Extraits de code commentés

1. Traitement de la connexion utilisateur.

```php
// Classe pour gérer l'authentification des utilisateurs
class Authentification {
    private $pdo; // Stocke la connexion PDO à la base de données

    public function __construct($pdo) {
        $this->pdo = $pdo; // Initialise la connexion PDO
    }

    // Fonction pour connecter un utilisateur avec email et mot de passe
    public function connecterUtilisateur($email, $password) {
        // On nettoie l'email pour éviter les soucis
        $email = trim($email);

        // Prépare la requête pour aller chercher l'utilisateur dans la base de données
        $sql = "SELECT * FROM users WHERE mail = :email";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute(['email' => $email]);

        // On regarde si on a trouvé un utilisateur avec cet email
        if ($stmt->rowCount() > 0) {
            $data = $stmt->fetch(); // On récupère les informations de l'utilisateur

            // On vérifie si le mot de passe donné correspond au mot de passe enregistré (qui est caché)
            if (password_verify($password, $data['mp'])) {
                // Si tout est bon, on se souvient de l'utilisateur en stockant des infos dans la session
                $_SESSION['email'] = $email;
                $_SESSION['role'] = $data['role'];
                $_SESSION['prenom'] = $data['prenom'];
                $_SESSION['connecte'] = true; // On ajoute une variable pour indiquer que l'utilisateur est connecté

                // On redirige l'utilisateur selon son rôle
                switch ($data['role']) {
                    case 'admin':
                        header('Location: admin/admin.php');
                        break;
                    case 'enseignant':
                        header('Location: prof.php');
                        break;
                    case 'etudiant':
                        header('Location: eleve.php');
                        break;
                    default:
                        echo "Rôle inconnu.";
                        exit();
                }
                exit();
            } else {
                // Mot de passe incorrect
                return "Mot de passe incorrect.";
            }
        } else {
            // Utilisateur non trouvé
            return "Utilisateur non trouvé.";
        }
    }
}
```

2. Utilisateur.

2.1. Création d’un utilisateur.

```php
// Traite l'inscription d'un nouvel utilisateur
private function processRegistration() {
    if (isset($_POST['ok'])) {
        $name = htmlspecialchars($_POST['name']);
        $firstname = htmlspecialchars($_POST['firstname']);
        $mail = filter_var($_POST['mail'], FILTER_SANITIZE_EMAIL);
        $password = $_POST['password'];
        $confirmPassword = $_POST['confirmpassword'];

        // Vérifie si l'email existe déjà
        $checkEmail = $this->pdo->prepare("SELECT COUNT(*) FROM users WHERE mail = ?");
        $checkEmail->execute([$mail]);
        if ($checkEmail->fetchColumn() > 0) {
            $this->message = "Cet email est déjà utilisé.";
        }
        // Vérifie que les mots de passe correspondent
        elseif ($password !== $confirmPassword) {
            $this->message = "Les mots de passe ne correspondent pas.";
        }
        // Vérifie la complexité du mot de passe
        elseif (strlen($password) < 8) {
            $this->message = "Le mot de passe doit contenir au moins 8 caractères.";
        } else {
            $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
            $sql = "INSERT INTO users (Nom, Prenom, mail, mp, role) VALUES (?, ?, ?, ?, 'etudiant')";
            $stmt = $this->pdo->prepare($sql);

            if ($stmt->execute([$name, $firstname, $mail, $hashedPassword])) {
                $this->message = "Utilisateur enregistré avec succès.";
            } else {
                $this->message = "Erreur lors de l'enregistrement de l'utilisateur.";
            }
        }
    }
}
```

