PDO IN THE MANAGEMENT OF WEB-BASED EDUCATIONAL INFRASTRUCTURE: CASE STUDY 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] National Technological Institute of Mexico/Puebla Technological Institute,violeta.martinez@puebla.tecnm.mx , https://orcid.org/0000-0003-1518-786X
[2] National Technological Institute of Mexico/Technological Institute of Puebla,susana.martinez@puebla.tecnm.mx , https://orcid.org/0009-0002-8510-5380
[3] National Technological Institute of Mexico/Puebla Technological Institute,efren.osorio@puebla.tecnm.mx , https://orcid.org/0000-0002-6902-1841
[4] National Technological Institute of Mexico/Puebla Technological Institute,teresa.luciano@puebla.tecnm.mx , https://orcid.org/0000-0003-3979-9369
[5] National Technological Institute of Mexico/Puebla Technological Institute,i20222166.19@puebla.tecnm.mx , https://orcid.org/0009-0008-1746-4633

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

Abstract -- This technological research paper describes the development methodology used in the creation of the "Atlas de Puebla de la Infraestructura Educativa" (Puebla Educational Infrastructure Atlas) system, a web application for managing and consulting information on educational facilities in the state of Puebla. It details a step-by-step approach to the coding process, highlighting the key phases, the technologies used, and the methodological decisions that ensured the functionality and scalability of the project. The objective is to offer a clear vision of how the application was built, from the initial conception of the modules to the implementation of its main features.

Keywords: software, Agile Methodology, PDO, School Management, 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.

INTRODUCTION

Information published by the newspaper EL UNIVERSAL PUEBLA (1) in its digital version states that more than 80% of educational institutions in the state are public schools.

According to data from the Mexican National Statistical Information System, there are 14,494 schools in the state's education system, of which 11,961 are public. It also highlights that the state of Puebla has the second highest demand for education, followed by Mexico City, receiving a large number of students annually from other states, mainly from central, southern, and southeastern Mexico.

The SEP of the state of Puebla has more than 2 million students enrolled at all educational levels for the year 2025.

Information regarding the infrastructure of all public schools in the state of Puebla is concentrated in the Puebla Administration Committee for the Construction of Educational Spaces, a decentralized body (CAPCEE). As a decentralized body dependent on the state government, it is responsible for formulating, applying, executing, and monitoring the construction, rehabilitation, equipping, and maintenance of physical educational infrastructure and special CPACEE data (2). It oversees the movable and immovable property used for education provided by the state. So far during this six-year term, educational works are underway at 32 schools with an investment of 68.5 million pesos focused on the construction and rehabilitation of classrooms and bathrooms and the installation of solar panels and rainwater collectors, according to data published by capcee.puebla.gob.mx.

State of the Art

Below is a review of some academic papers published by various universities in Latin America related to the use of PHP and its PDO extension.

The first, entitled "Segurança em Sistemas Web: Uma Análise da Extensão PDO como forma de Proteção de Sistemas PHP Contra Injeções de SQL" (3), Brazil, shows the successful implementation within a web system using PDO to prevent improper and external injection of SQL queries, with the intention of protecting data against unauthorized access, especially by malicious users, whose actions can cause significant damage to individuals and organizations. As a result, tests performed on the web login system validated that the use of prepared statements and the parameterization of user-entered information in database queries are highly effective in protecting systems against SQL injections. According to tests performed with manual injections through a form field or with a specialized tool, PDO proved to be 100% secure against SQL (Structured Query Language) injection attacks, provided that it is used correctly and data integrity is maintained. However, if commands are injected by the user and were not initially executed but were stored in the database, they can create high vulnerability to the system, known as "second-order injections." Therefore, it is recommended to apply parameterization to them. In summary, PDO provides a high-performance security layer for web systems.

Another academic paper entitled "Comparison between MySQL API and PDO API" (4) highlights the vulnerability of web applications distributed on the internet. Therefore, to significantly reduce attacks frequently injected through the browser address bar or HTML forms, a more up-to-date and parameterized API called PDO has been implemented. As a result of the comparison between MySQL API and PDO, through which a penetration test was automated using a web vulnerability scanning tool, the ZAP (Zed Attack Proxy) program, to verify whether the applications had suffered SQL injection attacks, it was demonstrated that PDO achieved 100% security against SQL injections.

