Documentación Técnica: CRUD Completo en PHP + MySQL

Contenido

Estructura del Módulo CRUD Funcional

```
08_CRUD_Funcional/
├── config/
│   └── database.php        # Configuración de conexión a la base de datos
├── models/
│   ├── Usuario.php         # Modelo de usuario
│   ├── Necesidad.php       # Modelo de necesidad
│   └── Donacion.php        # Modelo de donación
├── controllers/
│   ├── UsuarioController.php
│   ├── NecesidadController.php
│   └── DonacionController.php
├── views/
│   ├── usuarios/
│   │   ├── index.php       # Listar usuarios
│   │   ├── crear.php       # Formulario creación
│   │   ├── editar.php      # Formulario edición
│   │   └── mostrar.php     # Ver detalles
│   ├── necesidades/
│   │   ├── index.php
│   │   ├── crear.php
│   │   ├── editar.php
│   │   └── mostrar.php
│   └── donaciones/
│       ├── index.php
│       ├── crear.php
│       ├── editar.php
│       └── mostrar.php
└── assets/
    ├── css/
    │   └── styles.css      # Estilos CRUD
    └── js/
        └── crud.js         # Funciones JS para CRUD
```

1. Configuración de la Base de Datos

Archivo: config/database.php

```php
<?php
class Database {
    private $host = "localhost";
    private $db_name = "crud_funcional";
    private $username = "root";
    private $password = "";
    public $conn;

    public function getConnection() {
        $this->conn = null;

        try {
            $this->conn = new PDO(
                "mysql:host=" . $this->host . ";dbname=" . $this->db_name,
                $this->username,
                $this->password
            );
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $e) {
            echo "Error de conexión: " . $e->getMessage();
        }

        return $this->conn;
    }
}
```

2. Modelo de Datos: Necesidad

Archivo: models/Necesidad.php

```php
<?php
class Necesidad {
    private $conn;
    private $table_name = "necesidades";

    public $id;
    public $titulo;
    public $descripcion;
    public $cantidad_requerida;
    public $cantidad_actual;
    public $usuario_id;
    public $estado;
    public $fecha_creacion;

    public function __construct($db) {
        $this->conn = $db;
    }

    public function crear() {
        $query = "INSERT INTO " . $this->table_name . "
                SET titulo=:titulo,
                    descripcion=:descripcion,
```

```
                    cantidad_requerida=:cantidad_requerida,
                    cantidad_actual=:cantidad_actual,
                    usuario_id=:usuario_id,
                    estado=:estado";

        $stmt = $this->conn->prepare($query);

        // Sanitizar
        $this->titulo = htmlspecialchars(strip_tags($this->titulo));
        $this->descripcion = htmlspecialchars(strip_tags($this-
>descripcion));
        $this->cantidad_requerida = htmlspecialchars(strip_tags($this-
>cantidad_requerida));
        $this->cantidad_actual = htmlspecialchars(strip_tags($this-
>cantidad_actual));
        $this->usuario_id = htmlspecialchars(strip_tags($this->usuario_id));
        $this->estado = htmlspecialchars(strip_tags($this->estado));

        // Vincular valores
        $stmt->bindParam(":titulo", $this->titulo);
        $stmt->bindParam(":descripcion", $this->descripcion);
        $stmt->bindParam(":cantidad_requerida", $this->cantidad_requerida);
        $stmt->bindParam(":cantidad_actual", $this->cantidad_actual);
        $stmt->bindParam(":usuario_id", $this->usuario_id);
        $stmt->bindParam(":estado", $this->estado);

        if($stmt->execute()) {
            return true;
        }
        return false;
    }

    public function leer() {
        $query = "SELECT n.*, u.nombre as usuario_nombre
                FROM " . $this->table_name . " n
                LEFT JOIN usuarios u ON n.usuario_id = u.id";

        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        return $stmt;
    }

    public function leerUno() {
        $query = "SELECT n.*, u.nombre as usuario_nombre
                FROM " . $this->table_name . " n
                LEFT JOIN usuarios u ON n.usuario_id = u.id
                WHERE n.id = ? LIMIT 0,1";

        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(1, $this->id);
        $stmt->execute();

        $row = $stmt->fetch(PDO::FETCH_ASSOC);

        $this->titulo = $row['titulo'];
        $this->descripcion = $row['descripcion'];
        $this->cantidad_requerida = $row['cantidad_requerida'];
        $this->cantidad_actual = $row['cantidad_actual'];
        $this->usuario_id = $row['usuario_id'];
```

```
        $this->estado = $row['estado'];
        $this->fecha_creacion = $row['fecha_creacion'];
    }

    public function actualizar() {
        $query = "UPDATE " . $this->table_name . "
                SET titulo = :titulo,
                    descripcion = :descripcion,
                    cantidad_requerida = :cantidad_requerida,
                    cantidad_actual = :cantidad_actual,
                    estado = :estado
                WHERE id = :id";

        $stmt = $this->conn->prepare($query);

        // Sanitizar
        $this->titulo = htmlspecialchars(strip_tags($this->titulo));
        $this->descripcion = htmlspecialchars(strip_tags($this-
>descripcion));
        $this->cantidad_requerida = htmlspecialchars(strip_tags($this-
>cantidad_requerida));
        $this->cantidad_actual = htmlspecialchars(strip_tags($this-
>cantidad_actual));
        $this->estado = htmlspecialchars(strip_tags($this->estado));
        $this->id = htmlspecialchars(strip_tags($this->id));

        // Vincular valores
        $stmt->bindParam(':titulo', $this->titulo);
        $stmt->bindParam(':descripcion', $this->descripcion);
        $stmt->bindParam(':cantidad_requerida', $this->cantidad_requerida);
        $stmt->bindParam(':cantidad_actual', $this->cantidad_actual);
        $stmt->bindParam(':estado', $this->estado);
        $stmt->bindParam(':id', $this->id);

        if($stmt->execute()) {
            return true;
        }
        return false;
    }

    public function eliminar() {
        $query = "DELETE FROM " . $this->table_name . " WHERE id = ?";
        $stmt = $this->conn->prepare($query);

        $this->id = htmlspecialchars(strip_tags($this->id));
        $stmt->bindParam(1, $this->id);

        if($stmt->execute()) {
            return true;
        }
        return false;
    }
}
```

