EDUCACIÓN
IPSUMTEC
ISSN: 2594 - 2905

PDO EN LA GESTIÓN DE INFRAESTRUCTURA EDUCATIVA WEB: CASO DE ESTUDIO ATLAS DE PUEBLA

PDO IN THE MANAGEMENT OF WEB EDUCATIONAL INFRASTRUCTURE: CASE STUDY ATLAS DE PUEBLA

Martínez Ramírez Violeta, Martínez Rabanales Susana, Osorio Ramírez Efrén Armando, Luciano Machorro Teresa, Ara Mora Aldo Yahir

[1]Tecnológico Nacional de México/Instituto Tecnológico de Puebla, violeta.martinez@puebla.tecnm.mx, https://orcid.org/0000-0003-1518-786X
[2]Tecnológico Nacional de México/Instituto Tecnológico de Puebla, susana.martinez@puebla.tecnm.mx, https://orcid.org/0009-0002-8510-5380
[3]Tecnológico Nacional de México/Instituto Tecnológico de Puebla, efren.osorio@puebla.tecnm.mx, https://orcid.org/0000-0002-6902-1841
[4]Tecnológico Nacional de México/Instituto Tecnológico de Puebla, teresa.luciano@puebla.tecnm.mx, https://orcid.org/0000-0003-3979-9369
[5]Tecnológico Nacional de México/Instituto Tecnológico de Puebla, i20222166.19@puebla.tecnm.mx, https://orcid.org/0009-0008-1746-4633

https://doi.org/10.61117/ipsumtec.v8i2.421

Resumen -- El presente de investigación tecnológica, describe la metodología de desarrollo empleada en la creación del sistema "Atlas de Puebla de la Infraestructura Educativa", una aplicación web para la gestión y consulta de información sobre planteles educativos en el estado de Puebla. Se detalla un enfoque paso a paso en el proceso de codificación, resaltando las fases clave, las tecnologías utilizadas y las decisiones metodológicas que garantizaron la funcionalidad y escalabilidad del proyecto. El objetivo es ofrecer una visión clara de cómo se construyó la aplicación, desde la concepción inicial de los módulos hasta la implementación de sus características principales.

Palabras Clave: software, Metodología Ágil, PDO, Gestión de planteles, web.

Abstract -- The present technological research describes the development methodology used in creating the system "Atlas de Puebla de la Infraestructura Educativa", a web application for managing and consulting information about educational institutions in the state of Puebla. A step-by-step approach in the coding process is detailed, highlighting the key phases, the technologies used, and the methodological decisions that ensured the functionality and scalability of the project. The aim is to provide a clear view of how the application was built, from the initial conception of the modules to the implementation of its main features.

Key words – software, Agile Methodology, PDO, Management of facilities, web.

INTRODUCCIÓN

Información publicada por el diario EL UNIVERSAL PUEBLA (1) en su versión digital, afirma que más del 80% de planteles educativos en el estado, son escuelas públicas.

Según datos del Sistema Nacional de Información Estadística Mexicana refiere que hay 14 mil 494 escuelas en el sistema educativo de la entidad, de las cuales 11 mil 961. Así mismo, destaca que el estado de Puebla es el segundo estado con mayor demanda educativa tiene, seguido de la Ciudad de México, recibiendo anualmente una gran cantidad de estudiantes de otros estados principalmente del centro, sur y sureste mexicano.

La SEP del estado de Puebla para el año 2025, tiene registrados a más de 2 millones de estudiantes en todos los niveles educativos.

La información relativa a la infraestructura de todo plantel público en el estado de Puebla es concentrada en el Comité de Administración Poblano para la construcción de Espacios Educativos, un órgano descentralizado (CAPCEE). Como órgano descentralizado y dependiente del gobierno estatal es el encargado de formular, aplicar, ejecutar y vigila la construcción rehabilitación, equipamiento y mantenimiento de la Infraestructura física educativa y especial datos de CPACEE (2). Se atienden los muebles e inmuebles destinados a la educación impartida por el Estado. En lo que va de este sexenio gubernamental se tiene el proceso obras educativas en 32 planteles con una inversión de 68.5 millones de pesos enfocados a construcción y rehabilitación de aulas y sanitarios e instalación de paneles solares y captadores pluviales publica datos del capcee.puebla.gob.mx.

Estado del Arte

A continuación, se da un recorrido a algunos trabajos publicados como trabajos académicos de diversas universidades de Hispanoamérica relacionadas con el uso de PHP y su extensión PDO.

El primero de ellos titulado “Segurança em Sistemas Web: Uma Análise da Extensão PDO como forma de Proteção de Sistemas PHP Contra Injeções de SQL” (3), Brasil muestra la exitosa implementación dentro de un sistema web utilizando PDO para evitar la inyección indebida y externa de consultas en SQL, con la intención de proteger los datos contra el acceso no autorizado, especialmente por parte de usuarios maliciosos, cuyas acciones pueden causar daños significativos a personas y organizaciones. Como resultado se obtuvo mediante las pruebas ejecutadas en el sistema web de inicio de sesión, validaron que el uso de sentencias preparadas y la parametrización de la información introducida por el usuario en las consultas a bases de datos, son altamente eficaces en la protección de sistemas contra las inyecciones SQL. Según las pruebas hechas con inyecciones manuales por medio de un campo de formulario o con una herramienta especializada, PDO demostró ser 100 % seguro contra ataques de inyección SQL (Lenguaje de Consulta Estructurado), siempre que se utilice correctamente se mantiene la integridad de los datos. Sin embargo, si los comandos son inyectados por el usuario y que no se ejecutaron inicialmente, pero que se almacenaron en la base de datos pueden crear alta vulnerabilidad al sistema llamados “inyeciones de segundo orden”. Por lo tanto, se sugiere aplicar parametrización en ellas. Resumiendo, PDO proprociona una capa de seguridad de gran desempeño en los sistemas web.