The work published as "English auction system developed with PHP POO PDO + MYSQL" (5) in Ecuador, among its automated requirements, highlights its contribution to information management performance through a web system that performs and updates records securely thanks to PDO exceptions. In addition, it offered greater security within the system by preserving data integrity when granting roles and privileges to users within the application. Last but not least, it provides a reliability scheme where the application remains stable and ensures 100% trust in transactions performed by users and administrators throughout its entire lifetime.

In Bogotá, Colombia, the work "Design and Implementation of a POS System, with Product Inventory Management Module for Customers and User Profiles, Applying RUP Methodology" published in (6), opts for the development of code that includes the MVC (Model View Controller) pattern, created in three layers or levels, where the security layer is more important than the design layer within the web application created as a corporate image and online sales. All system managers implement the connection to the database with MySQL using the PHP PDO class to ensure successful connection and data integrity. Real-time point-of-sale transactions require protection of the information provided by the customer about their own information and digital purchases as a web portal.

A 2025 publication in Brazil, with the project entitled "SQL Injection: Understanding and Avoiding" (7), provides comparisons related to information management security in web applications, considering three strategies: data encryption, input validation, and the use of PHP Data Objects (PDO). The first of these, encryption, protects information by converting it into formats that are unreadable without the correct key. The second strategy applies input validation by blocking dangerous characters using white or black lists. In the last one, PDO offered a secure interface for SQL queries, using parameters that prevent the injection of malicious code. The results showed that, while encryption and validation significantly reduce risks, the most effective strategy was to implement PDO, as it encapsulates queries and prevents unauthorized updates. The best computational practice, then, is to combine these techniques to ensure the security of systems that interact with databases on the web.

Problem.

The "ATLAS DE PUEBLA" system arose from the need to centralize, organize, and facilitate access to crucial information about educational institutions in the state of Puebla, Mexico. Prior to this initiative, the management of data on infrastructure, services, and the operational status of schools was scattered, making it difficult to make informed decisions and conduct efficient oversight. This report documents the development methodology adopted to build this web application, emphasizing the iterative coding process and the practices implemented to ensure quality and compliance with requirements. The project is part of an effort to modernize and optimize educational administration through technological tools.

General objective

Creation of a web application for managing educational spaces in public schools in the state of Puebla using PHP Data Object.

Justification

The selection of technologies in the "Atlas de Puebla" software leaned toward PHP Data Object (PDO) due to its high capacity to provide database security, adaptability, and error handling.

It is an intelligent source in related database tables, effectively protecting against SQL injection by using prepared statements to separate those instructions from the data. In addition, it is designed to work with various databases and issue messages if something improper happens.

That is why it stands out as the most recommended and popular choice for interacting with databases in PHP.

Based on the above, the use of PDO is justified for the security of the database design and storage in the "Atlas Puebla" system.

PHP (Hypertext Preprocessor)

It is one of the most widely used tools in the world of web application design and creation. Its main feature lies in its strength as a programming language that offers several options for interacting with databases, with its vast functionalities for creating robust back ends predominating.

One of the flexible and powerful tools is the PDO extension, which promotes portability and information security.

Its goal is to create a consistent and secure way to access databases regardless of the type of Database Management System (DBMS) (8).

About PDO

The PHP website (9) defines PHP Data Object (PDO) as an extension that allows access to different databases that are popular among developers, such as MySQL/MariaDB, PostgreSQL, Oracle, MS SQL Server, SQLite, Firebird, DB2, Informix, etc., establishing portability between them. However, it does not evaluate the correctness of SQL queries, but it does offer security through prepared queries. Formally, PDO is a PHP library that works with object-oriented programs.

Choosing how to handle errors in PDO: 1. Without interrupting program execution, errors are saved and the type of error must be checked when making a query. 2. Errors are saved and a warning is issued. 3. In this option, the error exception is handled by the program itself.

Queries whose data source comes from forms must use prepared queries to avoid attacks that the QUERY method does not protect against. Otherwise, only an if...else conditional statement will be required (image 1). Return (image 2) or no records (image 3).

Image 1. *Query without data from forms*.

```
// EJECUCIÓN DE UNA CONSULTA SIN DATOS PROVENIENTES DE UN FORMULARIO
$pdo->query("consulta");
```

Source: *Own work* (2025).

