauricio Cardona

12vo BTP en Informática

Cañaveral, Cortés

3 de Abril,2025

MANUAL TÉCNICO - BURGER HOUSE

1. REQUISITOS DEL SISTEMA

------------------------

Para el correcto funcionamiento del sitio web de Burger House, es necesario contar con un entorno de desarrollo que cumpla con los siguientes requisitos técnicos. Estos elementos son fundamentales para garantizar tanto el desarrollo como la ejecución adecuada del proyecto.

```bash

# Verificación de requisitos del sistema

php -v # Debe mostrar PHP 7.4 o superior

apache2 -v # Debe mostrar Apache 2.4 o superior

mysql --version # Debe mostrar MySQL 5.7 o superior

# Verificación de extensiones PHP necesarias

php -m | grep -E 'pdo|mysqli|gd|json|mbstring'

# Verificación de permisos de directorios

ls -la /xampp/htdocs/hamburguesa

```

Explicación de los comandos:

- `php -v`: Muestra la versión de PHP instalada

- `apache2 -v`: Verifica la versión del servidor Apache

- `mysql --version`: Comprueba la versión de MySQL

- `php -m`: Lista todas las extensiones PHP instaladas

- `ls -la`: Muestra los permisos de los directorios

2. PROCESO DE DESARROLLO

-----------------------

El desarrollo del sitio web de Burger House se llevó a cabo siguiendo un proceso estructurado y metodológico, dividido en varias fases principales. Cada fase fue implementada de manera secuencial, permitiendo un desarrollo ordenado y mantenible.

2.1. Configuración Inicial

La fase inicial del proyecto se centró en establecer el entorno de desarrollo necesario y preparar la estructura base del proyecto. Esta etapa fue crucial para asegurar un desarrollo fluido y organizado.

```bash

# Estructura de directorios recomendada

mkdir -p hamburguesa/{images,pages,js,css}

cd hamburguesa

touch index.html style.css script.js

# Script de inicialización del proyecto

#!/bin/bash

PROJECT\_NAME="hamburguesa"

# Crear estructura de directorios

mkdir -p $PROJECT\_NAME/{images,pages,js,css}

# Crear archivos principales

touch $PROJECT\_NAME/index.html

touch $PROJECT\_NAME/style.css

touch $PROJECT\_NAME/script.js

# Crear archivos de páginas

for page in menu nosotros contacto; do

touch $PROJECT\_NAME/pages/$page.html

done

# Crear archivos JavaScript

touch $PROJECT\_NAME/js/{carrito,validacion,main}.js

# Crear archivos CSS

touch $PROJECT\_NAME/css/{style,responsive,components}.css

# Establecer permisos

chmod -R 755 $PROJECT\_NAME

```

Explicación del script:

- Crea una estructura de directorios completa

- Inicializa archivos principales

- Crea páginas secundarias

- Establece permisos adecuados

1. Instalación de XAMPP 3.0

```bash

# Comandos para verificar la instalación

xampp start

xampp status

# Verificar servicios

netstat -an | findstr "80"

netstat -an | findstr "3306"

# Verificar logs

tail -f /xampp/apache/logs/error.log

```

2. Creación de la estructura de carpetas

```bash

# Script para crear la estructura completa

#!/bin/bash

mkdir -p hamburguesa/{images,pages,js,css}

touch hamburguesa/index.html

touch hamburguesa/style.css

touch hamburguesa/script.js

touch hamburguesa/pages/{menu,nosotros,contacto}.html

# Crear archivos de configuración

cat > hamburguesa/.htaccess << 'EOL'

Options -Indexes

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)$ index.php?url=$1 [QSA,L]

EOL

# Crear archivo de configuración de base de datos

cat > hamburguesa/config/database.php << 'EOL'

<?php

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', '');

define('DB\_NAME', 'burger\_house');

EOL

```

3. Configuración del entorno

```json

// .vscode/settings.json

{

"editor.formatOnSave": true,

"editor.defaultFormatter": "esbenp.prettier-vscode",

"files.associations": {

"\*.php": "php",

"\*.html": "html",

"\*.css": "css",

"\*.js": "javascript"

},

"editor.codeActionsOnSave": {

"source.fixAll.eslint": true

},

"javascript.updateImportsOnFileMove.enabled": "always",

"typescript.updateImportsOnFileMove.enabled": "always",

"files.autoSave": "afterDelay",

"files.autoSaveDelay": 1000

}

```

```json

// .prettierrc

{

"semi": true,

"singleQuote": true,

"tabWidth": 4,

"printWidth": 100,

"trailingComma": "es5",

"bracketSpacing": true,

"arrowParens": "avoid"

}

```

2.2. Desarrollo Frontend

La fase de desarrollo frontend se centró en crear una interfaz de usuario atractiva y funcional. A continuación, se detallan los componentes principales y sus implementaciones.

1. Estructura HTML Base

Todos los archivos HTML siguen una estructura base común para mantener la consistencia:

```html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Burger House - [Nombre de Página]</title>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css" />

<link rel="stylesheet" href="../style.css">

<link rel="stylesheet" href="pages.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

</head>

<body>

<!-- Contenido específico de cada página -->

</body>

</html>

```

2. Componentes Reutilizables

Se desarrollaron componentes comunes para mantener la consistencia en todo el sitio:

a) Header Navigation:

```html

<header class="header">

<div class="menu container">

<a href="../index.html">

<img class="logo-1" src="../images/logo.svg" alt="Logo">

</a>

<input type="checkbox" id="menu"/>

<label for="menu">

<img src="../images/menu.png" class="menu-icono" alt="">

</label>

<nav class="navbar">

<!-- Contenido del menú -->

</nav>

</div>

</header>

```

b) Footer:

```html

<footer class="footer container">

<img class="logo-2" src="../images/logo.svg" alt="">

<div class="links">

<!-- Enlaces del footer -->

</div>

</footer>

```

3. Estilos CSS

Los estilos se organizan siguiendo una estructura modular:

```css

/\* Variables globales \*/

:root {

--primary-color: #ff0000;

--secondary-color: #333333;

--background-color: #ffffff;

--text-color: #222222;

--font-primary: 'Arial', sans-serif;

}

/\* Reset y estilos base \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Layout y componentes principales \*/

.container {

max-width: 1200px;

margin: 0 auto;

padding: 0 15px;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

/\* Estilos para tablets \*/

}

@media (max-width: 480px) {

/\* Estilos para móviles \*/

}

```

2.3. Sistema de Carrito

El carrito de compras se implementó utilizando JavaScript moderno y LocalStorage para persistencia de datos.

1. Estructura de Datos

```javascript

// Definición de la clase Producto

class Producto {

constructor(id, nombre, precio, imagen) {

this.id = id;

this.nombre = nombre;

this.precio = precio;

this.imagen = imagen;

this.cantidad = 1;

}

}

// Gestión del carrito

class Carrito {

constructor() {

this.items = [];

this.total = 0;

this.cargarLocalStorage();

}

agregarProducto(producto) {

// Implementación

}

eliminarProducto(id) {

// Implementación

}

actualizarCantidad(id, cantidad) {

// Implementación

}

calcularTotal() {

// Implementación

}

}

```

2. Funciones de Interfaz

```javascript

// Actualización de la UI del carrito

function actualizarCarritoUI() {

const carritoContainer = document.getElementById('carrito-items');

const totalElement = document.getElementById('carrito-total');

carritoContainer.innerHTML = '';

carrito.items.forEach(item => {

// Renderizado de items

});

totalElement.textContent = `Total: ${carrito.total.toFixed(2)} Lps`;

}

```

2.4. Sistema de Formularios

Los formularios implementan validación tanto en el frontend como en el backend.

1. Estructura Base de Formularios

```html

<form id="contactForm" action="https://formsubmit.co/mauriciocardona402@gmail.com" method="POST">

<!-- Campos ocultos de configuración -->

<input type="hidden" name="\_next" value="http://localhost/hamburguesa/pages/nosotros.html">

<input type="hidden" name="\_captcha" value="false">

<input type="hidden" name="\_template" value="table">

<!-- Campos del formulario -->

<div class="form-group">

<label for="nombre">Nombre Completo</label>

<input type="text" id="nombre" name="nombre" required>

</div>

<!-- Más campos según necesidad -->

</form>

```

2. Validación de Formularios

```javascript

document.getElementById('contactForm').addEventListener('submit', function(e) {

e.preventDefault();

if (validarFormulario()) {

this.submit();

}

});

function validarFormulario() {

// Implementación de validaciones

return true;

}

```

2.5. Optimización

La optimización del sitio se realizó en varias áreas clave para garantizar el mejor rendimiento posible.

