



ENSSAT
L A N N I O N

PROJET

PHP & CODEIGNITER

Création d'un serveur de vœux d'enseignements

Colin LEVERGER
Alexis ZANKOWITCH
Romain CARON

Informatique, Multimédia et
Réseaux
Promotion 2017

CONTACT : FirstLetterOfSurname SixFirstLettersOfLastName w/o accents [at] enssat [dot] fr

Destinataire : Jonathan CHEVELU

22 juin 2015

Table des matières

1	Introduction	2
2	Présentation du sujet	3
2.1	Prérequis	3
2.2	Le sujet	3
2.3	Gestion de projet	4
3	Cahier des spécifications fonctionnelles	6
3.1	L'utilisateur admin	6
3.2	Les fonctionnalités	6
3.3	Scenarii d'usage	8
3.3.1	Utilisateur "admin"	8
3.3.2	Utilisateur "normal"	8
4	Développement technique	10
4.1	Architecture globale de l'application	10
4.2	Quelques fonctions expliquées	11
4.2.1	le contrôleur "profile"	11
4.2.2	Recherche de modules	11
4.2.3	Les différents retours sur les données envoyées par l'utilisateur	13
5	Conclusion	14
5.1	Les limites de la solution employée	14
5.2	Recette	15
5.3	Bilan & Conclusion fonctionnelle	16
6	Annexes	17

1. Introduction

Dans le cadre du module Technologies Web proposé par l'ENSSAT formation Informatique, Multimédia et Réseaux en première année, nous avons été amenés à manipuler le framework "CodeIgniter" pour un projet noté. Nous devons en effet monter un projet à partir d'une commande d'un client fictif, et répondre au mieux à ses exigences. Cette année, il s'agissait de construire une application WEB permettant de gérer de manière intuitive et efficace les modules et leur attribution pour l'école ENSSAT. Les technologies que nous avons utilisées pour ce projet sont les suivantes :

- HTML & CSS
- CodeIgniter Framework PHP
- JavaScript & JQuery
- AJAX
- Bootstrap
- Git

Le présent document donnera un aperçu de la méthode que nous avons employée pour mener à bien ce projet. Il s'agit donc de développer le serveur de voeux en s'appuyant sur le framework imposé, et sur l'utilisation du modèle MVC. Pour rappel, le modèle MVC est un modèle d'architecture qui cherche à séparer nettement les couches de présentation (UI : User Interface), métier (BLL : Business Logic Layer) et d'accès aux données (DAL : Data Access Layer). Le but étant d'avoir une dépendance minimale entre les différentes couches de l'application ; ainsi les modifications effectuées sur n'importe quelle couche de l'application n'affectent pas les autres couches.¹

Nous allons tout d'abord présenter le sujet et les attentes du client ; nous allons ensuite expliciter l'organisation choisie et la gestion du projet en elle même. Nous allons dans une troisième partie parler du cahier des charges fonctionnel et des scénarii d'usages associés. À noter, ces scénarii nous serviront à tester notre application. La quatrième partie sera consacrée à l'étude technique de notre solution ; nous allons y expliciter quelques fonctions complexes que nous avons développées. Nous finirons enfin par conclure sur notre application, et expliquer quelles auraient été les améliorations possibles pour notre système.

1. Source : developpez.com

2. Présentation du sujet

2.1 Prérequis

Un module représente une matière. Exemple : il peut exister un module de Mathématiques, un module d'algo...

Confiance aux utilisateurs : Le client demandais lors de ce projet de considérer que les utilisateurs de ce site web seront cordiaux et sympathiques ; autrement dit, ils ne chercheront pas à tout prix et par n'importe quels moyens à détruire notre site ni à entrer dans le système... Nous avons fait le maximum en terme de vérifications diverses et variés mais nous n'avons pas pris le temps de verrouiller chacune de nos fonctions contre tout type d'attaques.

2.2 Le sujet

Au début de chaque année scolaire, il faut que les enseignants de l'ENSSAT se repartissent les modules, en fonction des différentes aspirations de chacun. Il était donc nécessaire de développer une application dédiée qui permette d'avoir une vue d'ensemble de l'état d'avancement de ce processus assez facilement. Concrètement, le principe est d'avoir un site web utilisable par tous les professeurs pour exprimer leurs voeux. Il faut impérativement que les utilisateurs *aillent à l'essentiel* : s'inscrire sur les modules qui les intéressent, se désinscrire de certains modules, afficher les modules de leurs collègues... L'une des utilisations les plus basiques sera l'action de se connecter, d'avoir un accès immédiat aux informations importantes, et de pouvoir de manière ludique et rapide effectuer des modifications sur son emploi du temps. Les professeurs de l'université ont tous différents statuts ; à ces statuts sont liées différentes charges horaires annuelles. Par exemple, un professeur sous le statut de titulaire aura 192 heures d'enseignement à effectuer chaque année ; s'il s'inscrit sur deux modules qui comportent 50 heures, sur un module qui comporte 45 heures et sur un module qui comporte 47 heures, son quota d'heure sera atteint. Il lui est à ce moment là encore possible de s'inscrire pour un nombre d'heure supérieur à celui prévu par son quota statutaire, cas exceptionnel.