Image 2. *Query that does not return a record*.

```
// 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
}
```

Source: *Own creation* (2025).

Image **3**. *Query that returns a record*.

```
// 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
}
```

Source: *Own work* (2025).

The term "Consultation" according to the portal manuais.iessanclemente.net (2018), is defined as a pre-compiled SQL statement that can be executed multiple times with the sole condition of sending its data to a server.

DEVELOPMENT

The development methodology chosen for "ATLAS DE PUEBLA" was a hybrid approach, combining elements of agile methodologies (such as Scrum or Kanban for task and iteration management) with structured initial planning for database architecture and module definition. This allowed for flexibility to adapt to changing requirements while maintaining a clear vision of the fundamental structure of the system.

The general phases included:

• Planning and Requirements Analysis: Definition of key functionalities (staff management, search, reports, users), identification of data entities and their relationships.

• Database Design: Creation of the relational schema.

• Application Architecture Design: Determination of file structure, modules, and design patterns (implicit MVC).

• Implementation (Iterative Coding): Progressive development of functionalities.

• Testing and Debugging: Constant verification of functionality and error correction.

• Deployment and Maintenance: Launch and ongoing support.

This report focuses on the Implementation phase (Iterative Coding), which is broken down below.

Step-by-Step Coding

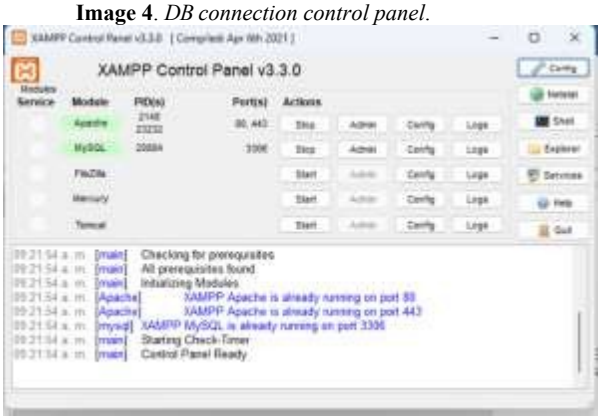
The coding of the "ATLAS DE PUEBLA" was carried out in a modular and layered manner, following a principle of incremental development.

Environment Configuration and Database Connection:

The first step in coding was to establish the development environment and connectivity. XAMPP

(Apache, MySQL, PHP) was used as the local server. See image 4.

Image 4. *DB connection control panel*.



Source: *Own work* (2025).

The database.php file was created to encapsulate the database connection settings (server, user, password, database name, and UTF-8 encoding) using PDO (PHP Data Objects). PDO was chosen for its robustness, security against SQL injection through prepared statements, and error handling. BASE_URL was defined to ensure proper navigation between pages. In addition, auth_check.php implemented an authentication system based on PHP sessions, included in all restricted pages to verify login and expiration due to inactivity.

Featured Code: PDO Connection (database.php)

This essential snippet shows how to establish a secure connection to the database, defining constants and options for PDO.

PHP

```
// Excerpt from 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 connection connection PDO: " . $e->getMessage());
    die("Error connecting to the server. Please try again later.");
}
// The definition of BASE_URL is assumed to occur here or in a similar file [MODIFICATION: COMMENT ADDED]
// define('BASE_URL', $protocol . $_SERVER['HTTP_HOST'] . '/atlas_puebla/');
?>
.
```

Database Schema Design and Implementation:

Before any visible HTML lines, the database was meticulously designed to support all functionalities. The atlas_puebla_db.sql script defined the structure of all tables, including schools (central information for each school), users (account management), roles (access levels), and support tables such as municipalities, localities, and cordes. Detail tables (detalle_hidrosanitario, detalle_proteccion_civil, detalle_servicios) were also created with one-to-one relationships with schools, as well as tables for managing files and visual material (archivos_plantel, galeria_fotos). Primary and foreign keys were defined with ON DELETE CASCADE or SET NULL actions to maintain referential integrity and improve data consistency.

Featured Code: Table structure for planteles and detail_proteccion_civil (atlas_puebla_db.sql)

These fragments illustrate the central table of campuses and one of its detail tables, civil_protection_details, showing the key relationships that ensure the consistency of the information.