Otro trabajo académico nombrado “Comparativo entre API MySQL e API PDO” (4), resalta la vulnerabilidad de las aplicaciones web difundidas en internet, por lo tanto, para disminuir considerablemente ataques inyectados frecuentemente a través de la barra de direcciones del navegador o formularios HTML, con la implementación de una API más actualizada y parametrizada denominada PDO. Como resultado de la comparativa entre API MySQL y PDO, por medio de la cual se automatizó una prueba de penetración utilizando una herramienta de escaneo de fragilidades web, el programa ZAP (Proxy de Ataque Zed), para verificar si las aplicaciones habían sufrido ataques de inyección SQL, se demostró que PDO logró el 100% de seguridad contra inyecciones SQL.

El trabajo publicado como “Sistema de subastas inglesa desarrollado con PHP POO PDO + MYSQL” (5) en el Ecuador, dentro de sus requerimientos automatizados obtenidos destaca su aporte en el rendimiento de gestión de la información mediante un sistema web que efectúa y actualiza los registros de una forma segura gracias a las excepciones PDO. Además, ofreció mayor seguridad dentro del sistema conservando la integridad de datos en el momento de la otorgación de roles y privilegios al usuario dentro de la aplicación. Por último, pero no menos importante, proporciona un esquema de fiabilidad en donde la aplicación se mantiene estable y asegura en las transacciones realizadas por usuarios y administradores 100% de confianza durante todo el tiempo vida útil.

En Bogotá Colombia el trabajo “Diseño e Implementación de un Sistema POS, con Módulo de Gestión de Inventario de Productos para Clientes y Perfiles de Usuario, Aplicando Metodología RUP” publicado en (6), opta para el desarrollo del código que incluya el patrón MVC (Modelo Vista Controlador), creado en tres capas o niveles, donde la de importancia preponderante es la capa de seguridad más que la de diseño dentro de la aplicación web creada como imagen corporativa y ventas en línea. Sobre todos los gestores del sistema se implementa la conexión a la base de datos con MySQL utilizando la clase PHP PDO para asegurar la conexión e integridad de los datos de forma exitosa. Las transacciones de punto de venta en tiempo real ameritan proteger la información suministrada por el cliente sobre su propia información y de la compra digital como portal en la web.

Publicación del 2025 en Brasil, con el proyecto titulado “SQL Injection: Entendendo e evitando” (7) permite conocer las comparativas relacionadas a la seguridad de gestión de información en aplicativos webs, se consideran tres estrategias: cifrado de datos, validación de entradas y el uso de PHP Data Objects (PDO). La primera de ellas, con cifrado protege la información al convertirla en formatos ilegibles sin la clave correcta. La segunda estrategia aplica una validación de entradas bloqueando caracteres peligrosos mediante listas blancas o negras. En la última, PDO ofreció una interfaz segura para consultas SQL, utilizando parámetros que evitan la inyección de código malicioso. Los resultados demostraron que, si bien el cifrado y la validación reducen los riesgos de manera significativa, la estrategia más efectiva fue implementar PDO, gracias a que encapsula las consultas y evita actualizaciones no autorizadas. La mejor de las prácticas computacionales entonces será la combinación de las técnicas para garantizar la seguridad de los sistemas que interactúan con bases de datos en la web.

Problemática.

El sistema "ATLAS DE PUEBLA" surge de la necesidad de centralizar, organizar y facilitar el acceso a información crucial sobre los planteles educativos en el estado de Puebla, México. Previo a esta iniciativa, la gestión de datos de infraestructura, servicios y estado operativo de las escuelas se encontraba dispersa, dificultando la toma de decisiones informadas y la supervisión eficiente. Este informe documenta la metodología de desarrollo adoptada para construir esta aplicación web, poniendo énfasis en el proceso iterativo de codificación y las prácticas implementadas para asegurar la calidad y el cumplimiento de los requisitos. El proyecto se enmarca en un esfuerzo por modernizar y optimizar la administración educativa a través de herramientas tecnológicas.

Objetivo general

Creación de un aplicativo web para la gestión de los espacios educativos de los planteles públicos en el estado de Puebla utilizando PHP Data Object.

Justificación

La selección de las tecnologías en el software “Atlas de Puebla “, se inclinó a PHP Data Object (PDO), por su alta capacidad de brindar seguridad en la base de datos, adaptabilidad y manejo de errores.

Es una fuente inteligente en las tablas de las bases de datos relacionadas, protegiendo eficazmente de la inyección SQL utilizando sentencias preparadas para ser separadas esas instrucciones de los datos. Además, está creado para trabajar diversas bases de datos y emitir mensajes si algo indebido sucede.

Es por ello, que se distingue por ser la más recomendada y popular para interactuar con base de datos en PHP.

Por lo anterior expuesto, se justifica utilizar PDO en la seguridad del diseño y almacenamiento de la base de datos en el sistema “Atlas Puebla”.

PHP (Hypertext Preprocessor)

Es una de las herramientas más socorridas en el mundo de deseño y creación de aplicaciones en la web, su característica principal radica en la fortaleza como lenguaje de programación que ofrece varias opciones para la interacción con bases de datos; en él predomina sus vastas funcionalidades para la creación del Back End robustas.