Il est important de garder à l'esprit que pour ce projet, l'utilisation de la base de donnée fournie par le client est un prérequis. Il faut que notre projet soit utilisable avec cette dernière avec le moins de modifications possible. En effet, la base utilisée est multi-usage, et pourra être sollicitée par d'autres applications en parallèle.

L'une des priorités pour nous en tant que concepteurs est d'imaginer un service simple et efficace ; il faut que le produit soit très facile d'utilisation, pour optimiser le temps des utilisateurs passé à se répartir les tâches. Il est donc demandé implicitement de mettre l'accent sur l'ergonomie et le design.

2.3 Gestion de projet

Lors de ce projet, il était important de bien s'organiser afin d'optimiser les phases de développement. Nous avons repris des éléments de la méthode agile pour le développement, en nous appuyant sur un travail collaboratif à l'aide de l'outil *git*. Nous avons donc décomposé les tâches en sprint et nous avons codé dans une optique de devops, en intégrant continuellement des nouvelles fonctionnalités. Cette méthode est utilisée dans énormément de projets dans le domaine de l'informatique (exemple : Spotify, Amazon...)

Nous avons décomposé les tâches et les différentes fonctionnalités attendues du site pendant les premières séances de projet. Il fallait en effet avoir une idée claire et précise des tâches à réaliser au plus tôt. Cette décomposition s'est accompagnée de la rédaction d'un cahier des spécifications fonctionnelles (cf annexes). Nous avons rédigé un certain nombre de scénarii dans ce dernier, et nous avons aussi et surtout décomposé chaque fonctionnalité du site une par une. Il s'agissait dans un premier temps, sans penser réellement à la technique employée, de définir les grands axes de développement. Alexis s'est occupé de la maquette du site web. Nous voulions notre interface épurée, nous nous sommes donc appuyés sur un flat design. Afin de gagner du temps lors du développement, nous nous sommes aussi appuyés sur le framework css *Bootstrap*. Il permet en effet de placer les éléments de rapidement, en se basant sur un système de grille. Dans un second temps, nous avons hiérarchisé les phases de développement. De notre cahier des charges ressortaient un certain nombre de fonctionnalités à développer : un panel de profil, un panel d'administrateur, etc. Nous avons donc réalisé un diagramme de Gantt pour avoir une vision d'ensemble du développement, et pour avoir une idée des échéances. Une fois le Gantt posé, nous nous sommes réparti les tâches en fonction de nos capacités. Nous avons tous développé certaines fonctionnalités en parallèle.

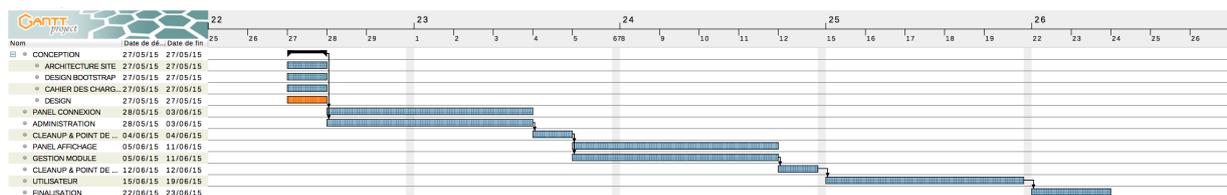


FIGURE 2.1 – Diagramme de Gantt, disponible en annexes

La répartition des tâches a été la suivante :

- Romain a travaillé sur le panel de connexion et d'inscription. Il s'est aussi occupé de trier les modules. Il a enfin participé à la création du module News.
- Colin a travaillé sur le profil et sur la page administrateur. Il s'est occupé de tester le site avec les scénarii d'usage.
- Alexis s'est occupé de l'intégration des nouvelles pages, a posé les structures et s'est occupé de construire l'arborescence du projet. En tant qu'expert JavaScript / JQuery, il s'est aussi occupé des animations. Il a enfin harmonisé le site.

