**RAPPORT DE PROJET**

**CREATION D’UNE APPLICATION WEB DE RENDU DE DEVOIR**

**Tuteur du projet :**

Laurent JOSSE

**Membres de l’équipe :**

Abdoulaye SOW

Mamadou Hambaliou DIALLO

**SOMMAIRE**

1. INTRODUCTION
2. CONTEXTE DU PROJET
3. CONCEPTION
   1. Langages et choix des outils
   2. Base de données
   3. Différentes étapes du développement
      1. Programmation
      2. Design
   4. Résultats obtenus
   5. Problèmes rencontrés
4. ANNEXES
   1. Mode d’emploi pour la mise en production.
5. **INTRODUCTION :**

Dans le cadre du projet thématique (module C605) nous avons choisi comme sujet « Application web de rendu de devoir » consistant à la création d’une application web de gestion des rendus de devoir en ligne. Il permettra de centraliser en un et même endroit l’ensemble des devoirs. Ceci pour faciliter la tâche à la fois aux professeurs et aux étudiants qui ne sont parfois pas inscrits en présentiel ou qui sont en alternance.

1. **CONTEXTE DU PROJET :**

Pour la réalisation de ce projet, un cahier de charge a été définit et validé avec le tuteur dans lequel toutes les spécificités techniques ont été définies (les fonctionnalités, l’architecture, les choix d’implémentation et les risques liés au projet).

1. **CONCEPTION :**
   1. **Langages et choix des outils:**

Pour la réalisation de ce projet nous avons choisi les langages suivants :

* **HTML (HyperText Mark-Up Langage)** : tout d’abord pour tout projet web il faut penser à utiliser le HTML pour la partie statique du site web. C’est un langage dit de balisage ou de structuration permettant la conception des pages web avec des balises de formatage. Il constitue le squelette de tout projet web. Nous avons dans notre cas utilisé les systèmes de façade de laravel.
* **CSS (Cascading Style Sheet ou feuilles de style en cascade) Bootsrap :** c’est un langage informatique complétant le HTML. Le framework utilisé est Bootstrap qui est un framework CSS qui permet de faire de la mise en forme ou de faire le design d’un site web. Nous l’avons choisi car plus facile d’utilisation que le CSS traditionnel car contient des classes déjà prédéfinies et simples d’utilisation.
* **Java script (Vue js):** qui est un langage web dynamique principalement utilisé dans les pages web interactives. Il interviendra par exemple pour recharger le contenu d’une page sans avoir à l’actualiser, à afficher des popups, etc … nous l’avons choisi pour la dynamisation des certaines pages et pour l’interaction avec les utilisateurs.
* **PHP (Le framework Laravel):** qui est un langage de programmation web principalement utilisé pour produire des pages web dynamiques. Laravel est un framework open source écrit en PHP, orienté objet et qui respecte le principe de MVC (modèle-vue-contrôleur). C’est ce framwork qui sera utilisé pour la réalisation de ce projet.

Notre choix a porté sur lui à cause de quelques critères suivants : :

* + **Architecture MVC :** cette logique assure la clarté entre la couche logique et la présentation et permet d’augmenter la performance de l’application ;
  + **Excellente documentation :** le fondateur et développeur principal du framework Taylor Otwell écrit lui-même la documentation. Il est organisé de façon à permettre au programmeur de facilement l’assimiler.
  + **Une approche modulaire :** le framework est construit à partir de plus de 20 bibliothèque différentes. Il adopte des principes de PHP modernes, qui permettent aux développeurs de créer des applications web modulaires ;
  + **Système de routing :** il permet de créer des liens. De cette façon, on peut appeler directement les noms des routes et laravel va automatiquement insérer l’URL ;
  + **Eloquent :** Laravel fournit une magnifique et simple implémentation de Active Record. C’est un programme qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet.
* **Mysql :** qui est un système de gestion gratuit de bases de données. Son avantage est qu’il est gratuit et déjà préinstallé dans la plupart des serveurs d’hébergement web.