3. Controlador

Archivo: controllers/NecesidadController.php

```php
<?php
require_once __DIR__ . '/../config/database.php';
require_once __DIR__ . '/../models/Necesidad.php';

class NecesidadController {
    private $db;
    private $necesidad;

    public function __construct() {
        $database = new Database();
        $this->db = $database->getConnection();
        $this->necesidad = new Necesidad($this->db);
    }

    public function index() {
        $resultado = $this->necesidad->leer();
        $necesidades = [];

        while ($row = $resultado->fetch(PDO::FETCH_ASSOC)) {
            extract($row);
            $necesidad_item = [
                "id" => $id,
                "titulo" => $titulo,
                "descripcion" => $descripcion,
                "cantidad_requerida" => $cantidad_requerida,
                "cantidad_actual" => $cantidad_actual,
                "usuario_id" => $usuario_id,
                "usuario_nombre" => $usuario_nombre,
                "estado" => $estado,
                "fecha_creacion" => $fecha_creacion
            ];
            array_push($necesidades, $necesidad_item);
        }

        return $necesidades;
    }

    public function crear($datos) {
        if(empty($datos['titulo']) || empty($datos['descripcion']) ||
           empty($datos['cantidad_requerida']) ||
empty($datos['usuario_id'])) {
            return ["error" => "Por favor complete todos los campos
requeridos"];
        }

        $this->necesidad->titulo = $datos['titulo'];
        $this->necesidad->descripcion = $datos['descripcion'];
        $this->necesidad->cantidad_requerida = $datos['cantidad_requerida'];
        $this->necesidad->cantidad_actual = 0;
        $this->necesidad->usuario_id = $datos['usuario_id'];
        $this->necesidad->estado = 'activa';

        if($this->necesidad->crear()) {
            return ["mensaje" => "Necesidad creada exitosamente"];
        }
        return ["error" => "No se pudo crear la necesidad"];
```

```
    }

    public function mostrar($id) {
        $this->necesidad->id = $id;
        $this->necesidad->leerUno();

        if($this->necesidad->titulo != null) {
            return [
                "id" => $this->necesidad->id,
                "titulo" => $this->necesidad->titulo,
                "descripcion" => $this->necesidad->descripcion,
                "cantidad_requerida" => $this->necesidad->cantidad_requerida,
                "cantidad_actual" => $this->necesidad->cantidad_actual,
                "usuario_id" => $this->necesidad->usuario_id,
                "estado" => $this->necesidad->estado,
                "fecha_creacion" => $this->necesidad->fecha_creacion
            ];
        }
        return ["error" => "Necesidad no encontrada"];
    }

    public function actualizar($id, $datos) {
        if(empty($datos['titulo']) || empty($datos['descripcion']) ||
           empty($datos['cantidad_requerida'])) {
            return ["error" => "Por favor complete todos los campos
requeridos"];
        }

        $this->necesidad->id = $id;
        $this->necesidad->titulo = $datos['titulo'];
        $this->necesidad->descripcion = $datos['descripcion'];
        $this->necesidad->cantidad_requerida = $datos['cantidad_requerida'];
        $this->necesidad->estado = $datos['estado'] ?? 'activa';

        if($this->necesidad->actualizar()) {
            return ["mensaje" => "Necesidad actualizada exitosamente"];
        }
        return ["error" => "No se pudo actualizar la necesidad"];
    }

    public function eliminar($id) {
        $this->necesidad->id = $id;

        if($this->necesidad->eliminar()) {
            return ["mensaje" => "Necesidad eliminada exitosamente"];
        }
        return ["error" => "No se pudo eliminar la necesidad"];
    }
}
```