Nous nous sommes réparti les tâches de manière équitable afin que chacun puisse développer tant dans le modèle, les vues ou les contrôleurs. Il nous semblait en effet important que chacun touche à tout pour pouvoir apprendre le plus possible sur le framework CodeIgniter et sur l'utilisation d'un modèle MVC.

Alexis s'est occupé de poser le design du site, dont vous pouvez voir la maquette ci-dessous.

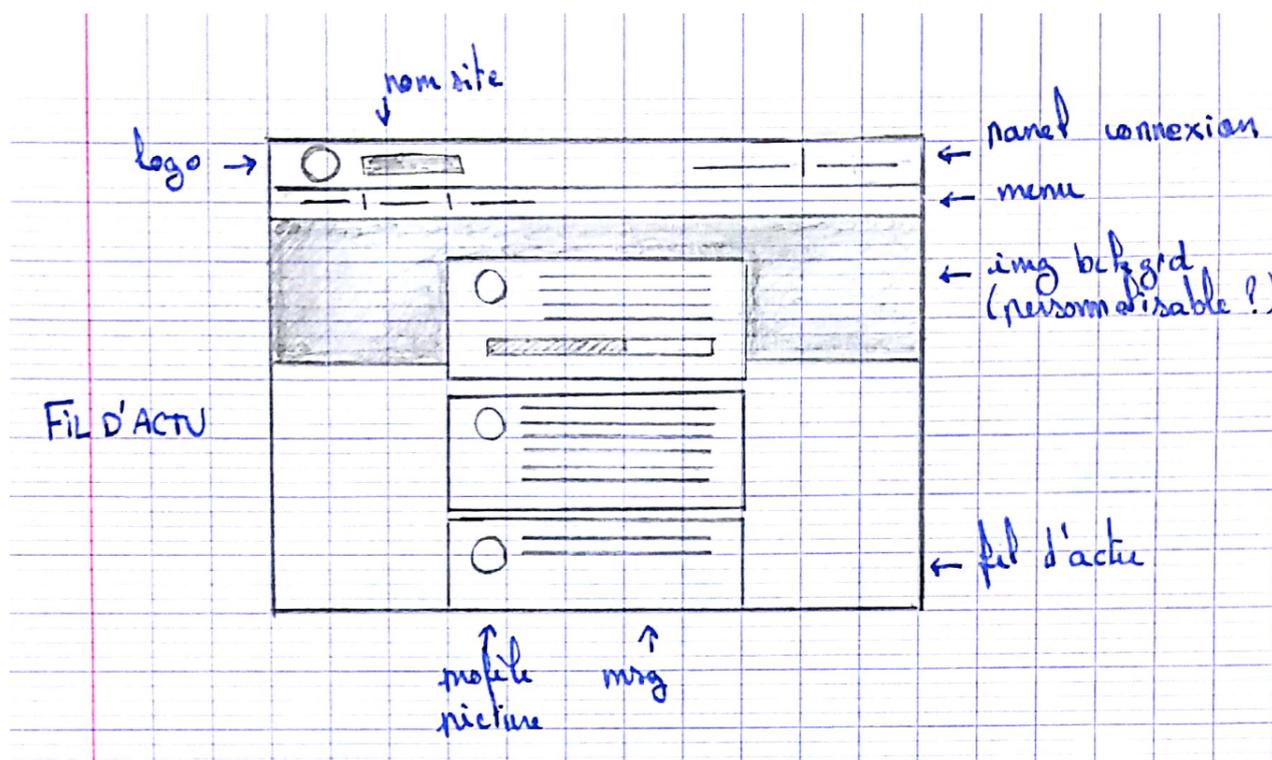


FIGURE 2.2 – Design du site dessiné par Alexis

3. Cahier des spécifications fonctionnelles

3.1 L'utilisateur admin

L'application aura deux types d'utilisateurs : utilisateur *basique* et *administrateur*. Il faut que l'administrateur puisse effectuer les actions suivantes :

- Création d'utilisateurs
- Modification d'un profil utilisateur
- Affectation d'un cours à un utilisateur
- Création de modules
- Acceptation d'un utilisateur qui s'est inscrit (aspect modération)

L'administrateur a bien évidemment tous les droits sur le site, et pourra lui même gérer son profil et ses modules.