2.2. Modification d’un utilisateur.

```php
// Traitement du formulaire de modification
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['update'])) {
    $userId = filter_var($_POST['userId'], FILTER_SANITIZE_NUMBER_INT);
    $nom = htmlspecialchars($_POST['name']);
    $prenom = htmlspecialchars($_POST['firstname']);
    $email = filter_var($_POST['mail'], FILTER_SANITIZE_EMAIL);
    $role = htmlspecialchars($_POST['role']);
    $classeId = filter_var($_POST['classe_id'], FILTER_SANITIZE_NUMBER_INT) ?: null;

    // Vérifier si l'email existe déjà pour un autre utilisateur
    $checkEmail = $pdo->prepare("SELECT COUNT(*) FROM users WHERE mail = ? AND IdUsers != ?");
    $checkEmail->execute([$email, $userId]);

    if ($checkEmail->fetchColumn() > 0) {
        $message = "Cet email est déjà utilisé par un autre utilisateur.";
    } else {
        // Vérifier si le mot de passe doit être mis à jour
        if (!empty($_POST['password']) && !empty($_POST['confirmpassword'])) {
            $password = $_POST['password'];
            $confirmPassword = $_POST['confirmpassword'];

            if ($password !== $confirmPassword) {
                $message = "Les mots de passe ne correspondent pas.";
            } elseif (strlen($password) < 8) {
                $message = "Le mot de passe doit contenir au moins 8 caractères.";
            } else {
                // Mise à jour avec nouveau mot de passe
                $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
                $sql = "UPDATE users SET Nom = ?, Prenom = ?, mail = ?, mp = ?, role = ?, classe_id = ? WHERE IdUsers = ?";
                $stmt = $pdo->prepare($sql);

                if ($stmt->execute([$nom, $prenom, $email, $hashedPassword, $role, $classeId, $userId])) {
                    $message = "Utilisateur mis à jour avec succès.";
                    // Récupérer les données mises à jour
                    $stmt = $pdo->prepare("SELECT * FROM users WHERE IdUsers = ?");
                    $stmt->execute([$userId]);
                    $userData = $stmt->fetch(PDO::FETCH_ASSOC);
                } else {
                    $message = "Erreur lors de la mise à jour de l'utilisateur.";
                }
            }
        } else {
            // Mise à jour sans changer le mot de passe
            $sql = "UPDATE users SET Nom = ?, Prenom = ?, mail = ?, role = ?, classe_id = ? WHERE IdUsers = ?";
            $stmt = $pdo->prepare($sql);

            if ($stmt->execute([$nom, $prenom, $email, $role, $classeId, $userId])) {
                $message = "Utilisateur mis à jour avec succès.";
                // Récupérer les données mises à jour
                $stmt = $pdo->prepare("SELECT * FROM users WHERE IdUsers = ?");
                $stmt->execute([$userId]);
                $userData = $stmt->fetch(PDO::FETCH_ASSOC);
            } else {
                $message = "Erreur lors de la mise à jour de l'utilisateur.";
            }
        }
    }
}
```

2.3. Suppression d’un utilisateur.

```php
<?php
include_once "connect_ddb.php";

if (isset($_GET['id'])) {
    $user_id = $_GET['id'];

    try {
        // Supprimer d'abord les enregistrements dans `planning` liés à l'utilisateur
        $sql_planning = "DELETE FROM planning WHERE prof_id = :id";
        $stmt_planning = $pdo->prepare($sql_planning);
        $stmt_planning->execute(['id' => $user_id]);

        // Ensuite, supprimer l'utilisateur
        $sql_user = "DELETE FROM users WHERE IdUsers = :id";
        $stmt_user = $pdo->prepare($sql_user);
        $stmt_user->execute(['id' => $user_id]);

        if ($stmt_user) {
            header("location:admin.php?message=DeleteSuccess");
        } else {
            header("location:admin.php?message=DeleteFail");
        }
    } catch (PDOException $e) {
        echo "Erreur : " . $e->getMessage();
    }
} else {
    header("location:admin.php?message=NoID");
}
?>
```