1. Optimización de Imágenes

```javascript

// Ejemplo de implementación de lazy loading

document.addEventListener("DOMContentLoaded", function() {

const images = document.querySelectorAll("[data-src]");

const imageObserver = new IntersectionObserver((entries, observer) => {

entries.forEach(entry => {

if (entry.isIntersecting) {

const img = entry.target;

img.src = img.dataset.src;

img.removeAttribute("data-src");

observer.unobserve(img);

}

});

});

images.forEach(img => imageObserver.observe(img));

});

```

Ejemplo de uso en HTML:

```html

<img data-src="../images/productos/hamburguesa1.jpg"

alt="Hamburguesa Clásica"

class="lazy-image"

src="placeholder.jpg">

```

2. Minificación de CSS

Ejemplo de CSS minificado:

```css

/\* Antes de minificar \*/

.header {

background-color: #ffffff;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

position: fixed;

width: 100%;

z-index: 1000;

}

/\* Después de minificar \*/

.header{background-color:#fff;box-shadow:0 2px 4px rgba(0,0,0,.1);position:fixed;width:100%;z-index:1000}

```

3. ESTRUCTURA DEL PROYECTO

-------------------------

```bash

# Estructura completa del proyecto

hamburguesa/

├── images/

│ ├── logo.svg

│ ├── menu.png

│ └── productos/

├── pages/

│ ├── bebidas.html

│ ├── burritos.html

│ ├── complementos.html

│ ├── forms.html

│ ├── hamburguesas.html

│ ├── menu.html

│ └── nosotros.html

├── js/

│ ├── carrito.js

│ ├── validacion.js

│ └── main.js

├── css/

│ ├── style.css

│ ├── responsive.css

│ └── components.css

├── config/

│ ├── database.php

│ └── config.php

├── includes/

│ ├── header.php

│ ├── footer.php

│ └── functions.php

├── .htaccess

├── index.html

├── style.css

└── script.js

```

Explicación de la estructura:

- `images/`: Almacena recursos visuales optimizados

- `pages/`: Contiene páginas secundarias del sitio

- `js/`: Archivos JavaScript organizados por funcionalidad

- `css/`: Estilos separados por componentes y responsividad

- `config/`: Archivos de configuración del sistema

- `includes/`: Componentes reutilizables en PHP

- `.htaccess`: Configuración de Apache

4. TECNOLOGÍAS UTILIZADAS

-------------------------

```html

<!-- Ejemplo de implementación de tecnologías -->

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Burger House</title>

<!-- CSS -->

<link rel="stylesheet" href="css/style.css">

<link rel="stylesheet" href="css/responsive.css">

<!-- JavaScript -->

<script src="js/main.js" defer></script>

<script src="js/carrito.js" defer></script>

<!-- Font Awesome -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

<!-- Swiper -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css">

</head>

```

```css

/\* style.css - Ejemplo de implementación CSS \*/

:root {

/\* Variables globales \*/

--color-primario: #ff0000;

--color-secundario: #333;

--color-texto: #222;

--espaciado-base: 1rem;

--fuente-principal: 'Arial', sans-serif;

}

/\* Reset y estilos base \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: var(--fuente-principal);

line-height: 1.6;

color: var(--color-texto);

}

/\* Componentes \*/

.btn {

display: inline-block;

padding: 0.5rem 1rem;

border: none;

border-radius: 4px;

cursor: pointer;

transition: all 0.3s ease;

}

.btn--primario {

background-color: var(--color-primario);

color: white;

}

.btn--primario:hover {

background-color: darken(var(--color-primario), 10%);

}

```

5. CONFIGURACIÓN DEL SERVIDOR

----------------------------

```apache

# httpd.conf (XAMPP)

<VirtualHost \*:80>

ServerName localhost

DocumentRoot "C:/xampp/htdocs"

<Directory "C:/xampp/htdocs">

Options Indexes FollowSymLinks MultiViews

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

# Configuración de seguridad

<Directory "C:/xampp/htdocs/hamburguesa">

Options -Indexes

AllowOverride All

Require all granted

# Protección de archivos sensibles

<FilesMatch "^\.">

Order allow,deny

Deny from all

</FilesMatch>

# Prevenir acceso a archivos de configuración

<FilesMatch "^(config\.php|database\.php)$">

Order allow,deny

Deny from all

</FilesMatch>

</Directory>

```

```php

// db\_config.php

<?php

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', '');

define('DB\_NAME', 'burger\_house');

class Database {

private static $instance = null;

private $conn;

private function \_\_construct() {

try {

$this->conn = new PDO(

"mysql:host=" . DB\_HOST . ";dbname=" . DB\_NAME,

DB\_USER,

DB\_PASS,

array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,

PDO::MYSQL\_ATTR\_INIT\_COMMAND => "SET NAMES utf8"

)

);

} catch(PDOException $e) {

echo "Error de conexión: " . $e->getMessage();

die();

}

}

public static function getInstance() {

if (self::$instance == null) {

self::$instance = new Database();

}

return self::$instance;

}

public function getConnection() {

return $this->conn;

}

}

?>

```

6. FUNCIONALIDADES PRINCIPALES

----------------------------

```javascript

// carrito.js

class Carrito {

constructor() {

this.items = [];

this.total = 0;

this.cargarLocalStorage();

this.observadores = [];

}

agregarProducto(producto) {

const existe = this.items.find(item => item.id === producto.id);

if (existe) {

existe.cantidad++;

} else {

this.items.push(producto);

}

this.guardarLocalStorage();

this.calcularTotal();

this.notificarObservadores();

}

eliminarProducto(id) {

this.items = this.items.filter(item => item.id !== id);

this.guardarLocalStorage();

this.calcularTotal();

this.notificarObservadores();

}

calcularTotal() {

this.total = this.items.reduce(

(total, item) => total + (item.precio \* item.cantidad),

0

);

}

guardarLocalStorage() {

localStorage.setItem('carrito', JSON.stringify(this.items));

}

cargarLocalStorage() {

const carrito = localStorage.getItem('carrito');

if (carrito) {

this.items = JSON.parse(carrito);

this.calcularTotal();

}

}

// Patrón Observer

agregarObservador(observador) {

this.observadores.push(observador);

}

notificarObservadores() {

this.observadores.forEach(observador => {

observador.actualizar(this);

});

}

}

// Implementación del observador

class CarritoUI {

constructor(carrito) {

this.carrito = carrito;

this.carrito.agregarObservador(this);

this.actualizarUI();

}

actualizar(carrito) {

this.actualizarUI();

}

actualizarUI() {

const carritoContainer = document.getElementById('carrito-items');

const totalElement = document.getElementById('carrito-total');

carritoContainer.innerHTML = '';

this.carrito.items.forEach(item => {

const itemElement = document.createElement('div');

itemElement.className = 'carrito-item';

itemElement.innerHTML = `

<img src="${item.imagen}" alt="${item.nombre}">

<div class="item-details">

<h3>${item.nombre}</h3>

<p>Precio: ${item.precio} Lps</p>

<p>Cantidad: ${item.cantidad}</p>

</div>

<button onclick="carrito.eliminarProducto(${item.id})">Eliminar</button>

`;

carritoContainer.appendChild(itemElement);

});

totalElement.textContent = `Total: ${this.carrito.total.toFixed(2)} Lps`;

}

}

```

7. MANTENIMIENTO

---------------

```javascript

// optimizacion.js

class Optimizador {

static async optimizarImagen(file) {

const options = {

maxSizeMB: 1,

maxWidthOrHeight: 1920,

useWebWorker: true

};

try {

const compressedFile = await imageCompression(file, options);

return compressedFile;

} catch (error) {

console.error('Error al comprimir imagen:', error);

throw error;

}

}

static validarImagen(file) {

const tiposPermitidos = ['image/jpeg', 'image/png', 'image/webp'];

const maxSize = 2 \* 1024 \* 1024; // 2MB

if (!tiposPermitidos.includes(file.type)) {

throw new Error('Tipo de archivo no permitido');

}

if (file.size > maxSize) {

throw new Error('Archivo demasiado grande');

}

return true;

}

static async procesarImagen(file) {

try {

// Validar imagen

this.validarImagen(file);

// Optimizar imagen

const imagenOptimizada = await this.optimizarImagen(file);

// Generar diferentes tamaños

const sizes = [800, 1200, 1600];

const imagenes = await Promise.all(

sizes.map(size => this.redimensionarImagen(imagenOptimizada, size))

);

return {

original: imagenOptimizada,

variantes: imagenes

};

} catch (error) {

console.error('Error en procesamiento de imagen:', error);

throw error;

}

}

static async redimensionarImagen(file, size) {

return new Promise((resolve, reject) => {

const img = new Image();

img.src = URL.createObjectURL(file);

img.onload = () => {

const canvas = document.createElement('canvas');

const ctx = canvas.getContext('2d');

canvas.width = size;

canvas.height = (img.height \* size) / img.width;

ctx.drawImage(img, 0, 0, canvas.width, canvas.height);

canvas.toBlob(blob => {

resolve(new File([blob], file.name, {

type: 'image/jpeg',

lastModified: Date.now()

}));

}, 'image/jpeg', 0.8);

};

img.onerror = reject;

});

}

}

```

8. FORMULARIOS

-------------

```javascript

// validacion.js

class FormValidator {

constructor(formId) {

this.form = document.getElementById(formId);

this.errors = [];

this.initializeValidation();

}

initializeValidation() {

this.form.addEventListener('submit', (e) => {

e.preventDefault();

this.errors = [];

if (this.validateForm()) {

this.form.submit();

} else {

this.showErrors();

}

});

// Validación en tiempo real

this.form.querySelectorAll('input, textarea').forEach(field => {

field.addEventListener('blur', () => {

this.validateField(field);

});

});

}

validateForm() {

const nombre = this.form.querySelector('#nombre').value;

const email = this.form.querySelector('#email').value;

const mensaje = this.form.querySelector('#mensaje').value;

return this.validateRequired(nombre, 'nombre') &&

this.validateEmail(email) &&

this.validateRequired(mensaje, 'mensaje');

}

validateEmail(email) {

const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!regex.test(email)) {

this.errors.push('El correo electrónico no es válido');

return false;

}

return true;

}

validateRequired(value, field) {

if (!value.trim()) {

this.errors.push(`El campo ${field} es requerido`);

return false;

}

return true;

}

validateField(field) {

const value = field.value.trim();

const fieldName = field.getAttribute('name');

// Limpiar errores previos

this.clearFieldErrors(field);

// Validar según el tipo de campo

switch(field.type) {

case 'email':

if (!this.validateEmail(value)) {

this.showFieldError(field, 'Correo electrónico inválido');

}

break;

case 'tel':

if (!this.validatePhone(value)) {

this.showFieldError(field, 'Número de teléfono inválido');

}

break;

default:

if (field.required && !value) {

this.showFieldError(field, 'Este campo es requerido');

}

}

}

showFieldError(field, message) {

const errorDiv = document.createElement('div');

errorDiv.className = 'error-message';

errorDiv.textContent = message;

field.parentNode.appendChild(errorDiv);

field.classList.add('error');

}

clearFieldErrors(field) {

const errorDiv = field.parentNode.querySelector('.error-message');

if (errorDiv) {

errorDiv.remove();

}

field.classList.remove('error');

}

showErrors() {

const errorDiv = document.createElement('div');

errorDiv.className = 'error-messages';

errorDiv.innerHTML = this.errors.map(error => `<p>${error}</p>`).join('');

this.form.insertBefore(errorDiv, this.form.firstChild);

}

}