3.2 Les fonctionnalités

Les différentes fonctionnalités que nous avons prévu sont les suivantes :

- Possibilité pour un utilisateur de s'inscrire sur le site web. S'il s'inscrit, l'administrateur pourra l'accepter ou non. Tant qu'il n'est pas accepté par un administrateur, un utilisateur n'a pas accès au site.
- Possibilité pour un utilisateur de se connecter à sa page personnelle et de gérer son profil. Il pourra modifier ses décharges, uploader / supprimer une image de profil, changer son password par défaut. Il pourra aussi afficher le nombre d'heures restantes pour qu'il ait entièrement consommé son quota d'heure.

Les administrateurs auront leur page dédiée. Celle-ci leur permettra de gérer beaucoup de fonctionnalité du site, telles que :

- Ajout/Suppression/Modification d'un module.
- Ajout/Suppression/Modification du contenu d'un module.
- Gestion des utilisateurs (modification, suppression, acceptation)
- Gestion des news.

Possibilité de rechercher des modules, en fonction de différents filtres :

- Promotion.
- Semestre.
- Module.
- Enseignant.
- Contenu sans enseignant.

Il sera possible pour l'utilisateur courant (qu'il soit administrateur ou non de l'application) de s'inscrire à un module. Le décompte de ses heures s'effectuera automatiquement, et la barre de progression du profil sera mise à jour. Il sera aussi possible pour l'utilisateur de se désinscrire d'un module.

Une notion apportée par notre conception était la notion de *News*. Il nous semblait intéressant d'afficher, sous forme de rapide brève, la dernière activité sur le site. Si un utilisateur s'est inscrit dans un module, une brève s'affichera alors sur la page d'accueil. Les brèves / news seront triées dans l'ordre chronologique. C'est aussi pour cette raison que nous avons rendu possible l'upload d'un avatar dans la page profil. Les news s'afficheront en effet avec l'image de l'utilisateur concerné.

Dans l'optique de rendre notre application agréable à utiliser, nous avons utilisé du JavaScript pour rendre les interactions fluides et efficaces.

3.3 Scenarii d'usage

3.3.1 Utilisateur "admin"

L'administrateur, vbarreau, se connecte à l'application, via l'interface web. Il ajoute un module à l'aide du panel prévu pour : module de "programmation web", pour les IMR1. Ce cours aura une durée de 45H00, décomposée en 3 parties de 15 minutes :

- "cours partie 1"
- "cours partie 2"
- "TP"

Le responsable de ce module sera **Arnaud DELHAY**. Il ajoute **Vincent BARREAUD** (lui-même) sur la partie "cours partie 1" et prend donc les 15 premières heures. Il ajoute lui-même un intervenant à la partie "cours partie 2", qui sera **Virginie THION**. La 3eme partie, "TP", est laissée libre. Il affiche tous les cours qu'il aura à préparer, puis se déconnecte à la fin de toutes ces opérations.

Un deuxième admin, **Romain CARON**, décide de se connecter pour modifier le cours précédemment créé :

- Ajout d'une partie de 2H00 : "conclusion"
- Retrait de 2H00 de la partie "TP"

3.3.2 Utilisateur "normal"

Un professeur, **Joseph PATUREL**, va effectuer les actions suivantes :

- S'inscrire sur le site :
 - Pseudo : jpaturel
 - Mdp : 1234
- S'identifier.
- Afficher tous les cours.
- Prendre la partie "TP" du module "programmation web".
- Afficher tous les cours qui lui sont attribués.
- Changer complètement d'avis et supprimer sa contribution sur ce cours.
- Se déconnecter.

Un autre professeur, **Alexis ZANKOWITCH**, qui est déjà inscrit, va effectuer les actions suivantes :

- Connexion sur le site :
 - Pseudo : azankowi
 - Mdp : azertyuiop
- Prendre l'intégralité (15H00) de la partie "TP" du module "programmation web".
- Afficher l'intégralité des cours proposés à l'ENSSAT.
- Afficher l'intégralité des cours qui lui sont attribués.
- Afficher l'intégralité des professeurs actifs.
- Se déconnecter.

Un dernier utilisateur, **Colin LEVERGER** se connecte à son profil. Il ajoute un avatar, change d'avis et le supprime, re-upload une image, change son mot de passe. Il affiche ensuite tous les modules pour le semestre 1. Il affiche ensuite tous les modules pour l'utilisateur "vbarraud". Il retourne enfin sur son profil, et ajoute sa décharge de 10 heures au système. Il se déconnecte enfin après toutes ces opérations.