Una de las herramientas flexible y potente es la extensión de PDO quien favorece la portabilidad y seguridad de la información.

Su objetivo es crear una forma coherente y segura de acceder a las bases de datos sin importar el tipo Sistema de Gestor de la BD (SGBD) (8).

Acerca de PDO

El sitio de PHP (9) define a PHP Data Object (PDO) como una extensión que permite a acceder a diferentes bases de datos de las más populares entre los desarrolladores como: MySQL/MariaBD, PostgreSQL, Oracle, MS SQL Server, SQLite, Firebird, DB2, Informix, etc.) estableciendo la portabilidad entre ellas.

Sin embargo, no evalúa la corrección de consultas SQL, pero sí ofrece seguridad mediante consultas preparadas.

Formalmente PDO es una biblioteca de PHP que trabaja con programas orientados objetos.

Elección de la forma del manejo de errores en PDO: 1. Sin interrumpir la ejecución del programa, los errores se guardan y se deberán de verificar qué tipo de error fue comprobarlo al momento de hacer una consultar. 2. Se guardan los errores y se emite un aviso de estos. 3.En esta opción, se maneja la excepción del error por el mismo programa.

Las consultas cuyo origen de datos proviene de formularios se deberán utilizar consultas preparadas para evitar ataques de los que el método QUERY no protege. Caso contrario, entonces solo requerirá de una sentencia condicional if…else (imagen 1). Devuelva (imagen 2) o no registros (imagen 3).

Imagen 1. *Consulta sin datos desde formularios*.

```
// EJECUCIÓN DE UNA CONSULTA SIN DATOS PROVENIENTES DE UN FORMULARIO

$pdo->query("consulta");
```

Fuente. *Elaboración propia* (2025).

Imagen 2. *Consulta que no retorna registro*.

```
// EJECUCIÓN DE UNA CONSULTA QUE NO VA A DEVOLVER REGISTROS

if (!$pdo->query("consulta")) {
    ...//Si la consulta falla
} else {
    ...//Si la consulta se ejecuta correctamente
}
```

Fuente. *Elaboración propia* (2025).

Imagen 3. *Consulta que si retorna registro*.

```
// EJECUCIÓN DE UNA CONSULTA QUE PUEDE DEVOLVER REGISTROS

$resultado = $pdo->query("consulta");
if (!$resultado) {
    ...//Si la consulta falla
} else {
    ...//Si la consulta se ejecuta correctamente
}
```

Fuente. *Elaboración propia* (2025).

El término “Consulta preparada”, según el portal manuais.iessanclemente.net (2018) se define como una sentencia SQL pre-compilada que puede ser ejecutada múltiples veces con la única condición de enviar sus datos a un servidor.

DESARROLLO

La metodología de desarrollo elegida para "ATLAS DE PUEBLA" fue un enfoque híbrido, combinando elementos de metodologías ágiles (como Scrum o Kanban para la gestión de tareas e iteraciones) con una planificación inicial estructurada para la arquitectura de la base de datos y la definición de módulos. Esto permitió una flexibilidad para adaptarse a los requisitos cambiantes, mientras se mantenía una visión clara de la estructura fundamental del sistema.

Las fases generales incluyeron:

• Planificación y Análisis de Requisitos: Definición de funcionalidades clave (gestión de planteles, búsqueda, reportes, usuarios), identificación de entidades de datos y sus relaciones.

• Diseño de Base de Datos: Creación del esquema relacional.

• Diseño de Arquitectura de la Aplicación: Determinación de la estructura de archivos, módulos y patrones de diseño (MVC implícito).

• Implementación (Codificación Iterativa): Desarrollo progresivo de las funcionalidades.

• Pruebas y Depuración: Verificación constante del funcionamiento y corrección de errores.

• Despliegue y Mantenimiento: Puesta en marcha y soporte continuo.

El foco de este informe es la fase de Implementación (Codificación Iterativa), que se desglosa a continuación.

Paso a Paso de la Codificación

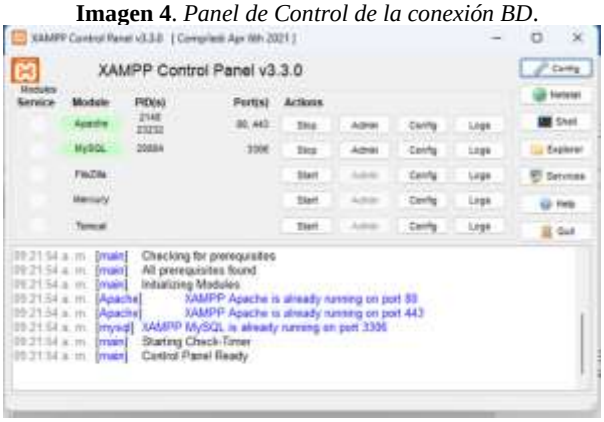
La codificación del "ATLAS DE PUEBLA" se llevó a cabo de manera modular y por capas, siguiendo un principio de desarrollo incremental.

Configuración del Entorno y Conexión a la Base de Datos:

El primer paso en la codificación fue establecer el entorno de desarrollo y la conectividad. Se utilizó XAMPP

(Apache, MySQL, PHP) como servidor local. Ver imagen 4.

Imagen 4. *Panel de Control de la conexión BD*.



Fuente. *Elaboración propia* (2025).