Pour les outils nous avons choisi les suivants :

* **Xampp :** c’est un ensemble de logiciels permettant de mettre facilement en place un serveur web local (apache, mysql, …), un serveur FTP et un serveur de messagerie électronique. Son avantage c’est sa souplesse, sa facilité et sa rapidité d’installation. C’est ce qui nous a poussé à le choisir.
* **Intellij IDEA :** comme IDE (environnement de développement intégré) de technologie java destiné au développement des logiciels informatiques. Nous l’avons choisi pour sa facilité d’utilisation (l’autocomplétion, la sauvegarde automatique, le versionning) et sa convivialité. Il a aussi un terminal intégré.
* **GitHub :** est service web d’hébergement et de gestion de développement de logiciels utilisation le logiciel de gestion de versions Git. Il est centré vers l’aspect social du développement et facilite en même temps le travail collaboratif. Nous avons choisi de l’utiliser pour faciliter le travail collaboratif et pour diminuer le risque de perdre notre projet.
* **Power Designer (anciennement Power AMC) :** qui est un logiciel informatique permettant de modéliser les traitements informatiques et leurs bases de données.
* **Maildev :** qui est un serveur SMTP permettant de simuler des envoi de mail aux développeurs quand ils sont en local.
  1. **Base de données :**

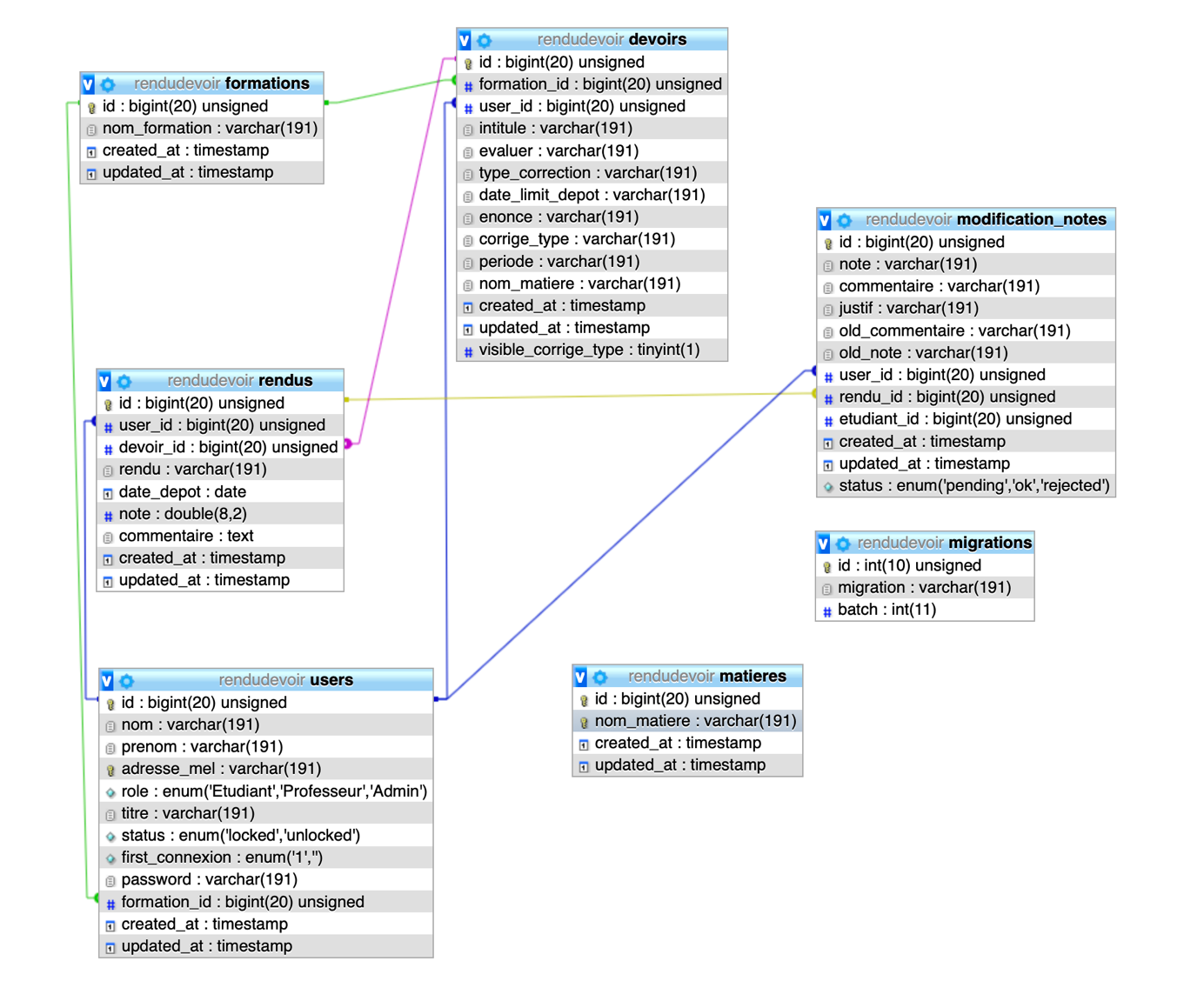
Comme pour tout vrai site web, la réalisation de ce projet a nécessité la création d’une base de données dans laquelle nous stockons les informations.