3. Matière.

3.1. Création d'une matière.

```php
// Crée une nouvelle matière dans la base de données
public function createMatiere($name) {
    // Vérifie si le nom de la matière est fourni
    if (!empty($name)) {
        try {
            // Prépare et exécute la requête d'insertion
            $sql = "INSERT INTO matiere (Name) VALUES (:subjectName)";
            $stmt = $this->pdo->prepare($sql);
            $stmt->execute(['subjectName' => htmlspecialchars($name)]);
            // Enregistre un message de succès dans la session
            $_SESSION['message'] = "<p class='text-success text-center'>Matière créée avec succès.</p>";
        } catch (PDOException $e) {
            // Enregistre un message d'erreur dans la session en cas d'exception
            $_SESSION['message'] = "<p class='text-danger text-center'>Erreur lors de la création de la matière.</p>";
        }
    } else {
        // Enregistre un message d'erreur si le nom de la matière est vide
        $_SESSION['message'] = "<p class='text-danger text-center'>Le nom de la matière est requis.</p>";
    }
    // Redirige pour éviter la double soumission du formulaire
    header("Location: adminmatiere.php");
    exit();
}
```

3.2. Modification d'une matière.

```php
    public function modifierMatiere($id, $nom) {
        $sql = "UPDATE matiere SET Name = :name WHERE id = :id";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute(['name' => htmlspecialchars($nom), 'id' => $id]);
    }
}

$matiereManager = new MatiereManager($pdo);
$message = "";
$matiere = null;

// Gestion de la soumission du formulaire de modification
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['updateMatiere'])) {
    if (isset($_POST['matiereId']) && is_numeric($_POST['matiereId']) && !empty($_POST['matiereName'])) {
        $matiereIdToUpdate = $_POST['matiereId'];
        $newMatiereName = $_POST['matiereName'];

        if ($matiereManager->modifierMatiere($matiereIdToUpdate, $newMatiereName)) {
            $message = "<p class='text-success text-center'>Matière mise à jour avec succès.</p>";
            // Récupérer la matière mise à jour pour afficher les nouvelles valeurs
            $matiere = $matiereManager->getMatiereById($matiereIdToUpdate);
        } else {
            $message = "<p class='text-danger text-center'>Erreur lors de la mise à jour de la matière.</p>"
        }
    } else {
        $message = "<p class='text-danger text-center'>Le nom de la matière est requis.</p>";
    }
}

// Récupérer l'ID de la matière à modifier depuis l'URL AU CHARGEMENT INITIAL
if (isset($_GET['id']) && is_numeric($_GET['id'])) {
    $matiereId = $_GET['id'];
    $matiere = $matiereManager->getMatiereById($matiereId);

    if (!$matiere) {
        $message = "<p class='text-danger text-center'>Matière non trouvée.</p>";

    }
} elseif (empty($message) && $_SERVER['REQUEST_METHOD'] != 'POST') {
    $message = "<p class='text-danger text-center'>ID de matière invalide.</p>";
}
```

3.3. Suppression d’une matière.

```php
include_once "connect_ddb.php"; // Vérifiez que la connexion fonctionne correctement

// Vérifiez si une demande de suppression a été envoyée
if (isset($_GET['delete_id'])) {
    $deleteId = intval($_GET['delete_id']); // Convertir en entier pour sécuriser l'entrée

    // Préparer et exécuter la requête de suppression
    $sql = "DELETE FROM matiere WHERE id = :id";
    $stmt = $pdo->prepare($sql);

    if ($stmt->execute(['id' => $deleteId])) {
        // Rediriger avec succès
        header("Location: adminmatiere.php?msg=success");
        exit();
    } else {
        // Rediriger avec un message d'erreur
        header("Location: adminmatiere.php?msg=error");
        exit();
    }
} else {
    // Si aucun ID n'est fourni, redirection directe
    header("Location: adminmatiere.php");
    exit();
}
?>
```

