qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| **Projet WEB**  **“Planning Par Semaine”**  **Cahier principal** : Cahier de conception  (Cahier 1/5) |

**KALWANT Steven FROMONT Didier BENKHALA Rania MOUGAMADOU Javid**

**LSI2**

Sommaire

Introduction

[I. Description du projet 4](#_Toc416356102)

[II. Présentations fonctionnelle et technique de l'application 5](#_Toc416356103)

[A. Aspects techniques 5](#_Toc416356104)

[1. Cas 1 : Consulter le planning par semaine d’une promotion 5](#_Toc416356105)

[2. Cas 2 : Se connecter en tant que responsable d’année 7](#_Toc416356106)

[3. Cas 3 : Editer un planning par semaine 8](#_Toc416356107)

[4. Cas 4 : Consulter la charge d’un enseignant 9](#_Toc416356108)

[5. Cas 5 : Ajouter/Modifier les descriptions de Modules 10](#_Toc416356109)

[6. Cas 6 : Calculer le service d’un enseignant 11](#_Toc416356110)

[7. Cas 7 : Exporter sous format CSV, iCal, ... 12](#_Toc416356111)

[8. Cas 8 : Exploiter les traces (pas implémenté) 12](#_Toc416356112)

[B. Recette 13](#_Toc416356113)

[C. Bilan 13](#_Toc416356114)

Conclusion

Introduction

Le projet Web « Planning par Semaine » consiste à mettre en œuvre toutes les étapes nécessaires pour qu’un utilisateur puisse visualiser le planning d’une promotion et un responsable d’année puisse éventuellement éditer le planning de sa propre promotion. Etant donné que le cahier des charges précise les différents cas d’utilisation et les contraintes techniques associées, notre première partie du projet avait pour but de mettre en place les concepts d'itérations et d'activités (analyse, conception, ébauche d'implémentation).

Ce rapport va mettre en lumière les différentes étapes d'implémentation des cas d’usage. Le plan de notre rapport se base sur trois axes principaux qui traitent de la description du projet, la présentation fonctionnelle et technique de notre application, et le bilan du projet.

# Description du projet

L'application que nous souhaitons réaliser se présente comme ceci : il s'agit d'une application qui est « Planning par Semaine » permettant principalement à un utilisateur de visualiser le planning d’une promotion. L’application permet également de se connecter en tant que responsable d’année ou directeur des études, d’éditer un planning d’une promotion (uniquement un RA ou DDE) et de pouvoir consulter la charge des enseignants par semaines.

Le système sera implémenté en utilisant le langage HTML, CSS, PHP et JavaScript. L'architecture envisagée pour l'implémentation est du type MVC.

Nous avions décidé de distinguer les 8 cas d’utilisations présentées comme ceci :

1er cas : Consulter le planning par semaine d’une promotion

* Envoi de la requête SQL vers la BD et afficher le résultat
* Afficher le taux de remplissage de la semaine

2ème cas : Se connecter en tant que responsable d’année

* Etablissement d’un système de Session /Cookie pour la page de connexion
* Changer/éditer le mot de passe ou son adresse mail
* Page de déconnexion
* Autoriser la modification du planning de la filière dont on est responsable uniquement
* Restreindre à la consultation seule des autres plannings

3ème cas : Editer un planning par semaine

* Déplacement des pavés (Drag & Drop) sur une semaine libre
* Effectuer un Commit pour sauvegarder les différents changements vers la BD

4ème cas : Consulter la charge d’un enseignant

* Voir les charges des enseignants par semaine

5ème cas : Ajouter/Modifier les descriptions de Modules

* Affecter un enseignant comme responsable d’un module
* Affecter un enseignant à un cours

6ème cas : Calculer le service d’un enseignant

* Voir le service des enseignants

7ème cas : Exporter sous format CSV, iCal, ...