El archivo database.php se creó para encapsular la configuración de la conexión a la base de datos (servidor, usuario, contraseña, nombre de la BD y codificación UTF-8) utilizando PDO (PHP Data Objects). PDO fue elegido por su robustez, seguridad contra inyección SQL a través de sentencias preparadas y manejo de errores. Se definió BASE_URL para asegurar la correcta navegación entre páginas. Además, auth_check.php implementó un sistema de autenticación basado en sesiones PHP, incluyéndose en todas las páginas restringidas para verificar el inicio de sesión y la expiración por inactividad.

Código Destacado: Conexión PDO (database.php)

Este fragmento esencial muestra cómo se establece la conexión segura a la base de datos, definiendo constantes y opciones para PDO.

PHP

```
// Fragmento de database.php
<?php
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'atlas_puebla_db');
define('DB_CHARSET', 'utf8mb4');

$options = [
   PDO::ATTR_ERRMODE                 => PDO::ERRMODE_EXCEPTION,
   PDO::ATTR_DEFAULT_FETCH_MODE      => PDO::FETCH_ASSOC,
   PDO::ATTR_EMULATE_PREPARES => false,
];

$dsn = "mysql:host=" . DB_SERVER . ";dbname=" . DB_NAME . ";charset=" . DB_CHARSET;

try {
   $pdo = new PDO($dsn, DB_USERNAME, DB_PASSWORD, $options);
} catch (\PDOException $e) {
   error_log("Error de conexión PDO: " . $e->getMessage());
   die("Error de conexión con el servidor. Por favor, intente más tarde.");
}
// La definición de BASE_URL se asume que ocurre aquí o en un archivo similar [MODIFICACIÓN: COMENTARIO AÑADIDO]
// define('BASE_URL', $protocol . $_SERVER['HTTP_HOST'] . '/atlas_puebla/');
?>
```

.

Diseño e Implementación del Esquema de Base de Datos:

Antes de cualquier línea de HTML visible, la base de datos fue diseñada meticulosamente para soportar todas las funcionalidades. El script atlas_puebla_db.sql definió la estructura de todas las tablas, incluyendo planteles (información central de cada escuela), usuarios (gestión de cuentas), roles (niveles de acceso), y tablas de apoyo como municipios, localidades, y cordes. También se crearon tablas de detalle (detalle_hidrosanitario, detalle_proteccion_civil, detalle_servicios) con relaciones uno a uno con planteles, y tablas para la gestión de archivos y material visual (archivos_plantel, galeria_fotos). Se definieron claves primarias y foráneas con acciones ON DELETE CASCADE o SET NULL para mantener la integridad referencial y mejorar la consistencia de los datos.

Código Destacado: Estructura de Tabla planteles y detalle_proteccion_civil (atlas_puebla_db.sql)

Estos fragmentos ilustran la tabla central de planteles y una de sus tablas de detalle, detalle_proteccion_civil, mostrando las relaciones clave que aseguran la coherencia de la información.

SQL

```
-- Fragmento de atlas_puebla_db.sql
CREATE TABLE `planteles` (
  `cct` varchar(20) NOT NULL,
  `nombre_escuela` varchar(255) NOT NULL,
  `id_municipio` int(11) DEFAULT NULL,
  `id_localidad` int(11) DEFAULT NULL,
  `id_corde` int(11) DEFAULT NULL,
  `id_director_asignado` int(11) DEFAULT NULL,
  -- ... otras columnas ...
  PRIMARY KEY (`cct`),
  KEY `id_municipio` (`id_municipio`),
  KEY `id_director_asignado` (`id_director_asignado`),
  CONSTRAINT `planteles_ibfk_1` FOREIGN KEY (`id_municipio`) REFERENCES `municipios` (`id_municipio`),
  CONSTRAINT `planteles_ibfk_4` FOREIGN KEY (`id_director_asignado`) REFERENCES `usuarios` (`id_usuario`) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Ejemplo de tabla de detalle
CREATE TABLE `detalle_proteccion_civil` (
```

```
 `id_pc_detalle` int(11) NOT NULL AUTO_INCREMENT,
 `cct` varchar(20) NOT NULL UNIQUE, -- Relación uno a uno
 `alarma_sismica` tinyint(1) DEFAULT 0,
 -- ... otras columnas ...
 PRIMARY KEY (`id_pc_detalle`),
 CONSTRAINT `detalle_proteccion_civil_ibfk_1` FOREIGN KEY (`cct`) REFERENCES `planteles` (`cct`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Desarrollo del Módulo de Autenticación y Acceso:

El punto de entrada del sistema se construyó con un enfoque de seguridad. La página index.php redirige al dashboard si el usuario ya está autenticado o muestra el formulario de login. El archivo login.php procesa las credenciales utilizando password_verify() para comparar el hash de la contraseña de forma segura, y maneja los estados de cuenta (activo/inactivo) y errores de credenciales. Finalmente, logout.php destruye la sesión garantizando un cierre seguro.

Código Destacado: Verificación de Contraseña y Sesión (login.php)

Este fragmento ilustra cómo se utiliza password_verify() para la validación segura de las credenciales y cómo se manejan las sesiones PHP, incluyendo la regeneración del ID de sesión para mayor seguridad. Ver imagen 5.

Imagen 5. *Autenticación para el acceso*.

Fuente. *Elaboración propia* (2025).

