🎯 **PHP hướng đối tượng (OOP)** sử dụng **PDO**, có đủ 4 phương thức chính:
`addUser()`, `getUsers()`, `updateUser()`, `deleteUser()` và một hàm khởi tạo để **kết nối cơ sở dữ liệu**.

🧱 1️⃣ Cấu trúc tổng quát

Chúng ta sẽ tạo file `UserModel.php` như sau 👇

```php
<?php
class UserModel {
    private $pdo;

    // ------------------------
    // 1️⃣ Hàm khởi tạo (constructor)
    // ------------------------
    public function __construct($host, $dbname, $username, $password) {
        $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8";
        try {
            $this->pdo = new PDO($dsn, $username, $password);
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            echo "✅ Kết nối CSDL thành công!<br>";
        } catch (PDOException $e) {
            die("❌ Lỗi kết nối: " . $e->getMessage());
        }
    }

    // ------------------------
    // 2️⃣ CREATE - Thêm người dùng mới
    // ------------------------
    public function addUser($name, $email) {
        $sql = "INSERT INTO users (name, email) VALUES (:name, :email)";
        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam(":name", $name, PDO::PARAM_STR);
        $stmt->bindParam(":email", $email, PDO::PARAM_STR);

        if ($stmt->execute()) {
            echo "✅ Đã thêm người dùng: $name ($email)<br>";
            return true;
        }
        return false;
    }

    // ------------------------
    // 3️⃣ READ - Lấy danh sách người dùng
    // ------------------------
    public function getUsers() {
        $sql = "SELECT * FROM users";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute();
```

```
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    // ------------------------
    // 4️⃣ UPDATE - Cập nhật thông tin người dùng
    // ------------------------
    public function updateUser($email, $newName) {
        $sql = "UPDATE users SET name = :name WHERE email = :email";
        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam(":name", $newName, PDO::PARAM_STR);
        $stmt->bindParam(":email", $email, PDO::PARAM_STR);

        if ($stmt->execute()) {
            echo "✅ Cập nhật người dùng ($email) thành công.<br>";
            return true;
        }
        return false;
    }

    // ------------------------
    // 5️⃣ DELETE - Xóa người dùng
    // ------------------------
    public function deleteUser($email) {
        $sql = "DELETE FROM users WHERE email = :email";
        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam(":email", $email, PDO::PARAM_STR);

        if ($stmt->execute()) {
            echo "🗑️ Đã xóa người dùng có email: $email<br>";
            return true;
        }
        return false;
    }
}
?>
```

🚀 2️⃣ File chạy thử: `test_crud.php`

```
<?php
require_once 'UserModel.php';

// Khởi tạo đối tượng
$userModel = new UserModel("localhost", "testdb", "root", "");

// 1️⃣ Thêm người dùng mới
$userModel->addUser("Nguyễn Văn A", "vana@example.com");
```

```php
$userModel->addUser("Trần Thị B", "tranb@example.com");

// 2️⃣ Lấy danh sách người dùng
echo "<h3>📋 Danh sách người dùng:</h3>";
$users = $userModel->getUsers();
foreach ($users as $user) {
    echo "{$user['id']} | {$user['name']} | {$user['email']}<br>";
}

// 3️⃣ Cập nhật người dùng
$userModel->updateUser("vana@example.com", "Nguyễn Văn C");

// 4️⃣ Xóa người dùng
$userModel->deleteUser("tranb@example.com");

// 5️⃣ Kiểm tra lại danh sách sau khi CRUD
echo "<h3>🔄 Danh sách sau khi CRUD:</h3>";
$users = $userModel->getUsers();
foreach ($users as $user) {
    echo "{$user['id']} | {$user['name']} | {$user['email']}<br>";
}
?>
```

📘 Giải thích tổng quan

Phương thức	Chức năng	SQL sử dụng	Trả về
`addUser($name, $email)`	Thêm người dùng mới	`INSERT INTO users`	`true/false`
`getUsers()`	Lấy toàn bộ danh sách người dùng	`SELECT * FROM users`	Mảng kết quả
`updateUser($email, $newName)`	Cập nhật tên theo email	`UPDATE users SET ...`	`true/false`
`deleteUser($email)`	Xóa người dùng theo email	`DELETE FROM users WHERE email = ...`	`true/false`

⚡ Ưu điểm của cách viết OOP này

✅ Dễ mở rộng: chỉ cần thêm các phương thức khác (như `findById()`, `searchUser()`, …).
✅ Dễ bảo trì: Mọi truy cập CSDL tập trung trong 1 lớp duy nhất.
✅ An toàn: vẫn giữ nguyên cơ chế `prepare → bindParam → execute`.
✅ Tái sử dụng: Có thể gọi class này ở nhiều nơi khác (trang admin, API, CLI, …).