* Exporter le planning

8ème cas : Exploiter les traces

* Notifie les opérations effectué à chaque commit
* Possibilité de défaire ce qui a été fait

# Présentations fonctionnelle et technique de l'application

## Aspects techniques

Nous avons mis en œuvre le design pattern de type M-V-C (modèle-vue-contrôleur) car l'organisation d'une interface graphique est délicate. L'idée est de bien séparer les données, la présentation et les traitements. Ce choix de design pattern est dû au fait qu'il apporte de la clarté par l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet.

*Figure n°1 : Le design pattern MVC*

La figure n°1 regroupe les fonctions nécessaires en 3 parties :

* Le modèle : contient les différentes classes métiers utilisées pour modéliser l’application
* La vue : le code HTML pout l’affichage utilisé pour l’interaction avec l’utilisateur.
* Le contrôleur : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

### Cas 1 : Consulter le planning par semaine d’une promotion

**Le déroulement :**

1. L’utilisateur sélectionne la promo dont il veut visualiser son planning.
2. Une requête avec la promo voulue est envoyée vers la BD.
3. Après traitement de la requête, on affiche les résultats sur la page HTML.
4. Le taux de remplissage est également affiché sur la page HTML

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

* Get\_promo.php : retourne la liste des promos en traitant avec le fichier responsable.xml (nous expliquerons ce choix ultérieurement)
* Query.php : contient les deux requêtes importantes pour ce cas.

Contrôleur :

* Index.php : Contrôleur de notre page Planning par Semaine

Vue :

* Index.php : Vue HTML de notre page Planning par Semaine

**Analyse et Implémentation :**

Au niveau de la requête SQL (envoyé par la couche d’abstraction PDO), nous nous sommes arrangés pour avoir un résultat sous la forme : (Module, Intervenant, Nature, Semaine i,Heure en Semaine i) et avec les rowspans associés dans le résultat pour Module et Intervenant. Nous avons pris également en compte les modules affectés par aucun enseignant (NULL) en affichant #N/A. La requête étant très longue, nous avons fait plusieurs jointures successives entre la table contenumodule, la table module, la table semaine, la table virtuelle restantesmodule et la table affectationsemaine. Une fois la requête traitée avec succès, nous affichons le résultat par le biais d’une boucle while dans la vue html. Pour chaque pavé de 2h, nous affichons un rectangle ‘div’ rouge et pour un pavé de 1h nous affichons un ‘div’ bleu.

Pour afficher les taux de remplissages de chaque semaine, nous avons envoyé une requête SQL plus légère que la précédente qui nous retourne un résultat sous la forme : (Semaine, HeuresAffectées, Public et nombreHeuresMax). Pour cela, nous avons effectué une jointure entre les tables : affectationsemaine, module et semaine. Ces taux de remplissages seront très utiles pour la suite lors d’un déplacement de pavé d’une semaine i à une semaine j.

Pour récupérer la liste de toutes les promos, nous utilisons la fonction get\_promo qui permettra à l’utilisateur de sélectionner une promo parmi cette liste à partir d’un formulaire select dans la vue HTML. Celui-ci traite avec la DAO responsable.xml car nous avons anticipé le fait que chaque promo possède un responsable d’année.

**Problèmes / Bilan :**

Pour ce cas, nous n’avons pas eu réellement de problèmes étant donné que l’on nous avait fourni un code d’origine déjà fonctionnel. Toutefois, nous avons été contraints de rajouter un filtre (au niveau des requêtes SQL) pour certains cours. Par exemple, la promo IMR1 et EII2 possède un module en commun de Réseaux appelée RESIMREII où il a fallu conserver uniquement les cours correspondant à leur promos uniquement (TD EII2 et TD IMR1).

De plus, nous avons désactivé le Drag and Drop fourni de base car l’utilisateur ne pourra pas éditer sa promo sauf s’il est connecté en tant que RA et qu’il modifie sa propre promo uniquement. Enfin nous avons anticipé également sur l’attribution des ID des pavés sur la vue HTML qui sera très utile pour le Cas 3.