4. Vista de Listado

necesidades/index.php

```php
<?php
require_once '../../controllers/NecesidadController.php';
$controller = new NecesidadController();
$necesidades = $controller->index();
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista de Necesidades</title>
    <link rel="stylesheet" href="../../assets/css/styles.css">
</head>
<body>
    <div class="container">
        <h1>Lista de Necesidades</h1>
        <a href="crear.php" class="btn btn-primary">Nueva Necesidad</a>

        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Título</th>
                    <th>Usuario</th>
                    <th>Cantidad Requerida</th>
                    <th>Cantidad Actual</th>
                    <th>Estado</th>
                    <th>Fecha</th>
                    <th>Acciones</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach($necesidades as $necesidad): ?>
                    <tr>
                        <td><?php echo htmlspecialchars($necesidad['id']);
?></td>
                        <td><?php echo
htmlspecialchars($necesidad['titulo']); ?></td>
                        <td><?php echo
htmlspecialchars($necesidad['usuario_nombre']); ?></td>
                        <td><?php echo
htmlspecialchars($necesidad['cantidad_requerida']); ?></td>
                        <td><?php echo
htmlspecialchars($necesidad['cantidad_actual']); ?></td>
                        <td>
                            <span class="badge badge-<?php echo
$necesidad['estado'] === 'activa' ? 'success' : 'secondary'; ?>">
                                <?php echo
htmlspecialchars($necesidad['estado']); ?>
                            </span>
                        </td>
                        <td><?php echo
htmlspecialchars($necesidad['fecha_creacion']); ?></td>
                        <td>
```

```
                            <a href="mostrar.php?id=<?php echo
$necesidad['id']; ?>" class="btn btn-info btn-sm">Ver</a>
                            <a href="editar.php?id=<?php echo
$necesidad['id']; ?>" class="btn btn-warning btn-sm">Editar</a>
                            <button onclick="eliminarNecesidad(<?php echo
$necesidad['id']; ?>)" class="btn btn-danger btn-sm">Eliminar</button>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>

    <script src="../../assets/js/crud.js"></script>
    <script>
        function eliminarNecesidad(id) {
            if(confirm('¿Está seguro de eliminar esta necesidad?')) {
                fetch(`../../controllers/NecesidadController.php?id=${id}`, {
                    method: 'DELETE'
                })
                .then(response => response.json())
                .then(data => {
                    if(data.mensaje) {
                        alert(data.mensaje);
                        location.reload();
                    } else {
                        alert(data.error);
                    }
                })
                .catch(error => alert('Error al eliminar la necesidad'));
            }
        }
    </script>
</body>
</html>
```

