Pertemuan 15: Aplikasi CRUD Lengkap

Overview

Pertemuan ini merupakan culmination dari seluruh course PHP OOP, dimana kita akan membangun aplikasi CRUD (Create, Read, Update, Delete) lengkap yang mengintegrasikan semua konsep yang telah dipelajari dari basic OOP hingga advanced design patterns.

Learning Objectives

Setelah mengikuti pertemuan ini, peserta akan mampu:

- Membangun aplikasi web lengkap menggunakan PHP OOP
- Mengimplementasikan arsitektur MVC (Model-View-Controller)
- Menggunakan PDO untuk database operations yang aman
- Menerapkan form validation dan error handling
- Mengimplementasikan authentication dan authorization
- Menggunakan session management dan security best practices
- Menerapkan file upload dan management
- Mengintegrasikan multiple design patterns dalam real application
- Melakukan testing dan debugging aplikasi

Materi

1. Project Overview: Task Management System

Fitur Utama Aplikasi:

- **User Management**: Registration, login, logout, profile management
- **Task Management**: Create, read, update, delete tasks
- **Category Management**: Organize tasks by categories
- **File Upload**: Attach files to tasks
- **Dashboard**: Overview dan statistics
- **Search & Filter**: Advanced task filtering
- **Responsive UI**: Bootstrap-based interface

Teknologi yang Digunakan:

- **Backend**: PHP 8+ dengan OOP principles
- **Database**: MySQL dengan PDO
- **Frontend**: HTML5, CSS3, Bootstrap 5, JavaScript
- **Security**: Password hashing, input validation, CSRF protection
- **Architecture**: MVC pattern dengan dependency injection

2. Database Design

ERD (Entity Relationship Diagram):

```sql
-- Users table
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    full_name VARCHAR(100) NOT NULL,
    avatar VARCHAR(255) NULL,
    role ENUM('admin', 'user') DEFAULT 'user',
    status ENUM('active', 'inactive') DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);

-- Categories table
CREATE TABLE categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT NULL,
    color VARCHAR(7) DEFAULT '#007bff',
    user_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Tasks table
CREATE TABLE tasks (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(200) NOT NULL,
    description TEXT NULL,
    status ENUM('todo', 'in_progress', 'completed') DEFAULT 'todo',
    priority ENUM('low', 'medium', 'high') DEFAULT 'medium',
    due_date DATE NULL,
    category_id INT NULL,
    user_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (category_id) REFERENCES categories(id) ON DELETE SET
NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Task attachments table
CREATE TABLE task_attachments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    task_id INT NOT NULL,
    filename VARCHAR(255) NOT NULL,
    original_name VARCHAR(255) NOT NULL,
    file_size INT NOT NULL,
    mime_type VARCHAR(100) NOT NULL,
```

```
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (task_id) REFERENCES tasks(id) ON DELETE CASCADE
);
```

3. Project Structure

Directory Organization:

```
pertemuan-15/
├── config/
│   ├── database.php
│   └── app.php
├── src/
│   ├── Models/
│   │   ├── User.php
│   │   ├── Task.php
│   │   ├── Category.php
│   │   └── TaskAttachment.php
│   ├── Controllers/
│   │   ├── BaseController.php
│   │   ├── AuthController.php
│   │   ├── TaskController.php
│   │   ├── CategoryController.php
│   │   └── DashboardController.php
│   ├── Services/
│   │   ├── AuthService.php
│   │   ├── TaskService.php
│   │   ├── FileUploadService.php
│   │   └── ValidationService.php
│   ├── Repositories/
│   │   ├── UserRepository.php
│   │   ├── TaskRepository.php
│   │   └── CategoryRepository.php
│   └── Core/
│       ├── Database.php
│       ├── Router.php
│       ├── Session.php
│       └── View.php
├── public/
│   ├── index.php
│   ├── assets/
│   │   ├── css/
│   │   ├── js/
│   │   └── uploads/
│   └── .htaccess
├── views/
│   ├── layouts/
│   │   ├── header.php
│   │   └── footer.php
│   ├── auth/
│   │   ├── login.php
```

```
│   │   └── register.php
│   ├── tasks/
│   │   ├── index.php
│   │   ├── create.php
│   │   ├── edit.php
│   │   └── view.php
│   └── dashboard/
│       └── index.php
└── composer.json
```

4. Core Architecture Components

A. Database Connection (Singleton Pattern)

```php
class Database
{
    private static ?self $instance = null;
    private PDO $connection;

    private function __construct()
    {
        // Database connection logic
    }

    public static function getInstance(): self
    {
        if (self::$instance === null) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    public function getConnection(): PDO
    {
        return $this->connection;
    }
}
```

B. Base Model (Active Record Pattern)

```php
abstract class BaseModel
{
    protected PDO $db;
    protected string $table;
    protected array $fillable = [];

    public function __construct()
    {
```

```
        $this->db = Database::getInstance()->getConnection();
    }

    abstract public function validate(array $data): array;

    public function create(array $data): int;
    public function find(int $id): ?array;
    public function update(int $id, array $data): bool;
    public function delete(int $id): bool;
    public function all(): array;
}
```