### 2. Cas 2 : Se connecter en tant que responsable d’année

**Le déroulement :**

1. L’utilisateur entre son login et son mot de passe dans les champs concernés.

1b. L’utilisateur peut également sélectionner l’option « Maintenir la connexion » qui gardera sa session connecté jusqu’à sa déconnexion. (Cookies)

1. Vérifie si l’utilisateur est un responsable d’année auprès de la DAO responsable.xml .
2. Vérifie le mot de passe saisie auprès de la base de donnée par le biais d’une mini requête
3. L’utilisateur est connecté en tant que RA et est redirigé vers la page de visualisation des plannings

5a. L’utilisateur peut modifier son adresse mail

5b. L’utilisateur peut modifier son mot de passe

6a. L’utilisateur peut se déconnecter.

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

* responsable.xml : DAO possédant tous les responsables d’années et leurs promos associées.

Contrôleur :

* login.php : Contrôleur de notre page de connexion
* logout.php : Contrôleur de notre page de déconnexion
* editmail.php : Contrôleur de notre page de modification d’adresse mail
* editmdp.php : Contrôleur de notre page de modification de mot de passe

Vue :

* login.php : Vue HTML de notre page de connexion
* editmail.php : Vue HTML de notre page de modification d’adresse mail
* editmdp.php : Vue HTML de notre page de modification de mot de passe

**Analyse et Implémentation :**

Nous avons mis en place simplement un système de Session pour la page de connexion. En effet, la session nous permettra par la suite d’avoir quelques informations concernant le responsable d’année connecté (ex : nom, prénom, promo associée, etc…) et également l’établissement de la connexion/déconnexion du RA. Nous utiliserons un système de cookie dans le cas où le RA souhaite rester connecté par le biais d’un check box « Maintenir la connexion ».

Une fois que l’utilisateur a saisi son login et son mot de passe, nous récupérons ses valeurs par le biais du formulaire POST et nous vérifions si son login est présent dans la DAO responsable.xml grâce à une requête XPATH. Ensuite nous envoyons une requête SQL vers la base de données pour comparer le mot de passe saisie et le mot de passe stocké dans la table enseignant. Si toutes les conditions sont satisfaites, alors la connexion est réussie et nous redirigeons le RA vers la page de plannings. Dans le cas contraire, un message d’erreur sera affiché.

Le responsable peut modifier son adresse mail (resp son mot de passe) par le biais d’une autre interface editmail.php (resp editmdp.php). Celui-ci devra saisir sa nouvelle adresse mail (resp son nouveau mot de passe) et le confirmer en la saisissant une 2ème fois. Dans le cas du changement de mot de passe, l’ancien mot de passe est exigé. Les modifications seront effectuées dans la DAO pour l’adresse mail et dans la BD pour le mot de passe.

Une variable $edit permettra l’autorisation d’éditer le planning d’une promo ou consulter sans modifications. Enfin, le RA peut se déconnecter via un bouton sur la barre de navigation située en haut. Ainsi, la session est détruite et les cookies également s’ils existent.

**Problèmes / Bilan :**

Pour ce cas, l’ajout du système de Session et de Cookies pour la connexion/déconnexion fut très simple. Nous avons sécurisé ce cas en ajoutant une protection qui consiste à rediriger l’utilisateur s’il tente d’accéder à la page de modification de mot de passe /adresse mail alors que celui-ci n’est pas connecté.

De plus, nous effectuons une vérification côté client (Javascript) des informations saisies lors de la connexion ou du changement de mot de passe /adresse mail. Cela permet d’avoir une meilleure gestion des erreurs et un bon contrôle sur les données échangées avec l’utilisateur. Toutefois, nous effectuons ces mêmes vérifications côté serveur également (PHP).

### 3. Cas 3 : Editer un planning par semaine

**Le déroulement :**

1. Prérequis : Le RA est connecté et visualise le planning de sa propre promo.
2. Le RA déplace un pavé sur une semaine libre.

1b. Il peut repositionner (supprimer) un pavé en double cliquant.