Dans notre cas il a fallu tout d’abord créer une table users qui contiendra l’ensemble des utilisateurs de la plateforme.

Ensuite nous avons ajouté les autres tables nécessaire au développement de l’application :

* La table formations : comme son nom l’indique contient les formations ;
* La table devoirs : comme son nom l’indique contient les devoirs ;
* La table rendus : contient les rendus de devoirs. C’est une table résultat de la liaison plusieurs à plusieurs entre les devoirs et les étudiants car un étudiant peut rendre un ou plusieurs devoirs et un devoirs peut être rendu par un ou plusieurs étudiants ;
* La table matieres : comme son nom l’indique contient la liste des matières ;
* La table migrations : contient l’ensemble des migrations pour faciliter le contrôle de la base de données permettant par exemple à l’équipe de partager le schema de la base de données.

Voir ci-dessous les tables avec leurs liens.



* 1. **Les différentes étapes du développement :**
     1. **La programmation :**

Nous avons dans un premier temps mis en place la structure du site selon l’architecture MVC (Models Views Controllers) où :

* **Le modèle (Model):** s’interface à la base qui contient les données à afficher ainsi que la logique en rapport avec les données comme les lectures, les relations et enregistrements et les modifications. Les modèles se trouvent dans le répertoire App et correspondent chacune à une table de notre base de données. Ils donnent plus de souplesse dans l’écriture des requête à travers Eloquent (*qui est un ORM pour Object Relational Mapping se plaçant en en interface entre un programme applicatif et une base de données relationnelle pour simuler un base de données orientée objet*) et la manipulation des données des tables. Eloquent est déjà intégré dans le framework laravel. Par convention les noms des tables sont déduits des noms des modèles sauf que les noms de tables sont en minuscules et au pluriel, les clés étrangères sont nommées en migrant le modèle au singulier suivi de « \_id » par exemple : un utilisateur a une relation avec devoir donc quand la clé primaire de users migre vers dervoirs et donne user\_id.

Pour la mise en relation des tables, nous utilisons les instructions :