```

9. RESOLUCIÓN DE PROBLEMAS

-------------------------

```javascript

// debug.js

class Debugger {

static logError(error, context) {

console.error(`Error en ${context}:`, error);

this.sendToMonitoring(error, context);

}

static sendToMonitoring(error, context) {

const errorData = {

message: error.message,

stack: error.stack,

context: context,

timestamp: new Date().toISOString(),

userAgent: navigator.userAgent,

url: window.location.href

};

// Enviar a servicio de monitoreo

fetch('/api/log-error', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(errorData)

}).catch(err => {

console.error('Error al enviar log:', err);

});

}

static checkSystemRequirements() {

const requirements = {

localStorage: typeof localStorage !== 'undefined',

sessionStorage: typeof sessionStorage !== 'undefined',

fetch: typeof fetch !== 'undefined',

webp: this.checkWebPSupport(),

cookies: navigator.cookieEnabled,

javascript: true

};

return requirements;

}

static checkWebPSupport() {

const elem = document.createElement('canvas');

if (elem.getContext && elem.getContext('2d')) {

return elem.toDataURL('image/webp').indexOf('data:image/webp') === 0;

}

return false;

}

static async testConnection() {

try {

const response = await fetch('/api/health-check');

return response.ok;

} catch (error) {

console.error('Error en conexión:', error);

return false;

}

}

static async checkDatabaseConnection() {

try {

const response = await fetch('/api/db-status');

const data = await response.json();

return data.status === 'ok';

} catch (error) {

console.error('Error en conexión a base de datos:', error);

return false;

}

}

}

```

10. RECOMENDACIONES

-----------------

```javascript

// recomendaciones.js

class OptimizadorSitio {

static async realizarBackup() {

const backup = {

timestamp: new Date().toISOString(),

configuracion: this.obtenerConfiguracion(),

datos: await this.obtenerDatos()

};

return this.guardarBackup(backup);

}

static obtenerConfiguracion() {

return {

version: '1.0.0',

ultimaActualizacion: new Date().toISOString(),

configuraciones: {

// Configuraciones del sitio

}

};

}

static async obtenerDatos() {

// Implementación para obtener datos del sitio

return {};

}

static async guardarBackup(backup) {

// Implementación para guardar backup

console.log('Backup realizado:', backup);

}

static async optimizarRecursos() {

// Optimizar imágenes

const imagenes = document.querySelectorAll('img[data-src]');

for (const img of imagenes) {

await this.optimizarImagen(img);

}

// Minificar CSS

const estilos = document.querySelectorAll('link[rel="stylesheet"]');

for (const estilo of estilos) {

await this.minificarCSS(estilo.href);

}

// Minificar JavaScript

const scripts = document.querySelectorAll('script[src]');

for (const script of scripts) {

await this.minificarJS(script.src);

}

}

static async optimizarImagen(img) {

// Implementación de optimización de imágenes

}

static async minificarCSS(url) {

// Implementación de minificación CSS

}

static async minificarJS(url) {

// Implementación de minificación JavaScript

}

}

```

11. CONTACTO SOPORTE

------------------

```php

// soporte.php

<?php

class Soporte {

private $email;

private $prioridad;

private $mensaje;

private $db;

public function \_\_construct($email, $mensaje, $prioridad = 'normal') {

$this->email = $email;

$this->mensaje = $mensaje;

$this->prioridad = $prioridad;

$this->db = Database::getInstance()->getConnection();

}

public function enviarSolicitud() {

$asunto = $this->prioridad === 'urgente' ?

'[URGENTE] Solicitud de Soporte' :

'Solicitud de Soporte';

$headers = "From: " . $this->email . "\r\n";

$headers .= "Reply-To: " . $this->email . "\r\n";

$headers .= "X-Mailer: PHP/" . phpversion();

// Enviar email

$emailEnviado = mail(

'mauriciocardona402@gmail.com',

$asunto,

$this->mensaje,

$headers

);

// Registrar en base de datos

if ($emailEnviado) {

$this->registrarSolicitud();

}

return $emailEnviado;

}

public function registrarSolicitud() {

try {

$stmt = $this->db->prepare(

"INSERT INTO solicitudes\_soporte (email, mensaje, prioridad, fecha)

VALUES (:email, :mensaje, :prioridad, NOW())"

);

$stmt->execute([

':email' => $this->email,

':mensaje' => $this->mensaje,

':prioridad' => $this->prioridad

]);

return true;

} catch (PDOException $e) {

error\_log("Error al registrar solicitud: " . $e->getMessage());

return false;

}

}

public function obtenerSolicitudes() {

try {

$stmt = $this->db->query(

"SELECT \* FROM solicitudes\_soporte

ORDER BY fecha DESC

LIMIT 10"

);

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

error\_log("Error al obtener solicitudes: " . $e->getMessage());

return [];

}

}

}

?>

```

12. SISTEMA DE AUTENTICACIÓN

--------------------------

```php

// auth.php

<?php

class Auth {

private $db;

private static $instance = null;

private function \_\_construct() {

$this->db = Database::getInstance()->getConnection();

}

public static function getInstance() {

if (self::$instance === null) {

self::$instance = new self();

}

return self::$instance;

}

public function login($email, $password) {

try {

$stmt = $this->db->prepare(

"SELECT \* FROM usuarios WHERE email = :email LIMIT 1"

);

$stmt->execute([':email' => $email]);

$usuario = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($usuario && password\_verify($password, $usuario['password'])) {

$\_SESSION['usuario\_id'] = $usuario['id'];

$\_SESSION['usuario\_nombre'] = $usuario['nombre'];

$\_SESSION['usuario\_rol'] = $usuario['rol'];

return true;

}

return false;

} catch (PDOException $e) {

error\_log("Error en login: " . $e->getMessage());

return false;

}

}

public function registrar($nombre, $email, $password) {

try {

$hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

$stmt = $this->db->prepare(

"INSERT INTO usuarios (nombre, email, password, rol)

VALUES (:nombre, :email, :password, 'usuario')"

);

return $stmt->execute([

':nombre' => $nombre,

':email' => $email,

':password' => $hashedPassword

]);

} catch (PDOException $e) {

error\_log("Error en registro: " . $e->getMessage());

return false;

}

}

public function verificarSesion() {

return isset($\_SESSION['usuario\_id']);

}

public function cerrarSesion() {

session\_destroy();

return true;

}

}

?>

```

13. SISTEMA DE PEDIDOS

---------------------

```php

// pedidos.php

<?php

class Pedido {

private $db;

private $id;

private $usuario\_id;

private $items;

private $total;

private $estado;

private $fecha;

public function \_\_construct($usuario\_id) {

$this->db = Database::getInstance()->getConnection();

$this->usuario\_id = $usuario\_id;

$this->items = [];

$this->total = 0;

$this->estado = 'pendiente';

$this->fecha = date('Y-m-d H:i:s');

}

public function agregarItem($producto\_id, $cantidad) {

try {

$stmt = $this->db->prepare(

"SELECT \* FROM productos WHERE id = :id LIMIT 1"

);

$stmt->execute([':id' => $producto\_id]);

$producto = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($producto) {

$this->items[] = [

'producto\_id' => $producto\_id,

'cantidad' => $cantidad,

'precio' => $producto['precio']

];

$this->calcularTotal();

return true;

}

return false;

} catch (PDOException $e) {

error\_log("Error al agregar item: " . $e->getMessage());

return false;

}

}

public function guardar() {

try {

$this->db->beginTransaction();

// Insertar pedido

$stmt = $this->db->prepare(

"INSERT INTO pedidos (usuario\_id, total, estado, fecha)

VALUES (:usuario\_id, :total, :estado, :fecha)"

);

$stmt->execute([

':usuario\_id' => $this->usuario\_id,

':total' => $this->total,

':estado' => $this->estado,

':fecha' => $this->fecha

]);

$this->id = $this->db->lastInsertId();

// Insertar items

$stmt = $this->db->prepare(

"INSERT INTO pedidos\_items (pedido\_id, producto\_id, cantidad, precio)

VALUES (:pedido\_id, :producto\_id, :cantidad, :precio)"

);

foreach ($this->items as $item) {

$stmt->execute([

':pedido\_id' => $this->id,

':producto\_id' => $item['producto\_id'],

':cantidad' => $item['cantidad'],

':precio' => $item['precio']

]);

}

$this->db->commit();

return true;

} catch (PDOException $e) {

$this->db->rollBack();

error\_log("Error al guardar pedido: " . $e->getMessage());

return false;

}

}

private function calcularTotal() {

$this->total = array\_reduce($this->items, function($total, $item) {

return $total + ($item['precio'] \* $item['cantidad']);

}, 0);

}

}

?>