```
PHP
// Fragmento de login.php
if ($user && password_verify($password, $user['password_hash'])) { // VERIFICACIÓN SEGURA DE CONTRASEÑA
   if ($user['estatus'] === 'ACTIVO') {
      session_regenerate_id(true); // Regenera el ID de sesión por seguridad
      $_SESSION["loggedin"] = true;
      $_SESSION["id_usuario"] = $user['id_usuario'];
      $_SESSION["nombre_completo"] = $user['nombre_completo'];
      $_SESSION["id_rol"] = $user['id_rol'];
      // Actualizar última conexión en BD
      $update_stmt = $pdo->prepare("UPDATE usuarios SET ultima_conexion = CURRENT_TIMESTAMP WHERE id_usuario = :id");
      $update_stmt->execute(['id' => $user['id_usuario']]);
      header("location: " . BASE_URL . "dashboard.php");
      exit;
   } else {
      $error = "La cuenta está inactiva.";
   }
} else {
   $error = "Usuario o contraseña incorrectos."; // Mensaje genérico por seguridad
}
```

Implementación del Dashboard y Perfil de Usuario:

Una vez el acceso estaba funcional, se desarrolló la interfaz principal. El dashboard.php actúa como punto de navegación central, mostrando opciones de acuerdo con el id_rol del usuario. Los archivos mi_perfil.php y mi_perfil_guardar.php permiten a los usuarios ver y actualizar su información personal y cambiar su contraseña, incluyendo la validación de la contraseña actual y el hashing de la nueva con password_hash().

Construcción del Módulo de Gestión de Planteles:

Este es el corazón del sistema, donde se registra y consulta la información de las escuelas. Ver imagen 6.

Imagen 6. *Registro y consulta de información de los planteles educativos del estado*.

Fuente. Elaboración propia (2025).

planteles_lista.php muestra un listado paginado de planteles, con la consulta SQL adaptándose según el rol del usuario (administrador ve todos, directores/capturistas ven solo los asignados), e incluye búsqueda por texto.

plantel_ficha_form.php y plantel_ficha_guardar.php implementan el formulario para crear o editar la ficha básica de un plantel, con una lógica de permisos crucial que permite a los directores registrar o editar sus propios planteles y a los administradores modificar cualquiera, incluyendo el estatus; maneja la inserción (INSERT) y actualización (UPDATE) de datos. La página

plantel_detalle.php centraliza toda la información de un plantel específico, organizando los datos en pestañas (Ficha Base, Espacios, Servicios, Protección Civil, Archivos, Galerías) mediante JavaScript para una mejor experiencia de usuario, y cargando datos con múltiples consultas individuales por sección. Finalmente, plantel_eliminar.php permite a los administradores eliminar un plantel, con un control para evitar la eliminación si existen registros asociados (debido a restricciones de claves foráneas) y maneja transacciones PDO para asegurar la atomicidad. Ver imagen 7.

Imagen 7. *Proceso de gestión para la integridad de la información*.

Fuente. *Elaboración propia* (2025).

.

Desarrollo de Sub-Módulos de Detalle (Espacios, Servicios, Protección Civil, Archivos, Galerías):

Cada sección del plantel_detalle.php tiene su propia lógica de gestión.

espacio_guardar.php y espacio_eliminar.php gestionan la adición y eliminación de registros en espacios_areas.

servicios_form.php y servicios_guardar.php permiten editar y guardar la información de detalle_hidrosanitario y detalle_servicios, utilizando ON DUPLICATE KEY UPDATE en el SQL para manejar eficientemente la inserción o actualización de estos registros relacionados. De manera similar, proteccion_civil_form.php y proteccion_civil_guardar.php manejan la información en detalle_proteccion_civil. Los archivos archivo_guardar.php y archivo_eliminar.php implementan la subida y eliminación de archivos, incluyendo el manejo de nombres seguros y rutas de almacenamiento (uploads/archivos/), borrando el archivo físico al eliminar el registro. Finalmente, galeria_guardar.php y foto_eliminar.php gestionan la subida de múltiples imágenes y su eliminación, utilizando el directorio uploads/galeria/. Ver imagen 8.

Imagen 8. *Proceso de gestión de la galería gráfica*.

.

Fuente. *Elaboración propia* (2025).

Código Destacado: Transacción y UPSERT para Detalles de Plantel (servicios_guardar.php)

Este ejemplo de código demuestra el uso de transacciones de base de datos (beginTransaction(), commit(), rollBack()) para asegurar que las operaciones de inserción/actualización de múltiples tablas relacionadas sean atómicas. La cláusula ON DUPLICATE KEY UPDATE permite insertar un nuevo registro si no existe, o actualizarlo si ya lo hace, simplificando la lógica de "upsert".

```
PHP
// Fragmento de servicios_guardar.php (ejemplo de transacción y UPSERT)
try {
    $pdo->beginTransaction(); // INICIA LA TRANSACCIÓN

    // Para detalle_hidrosanitario: INSERTA O ACTUALIZA DATOS HIDROSANITARIOS
    $sql_hidro = "INSERT INTO detalle_hidrosanitario (cct, fuente_agua, almacenamiento_agua, tipo_drenaje, sanitarios_hombres_wc, sanitarios_hombres_lavabos, sanitarios_mujeres_wc, sanitarios_mujeres_lavabos, observaciones)
                  VALUES (:cct, :fuente_agua, :almacenamiento_agua, :tipo_drenaje, :sanitarios_hombres_wc, :sanitarios_hombres_lavabos, :sanitarios_mujeres_wc, :sanitarios_mujeres_lavabos, :observaciones)
                  ON DUPLICATE KEY UPDATE fuente_agua=VALUES(fuente_agua), almacenamiento_agua=VALUES(almacenamiento_agua), tipo_drenaje=VALUES(tipo_drenaje), sanitarios_hombres_wc=VALUES(sanitarios_hombres_wc), sanitarios_hombres_lavabos=VALUES(sanitarios_hombres_lavabos), sanitarios_mujeres_wc=VALUES(sanitarios_mujeres_wc), sanitarios_mujeres_lavabos=VALUES(sanitarios_mujeres_lavabos), observaciones=VALUES(observaciones)";
    $pdo->prepare($sql_hidro)->execute($data_hidro);