* **belongsTo :** pour la mise en relation des tables dont la relation est de type 1 : 1 ;
* **belongsToMany :** pour la mise en relation des tables dont la relation est de type n : n.
* **hasMany :** pour la mise en relation des tables dont la relation est de type 1 : n . nous avons par exemple la table formations et la table users. Une formation peut avoir un ou plusieurs users.
  + **le modèle Devoir :**contient les méthodes formation, etudiants, … qui permettent de mettre en relation respectivement la table devoirs et les tables formations, etudiants ;
  + **le modèle Formation :** Contient une méthode users permettant de lister les étudiants inscrits dans une formation ;
  + **le modèle Matiere :** C’est l’interface d’accès et de manipulation (insertion, mise à jour du contenu, …) de la table matieres;
  + **Le modèle ModificationNote :** contient les méthodes professeur et etudiant qui permettent de mettre en relation respectivement la table modification\_notes et les tables professeurs et etudiants ;
  + **Le modèle Rendu :** contient une méthode devoir qui met en relation la table rendu et devoir.
  + **Le modèle User :** contient les méthodes formation, renduEtudiant, professeurs permettant de mettre en relation respectivement la table users aux tables formations, rendus, modification\_notes.
* **La vue (View):** contient la présentation de l’interface graphique. C’est la partie visible par l’utilisateur. Elle se sert du modèle et peut être un formulaire, un bouton, etc… Les vues sont dans le répertoire resources/views et sont groupées dans des sous répertoires .
  + **Le sous répertoire admin :** contient les vues aux quelles l’admin peut avoir accès. Autrement dit avec lesquelles il peut interagir. Il y a dans ce sous répertoire les sous répertoires:
    - approbation : qui redirige vers la vue d’approbation des demandes de modifications de notes émises par les professeurs;
    - formation : contenant la vue avec le formulaire de création d’une formation ;
    - home : contenant le menu de navigation de l’admin avec les liens redirigeant vers les formulaires respectifs. L’admin est redirigé dans ce répertoire dès qu’il est connecté ;
    - matiere : contenant les vues d’affichage, de création et de modification d’une matière ;
    - user : contenant les vues d’affichage, de création et de modifcation d’un utilisateur.
  + **Le sous répertoire auth :** contient les vues de connexion, d’inscription (disponible juste pour l’admin) et de réinitialisation de mot de passe à la première connexion.
  + **Le sous répertoire email :** contenant les envois de mail. Il contient les sous répertoires :
    - etudiant : pour l’envoi de mail à l’étudiant lorsque l’étudiant rend un devoir et l’envoi de mail lors que le professeur corrige son devoir.
    - users : pour l’envoi de mail à un utilisateur lorsque son compte est créé. On lui amène en même temps le mot de passe par défaut qui a été créé qu’il doit changer à sa première connexion.
    - prof : contient l’envoi de mail au professeur lors qu’un étudiant dépose un devoir.
  + **Le sous répertoire etudiant :** contenant les vues accessibles par l’étudiant. Il contient les sous répertoires :
    - Home : contenant la vue avec la liste des matières. C’est dans cette page qu’il est redirigé après connexion.
    - devoirs : contenant la vue avec la liste des devoirs de l’etudiant après qu’il ait cliqué sur une matière dans son home.
    - rendu : contenant la vue avec le formulaire pour rendre un devoir. Elle s’affiche lorsqu’un étudiant clique sur un devoir d’une matière.
  + **Le sous répertoire prof :** contenant les vues accessibles par le professeur. Il contient les sous répertoires :
    - Home : qui est la vue affichée lorsqu’il se connecte. Elle contient la liste des matières qu’il donne.
    - devoir : contenant les vues de création de devoir, la liste des devoirs d’une matière après qu’il ait cliqué sur une matière et la vue de correction lorsqu’il clique sur un devoir.
* **Le contrôleur (Controller):** contient la logique concernant les actions effectuées par l’utilisateur. C’est le module qui traite les actions des utilisateurs. Ils se trouvent dans le répertoire App/http/Controllers et Nous les avons regroupé par actions pouvant être effectuées.
  + **Admin :** permet d’administrer la plateforme dont entre autre la création de users, l’approbation de modifications de notes, la création de formations, ….
  + **Etudiant :** contenant la gestion des actions des utilisateurs dont entre autres l’affichage des devoir, le rendu des devoir. ;
  + **Prof :** contient la gestion des actions des professeurs dont entre autres la création de devoirs, la modification des notes, … ;
  + **Auth :** contient la gestion des authentifications dont entre autre la connexion, l’enregistrement, …
    1. **Le design :**

Pour le design nous sommes parti sur un design basique avec les couleurs et logos de l’université.

Pour la page de connexion nous avons utilisé une image de l’université comme fond avec un léger filtre bleu. Nous avons mis deux champs de saisi pour le login et mot de passe et un bouton pour se connecter avec le logo bleu de l’université en dessus des champs de saisi.