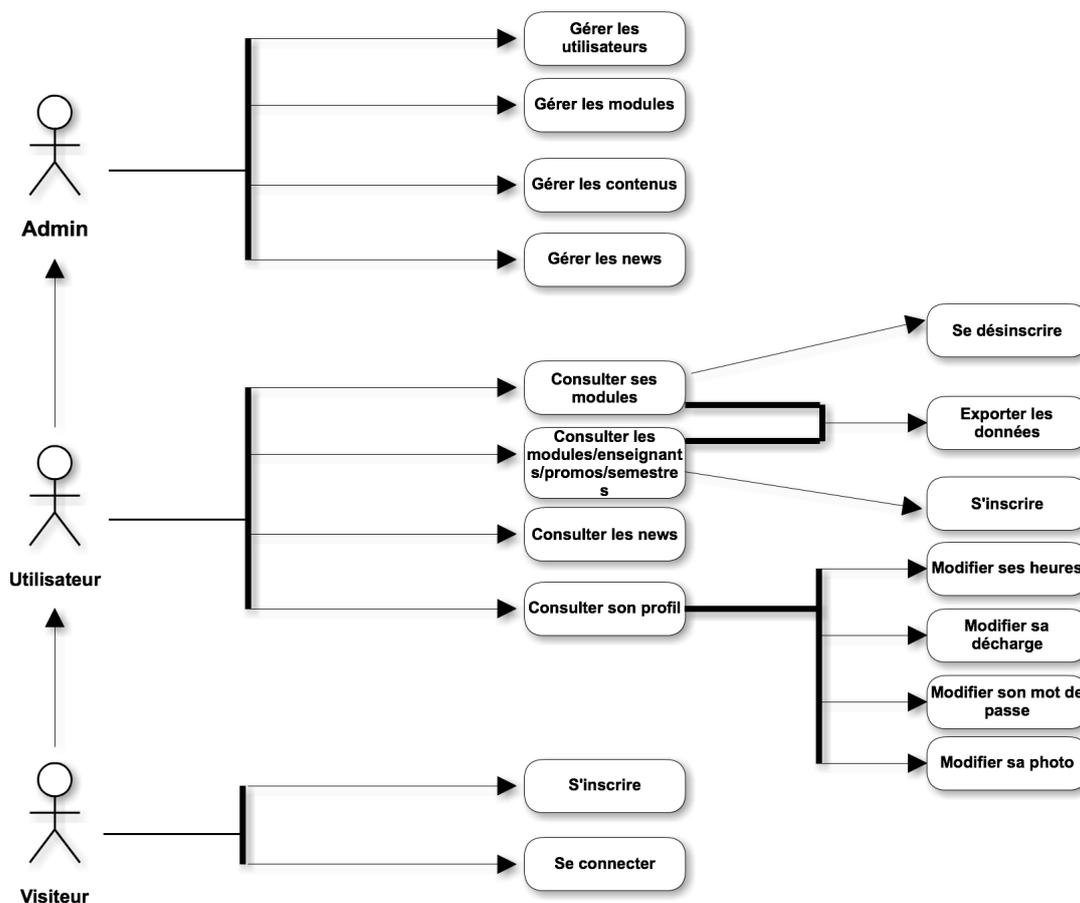


FIGURE 3.1 – Use case schéma

4. Développement technique

4.1 Architecture globale de l'application

Nous avons premièrement vérifié que l'utilisateur est bien connecté au site avant qu'il puisse utiliser un contrôleur. En effet, sans cette vérification il était possible pour n'importe quel utilisateur possédant l'URL des contrôleurs d'y accéder en la tapant dans la barre de recherche du navigateur. Cette vérification sera effectuée dans chaque constructeur.

Nous avons décomposé notre application en 6 contrôleurs, qui sont les suivants :

- Login - gère la connexion et l'inscription.
- admin - contrôleur de la zone d'administration.
- profile - gère le profil.
- homeNews - affichage des news selon filtre.
- modules - mes modules/recherche/reporting.
- upload - gère l'upload d'image.

Concrètement, il existera un contrôleur par page du site ; les contrôleurs connaîtront par contre plus ou moins tous les modèles car nous avons besoin pour chaque page du site de rapatrier des informations de différentes tables. Pour les modèles, qui sont au nombre de six, nous avons préféré segmenter le plus possible en fonction des requêtes et des tables à interroger, toujours dans un souci de clarté : le modèle users récupérera des informations sur les utilisateurs, le modèle décharge sur les décharges, etc.

