Page de garde

Table des matières

[1 Analyse du cahier des charges 3](#_Toc196302385)

[1.1 Contexte 3](#_Toc196302386)

[1.2 Objectifs 3](#_Toc196302387)

[1.3 Enjeux et défis 3](#_Toc196302388)

[1.4 Technologies utilisées 3](#_Toc196302389)

[1.5 Plan du rapport 3](#_Toc196302390)

[2 Conception 3](#_Toc196302391)

[2.1 Planification initiale 3](#_Toc196302392)

[2.2 Architecture de l’application 3](#_Toc196302393)

[2.3 Base de données 3](#_Toc196302394)

[2.4 Maquette du site 3](#_Toc196302395)

[3 Développement et implémentation 4](#_Toc196302396)

[3.1 Mise en place de l’environnement 4](#_Toc196302397)

[3.2 Base de données 4](#_Toc196302398)

[4 Tests 4](#_Toc196302399)

[4.1 Plan des tests à effectuer 4](#_Toc196302400)

[4.2 Résultats des tests 4](#_Toc196302401)

[5 Conclusion 5](#_Toc196302402)

[6 Annexes 5](#_Toc196302403)

[6.1 Résumé du rapport du TPI / version succincte de la documentation 5](#_Toc196302404)

[6.2 Webographie 5](#_Toc196302405)

[6.3 Journal de travail 5](#_Toc196302406)

[6.4 Manuel d'Installation 5](#_Toc196302407)

[6.5 Manuel d'Utilisation 5](#_Toc196302408)

# Analyse du cahier des charges

Petit texte introductif

## Contexte

## Objectifs

## Enjeux et défis du projet

## Technologies utilisées

## Structure du rapport

# Conception

## Planification initiale

## Architecture de l’application

Le projet est basé sur une architecture MCV (Modèle -Vue – Contrôleur) qui m’a permis d’organiser proprement mon code en séparant les logiques. Cette séparation facilite la compréhension du code pour un développeur qui reprendrai le site et permet de diminuer la redondance de code.



1. L’utilisateur demande une page.
2. Le contrôleur, si besoin, fait appel au modèle pour récupérer les informations requises depuis la base de données.
3. Le contrôleur va appeler la vue qui correspond à la demande de l’utilisateur et lui fournit les données nécessaires.
4. La page HTML demandée est retournée.

Ce concept est structuré dans mon code de cette manière :

**Fichier racine**

index.php se situe à la racine du projet et est utilisé comme point d’entrée de l’application.

**Fichier Routes**

Situé dans le dossier routes il s’occupe de rediriger les requêtes de l’utilisateur vers le bon contrôleur en fonction des paramètres de l’URL ($\_GET['controller']).

**Dossier Controller**

Comme son nom l’indique ce dossier contient tous les différents contrôleurs de l’application et notamment le controller.php qui instancie la classe parente abstraite dont héritent tous les autres contrôleurs.

**Dossier Model**

Ce dossier contient notamment le fichier database.php qui centralise la gestion des connexions à la base de donnée ainsi que toutes les requêtes SQL qui seront utilisées et appelés par les repository présents eux aussi dans ce dossier.

**Dossier View**

Regroupe tous les fichiers contenant de l’html comme le footer, la nav et le header mais aussi les différentes pages.

**Dossier ressources**

Contient toutes les images du site, mes fichiers css et js mais aussi tout les fichiers nécessaires au thème SB Admin 2.

**Dossier SSO**

Regroupe les fichiers qui s’occupent de gérer la connexion EduVaud.  
  
J’ai choisi cette structure car j’ai déjà eut l’occasion de l’utiliser lors du deuxième projet de préparation au TPI et qu’elle m’avait été fournie lors du module traitant le MVC Web.

## Base de données

## Maquette du site

# Développement et implémentation

## Mise en place de l’environnement

## Base de données

# Tests

## Plan des tests à effectuer

## Résultats des tests

# Conclusion

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Webographie

## Journal de travail

## Manuel d'Installation

## Manuel d'Utilisation