    // Para detalle_servicios: INSERTA O ACTUALIZA DATOS DE SERVICIOS BÁSICOS
```

```
    $sql_servicios = "INSERT INTO detalle_servicios (cct, electricidad_contrato, telefonia_fija, internet_acceso, gas_tipo, internet_tipo, observaciones)
                      VALUES (:cct, :electricidad_contrato, :telefonia_fija, :internet_acceso, :gas_tipo, :internet_tipo, :observaciones)
                      ON DUPLICATE KEY UPDATE electricidad_contrato=VALUES(electricidad_contrato), telefonia_fija=VALUES(telefonia_fija), internet_acceso=VALUES(internet_acceso), gas_tipo=VALUES(gas_tipo), internet_tipo=VALUES(internet_tipo), observaciones=VALUES(observaciones)";
    $pdo->prepare($sql_servicios)->execute($data_servicios);

    $pdo->commit(); // CONFIRMA LA TRANSACCIÓN SI AMBAS OPERACIONES FUERON EXITOSAS
    // ... (redirección exitosa) ...
} catch (PDOException $e) {
    $pdo->rollBack(); // DESHACE LA TRANSACCIÓN EN CASO DE ERROR
    error_log("Error al guardar servicios: " . $e->getMessage());
    // ... (redirección con mensaje de error) ...
}
```

Implementación del Buscador Avanzado:

Una funcionalidad central para la consulta de datos. buscador.php integra todos los filtros (texto, CORDE, municipio, nivel, sostenimiento, alarma sísmica). La construcción dinámica de la cláusula WHERE ($sql_where_parts) y el uso de sentencias preparadas son cruciales. Se abordó específicamente el manejo de la búsqueda insensible a mayúsculas/minúsculas para el texto (LOWER()) y la inclusión de registros NULL para el filtro de alarma sísmica, lo que mejora la precisión de los resultados. La paginación se integra con los filtros activos. Ver imagen 9.

Imagen 9. *Proceso de búsqueda avanzada*.

Fuente. *Elaboración propia* (2025).

Código Destacado: Lógica de Filtro de Texto y Alarma Sísmica (buscador.php)

Este fragmento ilustra la construcción dinámica de la cláusula WHERE para filtros múltiples. Se destaca la implementación de la búsqueda de texto insensible a mayúsculas/minúsculas usando LOWER () en la base de datos, y el manejo específico del filtro "Alarma Sísmica" para incluir tanto los valores '0' como los NULL, mejorando la precisión de los resultados.

PHP

```
// Fragmento de buscador.php (parte de la lógica de filtros)
// Filtro por texto (con LOWER para insensibilidad a mayúsculas/minúsculas)
if (!empty($filtros['busqueda'])) {
    $sql_where_parts[] = "(LOWER(p.cct) LIKE LOWER(:busqueda) OR LOWER(p.nombre_escuela) LIKE LOWER(:busqueda))";
    $params['busqueda'] = "%{$filtros['busqueda']}%";
}

// JOINS SQL
$sql_joins = "FROM planteles p LEFT JOIN municipios m ON p.id_municipio = m.id_municipio LEFT JOIN usuarios u ON p.id_director_asignado = u.id_usuario ";

// Filtro por Alarma Sísmica (maneja 0 y NULL para incluir planteles sin registro específico)
if ($filtros['alarma_sismica'] !== '') {
    $sql_joins .= "LEFT JOIN detalle_proteccion_civil pc ON p.cct = pc.cct ";
    if ($filtros['alarma_sismica'] === '1') {
        $sql_where_parts[] = "pc.alarma_sismica = 1";
    } elseif ($filtros['alarma_sismica'] === '0') {
        $sql_where_parts[] = "(pc.alarma_sismica = 0 OR pc.alarma_sismica IS NULL)";
    }
}