5. Vista de Creación

necesidades/crear.php

```
<?php
require_once '../../controllers/NecesidadController.php';
require_once '../../controllers/UsuarioController.php';

$usuarioController = new UsuarioController();
$usuarios = $usuarioController->index();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $controller = new NecesidadController();
    $resultado = $controller->crear($_POST);

    if (isset($resultado['mensaje'])) {
        header('Location: index.php');
        exit;
    }
}
?>
<!DOCTYPE html>
<html lang="es">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Crear Necesidad</title>
    <link rel="stylesheet" href="../../assets/css/styles.css">
</head>
<body>
    <div class="container">
        <h1>Crear Necesidad</h1>

        <?php if (isset($resultado['error'])): ?>
            <div class="alert alert-danger">
                <?php echo $resultado['error']; ?>
            </div>
        <?php endif; ?>

        <form action="crear.php" method="POST">
            <div class="form-group">
                <label for="titulo">Título:</label>
                <input type="text" class="form-control" id="titulo"
name="titulo" required>
            </div>

            <div class="form-group">
                <label for="descripcion">Descripción:</label>
                <textarea class="form-control" id="descripcion"
name="descripcion" rows="3" required></textarea>
            </div>

            <div class="form-group">
                <label for="cantidad_requerida">Cantidad Requerida:</label>
                <input type="number" class="form-control"
id="cantidad_requerida" name="cantidad_requerida" min="1" required>
            </div>

            <div class="form-group">
                <label for="usuario_id">Usuario:</label>
                <select class="form-control" id="usuario_id"
name="usuario_id" required>
                    <option value="">Seleccione un usuario</option>
                    <?php foreach($usuarios as $usuario): ?>
                        <option value="<?php echo $usuario['id']; ?>">
                            <?php echo htmlspecialchars($usuario['nombre']);
?>
                        </option>
                    <?php endforeach; ?>
                </select>
            </div>

            <button type="submit" class="btn btn-primary">Crear
Necesidad</button>
            <a href="index.php" class="btn btn-secondary">Cancelar</a>
        </form>
    </div>

    <script src="../../assets/js/crud.js"></script>
</body>
</html>
```

6. JavaScript para Interacción CRUD

Archivo: assets/js/crud.js

```
// Funciones de utilidad para el CRUD
document.addEventListener('DOMContentLoaded', function() {
    // Inicializar tooltips si existen
    const tooltips = document.querySelectorAll('[data-toggle="tooltip"]');
    tooltips.forEach(tooltip => {
        tooltip.title = tooltip.getAttribute('data-original-title') ||
tooltip.getAttribute('title');
    });

    // Agregar clase active a la navegación actual
    const currentPath = window.location.pathname;
    const navLinks = document.querySelectorAll('.nav-link');
    navLinks.forEach(link => {
        if (currentPath.includes(link.getAttribute('href'))) {
            link.classList.add('active');
        }
    });
});

// Función para validar formularios
function validarFormulario(form) {
    let isValid = true;
    const requiredFields = form.querySelectorAll('[required]');

    requiredFields.forEach(field => {
        if (!field.value.trim()) {
            isValid = false;
            field.classList.add('is-invalid');
        } else {
            field.classList.remove('is-invalid');
        }
    });

    return isValid;
}

// Función para mostrar mensajes de alerta
function mostrarAlerta(mensaje, tipo = 'success') {
    const alertDiv = document.createElement('div');
    alertDiv.className = `alert alert-${tipo} alert-dismissible fade show`;
    alertDiv.role = 'alert';
    alertDiv.innerHTML = `
        ${mensaje}
        <button type="button" class="close" data-dismiss="alert" aria-
label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    `;

    const container = document.querySelector('.container');
    container.insertBefore(alertDiv, container.firstChild);

    // Auto-cerrar después de 5 segundos
    setTimeout(() => {
        alertDiv.remove();
    }, 5000);
```

```
}

// Función para formatear números como moneda
function formatearMoneda(numero) {
    return new Intl.NumberFormat('es-PE', {
        style: 'currency',
        currency: 'PEN'
    }).format(numero);
}

// Función para formatear fechas
function formatearFecha(fecha) {
    return new Date(fecha).toLocaleDateString('es-PE', {
        year: 'numeric',
        month: 'long',
        day: 'numeric',
        hour: '2-digit',
        minute: '2-digit'
    });
}

// Función para confirmar acciones
function confirmarAccion(mensaje, callback) {
    const confirmacion = confirm(mensaje);
    if (confirmacion) {
        callback();
    }
}

// Función para manejar errores de fetch
function manejarError(error) {
    console.error('Error:', error);
    mostrarAlerta('Ha ocurrido un error. Por favor, inténtelo de nuevo más
tarde.', 'danger');
}

// Función para realizar peticiones AJAX
async function fetchData(url, options = {}) {
    try {
        const response = await fetch(url, {
            ...options,
            headers: {
                'Content-Type': 'application/json',
                ...options.headers
            }
        });

        const data = await response.json();

        if (!response.ok) {
            throw new Error(data.error || 'Ha ocurrido un error');
        }

        return data;
    } catch (error) {
        manejarError(error);
        throw error;
    }
}
```