Bạn có muốn mình **mở rộng class này thêm một phương thức `findUserByEmail()`** (để lấy thông tin chi tiết 1 người dùng) và **thêm cơ chế xử lý lỗi – trả về JSON cho API** không?
Điều đó sẽ biến lớp này thành **một module hoàn chỉnh có thể dùng cho REST API PHP**.

Hoàn hảo 🎯 — bây giờ ta sẽ **nâng cấp lớp `UserModel`** để:

1. Có thêm phương thức `findUserByEmail()` (tìm 1 người dùng cụ thể).
2. Có cơ chế **xử lý lỗi tập trung** và **trả về JSON** — rất hữu ích khi bạn muốn xây dựng **API RESTful PHP**.

🧱 1️⃣ Phiên bản hoàn chỉnh: `UserModel.php`

```php
<?php
class UserModel {
    private $pdo;

    // ------------------------
    // 🔷 Hàm khởi tạo (kết nối PDO)
    // ------------------------
    public function __construct($host, $dbname, $username, $password) {
        $dsn = "mysql:host=$host;dbname=$dbname;charset=utf8";

        try {
            $this->pdo = new PDO($dsn, $username, $password);
            $this->pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            $this->returnJsonError("Kết nối CSDL thất bại", $e->getMessage());
            exit;
        }
    }

    // ------------------------
    // 🔷 Hàm hỗ trợ: Trả về JSON
    // ------------------------
    private function returnJson($status, $message, $data = null) {
        header('Content-Type: application/json; charset=utf-8');
        echo json_encode([
            "status" => $status,
            "message" => $message,
            "data" => $data
        ], JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
    }

    // ------------------------
    // 🔷 Hàm hỗ trợ: Trả về JSON lỗi
    // ------------------------
    private function returnJsonError($message, $errorDetail = "") {
        $this->returnJson("error", $message, ["error_detail" => $errorDetail]);
    }
```

```php
    // ------------------------
    // 🟢 CREATE - Thêm người dùng
    // ------------------------
    public function addUser($name, $email) {
        try {
            $sql = "INSERT INTO users (name, email) VALUES (:name, :email)";
            $stmt = $this->pdo->prepare($sql);
            $stmt->bindParam(":name", $name, PDO::PARAM_STR);
            $stmt->bindParam(":email", $email, PDO::PARAM_STR);
            $stmt->execute();

            $this->returnJson("success", "Đã thêm người dùng thành công!", [
                "name" => $name,
                "email" => $email
            ]);
        } catch (PDOException $e) {
            $this->returnJsonError("Không thể thêm người dùng", $e->getMessage());
        }
    }

    // ------------------------
    // 🟢 READ - Lấy tất cả người dùng
    // ------------------------
    public function getUsers() {
        try {
            $stmt = $this->pdo->query("SELECT * FROM users ORDER BY id DESC");
            $users = $stmt->fetchAll(PDO::FETCH_ASSOC);
            $this->returnJson("success", "Lấy danh sách người dùng thành công", $users);
        } catch (PDOException $e) {
            $this->returnJsonError("Không thể lấy danh sách người dùng", $e-
>getMessage());
        }
    }

    // ------------------------
    // 🟢 FIND - Tìm người dùng theo email
    // ------------------------
    public function findUserByEmail($email) {
        try {
            $sql = "SELECT * FROM users WHERE email = :email";
            $stmt = $this->pdo->prepare($sql);
            $stmt->bindParam(":email", $email, PDO::PARAM_STR);
            $stmt->execute();
            $user = $stmt->fetch(PDO::FETCH_ASSOC);

            if ($user) {
                $this->returnJson("success", "Đã tìm thấy người dùng", $user);
            } else {
```

```
                $this->returnJson("error", "Không tìm thấy người dùng với email: $email");
            }
        } catch (PDOException $e) {
            $this->returnJsonError("Không thể tìm người dùng", $e->getMessage());
        }
    }

    // ------------------------
    // 🟢 UPDATE - Cập nhật tên người dùng theo email
    // ------------------------
    public function updateUser($email, $newName) {
        try {
            $sql = "UPDATE users SET name = :name WHERE email = :email";
            $stmt = $this->pdo->prepare($sql);
            $stmt->bindParam(":name", $newName, PDO::PARAM_STR);
            $stmt->bindParam(":email", $email, PDO::PARAM_STR);
            $stmt->execute();

            if ($stmt->rowCount() > 0) {
                $this->returnJson("success", "Cập nhật người dùng thành công", [
                    "email" => $email,
                    "new_name" => $newName
                ]);
            } else {
                $this->returnJson("error", "Không có người dùng nào được cập nhật (email không tồn tại)");
            }
        } catch (PDOException $e) {
            $this->returnJsonError("Không thể cập nhật người dùng", $e->getMessage());
        }
    }

    // ------------------------
    // 🟢 DELETE - Xóa người dùng theo email
    // ------------------------
    public function deleteUser($email) {
        try {
            $sql = "DELETE FROM users WHERE email = :email";
            $stmt = $this->pdo->prepare($sql);
            $stmt->bindParam(":email", $email, PDO::PARAM_STR);
            $stmt->execute();

            if ($stmt->rowCount() > 0) {
                $this->returnJson("success", "Xóa người dùng thành công", ["email" => $email]);
            } else {
                $this->returnJson("error", "Không tìm that người dùng để xóa");
```

```
            }
        } catch (PDOException $e) {
            $this->returnJsonError("Không thể xóa người dùng", $e->getMessage());
        }
    }
}
?>
```