Les modèles sont donc les suivants :

- users
- contenu
- décharges
- modulesmodel
- new

— uploadmodel

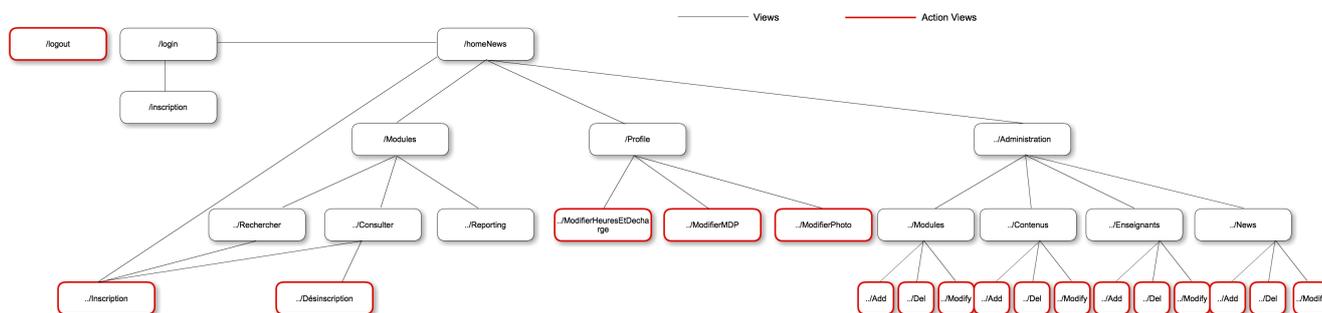


FIGURE 4.1 – Maquette organisation site (disponible en annexe dans une taille convenable !)

4.2 Quelques fonctions expliquées

4.2.1 le contrôleur "profile"

Pour la page de profil ¹, il fallait pouvoir effectuer quelques modifications : modifier la décharge de l'utilisateur, modifier son statutaire, modifier son mot de passe et modifier son image de profil. Il y a donc une fonction par fonctionnalité ici.

Nous avons besoin pour afficher sur chaque page d'une fonction générique qui retourne trois informations : le statutaire de l'utilisateur courant, le pourcentage d'accomplissement du statutaire, les heures prises du statutaire. Cette fonction était utilisée dans chacun des contrôleurs. Nous avons donc créé un contrôleur qui *héritait* de la classe `CI_Controller`, et défini la fonction `getPercentage($userName)` dans cette dernière. Cela nous permettait *d'éviter les doublons de code*.

4.2.2 Recherche de modules

Les fichiers utilisés sont : `modules.php` (contrôleur), `contenus.php` (modèles) et la vue `show-module.php`.

La recherche de module est multicritères (nom du module, promotion, enseignants, semestre, sans enseignants). Nous avons donc affiché ces derniers sous forme d'une liste déroulante (balise HTML `select`). Dans un souci de faciliter l'utilisation de notre application, nous avons intégré le plugin jquery chosen (<http://harvesthq.github.io/chosen/>) ce dernier rajoute un champ de recherche dans le `select` pour trouver plus facilement l'option qui nous intéresse. Une fois que l'utilisateur clique

1. Contrôleur "Profile" avec un e, langue anglaise oblige.

sur le bouton Rechercher, les valeurs des différentes listes déroulantes sont envoyées en post à notre contrôleur. La fonction appelée est `displayModuleContenus()`. Dans un premier temps la fonction stock les informations reçues en post dans un tableau. On récupère le type de recherche qui a été faite (promotion ou nom du module), cela nous permet de réafficher la recherche à l'état t-1. Nous interrogeons ensuite notre modèle pour récupérer les contenus de notre recherche. Nous utilisons une seule fonction pour interroger notre base de données. Nous traitons les différentes options possibles en intégrant des conditions à la requête : si l'utilisateur effectue sa recherche selon un semestre et un enseignant alors nous rajoutons des clauses `where` à la requête cela ce traduit en code de la manière suivante :

```
if ($array [ ' module ' ] )  
    $this ->db->where( ' module ' , $array [ ' module ' ] );
```

Afin de récupérer les différentes informations d'une partie nous joignons la table contenu et module. Pour récupérer le nom et prénom de l'enseignant assigné à un contenu nous utilisons un `left join`. De cette manière s'il n'y a pas d'enseignant de renseigné nous récupérons tout de même la ligne. Le modèle retourne à notre contrôleur un tableau de résultat. Nous stockons ensuite ces résultats dans la variable session de codeIgniter. Ensuite nous rappelons notre fonction `index` du contrôleur qui s'occupe de charger les différentes vues et transmet les différentes variables nécessaires à l'affichage. Nous avons choisi de sauvegarder les résultats dans la variable session de codeIgniter pour pouvoir afficher les résultats si l'utilisateur décide de s'inscrire.