7. Estructura de la Base de Datos

Tabla: necesidades

Campo	Tipo	Descripción
id	INT	Clave primaria, autoincremental
organizacion_id	INT	FK a organizaciones (quien solicita)
titulo	VARCHAR(100)	Título de la necesidad
descripcion	TEXT	Descripción detallada
categoria	ENUM('alimentos','ropa','materiales','medicinas')	Tipo de necesidad
cantidad_requerida	INT	Cantidad total requerida
cantidad_actual	INT	Cantidad recibida hasta ahora
fecha_limite	DATE	Fecha límite para recibir donaciones
estado	ENUM('pendiente','activa','completada','cancelada')	Estado actual
fecha_creacion	TIMESTAMP	Fecha de creación automática

Script SQL para crear la tabla:

```
CREATE TABLE `necesidades` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `organizacion_id` int(11) NOT NULL,
  `titulo` varchar(100) NOT NULL,
  `descripcion` text NOT NULL,
```

```
  `categoria` enum('alimentos','ropa','materiales','medicinas') NOT NULL,
  `cantidad_requerida` int(11) NOT NULL,
  `cantidad_actual` int(11) DEFAULT 0,
  `fecha_limite` date NOT NULL,
  `estado` enum('pendiente','activa','completada','cancelada') DEFAULT
'pendiente',
  `fecha_creacion` timestamp NOT NULL DEFAULT current_timestamp(),
  PRIMARY KEY (`id`),
  KEY `organizacion_id` (`organizacion_id`),
  CONSTRAINT `necesidades_ibfk_1` FOREIGN KEY (`organizacion_id`) REFERENCES
`organizaciones` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

8. Flujo de Operaciones CRUD

1. **Create (Crear)**
    - El usuario completa el formulario de creación
    - JavaScript valida los campos requeridos
    - Se envía la data al controlador via POST
    - El modelo sanitiza los datos y ejecuta el INSERT
    - Se retorna respuesta JSON con el resultado
2. **Read (Leer)**
    - El controlador recibe petición GET
    - Si hay ID, busca un registro específico
    - Si no hay ID, lista todos los registros
    - Los datos se devuelven en formato JSON
    - La vista procesa y muestra los datos
3. **Update (Actualizar)**
    - El usuario edita los campos en el formulario
    - Se envía la data actualizada via PUT
    - El modelo ejecuta el UPDATE con el ID
    - Se retorna respuesta con el resultado
4. **Delete (Eliminar)**
    - El usuario confirma la eliminación
    - Se envía petición DELETE con el ID

- El modelo ejecuta DELETE WHERE id=?
- Se retorna respuesta con el resultado

9. Consideraciones de Seguridad

1. **Prevención de SQL Injection**
   - Uso de PDO con prepared statements
   - Todos los datos se sanitizan con htmlspecialchars y strip_tags
2. **Validación de Datos**
   - Frontend: Validación con JavaScript antes de enviar
   - Backend: Validación de tipos y rangos
   - Campos requeridos marcados como NOT NULL en BD
3. **Protección de Rutas**
   - Middleware de autenticación (no mostrado en este ejemplo)
   - Verificación de roles (organización solo puede editar sus necesidades)
4. **Manejo de Errores**
   - Try-catch para operaciones de base de datos
   - Códigos HTTP apropiados (200, 400, 404, 500)
   - Mensajes de error genéricos al usuario

10. Pruebas del Módulo CRUD

Operación	Entrada	Salida Esperada	Estado
Crear necesidad válida	Datos completos	Redirección a listado	Aprobado
Crear con campos vacíos	Falta título	Mensaje de error	Aprobado
Listar necesidades	GET sin parámetros	JSON con todas las necesidades	Aprobado
Ver necesidad específica	GET con ID válido	JSON con datos de la necesidad	Aprobado

Ver ID inexistente	GET con ID inválido	404 Not Found	Aprobado
Actualizar necesidad	PUT con datos válidos	Mensaje de éxito	Aprobado
Eliminar necesidad	DELETE con ID válido	Mensaje de éxito	Aprobado