SQL

```
-- Fragment from atlas_puebla_db.sql
CREATE TABLE `planteles` (
  `cct` varchar(20) NOT NULL,
  `school_name` varchar(255) NOT NULL,
  `id_municipality` int(11) DEFAULT NULL,
  `id_localidad` int(11) DEFAULT NULL,
  `code_id` int(11) DEFAULT NULL,
  `assigned_principal_id` int(11) DEFAULT NULL,
  -- ... other columns ... PRIMARY
  KEY (`cct`),
  KEY `id_municipality` (`id_municipality`),
  KEY `assigned_director_id` (`assigned_director_id`),
CONSTRAINT `ibfk_1_staff` FOREIGN KEY (`municipality_id`) REFERENCES `municipalities` (`municipality_id`),
  CONSTRAINT `planteles_ibfk_4` FOREIGN KEY (`assigned_director_id`) REFERENCES `users` (`user_id`) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Example of detail table
CREATE TABLE `civil_protection_details` (
```

```
  `id_pc_detail`  int(11)  NOT  NULL
AUTO_INCREMENT,
  `cct` varchar(20) NOT NULL UNIQUE, -- One-to-one
relationship
  `seismic_alarm` tinyint(1) DEFAULT 0,
  -- ... other columns ...
  PRIMARY KEY (`id_pc_detail`),
  CONSTRAINT  `civil_protection_detail_ibfk_1`
FOREIGN KEY (`cct`) REFERENCES `staff` (`cct`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

Development of the Authentication and Access Module: The system entry point was built with a security focus. The index.php page redirects to the dashboard if the user is already authenticated or displays the login form. The login.php file processes the credentials using password_verify() to securely compare the password hash, and handles account statuses (active/inactive) and credential errors. Finally, logout.php destroys the session, ensuring a secure logout.

Featured Code: Password and Session Verification (login.php)

This snippet illustrates how password_verify() is used for secure credential validation and how PHP sessions are handled, including session ID regeneration for added security. See image 5.

Image 5. *Authentication for access*.

Source: *Own work* (2025).