C. MVC Controller Pattern

```
abstract class BaseController
{
    protected View $view;
    protected Session $session;

    public function __construct()
    {
        $this->view = new View();
        $this->session = Session::getInstance();
    }

    protected function requireAuth(): void;
    protected function redirect(string $url): void;
    protected function jsonResponse(array $data, int $code = 200): void;
}
```

5. Key Features Implementation

A. Authentication System

- User registration dengan validation
- Secure login dengan password hashing
- Session management
- Role-based access control
- Password reset functionality

B. CRUD Operations

- **Create**: Form handling dengan validation
- **Read**: Data listing dengan pagination
- **Update**: Edit forms dengan pre-filled data
- **Delete**: Soft delete dengan confirmation

C. Advanced Features

- **Search & Filter**: Dynamic query building
- **File Upload**: Multiple file handling dengan security
- **Image Processing**: Thumbnail generation
- **Export**: CSV/PDF export functionality
- **API Endpoints**: RESTful API untuk mobile apps

6. Design Patterns Integration

Patterns yang Digunakan:

1. **Singleton**: Database connection, Session management
2. **Factory**: Model creation, Service instantiation
3. **Repository**: Data access abstraction
4. **Strategy**: Validation strategies, Export formats
5. **Observer**: Event handling, Notifications
6. **Decorator**: Request/Response enhancement
7. **Command**: Action dispatching
8. **Builder**: Query building, Form building

7. Security Implementation

Security Measures:

- **Input Validation**: Server-side validation untuk semua input
- **SQL Injection Prevention**: Prepared statements
- **XSS Protection**: Output escaping
- **CSRF Protection**: Token-based protection
- **File Upload Security**: Type checking, size limits
- **Password Security**: Bcrypt hashing
- **Session Security**: Secure session handling

8. Testing Strategy

Testing Approach:

- **Unit Testing**: Individual class testing
- **Integration Testing**: Component interaction testing
- **Functional Testing**: End-to-end user workflows
- **Security Testing**: Vulnerability assessment

9. Performance Optimization

Optimization Techniques:

- **Database Indexing**: Proper index creation
- **Query Optimization**: Efficient SQL queries
- **Caching**: Result caching, Session caching
- **File Optimization**: Image compression, Asset minification
- **Code Optimization**: Autoloading, Dependency injection

10. Deployment Considerations

Production Setup:

- **Environment Configuration**: Development vs Production
- **Error Handling**: Logging dan monitoring
- **Backup Strategy**: Regular database backups
- **SSL Certificate**: HTTPS implementation
- **Server Configuration**: Apache/Nginx setup

Praktikum

Lab 1: Database Setup

1. Create database dan tables
2. Insert sample data
3. Test database connections

Lab 2: Core Classes

1. Implement Database class
2. Create Base Model
3. Build User model

Lab 3: Authentication

1. Registration system
2. Login functionality
3. Session management

Lab 4: Task CRUD

1. Task creation form
2. Task listing dengan pagination
3. Task editing dan deletion

Lab 5: Advanced Features

1. File upload system
2. Search dan filtering
3. Dashboard dengan statistics

Lab 6: Security & Testing

1. Implement security measures
2. Write unit tests
3. Perform security testing

Studi Kasus

Case Study 1: E-Learning Platform

Adaptasi aplikasi task management menjadi e-learning platform:

- Course management
- Student enrollment
- Assignment submission
- Grade management

Case Study 2: Inventory Management

Modifikasi untuk sistem inventory:

- Product management
- Stock tracking
- Purchase orders
- Supplier management

Project Deliverables

Yang Harus Diselesaikan:

1. **Functional Application**: Working CRUD application
2. **Documentation**: Technical dan user documentation
3. **Test Cases**: Comprehensive testing
4. **Deployment**: Local deployment setup
5. **Presentation**: Demo aplikasi dan code review

Evaluation Criteria

Penilaian berdasarkan:

1. **Code Quality** (30%): Clean code, OOP principles, design patterns
2. **Functionality** (25%): All features working properly
3. **Security** (20%): Security measures implementation
4. **User Experience** (15%): Interface design dan usability
5. **Documentation** (10%): Code comments dan documentation

Resources Tambahan

Tools & Libraries:

- **Composer**: Dependency management
- **PHPUnit**: Unit testing framework
- **Twig**: Template engine (optional)
- **Monolog**: Logging library
- **PHPMailer**: Email functionality

Documentation:

- PHP Official Documentation
- PDO Documentation
- Bootstrap Documentation
- MySQL Documentation

Security Resources:

- OWASP Top 10
- PHP Security Best Practices
- Secure Coding Guidelines

Next Session Preview

Pertemuan 16 (UAS - Final Exam) akan berupa:

- **Comprehensive Exam**: Teori dan praktik OOP PHP
- **Project Defense**: Presentasi aplikasi CRUD
- **Code Review**: Peer review dan feedback
- **Course Evaluation**: Assessment dan feedback
- **Certification**: Certificate of completion

Pertemuan 15 ini adalah puncak dari seluruh pembelajaran, dimana semua konsep dari Basic OOP, Advanced OOP, SOLID Principles, Design Patterns, dan Best Practices akan diintegrasikan dalam satu aplikasi real-world yang lengkap!