2. Les modifications effectuées sont stockés dans une variable temporaire.

3. Il valide ses modifications en appuyant sur le bouton Commit.

4. Une requête asynchrone AJAX est envoyée vers le serveur

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

Contrôleur :

* index.php : Contrôleur de notre page Planning par Semaine
* task.php : Code PHP effectuant une requête AJAX pour le commit
* grouptask.php : Code PHP effectuant une requête AJAX pour le déplacement d’un pavé d’un module groupé (voir ci-dessous)

Vue :

* index.php : Vue HTML de notre page Planning par Semaine
* edit.php : Code JS incluant le Drag & Drop et le commit

**Analyse et Implémentation :**

Nous avons réactivé la fonction Drag & Drop obtenu du code origine basé sur les #ID des pavés. Nous avions anticipé pour cette fonctionnalité en attribuant un #ID pour chaque pavé dans le cas 1. Les #ID des pavés se présentent sous la forme : «  Module\_Enseignant\_Partie\_no item\_typegroupe » où le pavé est caractérisé par une partie (CM,TD,TP,Projet) du module fait par un enseignant. Le no item donne une information sur le n-ième cours du module/partie et également s’il s’agit d’un pavé de 1 ou 2h. Le dernier champ typegroupe indique si le module est destiné a plusieurs promos (nous expliciterons plus tard l’intêret de ce champs). Les #ID des cases semaines se présentent  sous la forme : «  Module\_Enseignant\_Partie\_numero semaine ». Le déplacement d’un pavé (Drag & Drop) sera validé uniquement si les 3 premiers champs des #ID pavés et #ID semaine correspondent et nous stockerons ces informations dans une variable temporaire en JSON.

Toutefois, un pavé ne peut être déplacé dans une case semaine si elle est libre. C’est pourquoi nous comparerons pour chaque déplacement de pavé, le nombre d’heures libres d’une semaine avec la durée du pavé, grâce au taux d’affichage implémentés dans le cas 1. Si la semaine n’est pas vide, nous affichons un message d’erreur.

Lorsque le RA validera les modifications effectuées en appuyant sur le bouton Commit, nous allons traiter notre variable temporaire JSON en l’envoyant dans une requête asynchrone (AJAX) vers le fichier task.php .Celui-ci va effectuer plusieurs requête SQL du type « INSERT – UPDATE » vers la base de données plus particulièrement dans la table AffectationSemaine.

**Problèmes / Bilan :**

Pour ce cas, nous avons rencontré un problème avec le module RESIMREII. En effet, car un cours de CM est destiné en même temps à la promo EII et à la promo IMR. Si nous devons déplacer un heure de CM depuis le planning des EII dans une semaine i libre, il faut vérifier que la semaine i est également libre pour la promo IMR. Nous avons contourné le problème en effectuant une requête asynchrone AJAX pour déterminer si la semaine est libre pour toutes les promos sur les modules groupés.

Nous avons rajouté un Autocommit par le biais d’un SetInterval s’effectuant tous les 1 min 30 sec. Il s’agit de la tâche de fond de l’interface qui poste régulièrement les modifications (placement des pavés) vers le serveur.

### 4. Cas 4 : Consulter la charge d’un enseignant

**Le déroulement :**

1. Prérequis : Le RA est connecté.
2. Le RA accède à la page des charges des enseignants.

2. 2 requêtes SQL sont envoyées puis traité vers le serveur.

3. La nouvelle interface affiche un tableau où les lignes sont représentées par des enseignants, les colonnes par des semaines et les cases par des charges.

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

Contrôleur :

* charges.php : Contrôleur de notre page de charges des enseignants

Vue :

* charges.php : Vue HTML de notre page de charges des enseignants

**Analyse et Implémentation :**

Le RA accède à la page des charges des enseignants par un bouton situé sur la barre de navigation. Lors du chargement de la page, deux requêtes SQL sont envoyés vers le serveur. La première retourne simplement les semaines et la seconde retourne les charges des enseignants. Pour cela nous avons fait une jointure avec les tables : Enseignant, Semaine et Serviceenseignantparsemaine. La dernière table est une table virtuelle (VIEW) déjà fournie de base et contient les charges (positifs) de tous les enseignants. Pour récupérer les charges nulles, i.e. celles des enseignants qui n’ont pas encore fait un service, il a fallu faire une jointure. Une fois le résultat obtenu, nous l’affichons sous la forme d’un tableau.