```
PHP
// Excerpt from login.php
if  ($user  &&  password_verify($password,
$user['password_hash']))  {  //  SECURE
PASSWORD VERIFICATION
    if ($user['status'] === 'ACTIVE') {
        session_regenerate_id(true); // Regenerate the session
        ID
session ID for security
        $_SESSION["loggedin"] = true;
        $_SESSION["user_id"] = $user['user_id'];
        $_SESSION["full_name"]                    =
$user['full_name'];
        $_SESSION["role_id"] = $user['role_id'];
        // Update last connection in DB
        $update_stmt = $pdo->prepare("UPDATE users SET
last_connection = CURRENT_TIMESTAMP WHERE
user_id = :id");
        $update_stmt->execute(['id'                    =>
$user['user_id']]);
        header("location:  "  .  BASE_URL  .
"dashboard.php");
        exit;
    } else {
        $error = "The account is inactive.";
    }
} else {
    $error = "Incorrect username or password."; // Generic
message for security
}
```

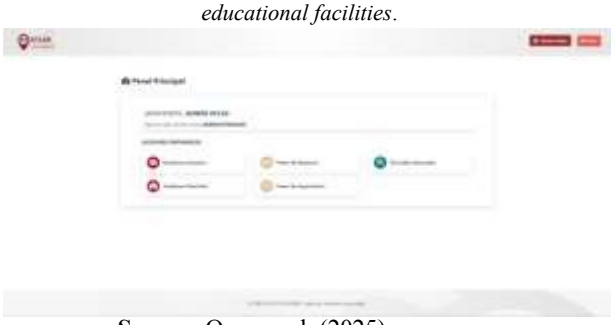
Dashboard and User Profile Implementation: Once access was functional, the main interface was developed. The dashboard.php acts as the central navigation point, displaying options according to the user's id_role.

The files mi_perfil.php and mi_perfil_guardar.php files allow users to view and update their personal information and change their password, including validation of the current password and hashing of the new one with password_hash().

Construction of the Campus Management Module:

This is the heart of the system, where school information is recorded and consulted. See image 6.

Image 6. *Registration and consultation of information on state educational facilities*.



Source: Own work (2025).

planteles_lista.php displays a paginated list of teams, with the SQL query adapting according to the user's role (administrator sees all, directors/data entry clerks see only those assigned), and includes text search.

plantel_ficha_form.php and plantel_ficha_guardar.php implement the form for creating or editing the basic file for a team, with crucial permission logic that allows managers to register or edit their own teams and administrators to modify any team, including its status; it handles data insertion (INSERT) and updating (UPDATE). The page

plantel_detalle.php centralizes all the information for a specific facility, organizing the data into tabs (Basic File, Spaces, Services, Civil Protection, Files, Galleries) using JavaScript for a better user experience, and loading data with multiple individual queries per section. Finally, plantel_eliminar.php allows administrators to delete a campus, with a control to prevent deletion if there are associated records (due to foreign key restrictions) and handles PDO transactions to ensure atomicity. See image 7.

Image 7. *Management process for information integrity*.

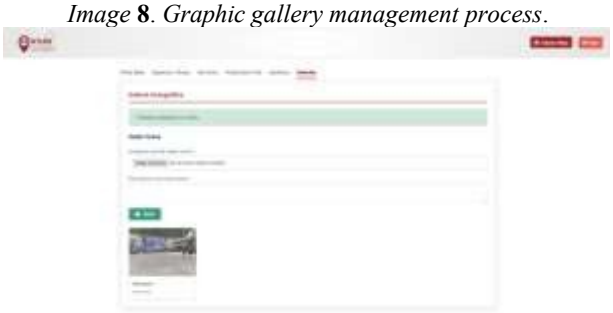
Source: *Own work* (2025).

.

Development of Detail Sub-Modules (Spaces, Services, Civil Protection, Archives, Galleries):

Each section of plantel_detalle.php has its own management logic.
espacio_guardar.php and espacio_eliminar.php manage the addition and deletion of records in espacios_areas.
servicios_form.php and servicios_guardar.php allow you to edit and save the information in detalle_hidrosanitario and detalle_servicios, using ON DUPLICATE KEY UPDATE in SQL to efficiently handle the insertion or updating of these related records. Similarly, proteccion_civil_form.php a nd proteccion_civil_guardar.php manage the information in detail_proteccion_civil. The files archivo_guardar.php and archivo_eliminar.php implement file uploads and deletions, including the handling of secure names and storage paths (uploads/archivos/), deleting the physical file when deleting the record. Finally, galeria_guardar.php and foto_eliminar.php manage the upload of multiple images and their deletion, using the uploads/galeria/ directory. See image 8.

Image **8**. *Graphic gallery management process*.



.

Source: *Own creation* (2025).

Featured Code: Transaction and UPSERT for Staff Details (servicios_guardar.php)

This code example demonstrates the use of database transactions (beginTransaction(), commit(), rollBack()) to ensure that insert/update operations on multiple related tables are atomic. The ON DUPLICATE KEY UPDATE clause allows you to insert a new record if it does not exist, or update it if it already does, simplifying the "upsert" logic.

```
PHP
// Excerpt from servicios_guardar.php (example of transaction and UPSERT)
try {
   $pdo->beginTransaction(); // START TRANSACTION

   // For detail_hydrosanitary: INSERT OR UPDATE HYDRO-SANITARY DATA
   $sql_hidro = "INSERT INTO detail_hydrosanitary (cct, water_source, water_storage, drainage_type, men's_toilets, men's_sinks, women's_toilets, women's_sinks, observations)
                 VALUES (:cct, :water_source, :water_storage, :drainage_type, :men's_toilets, :men's_washbasins, :women's_toilets, :women's_restroom_sinks, :observations)
                 ON DUPLICATE KEY UPDATE water_source=VALUES(water_source), water_storage=VALUES(water_storage
),
                 drainage_type=VALUES(drainage_type)
, men's_toilets=VALUES(men's_toilets), men's_toilets_sinks=VALUES(men's_toilets_sinks), women's_toilets=VALUES(women's_toilets), women's_toilets_sinks=VALUES(women's_toilets_sinks), observations=VALUES(observations)";
   $pdo->prepare($sql_hydro)->execute($data_hydro);

   // For service_details: INSERT OR UPDATE BASIC SERVICE DATA
```

```
    $sql_services = "INSERT INTO service_details (cct,
                electricity_contract,
                landline_telephone, internet_access, gas_type,
internet_type, comments)
                VALUES (:cct, :electricity_contract,
:landline_telephone, :internet_access, :gas_type,
:internet_type,
:comments)
                ON DUPLICATE KEY UPDATE
electricity_contract=VALUES(electricity_contract),
landline_telephone=VALUES(landline_telephone),
internet_access=VALUES(internet_access),
gas_type=VALUES(gas_type),
internet_type=VALUES(internet_type),
comments=VALUES(comments)";
    $pdo->prepare($sql_services)-
>execute($data_services);