4. Classe.

4.1. Création d’une classe.

```php
// Création d'une instance du gestionnaire de classes
$classeManager = new ClasseManager($pdo);
$message = "";

// Gestion de l'ajout d'une classe via le formulaire
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['createClass'])) {
    if (!empty($_POST['className'])) {
        if ($classeManager->ajouterClasse($_POST['className'])) {
            $message = "<p class='text-success text-center'>Classe créée avec succès.</p>";
        } else {
            $message = "<p class='text-danger text-center'>Erreur lors de la création de la classe.</p>";
        }
    } else {
        $message = "<p class='text-danger text-center'>Le nom de la classe est requis.</p>";
    }
    // Redirige pour éviter la double soumission du formulaire
    header("Location: " . $_SERVER['PHP_SELF']);
    exit();
}

// Récupère la liste des classes pour affichage
$classes = $classeManager->getClasses();
```

4.2. Modification d’une classe.

```php
    public function modifierClasse($id, $nom) {
        $sql = "UPDATE classe SET Name = :name WHERE id = :id";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute(['name' => htmlspecialchars($nom), 'id' => $id]);
    }
}

// Création d'une instance de la classe
$classeManager = new ClasseManager($pdo);
$message = "";
$classe = null; // Initialiser $classe à null

// Gestion de la soumission du formulaire de modification
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['updateClass'])) {
    if (isset($_POST['classId']) && is_numeric($_POST['classId']) && !empty($_POST['className'])) {
        $classIdToUpdate = $_POST['classId'];
        $newClassName = $_POST['className'];

        if ($classeManager->modifierClasse($classIdToUpdate, $newClassName)) {
            $message = "<p class='text-success text-center'>Modification effectuée avec succès.</p>";
            // Ne plus rediriger immédiatement pour afficher le message
            // header("Location: " . $_SERVER['PHP_SELF'] . "?id=" . $classIdToUpdate);
            // exit();
            // Récupérer la classe mise à jour pour afficher les nouvelles valeurs
            $classe = $classeManager->getClassById($classIdToUpdate);
        } else {
            $message = "<p class='text-danger text-center'>Erreur lors de la mise à jour de la classe.</p>";
        }
    } else {
        $message = "<p class='text-danger text-center'>Le nom de la classe est requis.</p>";
    }
}
```

4.3. Suppression d’une classe.

```php
// Vérifiez si une demande de suppression a été envoyée
if (isset($_GET['delete_id'])) {
    $deleteId = intval($_GET['delete_id']); // Convertir en entier pour sécuriser l'entrée

    // Préparer et exécuter la requête de suppression
    $sql = "DELETE FROM classe WHERE id = :id";
    $stmt = $pdo->prepare($sql);

    if ($stmt->execute(['id' => $deleteId])) {
        // Rediriger avec succès
        header("Location: classeadmin.php?msg=success");
        exit();
    } else {
        // Rediriger avec un message d'erreur
        header("Location: classeadmin.php?msg=error");
        exit();
    }
} else {
    // Si aucun ID n'est fourni, redirection directe
    header("Location: classeadmin.php");
    exit();
}
?>
```