$sql_where = "";
if (!empty($sql_where_parts)) {
    $sql_where = "WHERE " . implode(" AND ", $sql_where_parts);
}
// ... luego la preparación y ejecución de la consulta
```

3.8. Desarrollo de Módulos de Reportes y Supervisión:

Para la extracción y visualización de datos agregados, reportes_panel.php ofrece un menú a diferentes informes. Informes como:

1. reporte_planteles_municipio.php
2. reporte_estatus_planteles.php, y
3. reporte_infra_espacios.php

los cuales generan datos específicos con COUNT() y GROUP BY, incluyendo gráficas generadas con Chart.js para visualización.

exportar_csv.php permite exportar los datos a CSV. Además, los archivos supervision_panel.php, supervision_corde_detalle.php, y supervision_escuela_detalle.php proporcionan vistas de resumen y detalle sobre el avance de la captura de información a nivel de CORDE y de plantel, utilizando porcentajes de avance para el monitoreo.

Gestión de Usuarios (Administrador):
Este módulo es para la administración completa de usuarios.
usuarios_lista.php lista todos los usuarios con sus roles y estatus, permitiendo acciones de edición y eliminación. Los archivos usuario_form.php y usuario_guardar.php contienen los formularios y la lógica para crear nuevos usuarios o editar existentes, incluyendo el hashing de contraseñas y la validación de correos electrónicos únicos.
usuario_eliminar.php permite al administrador eliminar usuarios, con validaciones para evitar la autoeliminación y la eliminación de usuarios con dependencias en otras tablas (manejo de claves foráneas). Ver imagen 10.

Imagen 10. *Proceso de actualización de usuarios según dependencia*.

Fuente. *Elaboración propia* (2025).

DISCUSIÓN Y ANÁLISIS DE RESULTADOS

Consideraciones Técnicas y Buenas Prácticas
Durante el proceso de codificación, se adhirieron a varias buenas prácticas:
• Modularización: División del código en archivos PHP funcionales (database.php, auth_check.php, _guardar.php, _eliminar.php, etc.) para mejorar la mantenibilidad y reusabilidad.
• Seguridad: Uso extensivo de PDO con sentencias preparadas para prevenir inyecciones SQL. Hashing de contraseñas (password_hash()) para la protección de credenciales. Validaciones básicas de entrada (trim, htmlspecialchars).
• Manejo de Errores: Implementación de bloques try-catch para capturar excepciones de PDO y registrar errores críticos (error_log()), lo que facilita la depuración y mejora la robustez de la aplicación.
• Experiencia de Usuario: Uso de CSS para un diseño responsivo y consistente, y JavaScript para mejoras interactivas (ej. mostrar/ocultar filtros, paginación).
• Consistencia de Datos: El diseño de la base de datos con claves foráneas y la lógica ON DUPLICATE KEY UPDATE en PHP para las secciones de detalle aseguran la integridad y unicidad de los datos.

Discusión.
En relación con el objetivo del CAPCEE de reducir gastos de operación en los sistemas de información de la unidad, es importante atender el tema de seguridad al mismo tiempo que la reducción de costos en su implementación y mantenimiento de igual forma que el trabajo “Sistema web en la gestión de mantenimiento del taller automotriz VIP2CARS en el año 2024” en Ecuador (10), el cual muestra el desarrollo de un sistema digital basado en PHP, MySQL y JavaScript, diseñado bajo el modelo MVC (Modelo Vista Controlador) para optimizar el flujo de trabajo, reducir errores y mejorar la trazabilidad de los servicios. Donde su principal objetivo es eliminar errores creado con herramientas OSS (Open Source) para el Back End como: PHP, MySQLi y PDO que aseguran su atomicidad, consistencia, aislamiento y durabilidad (ACID) de la información. Todo ello, redujo notablemente la inversión financiera, aumenta la rentabilidad y eficacia operativa del negocio.
Las ventajas de elegir como mejor opción a PHP y las extensiones de PDO como herramienta de desarrollo y seguridad va respaldada con otros sistemas exitosamente implementados, como el presentado como solución a mercados artesanales en Chile con el proyecto “Aplicación web de sistema de inventario para tiendas de artesanía del mercado de Chillán” (11), aplicación web basada con la estructura de YII Framework como entorno de desarrollo, el cual provee seguridad de los datos con aporte de algunos patrones, tanto para la arquitectura, como para el diseño y la interacción de cliente-servidor gracias PHP con su API PDO.
El reciente aplicativo mostrado en el trabajo “Desenvolvimento de Meios Digitais para a Biblioteca e aos Alunos: Aplicação Web.” (12) Controla la información de la biblioteca en el plantel educativo Centro Paula Souza Etec Irmã Agostina en Brasil, bajo tecnología de Back End con SQL Server de Windows, por su versatilidad multiplataforma PHP y PDO, aprovechando su capacidad de ejecutar consultas en diferentes SGBD con las mismas clases y métodos.

CONCLUSIONES

El desarrollo del sistema "ATLAS DE PUEBLA" siguió una metodología iterativa y modular, permitiendo la construcción progresiva de funcionalidades complejas sobre una base de datos bien estructurada y una conexión segura. La atención a la seguridad, la robustez del código y la usabilidad fue prioritaria en cada etapa de la codificación.
El sistema resultante es una herramienta integral capaz de centralizar y gestionar eficientemente la información de planteles educativos, contribuyendo significativamente a la administración y supervisión en el estado de Puebla.
El enfoque paso a paso en la codificación, desde la configuración inicial hasta la implementación de cada módulo, demostró ser eficaz para gestionar la complejidad del proyecto y entregar un producto funcional y mantenible.
Mejoras del Sistema "Atlas de Puebla de la Infraestructura Educativa".