    $pdo->commit(); // CONFIRMS THE TRANSACTION
IF BOTH OPERATIONS WERE SUCCESSFUL
  // ... (redirection successful) ...
  } catch (PDOException $e) {
    $pdo->rollBack(); // ROLL BACK THE
TRANSACTION IN CASE OF ERROR
    error_log("Error when saving services: " .
    $e-
>getMessage());
    // ... (redirection with error message) ...
}
```

Implementation of the Advanced Search Engine:

A central feature for data queries. buscador.php integrates all filters (text, CORDE, municipality, level, sustainability, seismic alarm). The dynamic construction of the WHERE clause ($sql_where_parts) and the use of prepared statements are crucial. Specific attention was given to handling case-insensitive searches for text (LOWER()) and the inclusion of NULL records for the seismic alarm filter, which improves the accuracy of the results. Pagination is integrated with the active filters. See image 9.

Image 9. *Advanced search process*.

Source: *Own work* (2025).

Featured Code: Text Filter and Seismic Alarm Logic (search.php)

This fragment illustrates the dynamic construction of the WHERE clause for multiple filters. The Implementation of case-insensitive text search using LOWER() in the database, and specific handling of the "Seismic Alarm" filter to include both '0' and NULL values, improving the accuracy of the results.

PHP

```
// Fragment from search.php (part of the filter logic)
// Text filter (with LOWER for case insensitivity)
if (!empty($filters['search'])) {
    $sql_where_parts[] = "(LOWER(p.cct) LIKE
LOWER(:search) OR LOWER(p.school_name) LIKE
LOWER(:search))";
    $params['search'] = "%{$filters['search']}%";
}

// JOINS SQL
$sql_joins = "FROM campuses p LEFT JOIN
municipalities m ON p.municipality_id =
m.municipality_id LEFT JOIN users u ON
p.assigned_director_id = u.user_id ";

// Filter by Seismic Alarm (handles 0 and NULL to include
campuses without a specific record)
if ($filters['seismic_alarm'] !== '') {
    $sql_joins .= "LEFT JOIN civil_protection_details pc
ON p.cct = pc.cct";
    if ($filters['seismic_alarm'] === '1') {
        $sql_where_parts[] = "pc.seismic_alarm = 1";
    } elseif ($filters['seismic_alarm'] === '0') {
        $sql_where_parts[] = "(pc.seismic_alarm = 0 OR
pc.seismic_alarm IS NULL)";
    }
}