5. Cours.

5.1. Création d'un cours.

```php
class GestionCoursAdmin {
    private $pdo;

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

    public function getClasses() {
        $stmt = $this->pdo->query("SELECT Id, Name FROM classe");
        return $stmt->fetchAll();
    }

    public function getMatieres() {
        $stmt = $this->pdo->query("SELECT Id, Name FROM matiere");
        return $stmt->fetchAll();
    }

    public function getProfesseurs() {
        $stmt = $this->pdo->query("SELECT IdUsers, CONCAT(Nom, ' ', Prenom) AS FullName FROM users WHERE role = 'enseignant'");
        return $stmt->fetchAll();
    }

    public function creerCours($classe_id, $matiere_id, $prof_id, $debut, $fin) {
        $sql = "INSERT INTO planning (classe_id, matiere_id, prof_id, debut_du_cours, fin_du_cours) VALUES (:classe_id, :matiere_id, :prof_id, :debut, :fin)";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute(['classe_id' => $classe_id, 'matiere_id' => $matiere_id, 'prof_id' => $prof_id, 'debut' => $debut, 'fin' => $fin]);
    }

    public function getCours() {
        $sql = "SELECT p.Id, c.Name AS Classe, m.Name AS Matiere, CONCAT(u.Nom, ' ', u.Prenom) AS Professeur, p.debut_du_cours, p.fin_du_cours FROM planning p JOIN classe c ON p.classe_
        return $this->pdo->query($sql)->fetchAll();
    }

    public function supprimerCours($id) {
        $sql = "DELETE FROM planning WHERE Id = :id";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute(['id' => $id]);
    }
```

```php
    public function getCoursDetails($planning_id) {
        $sql = "SELECT p.Id, c.Name AS Classe, m.Name AS Matiere, CONCAT(prof.Nom, ' ', prof.Prenom) AS Professeur, p.debut_du_cours, p.fin_du_cours
                FROM planning p
                JOIN classe c ON p.classe_id = c.Id
                JOIN matiere m ON p.matiere_id = m.Id
                JOIN users prof ON p.prof_id = prof.IdUsers
                WHERE p.Id = :planning_id";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute(['planning_id' => $planning_id]);
        return $stmt->fetch();
    }

    public function getElevesStatutsParCours($planning_id) {
        $sql = "SELECT
                    u.idUsers,
                    u.Nom AS EleveNom,
                    u.Prenom AS ElevePrenom,
                    s.statut_presence
                FROM users u
                JOIN classe c ON u.classe_id = c.Id
                LEFT JOIN signature s ON u.idUsers = s.user_id AND s.planning_id = :planning_id
                WHERE u.role = 'etudiant'
                  AND u.classe_id = (SELECT classe_id FROM planning WHERE Id = :planning_id)
                ORDER BY u.Nom, u.Prenom";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute(['planning_id' => $planning_id]);
        return $stmt->fetchAll();
    }
}
```

5.2. Modification d’un cours.

```php
class GestionCours {
    private $pdo;

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

    public function getCoursById($id) {
        $sql = "SELECT Id, classe_id, matiere_id, prof_id, debut_du_cours, fin_du_cours FROM planning WHERE Id = :id";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute(['id' => $id]);
        return $stmt->fetch();
    }

    public function getClasses() {
        $stmt = $this->pdo->query("SELECT Id, Name FROM classe");
        return $stmt->fetchAll();
    }

    public function getMatieres() {
        $stmt = $this->pdo->query("SELECT Id, Name FROM matiere");
        return $stmt->fetchAll();
    }

    public function getProfesseurs() {
        $stmt = $this->pdo->query("SELECT IdUsers, CONCAT(Nom, ' ', Prenom) AS FullName FROM users WHERE role = 'enseignant'");
        return $stmt->fetchAll();
    }

    public function modifierCours($id, $classe_id, $matiere_id, $prof_id, $debut, $fin) {
        $sql = "UPDATE planning SET classe_id = :classe_id, matiere_id = :matiere_id, prof_id = :prof_id, debut_du_cours = :debut, fin_du_cours = :fin WHERE Id = :id";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute(['id' => $id, 'classe_id' => $classe_id, 'matiere_id' => $matiere_id, 'prof_id' => $prof_id, 'debut' => $debut, 'fin' => $fin]);
    }
}
```

5.3. Suppression d’un cours.

```php
// Vérifiez si une demande de suppression a été envoyée
if (isset($_GET['delete_id'])) {
    $deleteId = intval($_GET['delete_id']); // Convertir en entier pour sécuriser l'entrée

    // Préparer et exécuter la requête de suppression
    $sql = "DELETE FROM planning WHERE Id = :id"; // Utilisez Id et non id (erreur de casse)
    $stmt = $pdo->prepare($sql);

    if ($stmt->execute(['id' => $deleteId])) {
        // Rediriger avec succès
        header("Location: coursadmin.php?msg=success");
        exit();
    } else {
        // Rediriger avec un message d'erreur
        header("Location: coursadmin.php?msg=error");
        exit();
    }
} else {
    // Si aucun ID n'est fourni, redirection directe
    header("Location: coursadmin.php?msg=invalid_id");
    exit();
}
?>
```