Corto plazo: Se proyecta la adición de un módulo llamado Ventanilla Única, que agilice la gestión de trámites de las escuelas. El módulo se coordinará con la

Secretaría de Educación Pública (SEP), con el objetivo de evitar filas e inversión de tiempo prologando en su atención personalizada.

Mediano plazo: Podrá incorporarse funcionalidades de geolocalización avanzada, integrando mapas interactivos que permitan visualizar la distribución de los planteles y sus datos en tiempo real. La integración con otras plataformas educativas automatizará la actualización de información demográfica (como el total de alumnos o docentes), reduciendo la carga de trabajo manual.

Largo plazo: Evolución a una plataforma digital de análisis predictivo, utilizando los datos recopilados para proyectar necesidades de infraestructura, mantenimiento y seguridad, contribuyendo a una planificación educativa más estratégica y proactiva.

El futuro de PDO, según especialistas en tecnología su portabilidad, hace de PDO una API de bajo nivel, es "el método apropiado para trabajar con bases de datos en código moderno y profesional" según el sitio de SitePoint, (13), con demanda en crecimiento ya que proporciona flexibilidad, seguridad y flexibilidad para acceder a diferentes bases de datos lo afirma PHP (14).

Su mayor fortaleza está en la capa de seguridad, PDO ofrece una capa de seguridad adicional con sus sentencias preparadas concluye CEIBOO (15).

BIBLIOGRAFÍA

[1] Bretón Ángeles. ¿Cuántas escuelas públicas hay en Puebla? El Universal Puebla [Internet]. 2024 mayo 20 [citado 2025 jul 31]. Disponible en: https://www.eluniversalpuebla.com.mx/educacion/cuantas-escuelas-publicas-hay-en-puebla/

[2] CAPCEE. Quiénes somos [Internet]. 2024 [citado 2025 ago 1]. Disponible en: https://capcee.puebla.gob.mx/

[3] Folchini Sebbe VH. Segurança em Sistemas Web: Uma Análise da Extensão PDO como forma de Proteção de Sistemas PHP Contra Injeções de SQL [Internet]. Passo Fundo (BR): Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense - IFSUL; 2014 [citado 2025 ago 1]. Disponible en: https://painel.passofundo.ifsul.edu.br/uploads/arq/20160331163722187465530 1.pdf

[4] Nakagawa WT. Segurança em aplicação Web: comparativo entre API MySQL e API PDO [Internet]. Americana (BR): Faculdade de Tecnologia de Americana; 2017 [citado 2025 ago 1]. Disponible en: http://ric.cps.sp.gov.br/handle/123456789/1916

[5] Mendoza I, Franco V, Mera A, Muñoz V, Quiroz D. Sistema de subastas inglesa desarrollado con PHP POO PDO + MYSQL. Nexos Científicos [Internet]. 2022;6(2):42-6 [citado 2025 ago 1]. Disponible en: https://nexoscientificos.vidanueva.edu.ec/index.php/ojs/article/view/60/208

[6] Villamil Rodríguez JH. Diseño e Implementación de un Sistema POS, con Módulo de Gestión de Inventario de Productos para Clientes y Perfiles de Usuario, Aplicando Metodología RUP [Internet]. Bogotá (CO): Universidad Nacional Abierta y a Distancia; 2021 [citado 2025 ago 1]. Disponible en: https://repository.unad.edu.co/bitstream/handle/10596/41806/jhvillamilr.pdf

[7] Bastos, R. R., & de Magalhães, F. B. (2025). SQL Injection: Entendendo e evitando. Cuadernos de Educación y Desarrollo, 17(3), e7856-e7856. Mestre em Ciência da ComputaçãoInstituição: Universidade Federal de Pelotas. Pelotas, Brasil [consultado: 1 de agosto de 2025] Disponible en: https://ojs.cuadernoseducacion.com/ojs/index.php/ced/article/view/7856/5465

[8] Inaba. Uso de PDO (PHP Data Objects) para la conexión y gestión de bases de datos en PHP [Internet]. República Dominicana; 2025 [citado 2025 jul 31]. Disponible en: https://www.inabaweb.com/uso-de-pdo-php-data-objects-para-la-conexion-y-gestion-de-bases-de-datos-en-php/

[9] PHP. Introducción [Internet]. 2025 [citado 2025 ago 1]. Disponible en: https://www.php.net/manual/es/intro.pdo.php

[10] Melendez Cahuas JA. Sistema web en la gestión de mantenimiento del taller automotriz VIP2CARS en el año 2024 [Internet]. Lima (PE): Universidad Tecnológica del Perú; 2025 [citado 2025 ago 1]. Disponible en: https://repositorio.utp.edu.pe/handle/20.500.12867/12873

[11] González L, Eduardo J. Aplicación web de sistema de inventario para tiendas de artesanía del mercado de Chillán [Internet]. Chillán (CL): Universidad del Bío-Bío; 2015 [citado 2025 ago 1]. Disponible en: http://repobib.ubiobio.cl/jspui/handle/123456789/648

[12] Cepeda GV, Santos EN, Santos JP, Yamamoto DH. Desenvolvimento de meios digitais para a biblioteca e os alunos: aplicação web [Internet]. 2023 [citado 2025 ago 1]. Disponible en: http://ric-cps.eastus2.cloudapp.azure.com/bitstream/123456789/12964/1/Desenvolvimento%20de%20meios%20digitais%20para%20a%20Biblioteca.pdf

[13] sitePoint. Reintroduciendo la PDO – La forma correcta de acceder a las bases de datos en PHP [Internet]. 2025 [citado 2025 ago 1]. Disponible en: https://www.sitepoint.com/re-introducing-pdo-the-right-way-to-access-databases-in-php/

[14] PHP. Introducción [Internet]. 2025 [citado 2025 ago 1]. Disponible en: https://www.php.net/manual/es/intro.pdo.php

[15] CEIBOO. ¿Qué usar en un nuevo proyecto de PHP: PDO o MySQLi? [Internet]. 2021 [citado 2025 ago 1]. Disponible en: https://www.ceiboo.com/blog/que-usar-en-un-nuevo-proyecto-de-php-pdo-o-mysqli-5

Tabla de Roles de Contribución

Rol	Autor (es)
Conceptualización	Efrén Armando Osorio Ramírez
Curación de datos	Aldo Yahir Ara Mora

Metodología	Susana Martínez Rabanales
Administración del proyecto	Susana Martínez Rabanales
Recursos	Revisión y edición – Violeta Martínez Ramírez
Software	Aldo Yahir Ara Mora Ramos
Supervisión	Efrén Armando Osorio Ramírez
Validación	Aldo Yahir Ara Mora
Visualización	Teresa Luciano Machorro
Redacción	Borrador original – Violeta Martínez Ramírez
Redacción	Revisión y edición – Violeta Martínez Ramírez