```

14. SISTEMA DE NOTIFICACIONES

---------------------------

```javascript

// notificaciones.js

class NotificacionSistema {

constructor() {

this.notificaciones = [];

this.observadores = [];

}

static getInstance() {

if (!NotificacionSistema.instance) {

NotificacionSistema.instance = new NotificacionSistema();

}

return NotificacionSistema.instance;

}

agregarNotificacion(tipo, mensaje, duracion = 5000) {

const notificacion = {

id: Date.now(),

tipo,

mensaje,

duracion

};

this.notificaciones.push(notificacion);

this.mostrarNotificacion(notificacion);

this.notificarObservadores();

}

mostrarNotificacion(notificacion) {

const notificacionElement = document.createElement('div');

notificacionElement.className = `notificacion notificacion--${notificacion.tipo}`;

notificacionElement.innerHTML = `

<div class="notificacion\_\_contenido">

<p>${notificacion.mensaje}</p>

<button class="notificacion\_\_cerrar">&times;</button>

</div>

`;

document.body.appendChild(notificacionElement);

// Animación de entrada

requestAnimationFrame(() => {

notificacionElement.classList.add('notificacion--visible');

});

// Auto-cerrar

setTimeout(() => {

this.cerrarNotificacion(notificacionElement);

}, notificacion.duracion);

// Botón de cerrar

notificacionElement.querySelector('.notificacion\_\_cerrar')

.addEventListener('click', () => {

this.cerrarNotificacion(notificacionElement);

});

}

cerrarNotificacion(elemento) {

elemento.classList.remove('notificacion--visible');

elemento.addEventListener('transitionend', () => {

elemento.remove();

});

}

agregarObservador(observador) {

this.observadores.push(observador);

}

notificarObservadores() {

this.observadores.forEach(observador => {

observador.actualizar(this.notificaciones);

});

}

}

// Estilos para las notificaciones

const estilos = `

.notificacion {

position: fixed;

top: 20px;

right: 20px;

padding: 15px;

border-radius: 4px;

background: white;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

transform: translateX(120%);

transition: transform 0.3s ease;

z-index: 1000;

}

.notificacion--visible {

transform: translateX(0);

}

.notificacion--exito {

border-left: 4px solid #4CAF50;

}

.notificacion--error {

border-left: 4px solid #f44336;

}

.notificacion--info {

border-left: 4px solid #2196F3;

}

.notificacion\_\_contenido {

display: flex;

align-items: center;

justify-content: space-between;

}

.notificacion\_\_cerrar {

background: none;

border: none;

font-size: 20px;

cursor: pointer;

padding: 0 5px;

}

`;

// Agregar estilos al documento

const styleSheet = document.createElement('style');

styleSheet.textContent = estilos;

document.head.appendChild(styleSheet);

```

15. SISTEMA DE CACHÉ

------------------

```php

// cache.php

<?php

class Cache {

private static $instance = null;

private $cache\_dir;

private $tiempo\_expiracion;

private function \_\_construct() {

$this->cache\_dir = \_\_DIR\_\_ . '/cache/';

$this->tiempo\_expiracion = 3600; // 1 hora

$this->inicializarCache();

}

public static function getInstance() {

if (self::$instance === null) {

self::$instance = new self();

}

return self::$instance;

}

private function inicializarCache() {

if (!file\_exists($this->cache\_dir)) {

mkdir($this->cache\_dir, 0777, true);

}

}

public function set($key, $data, $tiempo = null) {

$tiempo = $tiempo ?? $this->tiempo\_expiracion;

$cache\_data = [

'tiempo' => time() + $tiempo,

'data' => $data

];

$file = $this->cache\_dir . md5($key) . '.cache';

return file\_put\_contents($file, serialize($cache\_data));

}

public function get($key) {

$file = $this->cache\_dir . md5($key) . '.cache';

if (!file\_exists($file)) {

return false;

}

$cache\_data = unserialize(file\_get\_contents($file));

if (time() > $cache\_data['tiempo']) {

unlink($file);

return false;

}

return $cache\_data['data'];

}

public function delete($key) {

$file = $this->cache\_dir . md5($key) . '.cache';

if (file\_exists($file)) {

return unlink($file);

}

return false;

}

public function limpiar() {

$files = glob($this->cache\_dir . '\*.cache');

foreach ($files as $file) {

unlink($file);

}

return true;

}

}

?>

```

16. SISTEMA DE LOGS

-----------------

```php

// logger.php

<?php

class Logger {

private static $instance = null;

private $log\_file;

private $nivel\_log;

const DEBUG = 1;

const INFO = 2;

const WARNING = 3;

const ERROR = 4;

private function \_\_construct() {

$this->log\_file = \_\_DIR\_\_ . '/logs/app.log';

$this->nivel\_log = self::DEBUG;

$this->inicializarLog();

}

public static function getInstance() {

if (self::$instance === null) {

self::$instance = new self();

}

return self::$instance;

}

private function inicializarLog() {

if (!file\_exists(dirname($this->log\_file))) {

mkdir(dirname($this->log\_file), 0777, true);

}

}

public function log($mensaje, $nivel = self::INFO) {

if ($nivel < $this->nivel\_log) {

return;

}

$fecha = date('Y-m-d H:i:s');

$nivel\_str = $this->getNivelString($nivel);

$log\_entry = "[$fecha] [$nivel\_str] $mensaje" . PHP\_EOL;

file\_put\_contents($this->log\_file, $log\_entry, FILE\_APPEND);

}

private function getNivelString($nivel) {

switch ($nivel) {

case self::DEBUG:

return 'DEBUG';

case self::INFO:

return 'INFO';

case self::WARNING:

return 'WARNING';

case self::ERROR:

return 'ERROR';

default:

return 'UNKNOWN';

}

}

public function debug($mensaje) {

$this->log($mensaje, self::DEBUG);

}

public function info($mensaje) {

$this->log($mensaje, self::INFO);

}

public function warning($mensaje) {

$this->log($mensaje, self::WARNING);

}

public function error($mensaje) {

$this->log($mensaje, self::ERROR);

}

public function setNivelLog($nivel) {

$this->nivel\_log = $nivel;

}

}

?>

```

17. SISTEMA DE VALIDACIÓN DE DATOS

--------------------------------

```php

// validator.php

<?php

class Validator {

private $errores = [];

private $reglas = [];

private $datos = [];

public function \_\_construct($datos) {

$this->datos = $datos;

}

public function setReglas($reglas) {

$this->reglas = $reglas;

return $this;

}

public function validar() {

foreach ($this->reglas as $campo => $reglas) {

foreach ($reglas as $regla) {

$this->aplicarRegla($campo, $regla);

}

}

return empty($this->errores);

}

private function aplicarRegla($campo, $regla) {

$valor = $this->datos[$campo] ?? null;

if (is\_string($regla)) {

$regla = ['tipo' => $regla];

}

switch ($regla['tipo']) {

case 'requerido':

if (empty($valor)) {

$this->agregarError($campo, "El campo $campo es requerido");

}

break;

case 'email':

if (!filter\_var($valor, FILTER\_VALIDATE\_EMAIL)) {

$this->agregarError($campo, "El email no es válido");

}

break;

case 'min':

if (strlen($valor) < $regla['valor']) {

$this->agregarError($campo, "El campo $campo debe tener al menos {$regla['valor']} caracteres");

}

break;

case 'max':

if (strlen($valor) > $regla['valor']) {

$this->agregarError($campo, "El campo $campo no debe exceder {$regla['valor']} caracteres");

}

break;

case 'numerico':

if (!is\_numeric($valor)) {

$this->agregarError($campo, "El campo $campo debe ser numérico");

}

break;

case 'fecha':

if (!strtotime($valor)) {

$this->agregarError($campo, "El campo $campo debe ser una fecha válida");

}

break;

}

}

private function agregarError($campo, $mensaje) {

if (!isset($this->errores[$campo])) {

$this->errores[$campo] = [];

}

$this->errores[$campo][] = $mensaje;

}

public function getErrores() {

return $this->errores;

}

public function getPrimerError($campo) {

return $this->errores[$campo][0] ?? null;

}

}

?>

```

18. SISTEMA DE ENCRIPTACIÓN

-------------------------

```php

// encryption.php

<?php

class Encryption {

private static $instance = null;

private $key;

private $cipher = 'aes-256-cbc';

private function \_\_construct() {

$this->key = hash('sha256', 'tu\_clave\_secreta', true);

}

public static function getInstance() {

if (self::$instance === null) {

self::$instance = new self();

}

return self::$instance;

}

public function encrypt($data) {

$iv = openssl\_random\_pseudo\_bytes(openssl\_cipher\_iv\_length($this->cipher));

$encrypted = openssl\_encrypt($data, $this->cipher, $this->key, 0, $iv);

return base64\_encode($iv . $encrypted);

}

public function decrypt($data) {

$data = base64\_decode($data);

$ivLength = openssl\_cipher\_iv\_length($this->cipher);

$iv = substr($data, 0, $ivLength);

$encrypted = substr($data, $ivLength);

return openssl\_decrypt($encrypted, $this->cipher, $this->key, 0, $iv);

}

public function hash($data) {

return password\_hash($data, PASSWORD\_BCRYPT);

}

public function verifyHash($data, $hash) {

return password\_verify($data, $hash);

}

public function generateToken($length = 32) {

return bin2hex(random\_bytes($length));

}

}

?>