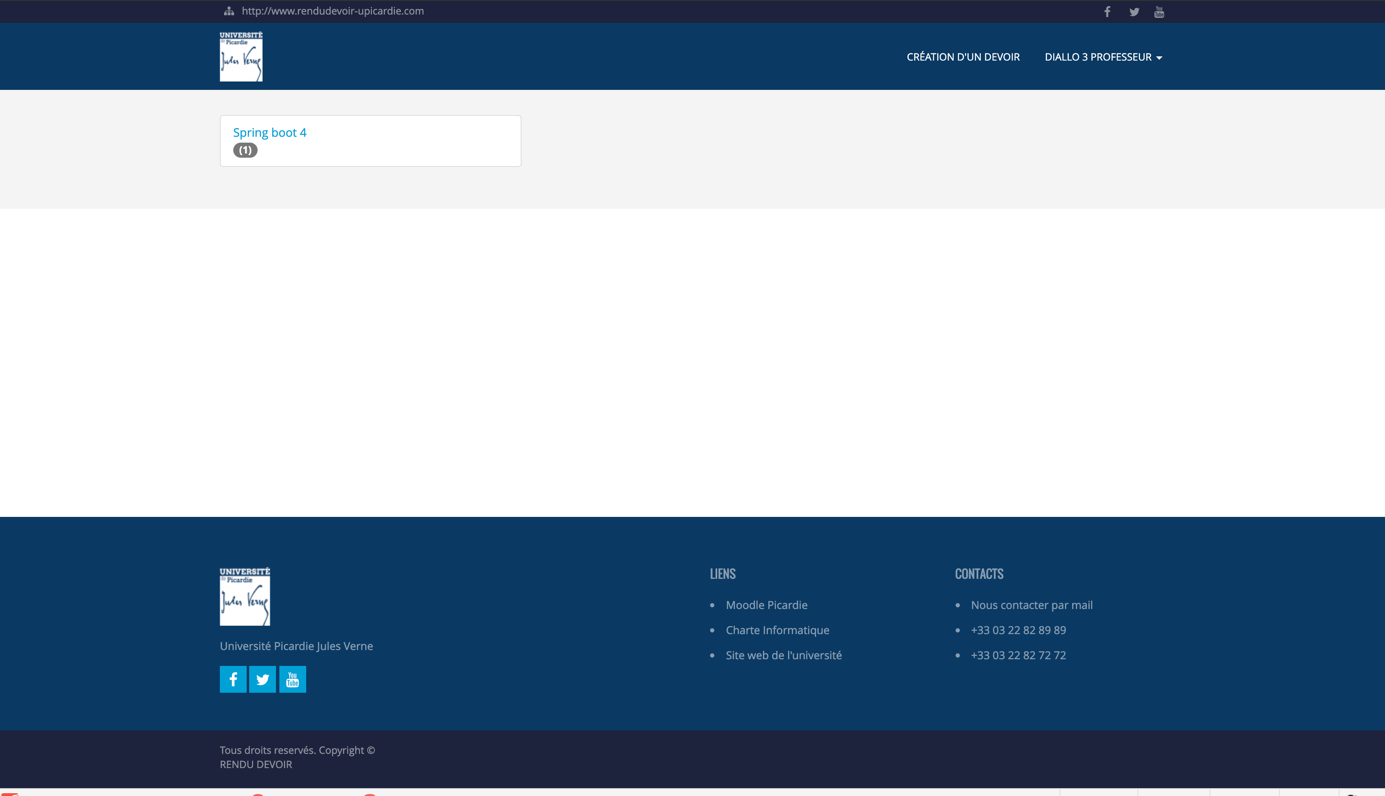
Pour toutes les autres pages, nous avons un entête bleu avec le logo de l’université à gauche de la page et à droite le nom de l’utilisateur connecté avec son rôle. Le contenu du milieu change en fonction de l’utilisateur connecté et des actions. Nous avons en bas de la page une bande bleu avec le logo de l’université à gauche en bas duquel nous avons les liens des pages de l’université sur certains réseaux sociaux, les liens utiles au centre et à droite les contacts :