$sql_where = "";
if (!empty($sql_where_parts)) {
    $sql_where = "WHERE " . implode(" AND ",
$sql_where_parts);
}
// ... then prepare and execute the query
```

3.8. Development of Reporting and Monitoring Modules:

For the extraction and visualization of aggregated data, reports_panel.php offers a menu with different reports. Reports such as:

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

which generate specific data with COUNT() and GROUP BY, including graphs generated with Chart.js for visualization.

export_csv.php allows data to be exported to CSV. In addition, the files supervision_panel.php, supervision_corde_detail.php, and supervision_escuela_detalle.php provide summary and detailed views of the progress of information capture at the CORDE and campus levels, using progress percentages for monitoring.

User Management (Administrator):
This module is for complete user administration.
usuarios_lista.php lists all users with their roles and status, allowing editing and deletion actions. The files usuario_form.php and usuario_guardar.php contain the forms and logic for creating new users or editing existing ones, including password hashing and unique email validation.
usuario_eliminar.php allows the administrator to delete users, with validations to prevent self-deletion and the deletion of users with dependencies in other tables (foreign key handling). See image 10.

Image 10. *User update process according to dependency*.

Source: *Own work* (2025).

DISCUSSION AND ANALYSIS OF RESULTS

Technical Considerations and Best Practices
During the coding process, several best practices were followed:
• Modularization: Division of code into functional PHP files (database.php, auth_check.php, _save.php, _delete.php, etc.) to improve maintainability and reusability.
• Security: Extensive use of PDO with prepared statements to prevent SQL injections. Password hashing (password_hash()) for credential protection. Basic input validations (
trim, htmlspecialchars).
• Error Handling: Implementation of try-catch blocks to capture PDO exceptions and log critical errors (error_log()), which facilitates debugging and improves application robustness.
• User Experience: Use of CSS for responsive and consistent design, and JavaScript for interactive enhancements (e.g., show/hide filters, pagination).
• Data Consistency: Database design with foreign keys and ON DUPLICATE KEY UPDATE logic in PHP for detail sections ensure data integrity and uniqueness.

Discussion.
In relation to CAPCEE's objective of reducing operating costs in the unit's information systems, it is important to address the issue of security at the same time. time that the reduction in implementation and maintenance costs is similar to the work "Web system for maintenance management at the VIP2CARS automotive workshop in 2024" in Ecuador (10), which shows the development of a digital system based on PHP, MySQL, and JavaScript, designed under the MVC (Model View Controller) model to optimize workflow, reduce errors, and improve service traceability. Its main objective is to eliminate errors created with OSS (Open Source) tools for the Back End such as: PHP, MySQLi, and PDO, which ensure the atomicity, consistency, isolation, and durability (ACID) of the information. All of this significantly reduced the financial investment and increased the profitability and operational efficiency of the business.
The advantages of choosing PHP and PDO extensions as the best option for development and security are backed by other successfully implemented systems, such as the one presented as a solution for artisan markets in Chile with the project "Web application inventory system for artisan shops in the Chillán market" (11), a web application based on the YII Framework structure as a development environment, which provides data security with the contribution of certain patterns, both for the architecture and for the design and client-server interaction thanks to PHP with its PDO API.
The recent application shown in the work "Development of Digital Media for the Library and Students: Web Application." (12) controls library information at the Centro Paula Souza Etec Irmã Agostina educational campus in Brazil, using Back End technology with Windows SQL Server, due to its multi-platform PHP and PDO versatility, taking advantage of its ability to execute queries in different DBMSs with the same classes and methods.

CONCLUSIONS

The development of the "ATLAS DE PUEBLA" system followed an iterative and modular methodology, allowing for the progressive construction of complex functionalities on a well-structured database and a secure connection. Attention to security, code robustness, and usability was a priority at every stage of coding.
The resulting system is a comprehensive tool capable of centralizing and efficiently managing information from educational institutions, contributing significantly to administration and supervision in the state of Puebla. The step-by-step approach to coding, from initial configuration to the implementation of each module,demonstrated to be effective for manage the complexity of the project and deliver a functional and maintainable product.
Improvements to the System "Atlas Atlas of Puebla Educational Infrastructure."
Short term: The addition of a module called Ventanilla Única (One-Stop Shop) is planned, which will streamline the management of school procedures. The module will be coordinated with the