```

19. SISTEMA DE GESTIÓN DE IMÁGENES

--------------------------------

```php

// ImageManager.php

<?php

class ImageManager {

private $allowed\_types = ['image/jpeg', 'image/png', 'image/webp'];

private $max\_size = 5242880; // 5MB

private $upload\_dir;

private $quality = 80;

public function \_\_construct($upload\_dir = 'uploads/') {

$this->upload\_dir = rtrim($upload\_dir, '/') . '/';

$this->createUploadDir();

}

private function createUploadDir() {

if (!file\_exists($this->upload\_dir)) {

mkdir($this->upload\_dir, 0777, true);

}

}

public function uploadImage($file, $custom\_name = null) {

try {

$this->validateImage($file);

$extension = pathinfo($file['name'], PATHINFO\_EXTENSION);

$filename = $custom\_name ? $custom\_name . '.' . $extension :

uniqid('img\_') . '.' . $extension;

$destination = $this->upload\_dir . $filename;

// Optimizar imagen antes de guardar

$optimized\_image = $this->optimizeImage($file['tmp\_name']);

if ($optimized\_image) {

imagejpeg($optimized\_image, $destination, $this->quality);

imagedestroy($optimized\_image);

// Crear versiones en diferentes tamaños

$this->createThumbnails($destination);

return [

'success' => true,

'filename' => $filename,

'path' => $destination

];

}

return [

'success' => false,

'error' => 'Error al optimizar la imagen'

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

private function validateImage($file) {

if (!isset($file['tmp\_name']) || !is\_uploaded\_file($file['tmp\_name'])) {

throw new Exception('Archivo no válido');

}

if (!in\_array($file['type'], $this->allowed\_types)) {

throw new Exception('Tipo de archivo no permitido');

}

if ($file['size'] > $this->max\_size) {

throw new Exception('El archivo excede el tamaño máximo permitido');

}

}

private function optimizeImage($source) {

$info = getimagesize($source);

switch ($info['mime']) {

case 'image/jpeg':

$image = imagecreatefromjpeg($source);

break;

case 'image/png':

$image = imagecreatefrompng($source);

break;

case 'image/webp':

$image = imagecreatefromwebp($source);

break;

default:

return false;

}

// Mantener transparencia para PNG

if ($info['mime'] === 'image/png') {

imagepalettetotruecolor($image);

imagealphablending($image, true);

imagesavealpha($image, true);

}

return $image;

}

private function createThumbnails($source) {

$sizes = [

'small' => 150,

'medium' => 300,

'large' => 600

];

foreach ($sizes as $name => $size) {

$this->createThumbnail($source, $name, $size);

}

}

private function createThumbnail($source, $prefix, $max\_size) {

$info = getimagesize($source);

$width = $info[0];

$height = $info[1];

// Calcular nuevas dimensiones

if ($width > $height) {

$new\_width = $max\_size;

$new\_height = floor($height \* ($max\_size / $width));

} else {

$new\_height = $max\_size;

$new\_width = floor($width \* ($max\_size / $height));

}

$thumb = imagecreatetruecolor($new\_width, $new\_height);

// Mantener transparencia

imagealphablending($thumb, false);

imagesavealpha($thumb, true);

$source\_image = imagecreatefromjpeg($source);

imagecopyresampled(

$thumb, $source\_image,

0, 0, 0, 0,

$new\_width, $new\_height,

$width, $height

);

$path\_info = pathinfo($source);

$thumbnail\_path = $path\_info['dirname'] . '/' .

$prefix . '\_' . $path\_info['basename'];

imagejpeg($thumb, $thumbnail\_path, $this->quality);

imagedestroy($thumb);

imagedestroy($source\_image);

}

}

?>

```

20. SISTEMA DE BÚSQUEDA AVANZADA

------------------------------

```php

// SearchEngine.php

<?php

class SearchEngine {

private $db;

private $table;

private $searchable\_fields;

private $order\_by;

private $limit;

private $offset;

public function \_\_construct($table, $searchable\_fields) {

$this->db = Database::getInstance()->getConnection();

$this->table = $table;

$this->searchable\_fields = $searchable\_fields;

$this->limit = 10;

$this->offset = 0;

}

public function search($query, $filters = [], $sort = []) {

try {

$where\_conditions = [];

$params = [];

// Procesar términos de búsqueda

if (!empty($query)) {

$search\_conditions = [];

foreach ($this->searchable\_fields as $field) {

$search\_conditions[] = "$field LIKE :search\_$field";

$params["search\_$field"] = "%$query%";

}

$where\_conditions[] = '(' . implode(' OR ', $search\_conditions) . ')';

}

// Procesar filtros

foreach ($filters as $field => $value) {

if (in\_array($field, $this->searchable\_fields)) {

if (is\_array($value)) {

$where\_conditions[] = "$field IN (" .

implode(',', array\_fill(0, count($value), '?')) . ")";

$params = array\_merge($params, $value);

} else {

$where\_conditions[] = "$field = ?";

$params[] = $value;

}

}

}

// Construir consulta SQL

$sql = "SELECT \* FROM {$this->table}";

if (!empty($where\_conditions)) {

$sql .= " WHERE " . implode(' AND ', $where\_conditions);

}

// Ordenamiento

if (!empty($sort)) {

$order\_clauses = [];

foreach ($sort as $field => $direction) {

if (in\_array($field, $this->searchable\_fields)) {

$direction = strtoupper($direction) === 'DESC' ? 'DESC' : 'ASC';

$order\_clauses[] = "$field $direction";

}

}

if (!empty($order\_clauses)) {

$sql .= " ORDER BY " . implode(', ', $order\_clauses);

}

}

// Paginación

$sql .= " LIMIT ? OFFSET ?";

$params[] = $this->limit;

$params[] = $this->offset;

// Ejecutar consulta

$stmt = $this->db->prepare($sql);

$stmt->execute($params);

return [

'success' => true,

'results' => $stmt->fetchAll(PDO::FETCH\_ASSOC),

'total' => $this->getTotalResults($where\_conditions, $params)

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

private function getTotalResults($where\_conditions, $params) {

$sql = "SELECT COUNT(\*) as total FROM {$this->table}";

if (!empty($where\_conditions)) {

$sql .= " WHERE " . implode(' AND ', $where\_conditions);

}

$stmt = $this->db->prepare($sql);

$stmt->execute($params);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['total'];

}

public function setLimit($limit) {

$this->limit = max(1, (int)$limit);

return $this;

}

public function setOffset($offset) {

$this->offset = max(0, (int)$offset);

return $this;

}

}

?>

```

21. SISTEMA DE EXPORTACIÓN DE DATOS

--------------------------------

```php

// DataExporter.php

<?php

class DataExporter {

private $data;

private $format;

private $filename;

public function \_\_construct($data, $format = 'csv') {

$this->data = $data;

$this->format = strtolower($format);

$this->filename = 'export\_' . date('Y-m-d\_H-i-s');

}

public function export() {

switch ($this->format) {

case 'csv':

return $this->exportCSV();

case 'excel':

return $this->exportExcel();

case 'json':

return $this->exportJSON();

case 'pdf':

return $this->exportPDF();

default:

throw new Exception('Formato no soportado');

}

}

private function exportCSV() {

if (empty($this->data)) {

throw new Exception('No hay datos para exportar');

}

$output = fopen('php://temp', 'r+');

// Escribir BOM para UTF-8

fprintf($output, chr(0xEF).chr(0xBB).chr(0xBF));

// Escribir encabezados

fputcsv($output, array\_keys(reset($this->data)));

// Escribir datos

foreach ($this->data as $row) {

fputcsv($output, $row);

}

rewind($output);

$csv = stream\_get\_contents($output);

fclose($output);

return [

'success' => true,

'filename' => $this->filename . '.csv',

'content' => $csv,

'type' => 'text/csv'

];

}

private function exportJSON() {

return [

'success' => true,

'filename' => $this->filename . '.json',

'content' => json\_encode($this->data, JSON\_PRETTY\_PRINT),

'type' => 'application/json'

];

}

private function exportExcel() {

require\_once 'vendor/autoload.php';

$spreadsheet = new \PhpOffice\PhpSpreadsheet\Spreadsheet();

$sheet = $spreadsheet->getActiveSheet();

// Escribir encabezados

$headers = array\_keys(reset($this->data)));

$col = 1;

foreach ($headers as $header) {

$sheet->setCellValueByColumnAndRow($col++, 1, $header);

}

// Escribir datos

$row = 2;

foreach ($this->data as $rowData) {

$col = 1;

foreach ($rowData as $value) {

$sheet->setCellValueByColumnAndRow($col++, $row, $value);

}

$row++;

}

// Autoajustar columnas

foreach (range('A', $sheet->getHighestColumn()) as $col) {

$sheet->getColumnDimension($col)->setAutoSize(true);

}

// Crear archivo

$writer = new \PhpOffice\PhpSpreadsheet\Writer\Xlsx($spreadsheet);

$temp\_file = tempnam(sys\_get\_temp\_dir(), 'excel\_');

$writer->save($temp\_file);

$content = file\_get\_contents($temp\_file);

unlink($temp\_file);

return [

'success' => true,

'filename' => $this->filename . '.xlsx',

'content' => $content,

'type' => 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'

];

}

private function exportPDF() {

require\_once 'vendor/autoload.php';

$pdf = new \TCPDF(PDF\_PAGE\_ORIENTATION, PDF\_UNIT, PDF\_PAGE\_FORMAT, true, 'UTF-8', false);

// Configurar documento

$pdf->SetCreator(PDF\_CREATOR);

$pdf->SetAuthor('Burger House');

$pdf->SetTitle('Exportación de datos');

// Configurar márgenes

$pdf->SetMargins(PDF\_MARGIN\_LEFT, PDF\_MARGIN\_TOP, PDF\_MARGIN\_RIGHT);

$pdf->SetHeaderMargin(PDF\_MARGIN\_HEADER);

$pdf->SetFooterMargin(PDF\_MARGIN\_FOOTER);

// Agregar página

$pdf->AddPage();

// Crear tabla HTML

$html = '<table border="1" cellpadding="4">';

// Encabezados

$html .= '<tr>';

foreach (array\_keys(reset($this->data)) as $header) {

$html .= "<th>$header</th>";

}

$html .= '</tr>';

// Datos

foreach ($this->data as $row) {

$html .= '<tr>';

foreach ($row as $value) {

$html .= "<td>$value</td>";

}

$html .= '</tr>';

}

$html .= '</table>';

// Escribir HTML

$pdf->writeHTML($html, true, false, true, false, '');

return [

'success' => true,

'filename' => $this->filename . '.pdf',

'content' => $pdf->Output('', 'S'),

'type' => 'application/pdf'

];

}

public function setFilename($filename) {

$this->filename = $filename;

return $this;

}

}

?>