* 1. **Résultats obtenu :**

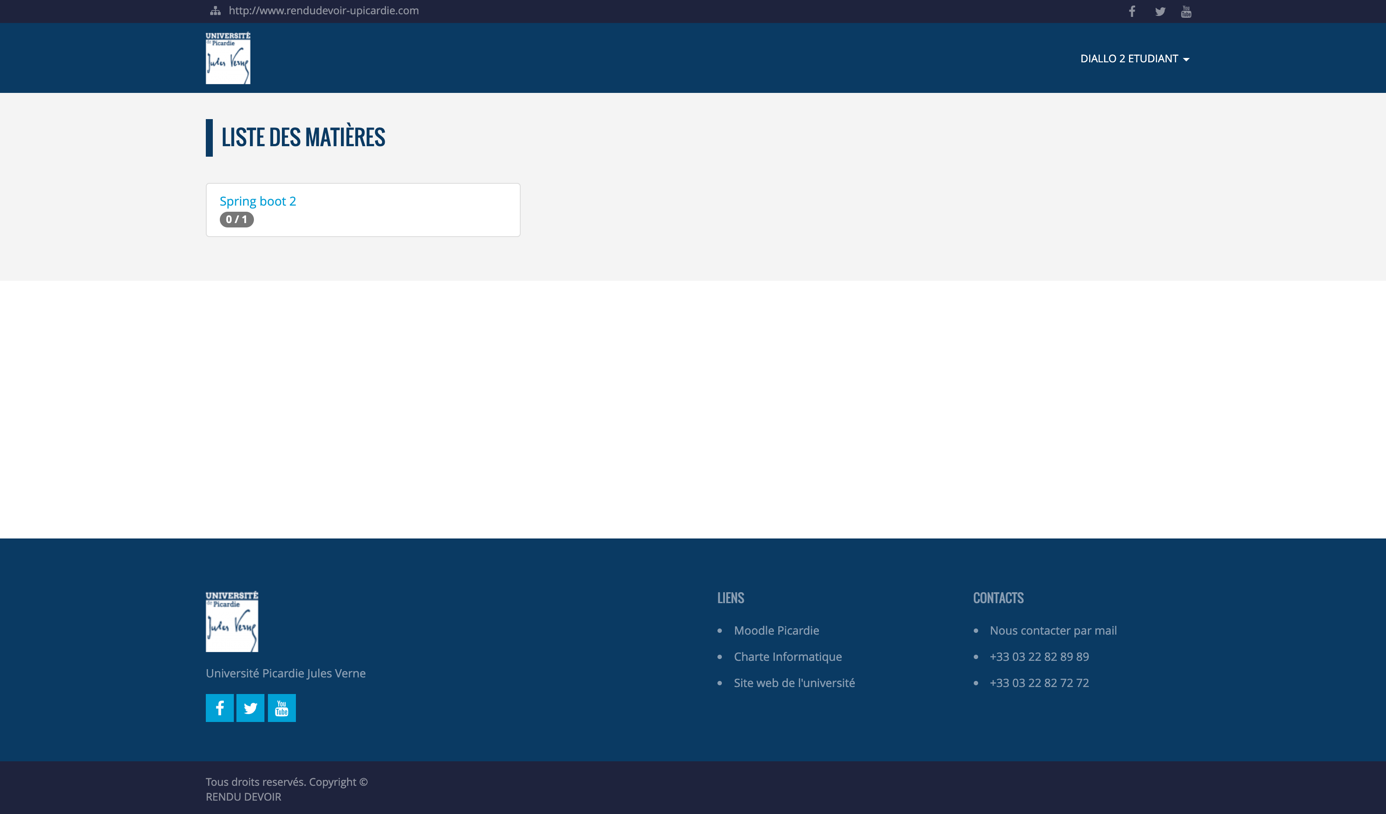
Page de connexion



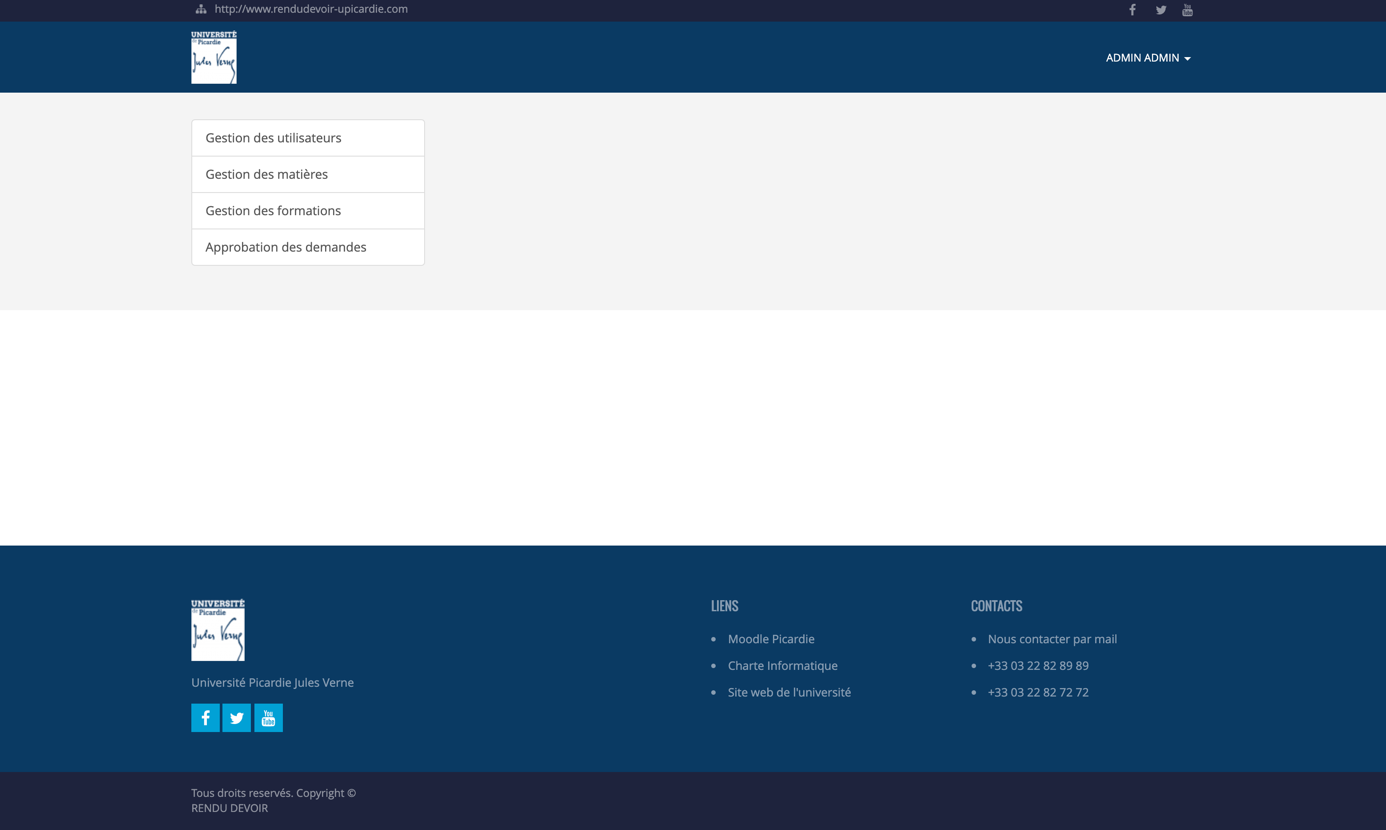
Page d’accueil pour un professeur :