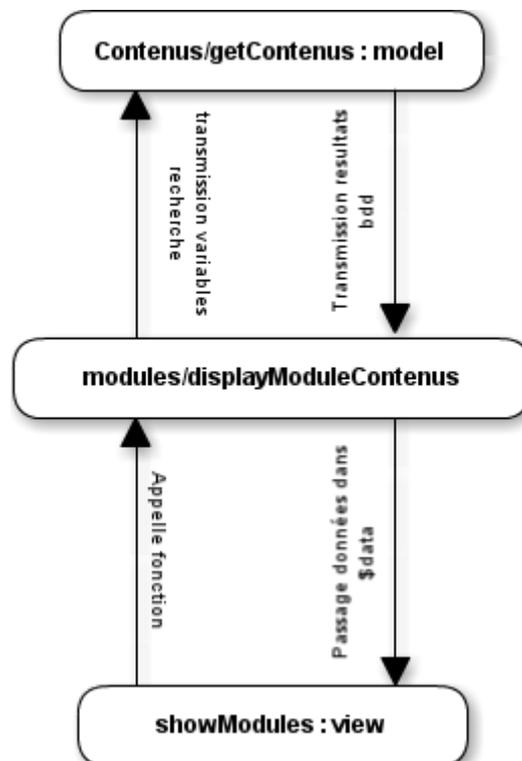


FIGURE 4.2 – Schéma recherche

4.2.3 Les différents retours sur les données envoyées par l'utilisateur

Lorsqu'un utilisateur envoie des données par formulaire, il nous semblait très important de lui donner un retour sur son action. Nous avons donc affiché un retour. S'il y a un problème, un message est généré pour expliquer l'erreur que que l'utilisateur a fait dans le formulaire.

Ci-dessous, deux images explicitant ce processus.