```

22. SISTEMA DE GENERACIÓN DE REPORTES

----------------------------------

```php

// ReportGenerator.php

<?php

class ReportGenerator {

private $db;

private $start\_date;

private $end\_date;

private $report\_type;

public function \_\_construct($start\_date = null, $end\_date = null) {

$this->db = Database::getInstance()->getConnection();

$this->start\_date = $start\_date ?? date('Y-m-d', strtotime('-30 days'));

$this->end\_date = $end\_date ?? date('Y-m-d');

}

public function generateReport($type) {

$this->report\_type = $type;

switch ($type) {

case 'ventas':

return $this->generateSalesReport();

case 'productos':

return $this->generateProductsReport();

case 'clientes':

return $this->generateCustomersReport();

case 'inventario':

return $this->generateInventoryReport();

default:

throw new Exception('Tipo de reporte no válido');

}

}

private function generateSalesReport() {

try {

// Ventas por día

$daily\_sales = $this->db->prepare("

SELECT

DATE(fecha) as fecha,

COUNT(\*) as total\_pedidos,

SUM(total) as total\_ventas

FROM pedidos

WHERE fecha BETWEEN ? AND ?

GROUP BY DATE(fecha)

ORDER BY fecha

");

$daily\_sales->execute([$this->start\_date, $this->end\_date]);

// Productos más vendidos

$top\_products = $this->db->prepare("

SELECT

p.nombre,

COUNT(pi.producto\_id) as cantidad\_vendida,

SUM(pi.precio \* pi.cantidad) as total\_ventas

FROM pedidos\_items pi

JOIN productos p ON p.id = pi.producto\_id

JOIN pedidos pe ON pe.id = pi.pedido\_id

WHERE pe.fecha BETWEEN ? AND ?

GROUP BY pi.producto\_id

ORDER BY cantidad\_vendida DESC

LIMIT 10

");

$top\_products->execute([$this->start\_date, $this->end\_date]);

// Métodos de pago

$payment\_methods = $this->db->prepare("

SELECT

metodo\_pago,

COUNT(\*) as total\_pedidos,

SUM(total) as total\_ventas

FROM pedidos

WHERE fecha BETWEEN ? AND ?

GROUP BY metodo\_pago

");

$payment\_methods->execute([$this->start\_date, $this->end\_date]);

return [

'success' => true,

'data' => [

'ventas\_diarias' => $daily\_sales->fetchAll(PDO::FETCH\_ASSOC),

'productos\_top' => $top\_products->fetchAll(PDO::FETCH\_ASSOC),

'metodos\_pago' => $payment\_methods->fetchAll(PDO::FETCH\_ASSOC)

]

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

private function generateProductsReport() {

try {

// Productos y su rendimiento

$products = $this->db->prepare("

SELECT

p.id,

p.nombre,

p.precio,

p.categoria,

COUNT(pi.producto\_id) as veces\_vendido,

COALESCE(SUM(pi.cantidad), 0) as unidades\_vendidas,

COALESCE(SUM(pi.precio \* pi.cantidad), 0) as total\_ventas

FROM productos p

LEFT JOIN pedidos\_items pi ON p.id = pi.producto\_id

LEFT JOIN pedidos pe ON pe.id = pi.pedido\_id

WHERE pe.fecha BETWEEN ? AND ? OR pe.fecha IS NULL

GROUP BY p.id

ORDER BY unidades\_vendidas DESC

");

$products->execute([$this->start\_date, $this->end\_date]);

return [

'success' => true,

'data' => [

'productos' => $products->fetchAll(PDO::FETCH\_ASSOC)

]

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

private function generateCustomersReport() {

try {

// Clientes más frecuentes

$top\_customers = $this->db->prepare("

SELECT

u.id,

u.nombre,

u.email,

COUNT(p.id) as total\_pedidos,

SUM(p.total) as total\_gastado,

AVG(p.total) as promedio\_pedido,

MAX(p.fecha) as ultima\_compra

FROM usuarios u

LEFT JOIN pedidos p ON u.id = p.usuario\_id

WHERE p.fecha BETWEEN ? AND ? OR p.fecha IS NULL

GROUP BY u.id

ORDER BY total\_pedidos DESC

");

$top\_customers->execute([$this->start\_date, $this->end\_date]);

return [

'success' => true,

'data' => [

'clientes' => $top\_customers->fetchAll(PDO::FETCH\_ASSOC)

]

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

private function generateInventoryReport() {

try {

// Estado del inventario

$inventory = $this->db->prepare("

SELECT

p.id,

p.nombre,

p.precio,

p.stock\_actual,

p.stock\_minimo,

p.stock\_maximo,

CASE

WHEN p.stock\_actual <= p.stock\_minimo THEN 'bajo'

WHEN p.stock\_actual >= p.stock\_maximo THEN 'alto'

ELSE 'normal'

END as nivel\_stock

FROM productos p

ORDER BY

CASE

WHEN p.stock\_actual <= p.stock\_minimo THEN 1

WHEN p.stock\_actual >= p.stock\_maximo THEN 3

ELSE 2

END,

p.nombre

");

$inventory->execute();

return [

'success' => true,

'data' => [

'inventario' => $inventory->fetchAll(PDO::FETCH\_ASSOC)

]

];

} catch (Exception $e) {

return [

'success' => false,

'error' => $e->getMessage()

];

}

}

public function setDateRange($start\_date, $end\_date) {

$this->start\_date = $start\_date;

$this->end\_date = $end\_date;

return $this;

}

}

?>

```

EXPLICACIONES DETALLADAS DE LOS SISTEMAS

======================================

1. SISTEMA DE GESTIÓN DE IMÁGENES (ImageManager.php)

-------------------------------------------------

Este sistema se encarga del manejo completo de imágenes en la aplicación. Aquí está el desglose de cada componente:

```php

class ImageManager {

private $allowed\_types = ['image/jpeg', 'image/png', 'image/webp'];

private $max\_size = 5242880; // 5MB

private $upload\_dir;

private $quality = 80;

}

Explicación de propiedades:

- `allowed\_types`: Define los tipos MIME permitidos para las imágenes

- `max\_size`: Establece el tamaño máximo de archivo (5MB)

- `upload\_dir`: Directorio donde se guardarán las imágenes

- `quality`: Calidad de compresión JPEG (80%)

Métodos principales:

1. Constructor y configuración:

```php

public function \_\_construct($upload\_dir = 'uploads/') {

$this->upload\_dir = rtrim($upload\_dir, '/') . '/';

$this->createUploadDir();

}

```

- Inicializa el directorio de carga

- Asegura que el directorio termine con '/'

- Crea el directorio si no existe

2. Validación de imágenes:

```php

private function validateImage($file) {

// Validaciones de seguridad

}

```

- Verifica que el archivo sea válido

- Comprueba el tipo MIME permitido

- Valida el tamaño máximo

3. Optimización de imágenes:

```php

private function optimizeImage($source) {

// Proceso de optimización

}

```

- Detecta el tipo de imagen

- Crea una nueva imagen optimizada

- Mantiene la transparencia en PNG

- Ajusta la calidad

4. Creación de miniaturas:

```php

private function createThumbnails($source) {

$sizes = [

'small' => 150,

'medium' => 300,

'large' => 600

];

}

```

- Genera versiones en diferentes tamaños

- Mantiene la proporción de aspecto

- Optimiza cada versión

2. SISTEMA DE BÚSQUEDA AVANZADA (SearchEngine.php)

----------------------------------------------

Sistema que implementa búsqueda flexible y eficiente en la base de datos:

```php

class SearchEngine {

private $db;

private $table;

private $searchable\_fields;

private $order\_by;

private $limit;

private $offset;

}

Componentes principales:

1. Constructor:

```php

public function \_\_construct($table, $searchable\_fields) {

$this->db = Database::getInstance()->getConnection();

$this->table = $table;

$this->searchable\_fields = $searchable\_fields;

}

```

- Inicializa la conexión a la base de datos

- Define la tabla objetivo

- Establece campos buscables

2. Método de búsqueda:

```php

public function search($query, $filters = [], $sort = []) {

// Implementación de búsqueda

}

```

Funcionalidades:

- Búsqueda por texto libre

- Filtros personalizados

- Ordenamiento configurable

- Paginación automática

- Conteo de resultados

3. Construcción de consultas:

```php

// Ejemplo de construcción de WHERE

$where\_conditions[] = "$field LIKE :search\_$field";

$params["search\_$field"] = "%$query%";

```

- Genera consultas SQL seguras

- Previene inyección SQL

- Soporta búsqueda parcial

3. SISTEMA DE EXPORTACIÓN (DataExporter.php)

----------------------------------------

Sistema para exportar datos en múltiples formatos:

```php

class DataExporter {

private $data;

private $format;

private $filename;

}

Formatos soportados:

1. CSV:

```php

private function exportCSV() {

// Implementación CSV

}

```

- Genera archivos CSV UTF-8

- Maneja encabezados automáticamente

- Escapa caracteres especiales

2. Excel:

```php

private function exportExcel() {

// Implementación Excel

}

```

- Usa PhpSpreadsheet

- Formato profesional

- Autoajuste de columnas

3. PDF:

```php

private function exportPDF() {

// Implementación PDF

}

```

- Usa TCPDF

- Tablas formateadas

- Encabezados y pies de página

4. SISTEMA DE REPORTES (ReportGenerator.php)

----------------------------------------

Sistema de generación de reportes empresariales:

```php

class ReportGenerator {

private $db;

private $start\_date;

private $end\_date;

private $report\_type;

}

Tipos de reportes:

1. Reporte de Ventas:

```php

private function generateSalesReport() {

// Implementación de reporte de ventas

}

```

Incluye:

- Ventas diarias

- Productos más vendidos

- Métodos de pago

- Totales y promedios

2. Reporte de Productos:

```php

private function generateProductsReport() {

// Implementación de reporte de productos

}

```

Muestra:

- Rendimiento de productos

- Estadísticas de ventas

- Categorías populares

3. Reporte de Clientes:

```php

private function generateCustomersReport() {

// Implementación de reporte de clientes

}

```

Analiza:

- Clientes frecuentes

- Patrones de compra

- Valor promedio de pedido

4. Reporte de Inventario:

```php

private function generateInventoryReport() {

// Implementación de reporte de inventario

}

```

Controla:

- Niveles de stock

- Alertas de reposición

- Estado de productos

EJEMPLOS DE USO PRÁCTICO

-----------------------

1. Gestión de Imágenes:

```php

// Subir y optimizar una imagen de producto

$imageManager = new ImageManager('uploads/productos/');

$result = $imageManager->uploadImage($\_FILES['producto\_imagen']);

if ($result['success']) {

// La imagen se subió correctamente

$rutaImagen = $result['path'];

$nombreArchivo = $result['filename'];

// Actualizar la base de datos

$stmt = $db->prepare("UPDATE productos SET imagen = ? WHERE id = ?");

$stmt->execute([$rutaImagen, $producto\_id]);

}

```

2. Búsqueda Avanzada:

```php

// Búsqueda de productos con filtros

$search = new SearchEngine('productos', ['nombre', 'categoria', 'descripcion']);

// Configurar búsqueda

$resultados = $search->setLimit(20)

->setOffset(0)

->search(

'hamburguesa', // término de búsqueda

[

'categoria' => 'principal',

'precio' => ['>', 100]

],

['nombre' => 'ASC']

);

// Procesar resultados

if ($resultados['success']) {

foreach ($resultados['results'] as $producto) {

// Mostrar productos

}

}

```

3. Exportación de Datos:

```php

// Exportar lista de productos a Excel

$productos = $db->query("SELECT \* FROM productos")->fetchAll();

$exporter = new DataExporter($productos, 'excel');

$resultado = $exporter->setFilename('catalogo\_productos\_' . date('Y-m-d'))

->export();

if ($resultado['success']) {

// Descargar archivo

header('Content-Type: ' . $resultado['type']);

header('Content-Disposition: attachment; filename="' . $resultado['filename'] . '"');

echo $resultado['content'];

}

```

4. Generación de Reportes:

```php

// Generar reporte mensual de ventas

$reportGen = new ReportGenerator(

date('Y-m-01'), // primer día del mes

date('Y-m-t') // último día del mes

);

$reporte = $reportGen->generateReport('ventas');

if ($reporte['success']) {

$ventas\_diarias = $reporte['data']['ventas\_diarias'];

$productos\_top = $reporte['data']['productos\_top'];

$metodos\_pago = $reporte['data']['metodos\_pago'];

// Procesar y mostrar datos

foreach ($ventas\_diarias as $venta) {

echo "Fecha: {$venta['fecha']} - Total: {$venta['total\_ventas']}";

}

}

```

CONSIDERACIONES DE SEGURIDAD

--------------------------

1. Gestión de Imágenes:

- Validación de tipos MIME

- Límites de tamaño

- Nombres de archivo seguros

- Permisos de directorio

2. Búsqueda:

- Prevención de inyección SQL

- Validación de campos

- Límites de consulta

3. Exportación:

- Validación de datos

- Control de acceso

- Límites de memoria

4. Reportes:

- Filtrado de datos sensibles

- Control de acceso

- Optimización de consultas

OPTIMIZACIÓN Y RENDIMIENTO

------------------------

1. Imágenes:

- Compresión automática

- Caché de miniaturas

- Carga diferida

2. Búsqueda:

- Índices de base de datos

- Paginación eficiente

- Caché de resultados

3. Exportación:

- Procesamiento por lotes

- Gestión de memoria

- Compresión de archivos

4. Reportes:

- Consultas optimizadas

- Caché de reportes

- Ejecución asíncrona

```

23. SISTEMA DE API REST

---------------------

```php

// ApiController.php

class ApiController {

private $request;

private $response;

private $method;

private $endpoints = [

'productos' => ['GET', 'POST', 'PUT', 'DELETE'],

'pedidos' => ['GET', 'POST', 'PUT'],

'usuarios' => ['GET', 'POST', 'PUT']

];

public function \_\_construct() {

$this->request = $\_REQUEST;

$this->method = $\_SERVER['REQUEST\_METHOD'];

header('Content-Type: application/json');

}

public function handleRequest($endpoint) {

if (!isset($this->endpoints[$endpoint])) {

return $this->sendError('Endpoint no encontrado', 404);

}

if (!in\_array($this->method, $this->endpoints[$endpoint])) {

return $this->sendError('Método no permitido', 405);

}

switch ($this->method) {

case 'GET':

return $this->handleGet($endpoint);

case 'POST':

return $this->handlePost($endpoint);

case 'PUT':

return $this->handlePut($endpoint);

case 'DELETE':

return $this->handleDelete($endpoint);

}

}

private function handleGet($endpoint) {

try {

$id = isset($\_GET['id']) ? $\_GET['id'] : null;

$db = Database::getInstance()->getConnection();

if ($id) {

$stmt = $db->prepare("SELECT \* FROM $endpoint WHERE id = ?");

$stmt->execute([$id]);

$data = $stmt->fetch(PDO::FETCH\_ASSOC);

} else {

$stmt = $db->query("SELECT \* FROM $endpoint");

$data = $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

return $this->sendResponse($data);

} catch (Exception $e) {

return $this->sendError($e->getMessage());

}

}

private function handlePost($endpoint) {

try {

$data = json\_decode(file\_get\_contents('php://input'), true);

if (!$data) {

return $this->sendError('Datos inválidos');

}

$db = Database::getInstance()->getConnection();

$columns = implode(', ', array\_keys($data));

$values = implode(', ', array\_fill(0, count($data), '?'));

$stmt = $db->prepare("INSERT INTO $endpoint ($columns) VALUES ($values)");

$stmt->execute(array\_values($data));

return $this->sendResponse([

'id' => $db->lastInsertId(),

'message' => 'Registro creado exitosamente'

], 201);

} catch (Exception $e) {

return $this->sendError($e->getMessage());

}

}

private function sendResponse($data, $code = 200) {

http\_response\_code($code);

echo json\_encode([

'status' => 'success',

'data' => $data

]);

}

private function sendError($message, $code = 400) {

http\_response\_code($code);

echo json\_encode([

'status' => 'error',

'message' => $message

]);

}

}

```

Explicación del Sistema API REST:

- Maneja peticiones HTTP (GET, POST, PUT, DELETE)

- Implementa endpoints para productos, pedidos y usuarios

- Validación de métodos y endpoints

- Respuestas en formato JSON

- Manejo de errores y códigos HTTP

24. SISTEMA DE CACHÉ AVANZADO

--------------------------

```php

// CacheManager.php

class CacheManager {

private $redis;

private $memcached;

private $file\_cache;

private $strategy;

private static $instance = null;

private function \_\_construct($strategy = 'redis') {

$this->strategy = $strategy;

$this->initializeCache();

}

public static function getInstance($strategy = 'redis') {

if (self::$instance === null) {

self::$instance = new self($strategy);

}

return self::$instance;

}

private function initializeCache() {

switch ($this->strategy) {

case 'redis':

$this->redis = new Redis();

$this->redis->connect('127.0.0.1', 6379);

break;

case 'memcached':

$this->memcached = new Memcached();

$this->memcached->addServer('localhost', 11211);

break;

case 'file':

$this->file\_cache = new FileCache();

break;

}

}

public function get($key) {

$value = null;

switch ($this->strategy) {

case 'redis':

$value = $this->redis->get($key);

break;

case 'memcached':

$value = $this->memcached->get($key);

break;

case 'file':

$value = $this->file\_cache->get($key);

break;

}

if ($value !== false) {

$this->logCacheHit($key);

return unserialize($value);

}

$this->logCacheMiss($key);

return null;

}

public function set($key, $value, $ttl = 3600) {

$serialized = serialize($value);

switch ($this->strategy) {

case 'redis':

return $this->redis->setex($key, $ttl, $serialized);

case 'memcached':

return $this->memcached->set($key, $serialized, time() + $ttl);

case 'file':

return $this->file\_cache->set($key, $serialized, $ttl);

}

}

public function delete($key) {

switch ($this->strategy) {

case 'redis':

return $this->redis->del($key);

case 'memcached':

return $this->memcached->delete($key);

case 'file':

return $this->file\_cache->delete($key);

}

}

public function flush() {

switch ($this->strategy) {

case 'redis':

return $this->redis->flushAll();

case 'memcached':

return $this->memcached->flush();

case 'file':

return $this->file\_cache->flush();

}

}

private function logCacheHit($key) {

Logger::getInstance()->info("Cache hit: $key");

}

private function logCacheMiss($key) {

Logger::getInstance()->info("Cache miss: $key");

}

}

```

Explicación del Sistema de Caché:

- Implementa múltiples estrategias (Redis, Memcached, File)

- Patrón Singleton para gestión de instancias

- Serialización de datos

- Control de tiempo de vida (TTL)

- Logging de hits y misses

25. SISTEMA DE EVENTOS Y NOTIFICACIONES

-----------------------------------

```php

// EventManager.php

class EventManager {

private $events = [];

private $websocket;

private static $instance = null;

private $logger;

private function \_\_construct() {

$this->logger = Logger::getInstance();

$this->initializeWebSocket();

}

public static function getInstance() {

if (self::$instance === null) {

self::$instance = new self();

}

return self::$instance;

}

public function subscribe($event, $callback) {

if (!isset($this->events[$event])) {

$this->events[$event] = [];

}

$this->events[$event][] = $callback;

$this->logger->info("Nuevo suscriptor para evento: $event");

}

public function emit($event, $data = null) {

$this->logger->info("Emitiendo evento: $event");

if (isset($this->events[$event])) {

foreach ($this->events[$event] as $callback) {

try {

call\_user\_func($callback, $data);

} catch (Exception $e) {

$this->logger->error("Error en callback de evento: " . $e->getMessage());

}

}

}

if ($this->websocket) {

$this->broadcastToWebSocket($event, $data);