Ministry of Public Education (SEP), with the aim of avoiding queues and saving time by providing personalized attention.

Medium term: Advanced geolocation features can be incorporated, integrating interactive maps that allow the distribution of campuses and their data to be viewed in real time. Integration with other educational platforms will automate the updating of demographic information (such as the total number of students or teachers), reducing the manual workload.

Long term: Evolution to a predictive analytics digital platform, using the data collected to project infrastructure, maintenance, and security needs, contributing to more strategic and proactive educational planning.

The future of PDO, according to technology specialists, is its portability, which makes PDO a low-level API. It is "the appropriate method for working with databases in modern and professional code," according to the SitePoint website (13), with growing demand as it provides flexibility, security, and flexibility to access different databases, according to PHP (14). Its greatest strength lies in the security layer, with PDO offering an additional layer of security with its prepared statements, concludes CEIBOO (15).

BIBLIOGRAPHY

[1] Bretón Ángeles. How many public schools are there in Puebla? El Universal Puebla [Internet]. May 20, 2024 [cited July 31, 2025]. Available at: https://www.eluniversalpuebla.com.mx/educacion/cuant as-escuelas-publicas-hay-en-puebla/

[2] CAPCEE. Who we are [Internet]. 2024 [cited 2025 Aug 1]. Available at: https://capcee.puebla.gob.mx/

[3] Folchini Sebbe VH. Security in Web Systems: An Analysis of the PDO Extension as a Means of Protecting PHP Systems Against SQL Injections [Internet]. Passo Fundo (BR): Federal Institute of Education, Science, and Technology of Rio Grande do Sul - IFSUL; 2014 [cited 2025 Aug 1]. Available at: https://painel.passofundo.ifsul.edu.br/uploads/arq/20160 3311637221874655301.pdf

[4] Nakagawa WT. Web application security: comparison between MySQL API and PDO API [Internet]. Americana (BR): Americana College of Technology; 2017 [cited 2025 Aug 1]. Available from: http://ric.cps.sp.gov.br/handle/123456789/1916

[5] Mendoza I, Franco V, Mera A, Muñoz V, Quiroz D. English auction system developed with PHP POO PDO + MYSQL. Nexos Científicos [Internet]. 2022;6(2):42-6 [cited 2025 Aug 1]. Available from: https://nexoscientificos.vidanueva.edu.ec/index.php/ojs/ article/view/60/208

[6] Villamil Rodríguez JH. Design and Implementation of a POS System, with Product Inventory Management Module for Customers and User Profiles, Applying RUP Methodology [Internet]. Bogotá (CO): National Open and Distance University; 2021 [cited 2025 Aug 1]. Available at: https://repository.unad.edu.co/bitstream/handle/10596/4 1806/jhvillamilr.pdf

[7] Bastos, R. R., & de Magalhães, F. B. (2025). SQL Injection: Understanding and avoiding. Education and Development Notebooks, 17(3), e7856-e7856. Master in Computer Science Institution: Federal University of Pelotas. Pelotas, Brazil [accessed: August 1, 2025] Available at: https://ojs.cuadernoseducacion.com/ojs/index.php/ced/ar ticle/view/7856/5465

[8] Inaba. Use of PDO (PHP Data Objects) for database connection and management in PHP [Internet]. Dominican Republic ; 2025 [cited 2025 2025 Jul 31]. Available at: https://www.inabaweb.com/uso-de-pdo-php-data-objects-para-la-conexion-y-gestion-de-bases-de-datos-en-php/

[9] PHP. Introduction [Internet]. 2025 [cited 2025 Aug 1]. Available at: https://www.php.net/manual/es/intro.pdo.php

[10] Melendez Cahuas JA. Web system for maintenance management at the VIP2CARS automotive workshop in 2024 [Internet]. Lima (PE): Technological University of Peru; 2025 [cited 2025 Aug 1]. Available at: https://repositorio.utp.edu.pe/handle/20.500.12867/1287 3

[11] González L, Eduardo J. Web application of inventory system for craft shops in the Chillán market [Internet]. Chillán (CL): University of Bío-Bío; 2015 [cited 2025 Aug 1]. Available at: http://repobib.ubiobio.cl/jspui/handle/123456789/648

[12] Cepeda GV, Santos EN, Santos JP, Yamamoto DH. Development of digital resources for the library and students: web application [Internet]. 2023 [cited 2025 Aug 1]. Available at: http://ric-cps.eastus2.cloudapp.azure.com/bitstream/123456789/1 2964/1/Desenvolvimento%20de%20meios%20digitais% 20para%20a%20Biblioteca.pdf

[13] sitePoint. Reintroducing PDO – The right way to access databases in PHP [Internet]. 2025 [cited 2025 Aug 1]. Available at: https://www.sitepoint.com/re-introducing-pdo-the-right-way-to-access-databases-in-php/

[14] PHP. Introduction [Internet]. 2025 [cited 2025 Aug 1]. Available at: https://www.php.net/manual/es/intro.pdo.php

[15] CEIBOO. What to use in a new PHP project: PDO or MySQLi? [Internet]. 2021 [cited 2025 Aug 1]. Available at: https://www.ceiboo.com/blog/que-usar-en-un-nuevo-proyecto-de-php-pdo-o-mysqli-5

Contribution Role Table

Role	Author(s)
Conceptualization	Efrén Armando Osorio Ramírez
Data curation	Aldo Yahir Ara Mora

Methodology	Susana Martínez Rabanales
Project management	Susana Martínez Rabanales
Resources	Review and editing – Violeta Martínez Ramírez
Software	Aldo Yahir Ara Mora Ramos
Supervision	Efrén Armando Osorio Ramírez
Validation	Aldo Yahir Ara Mora
Visualization	Teresa Luciano Machorro
Writing	Original draft – Violeta Martínez Ramírez
Writing	Revision and editing – Violeta Martínez Ramírez