⚙️ 2️⃣ File chạy thử: `api_test.php`

Ví dụ đơn giản giả lập các thao tác API bằng URL hoặc form.

```
<?php
require_once "UserModel.php";

// Khởi tạo kết nối
$userModel = new UserModel("localhost", "testdb", "root", "");

// Giả lập hành động từ tham số ?action=
$action = $_GET['action'] ?? 'list';
switch ($action) {
    case 'add':
        $userModel->addUser("Nguyễn Văn A", "vana@example.com");
        break;
    case 'list':
        $userModel->getUsers();
        break;
    case 'find':
        $email = $_GET['email'] ?? 'vana@example.com';
        $userModel->findUserByEmail($email);
        break;
    case 'update':
        $userModel->updateUser("vana@example.com", "Nguyễn Văn C");
        break;
    case 'delete':
        $userModel->deleteUser("vana@example.com");
        break;
    default:
        echo json_encode(["status" => "error", "message" => "Hành động không hợp lệ"]);
}
?>
```

🧪 Cách chạy thử

Giả sử bạn lưu các file trong thư mục `htdocs/pdo_api/`, bạn có thể truy cập các URL sau:

Chức năng	URL chạy thử	Kết quả JSON
Lấy danh sách	`http://localhost/pdo_api/api_test.php?action=list`	Danh sách người dùng
Thêm người dùng	`http://localhost/pdo_api/api_test.php?action=add`	JSON báo thêm thành công
Tìm người dùng	`http://localhost/pdo_api/api_test.php?action=find&email=vana@example.com`	Trả về thông tin user
Cập nhật	`http://localhost/pdo_api/api_test.php?action=update`	JSON báo cập nhật
Xóa	`http://localhost/pdo_api/api_test.php?action=delete`	JSON báo xóa

📦 Mẫu JSON trả về

✅ Thành công

```
{
  "status": "success",
  "message": "Đã tìm thấy người dùng",
  "data": {
    "id": 1,
    "name": "Nguyễn Văn A",
    "email": "vana@example.com"
  }
}
```

❌ Thất bại

```
{
  "status": "error",
  "message": "Không tìm thấy người dùng với email: vana@example.com"
}
```

💡 Lợi ích của phiên bản này

✅ Có thể dùng trực tiếp làm **REST API PHP** (Frontend hoặc Postman gọi đều được).
✅ Mọi phản hồi đều là **JSON chuẩn**, dễ xử lý bằng JavaScript.
✅ Có cơ chế **try-catch** xử lý lỗi PDO rõ ràng.
✅ Dễ mở rộng thêm các endpoint khác (VD: `searchUserByName`, `countUsers`, `paginateUsers`, ...).