}

}

private function broadcastToWebSocket($event, $data) {

try {

$this->websocket->broadcast([

'event' => $event,

'data' => $data,

'timestamp' => date('Y-m-d H:i:s')

]);

} catch (Exception $e) {

$this->logger->error("Error en broadcast WebSocket: " . $e->getMessage());

}

}

private function initializeWebSocket() {

try {

$this->websocket = new WebSocket([

'port' => 8080,

'host' => '0.0.0.0'

]);

} catch (Exception $e) {

$this->logger->error("Error iniciando WebSocket: " . $e->getMessage());

}

}

public function removeSubscriber($event, $callback) {

if (isset($this->events[$event])) {

$key = array\_search($callback, $this->events[$event]);

if ($key !== false) {

unset($this->events[$event][$key]);

$this->logger->info("Suscriptor removido del evento: $event");

}

}

}

}

```

Explicación del Sistema de Eventos:

- Implementa el patrón Observer

- Soporte para WebSocket

- Manejo de múltiples suscriptores

- Logging de eventos

- Manejo de errores en callbacks

26. SISTEMA DE ANÁLISIS Y ESTADÍSTICAS

----------------------------------

```php

// Analytics.php

class Analytics {

private $db;

private $cache;

private $metrics = [

'ventas\_diarias',

'productos\_populares',

'tiempo\_entrega',

'satisfaccion\_cliente'

];

public function \_\_construct() {

$this->db = Database::getInstance();

$this->cache = CacheManager::getInstance();

}

public function trackEvent($event, $data) {

try {

$this->db->insert('analytics\_events', [

'event' => $event,

'data' => json\_encode($data),

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

'user\_agent' => $\_SERVER['HTTP\_USER\_AGENT'] ?? null,

'ip\_address' => $\_SERVER['REMOTE\_ADDR'] ?? null

]);

$this->updateMetrics($event, $data);

$this->notifyIfThreshold($event, $data);

} catch (Exception $e) {

Logger::getInstance()->error("Error tracking event: " . $e->getMessage());

}

}

public function getMetrics($metric, $period = 'day') {

$cache\_key = "metrics:{$metric}:{$period}";

if ($cached = $this->cache->get($cache\_key)) {

return $cached;

}

$data = $this->calculateMetrics($metric, $period);

$this->cache->set($cache\_key, $data, 3600);

return $data;

}

private function calculateMetrics($metric, $period) {

$end\_date = date('Y-m-d H:i:s');

$start\_date = $this->getStartDate($period);

switch ($metric) {

case 'ventas\_diarias':

return $this->calculateDailySales($start\_date, $end\_date);

case 'productos\_populares':

return $this->calculatePopularProducts($start\_date, $end\_date);

case 'tiempo\_entrega':

return $this->calculateDeliveryTimes($start\_date, $end\_date);

case 'satisfaccion\_cliente':

return $this->calculateCustomerSatisfaction($start\_date, $end\_date);

}

}

private function getStartDate($period) {

switch ($period) {

case 'day':

return date('Y-m-d H:i:s', strtotime('-1 day'));

case 'week':

return date('Y-m-d H:i:s', strtotime('-1 week'));

case 'month':

return date('Y-m-d H:i:s', strtotime('-1 month'));

default:

return date('Y-m-d H:i:s', strtotime('-1 day'));

}

}

private function notifyIfThreshold($event, $data) {

$thresholds = [

'ventas\_diarias' => 1000,

'tiempo\_entrega' => 45

];

if (isset($thresholds[$event]) && $data > $thresholds[$event]) {

EventManager::getInstance()->emit('threshold\_exceeded', [

'metric' => $event,

'value' => $data,

'threshold' => $thresholds[$event]

]);

}

}

}

```

Explicación del Sistema de Análisis:

- Tracking de eventos en tiempo real

- Cálculo de métricas personalizadas

- Caché de resultados

- Notificaciones de umbrales

- Períodos configurables

27. SISTEMA DE GESTIÓN DE ERRORES

------------------------------

```php

// ErrorHandler.php

class ErrorHandler {

private $logger;

private $notifier;

private $environment;

private static $instance = null;

private function \_\_construct($environment = 'production') {

$this->environment = $environment;

$this->logger = Logger::getInstance();

$this->notifier = new NotificationSystem();

$this->registerHandlers();

}

public static function getInstance($environment = 'production') {

if (self::$instance === null) {

self::$instance = new self($environment);

}

return self::$instance;

}

private function registerHandlers() {

set\_error\_handler([$this, 'handleError']);

set\_exception\_handler([$this, 'handleException']);

register\_shutdown\_function([$this, 'handleShutdown']);

}

public function handleError($errno, $errstr, $errfile, $errline) {

$error = [

'type' => $this->getErrorType($errno),

'message' => $errstr,

'file' => $errfile,

'line' => $errline,

'trace' => debug\_backtrace(DEBUG\_BACKTRACE\_IGNORE\_ARGS)

];

$this->logError($error);

if ($this->environment === 'production') {

$this->notifyTeam($error);

$this->displayProductionError();

} else {

$this->displayError($error);

}

return true;

}

public function handleException($exception) {

$error = [

'type' => get\_class($exception),

'message' => $exception->getMessage(),

'file' => $exception->getFile(),

'line' => $exception->getLine(),

'trace' => $exception->getTrace()

];

$this->logError($error);

if ($this->environment === 'production') {

$this->notifyTeam($error);

$this->displayProductionError();

} else {

$this->displayError($error);

}

}

public function handleShutdown() {

$error = error\_get\_last();

if ($error !== null && in\_array($error['type'], [E\_ERROR, E\_PARSE, E\_CORE\_ERROR])) {

$this->handleError(

$error['type'],

$error['message'],

$error['file'],

$error['line']

);

}

}

private function logError($error) {

$this->logger->error(json\_encode($error));

}

private function notifyTeam($error) {

if ($this->isErrorCritical($error)) {

$this->notifier->sendAlert([

'subject' => 'Error Crítico en Producción',

'error' => $error,

'server' => $\_SERVER,

'timestamp' => date('Y-m-d H:i:s')

]);

}

}

private function isErrorCritical($error) {

return in\_array($error['type'], [

'E\_ERROR',

'E\_PARSE',

'E\_CORE\_ERROR',

'E\_COMPILE\_ERROR',

'E\_USER\_ERROR'

]);

}

private function getErrorType($errno) {

switch ($errno) {

case E\_ERROR:

return 'E\_ERROR';

case E\_WARNING:

return 'E\_WARNING';

case E\_PARSE:

return 'E\_PARSE';

case E\_NOTICE:

return 'E\_NOTICE';

default:

return 'UNKNOWN';

}

}

private function displayProductionError() {

header('HTTP/1.1 500 Internal Server Error');

include 'templates/error500.php';

exit;

}

private function displayError($error) {

header('HTTP/1.1 500 Internal Server Error');

include 'templates/error\_debug.php';

exit;

}

}

```

Explicación del Sistema de Errores:

- Manejo centralizado de errores

- Diferentes niveles de error

- Logging automático

- Notificaciones al equipo

- Modo debug y producción

EJEMPLOS DE USO

--------------

1. API REST:

```php

// Uso del API

$api = new ApiController();

$api->handleRequest('productos');

// Ejemplo de petición GET

curl -X GET http://api.burgerhouse.com/productos

// Ejemplo de petición POST

curl -X POST http://api.burgerhouse.com/productos \

-H "Content-Type: application/json" \

-d '{"nombre":"Hamburguesa Especial","precio":150}'

```

2. Caché:

```php

// Uso del sistema de caché

$cache = CacheManager::getInstance();

// Guardar datos

$cache->set('menu\_principal', $menu\_data, 3600);

// Recuperar datos

$menu = $cache->get('menu\_principal');

```

3. Eventos:

```php

// Uso del sistema de eventos

$events = EventManager::getInstance();

// Suscribirse a evento

$events->subscribe('nuevo\_pedido', function($data) {

// Notificar a cocina

// Actualizar dashboard

});

// Emitir evento

$events->emit('nuevo\_pedido', [

'id' => 123,

'items' => ['Hamburguesa', 'Papas']

]);

```

4. Analytics:

```php

// Uso del sistema de análisis

$analytics = new Analytics();

// Trackear evento

$analytics->trackEvent('venta\_completada', [

'monto' => 250,

'productos' => 3

]);

// Obtener métricas

$ventas = $analytics->getMetrics('ventas\_diarias', 'week');

```

5. Manejo de Errores:

```php

// Inicializar manejador de errores

$errorHandler = ErrorHandler::getInstance('development');

// El manejo de errores es automático

try {

// Código que puede fallar

throw new Exception('Error de prueba');

} catch (Exception $e) {

// El error será manejado automáticamente

}

```

CONSIDERACIONES DE SEGURIDAD

--------------------------

1. API REST:

- Autenticación mediante tokens

- Validación de datos de entrada

- Rate limiting

- CORS configurado correctamente

2. Caché:

- Datos sensibles no cacheados

- TTL apropiados

- Limpieza periódica

3. Eventos:

- Validación de suscriptores

- Timeout en callbacks

- Manejo de errores en broadcast

4. Analytics:

- Anonimización de datos

- Retención limitada

- Acceso controlado

5. Errores:

- No exponer información sensible

- Logs seguros

- Notificaciones controladas

OPTIMIZACIÓN Y RENDIMIENTO

------------------------

1. API:

- Caché de respuestas

- Compresión GZIP

- Paginación eficiente

2. Caché:

- Estrategias múltiples

- Precarga de datos

- Invalidación selectiva

3. Eventos:

- Procesamiento asíncrono

- Batch de notificaciones

- Conexiones persistentes

4. Analytics:

- Procesamiento en background

- Agregación de datos

- Índices optimizados

5. Errores:

- Logging asíncrono

- Agrupación de errores

- Rotación de logs