Case Study: Online Job Board Using MySQL and MVC Architecture

Prepared by: Ismael, Gérard, Achraf

October 26, 2025

Project brief source:	T-WEB-501_project.pdf (uploaded by user).
Database technology:	MySQL
Application architecture:	MVC (Model–View–Controller)

Abstract

This case study presents the design and implementation of an online job board application using a MySQL relational database and an MVC architecture. It highlights how the system separates data management, business logic, and user interfaces to achieve modularity, maintainability, and scalability.

Contents

1. Executive summary

The project involves building an online job board that enables companies to post advertisements and job seekers to view and apply for positions. The application integrates a MySQL database with a modular MVC back-end structure and a responsive front-end.

The system ensures:

- Clean separation between database (Model), application logic (Controller), and interface (View).
- Dynamic front-end updates using API endpoints.
- Secure authentication and an admin dashboard for data management.

2. Architecture overview: MVC pattern

The MVC architecture is used to structure the project into three core components:

Model: Handles all interactions with the MySQL database (queries, CRUD operations, data validation).

View: Represents the user interface — HTML/CSS templates and front-end components.

Controller: Contains business logic, receives HTTP requests, interacts with Models, and selects the appropriate View to render.

This separation ensures modular development:

- Developers can modify the UI without changing back-end logic.
- Database operations remain consistent and secure through the Model layer.
- Controllers orchestrate all communication between the View and Model.

3. Technology stack

Programming language: PHP (Laravel / pure MVC PHP framework) or Node.js (Express MVC pattern).

Database: MySQL (relational database management system).

Front-end: HTML5, CSS3, JavaScript (and Bootstrap for responsive design).

Server: Apache or Nginx.

Authentication: Sessions or JWT for login management.

Testing: PHPUnit or Postman for endpoint validation.

4. Database design (MySQL)

4.1. Relational schema

The MySQL database stores jobs, companies, users, and applications.

Listing 1: MySQL Schema Example

```
CREATE TABLE companies (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  website VARCHAR(255),
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE advertisements (
  id INT AUTO_INCREMENT PRIMARY KEY,
  company_id INT NOT NULL,
  title VARCHAR(255) NOT NULL,
  short_description VARCHAR(512),
  full_description TEXT,
  wages VARCHAR(100),
  place VARCHAR(255),
  working_time VARCHAR(100),
  posted_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  expires_at TIMESTAMP NULL,
  FOREIGN KEY (company_id) REFERENCES companies(id)
);

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(150) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  role ENUM('user','admin') DEFAULT 'user',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE people (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT,
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  email VARCHAR(255),
  phone VARCHAR(50),
  FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE applications (
  id INT AUTO_INCREMENT PRIMARY KEY,
  advertisement_id INT NOT NULL,
  person_id INT NOT NULL,
  message TEXT,
  resume_url VARCHAR(255),
  applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (advertisement_id) REFERENCES advertisements(id),
  FOREIGN KEY (person_id) REFERENCES people(id)
);
```

5. Application flow (MVC logic)

1. The user sends an HTTP request (e.g., view all jobs or apply for one).
2. The Controller receives the request, interacts with the Model to retrieve or update data.

3. The Controller sends data to the View, which renders HTML back to the browser.

Example controller flow:

Listing 2: Controller Example

```
public function showJobs() {
  $jobs = JobModel::getAllJobs();
  require_once 'views/jobs/list.php';
}
```

6. Front-end and user interaction

- The home page lists available job offers from the database.
- Clicking "Learn more" fetches detailed job information dynamically.
- Authenticated users can click "Apply" to submit their application form.
- Form data is validated and inserted into the MySQL database via the Model.

7. Authentication and admin dashboard

- Registration and login pages are handled by the AuthController.
- Passwords are hashed before being stored in MySQL using `password_hash()`.
- Admin users can access CRUD interfaces for managing ads, users, and applications.

8. Security measures

- SQL queries use prepared statements (PDO) to prevent SQL injection.
- Authentication tokens or sessions prevent unauthorized access.
- Form inputs are validated both client-side and server-side.

9. Testing and deployment

- Each Controller and Model function is unit-tested for correctness.
- Database migrations ensure reproducibility between environments.
- Deployment uses Apache + PHP + MySQL on a Linux environment or Docker setup.

10. Conclusion

Using MySQL with an MVC model ensures clean separation between logic, data, and presentation layers. This architecture supports scalability, maintainability, and secure handling of user data. The resulting online job board provides a reliable platform for employers and applicants alike.