🚗 May-IT

Application Web de Gestion d'Auto-École

Documentation Technique Complète

Auteure	Youma DRAME
Formation	BTS SIO SLAM — École IRIS Paris
Période	Janvier 2026
Technologies	PHP 8, MVC-POO, MySQL, HTML5/CSS3, JavaScript
Type	Application web full-stack — Gestion d'auto-école
Portfolio	youmadrame.com

1. Contexte & Objectifs

May-IT est une application web complète destinée à la gestion administrative d'une auto-école. Le projet répond à un besoin réel de digitaliser la gestion des candidats, moniteurs, véhicules, leçons et paiements, avec un tableau de bord administrateur centralisé.

1.1 Problématique

Les auto-écoles traditionnelles gèrent leurs dossiers manuellement (papier ou tableurs). L'objectif était de créer une solution web intuitive permettant une gestion centralisée, sécurisée et accessible depuis n'importe quel navigateur.

1.2 Objectifs du projet

Objectifs fonctionnels	
►	Gérer les candidats : inscription, suivi, résultats
►	Gérer les moniteurs : profils, disponibilités, affectations
►	Gérer les véhicules et les formules tarifaires
►	Planifier les leçons de conduite
►	Proposer un quiz de code de la route interactif
►	Afficher des statistiques en temps réel via un dashboard
►	Sécuriser l'accès par authentification avec rôles

2. Architecture Technique

2.1 Stack technique

Front-end	HTML5, CSS3, JavaScript (vanilla)
Back-end	PHP 8 — Architecture MVC (Modèle-Vue-Contrôleur)
Base de données	MySQL / MariaDB
Serveur local	XAMPP (Apache + MySQL)
Versioning	Git / GitHub

2.2 Architecture MVC

L'application suit le patron de conception MVC (Modèle-Vue-Contrôleur) qui sépare clairement les responsabilités :

- Modèle (Model) : classes PHP gérant les requêtes SQL et la logique métier
- Vue (View) : fichiers HTML/PHP affichant les données à l'utilisateur
- Contrôleur (Controller) : scripts PHP traitant les actions utilisateur et reliant Modèle et Vue

2.3 Structure des fichiers

Organisation du projet
► index.php — Point d'entrée unique (routing)
► controllers/ — Contrôleurs (CandidatController, MoniteurController...)
► models/ — Modèles (Candidat, Moniteur, Vehicule, Formule...)
► views/ — Vues HTML/PHP par module
► assets/ — CSS, JavaScript, images
► config/ — Configuration base de données

3. Base de Données

3.1 Modèle de données

La base de données MySQL contient les tables suivantes, reliées par des clés étrangères :

Tables principales
► candidats (id, nom, prenom, email, telephone, date_naissance, statut)
► moniteurs (id, nom, prenom, email, telephone, permis_depuis, actif)
► vehicules (id, marque, modele, immatriculation, annee, disponible)
► formules (id, nom, description, heures, prix, type)
► lecons (id, id_candidat, id_moniteur, id_vehicule, date, heure, duree, statut)

►	paiements (id, id_candidat, id_formule, montant, date, mode_paiement, statut)
►	questions_code (id, question, reponse_a/b/c/d, bonne_reponse, categorie)
►	users (id, login, password_hash, role, id_reference)

3.2 Exemples de requêtes SQL

Requête — Statistiques du dashboard :

```
SELECT COUNT(*) AS nb_candidats FROM candidats WHERE statut='actif'; SELECT
SUM(montant) AS revenus FROM paiements WHERE MONTH(date)=MONTH(NOW()); SELECT COUNT(*)
AS lecons_aujourd_hui FROM lecons WHERE DATE(date)=CURDATE();
```

4. Fonctionnalités Développées

4.1 Dashboard Administrateur

Le tableau de bord affiche en temps réel :

- Nombre de candidats inscrits
- Nombre de moniteurs actifs
- Revenus du mois en cours
- Leçons planifiées aujourd'hui
- Formules disponibles
- Actions rapides : gérer candidats, moniteurs, planning