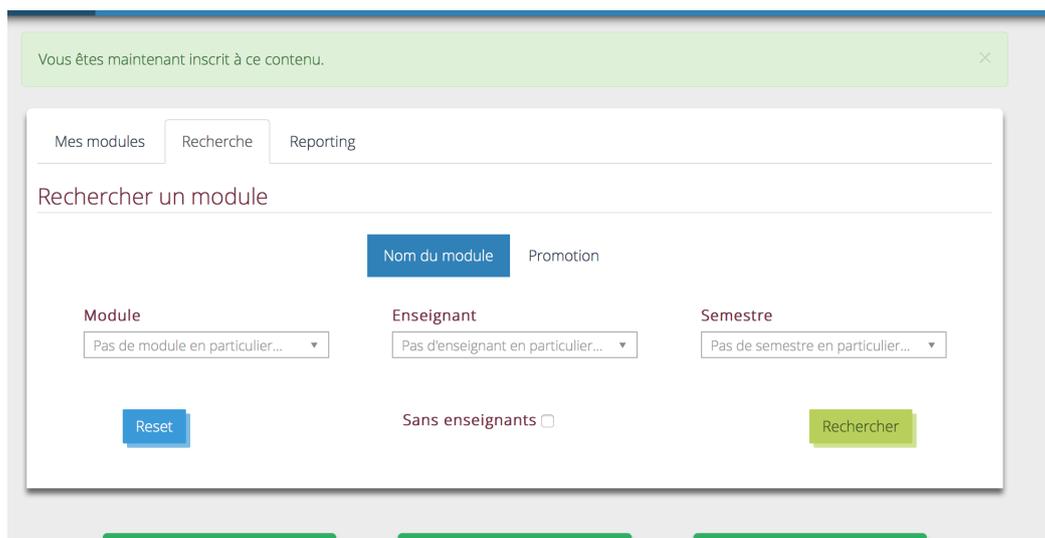


FIGURE 4.3 – Retour positif

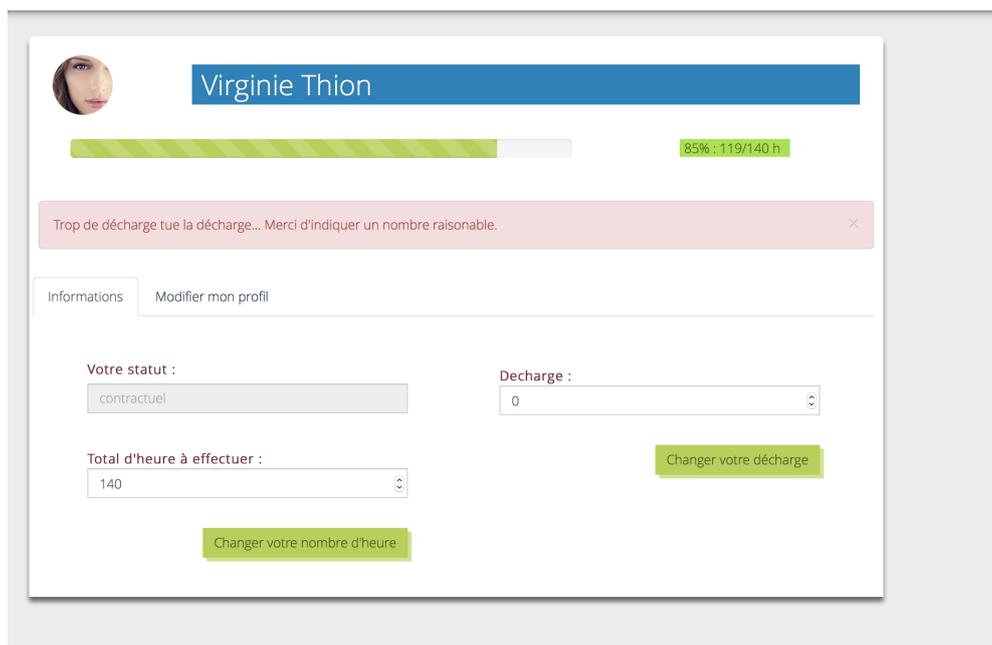


FIGURE 4.4 – Retour négatif

5. Conclusion

5.1 Les limites de la solution employée

Nous avons réussi à répondre au cahier des charges de manière quasi exhaustive. Le reproche que nous pourrions faire à notre application est le fait que l'affichage peut parfois prendre beaucoup de place sur l'écran. L'information est en effet mise en page dans un format *assez peu compact*. Il aurait été possible de développer une interface plus "petite".

5.2 Recette

FONCTIONNALITES	ATTENDUE	NON ATTENDUE	IMPLEMENTATION
Panel de connexion :			
Inscription utilisateur		x	x
Connexion utilisateur	x		x
Déconnexion utilisateur	x		x
Profil :			
Modification statutaire	x		x
Modification décharge	x		x
Vérification cohérence des données		x	NON
Upload image profil		x	x
Suppression image profil		x	x
Modification password		x	x
Administration :			
Création d'un module	x		x
Suppression d'un module	x		x
Ajout d'un contenu à un module	x		x
Suppression d'un contenu à un module	x		x
Modification d'un module	x		x
Acceptation d'un utilisateur inscrit		x	x
Ajout d'un utilisateur	x		x
Suppression d'un utilisateur	x		x
Modification d'un utilisateur	x		x
Reinitialisation mot de passe		x	x
New :			
Création news		x	x
Suppression news		x	x
Modification news		x	x
Publication automatique news		x	x
Recherche modules :			
Recherche module selon différents critères	x		x
Inscription module recherché	x		x
Désinscription module recherché	x		x
Export CVS	x		x
Reporting & statistiques		x	x
BDD :			
Persistance des données	x		x
Autre :			
Vérification de toutes les données utilisateur		x	NON
Ergonomie		x	x
Design		x	x

FIGURE 5.1 – Recette de notre solution

Pour valider chacun des points ci-dessus, nous avons effectué des tests poussés sur chacune des fonctionnalités du site. Les use case définis dans le cahier des spécifications ont tous été testés et approfondis.

5.3 Bilan & Conclusion fonctionnelle

Nous avons réussi à créer un site web fonctionnel et facile à utiliser pour l'utilisateur ; nous avons travaillé par itération, et nous étions en avance sur le Gantt défini initialement ; nous avons en effet profité des longs week end du mois de Mai pour avancer sur le projet. Le site est utilisable avec la base de donnée fournie dès le départ, et nous avons ajouté des fonctionnalités qui permettent de rendre l'information accessible plus rapidement. Le développement des news et l'affichage du reporting permet d'avoir une valeur ajoutée concernant le design et l'ergonomie de l'application. Pour améliorer le projet et pouvoir le mettre en production, il faut mettre l'accent sur les vérifications des données envoyées par les utilisateurs. Considérer que nous vivons dans une utopie où les utilisateurs auront tous une utilisation normale du site n'est bien évidemment pas viable...

6. Annexes

Disponible en annexes dans le dossier ANNEXES de notre fichier rar :

- Diagramme de Gantt, format PDF.
- Schéma d'organisation du site format PNG.
- Recette format PDF.
- Cahier des spécification fonctionnelles format PDF.