Page d’accueil pour un étudiant :



Page d’accueil d’un administrateur :



* 1. **Problèmes rencontrés :**

Comme dans la majorité des développements, nous avons rencontré quelques problèmes qui sont entre autres :

* Ajout de tables supplémentaires dans la base de données : nous avons ajouté des tables supplémentaires dans la base de données pour gérer des fonctionnalité comme l’ajout de la fonctionnalité de modification d’une note ;
* Problème de taille de fichiers à uploader : nous avons été confronté au problème d’upload de fichier à cause de leur taille. Nous essayions de télécharger des fichiers de plus de 2 Mo alors que dans les paramètres par défaut la taille est fixée à 2 Mo. Nous avons donc affiché un message à l’utilisateur pour qu’il sache que la taille maximale est de 2 Mo. Nous avons choisi de ne pas modifier la taille pour ne pas que le serveur ne soit pas saturé.
* Cohabitation de vue js et de java script : à un moment du projet nous avons été obligé d’utiliser java script pour dynamiser plus certaines pages par exemple au niveau de la modification de la note pour la saisie de la nouvelle note et du commentaire. Nous avons par contre continué à utiliser vue js car il est déjà installé et configuré dans laravel.

1. **ANNEXES**
   1. **Mode d’emploi pour la mise en production :**

Le mode recommandé est celui de la configuration via git ou par transfert FTP (avec accès au terminal pour lancer des commandes SSH).

**Avec Transfert FTP (accès terminal) :**

1. Archiver le dossier local sous une archive zip.
2. Installer un outil de transfert de fichier comme Filezilla. Mettre les paramètres de connexion au serveur distant.
3. Se connecter sur le site via ssh et vérifier que git est bien installé avec git --version
4. Créer un répertoire portant le nom de l’App sur le home: cd ; mkdir nomApp ;
5. Copier le fichier vers ce répertoire sur le site et le de zipper.
6. Remplacer le dossier public\_html par un lien symbolique : rm –rf public\_html ; ln –s nomApp/public public\_html
7. Dans le dossier nomApp, ouvrir le fichier composer.json et se rassurer de la concordance de la case sur les liens.
8. Lancer composer install
9. Ouvrir le fichier .env et modifier les lignes APP\_ENV=local en APP\_ENV=production et APP\_DEBUG=true en APP\_DEBUG=false. Rajouter les autres infos de configuration de la base de données ainsi que l’émail.
10. Dans le dossier nomApp lancer composer up.

**Avec Git :** Je vous recommande ce lien qui est très explicite : https://nindohost.com/blog/application-laravel-hebergement-cpanel/