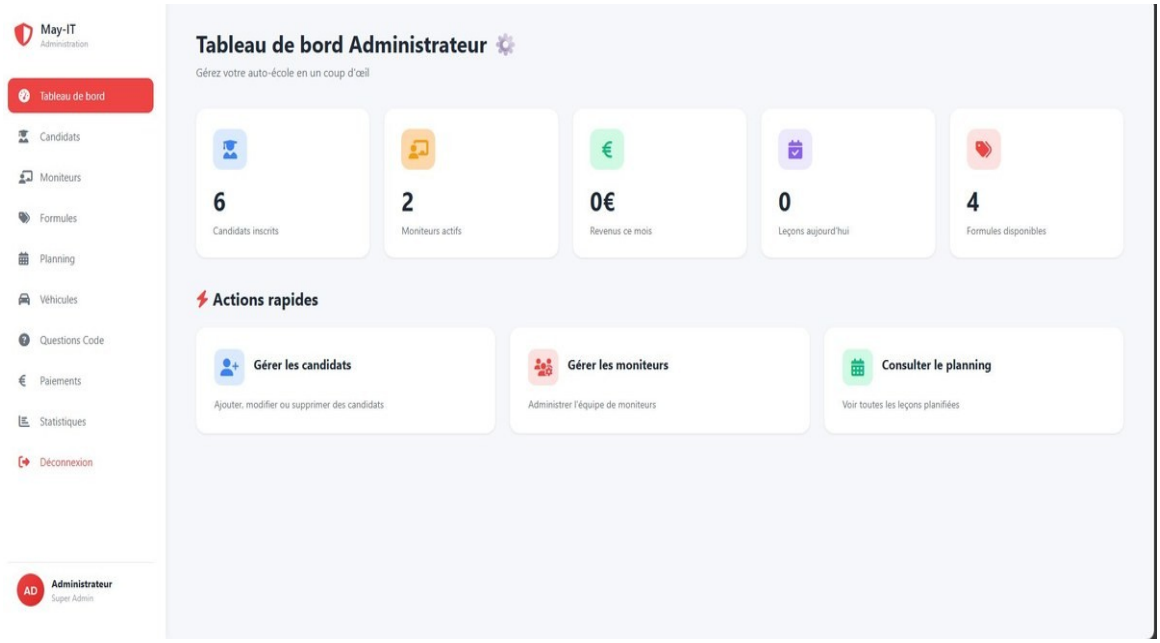

Fig. 1 — Tableau de bord administrateur May-IT

4.2 Statistiques et Analyses

Le module statistiques propose :

- Graphique d'inscriptions par mois (courbe)
- Répartition des formules choisies (camembert)

- Indicateurs : total candidats, moniteurs actifs, leçons ce mois, revenus potentiels

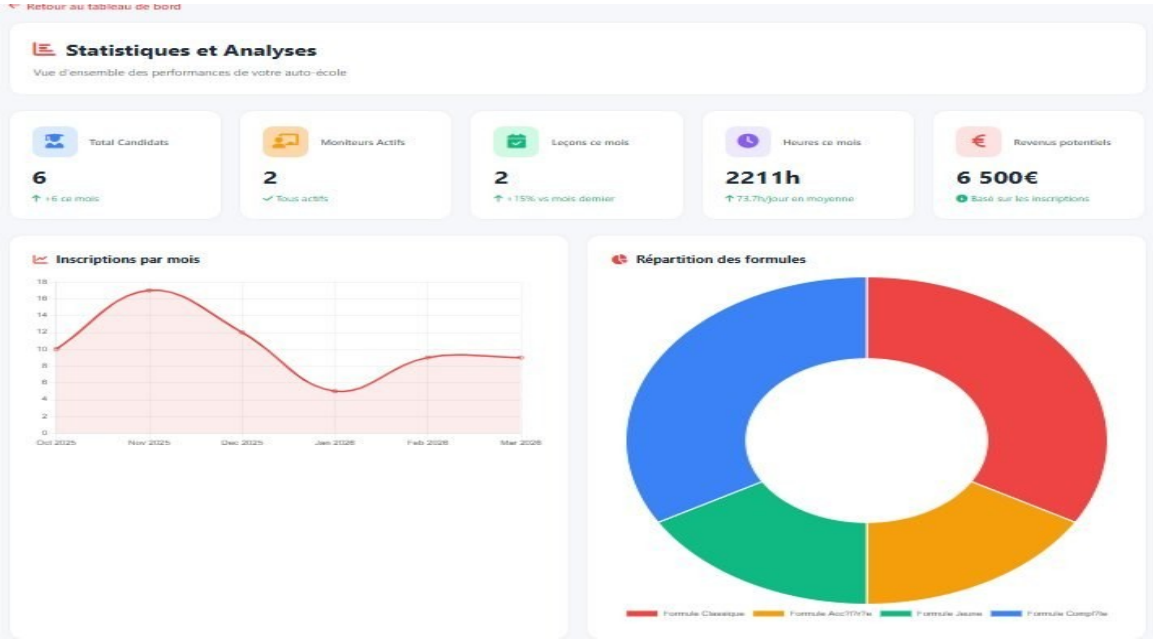

Fig. 2 — Module statistiques et analyses

4.3 Authentification & Sécurité

- Hachage des mots de passe (password_hash PHP)
- Sessions sécurisées avec régénération d'ID
- Contrôle des rôles (Super Admin / Moniteur / Candidat)
- Protection contre les injections SQL (requêtes préparées PDO)

5. Modélisation UML

5.1 Diagramme des cas d'utilisation

Acteurs & cas d'utilisation
► Administrateur : gérer candidats, moniteurs, véhicules, formules, planning, paiements, statistiques
► Moniteur : consulter son planning, voir ses candidats affectés
► Candidat : s'inscrire, répondre au quiz code, consulter son dossier

5.2 Diagramme de classes simplifié

Classes principales
► Candidat (id, nom, prenom, email, statut) — méthodes: inscrire(), modifier(), supprimer()
► Moniteur (id, nom, prenom, permis) — méthodes: affecter(), getPlanning()
► Vehicule (id, marque, immatriculation) — méthodes: reserver(), liberer()
► Lecon (id, date, duree, statut) — méthodes: planifier(), annuler()
► Paiement (id, montant, mode) — méthodes: enregistrer(), rembourser()

6. Tests Réalisés

Test fonctionnel 1	Création d'un candidat → Vérification en base de données ✅
Test fonctionnel 2	Authentification avec mauvais mot de passe → Accès refusé ✅
Test fonctionnel 3	Planification d'une leçon → Apparition dans le planning ✅
Test fonctionnel 4	Quiz code de la route → Calcul du score correct ✅
Test sécurité	Injection SQL dans formulaire → Bloquée par PDO ✅
Test responsive	Affichage sur mobile (375px) → Interface adaptée ✅

7. Bilan & Compétences Mobilisées

Compétences BTS SIO mobilisées	
►	Gérer le patrimoine informatique — gestion BDD, structure fichiers projet
►	Travailler en mode projet — planification, découpage en fonctionnalités
►	Mettre à disposition un service informatique — déploiement local XAMPP

Ce projet m'a permis de maîtriser l'architecture MVC en PHP, la gestion de sessions sécurisées, la conception de bases de données relationnelles complexes et la réalisation de tableaux de bord dynamiques avec statistiques visuelles.