**Problèmes / Bilan :**

Ce cas d’utilisation est le plus simple parmi les autres. Nous avons sécurisé ce cas en ajoutant une protection qui consiste à rediriger l’utilisateur s’il tente d’accéder à cette page alors que celui-ci n’est pas connecté.

### 5. Cas 5 : Ajouter/Modifier les descriptions de Modules

**Le déroulement :**

1. Prérequis : Le DDE est connecté.
2. Le DDE accède à la page d’affectation des enseignants

2. Il sélectionne un module.

3. Une requête est envoyée vers le serveur puis nous affichons sous la forme d’un tableau le responsable du module et les enseignants affectés au cours de ce module (Module/Partie).

4a. Il peut affecter un enseignant comme responsable du module.

4b. Il peut affecter un enseignant à un cours de ce module (Module/Partie).

5. Toutes les modifications effectuées sont enregistrés dans une variable temporaire JSON

6. Il valide ses modifications en cliquant sur le bouton « Valider »

7. Une requête asynchrone AJAX est envoyée vers le serveur

8. Un message de confirmation est affiché et la page est rafraichit.

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

* Get\_enseignant.php : retourne la liste des enseignants
* Get\_modules.php : retourne la liste des modules

Contrôleur :

* affectermodules.php : Contrôleur de notre page d’affectation des enseignants
* updateenseignant.php : Code PHP effectuant une requête AJAX pour la validation

Vue :

* affectermodules.php : Vue HTML de notre page d’affectation des enseignants

**Analyse et Implémentation :**

Tout d’abord, nous avons rajouté une nouvelle catégorie d’utilisateur : le directeur des études (DDE). Il s’agit tout simplement d’un super-utilisateur. Nous avons dû modifier le cas 2 de telle sorte qu’un DDE puisse se connecter et modifier le cas 3 car il peut éditer tous les plannings de toutes les promotions.

Pour ce cas d’utilisation, lorsque le DDE accède à la page d’affectation des enseignants, il peut choisir le module par le biais d’un formulaire et grâce à la fonction get\_modules. Nous envoyons donc deux requête SQL vers le serveur pour qu’il nous retourne le responsable du module et les enseignants affectés au cours de ce module (Module/Partie). La 1ère requête simple utilise seulement le champ « responsable » de la table Module et la 2ème requête utilise les champs : partie et enseignant de la table ContenuModule. Ensuite nous affichons le résultat sous la forme d’un tableau.

Nous ajoutons une nouvelle colonne dans notre tableau où pour un autre formulaire select contenant la liste des enseignants récupérés grâce à la fonction get\_enseignant. Le DDE pourra affecter un enseignant grâce à ce formulaire. Toutes les modifications sont enregistrées dans une variable temporaire. Lorsque que le DDE confirmera ces modifications en cliquant sur le bouton Valider (celui-ci est similaire à un bouton Commit), nous enverrons une requête asynchrone AJAX vers updateenseignant.php qui effectuera des requêtes SQL du type update dans la BD. Ensuite, nous affichons un message de confirmation puis nous rafraichissons la page.

**Problèmes / Bilan :**

Pour implémenter ce cas, nous nous sommes inspirés du cas 3 pour les requêtes asynchrones. Nous aurions pu rajouter une fonctionnalité permettant de désaffecter un enseignant (NULL) d’un cours ou de la responsabilité d’un module et également l’ajout/supression des parties. Cela n’a pas pu être fait par manque de temps. Toutefois, nous avons encore sécurisé ce cas en ajoutant une protection qui consiste à rediriger l’utilisateur s’il tente d’accéder à cette page alors que celui-ci n’est pas un DDE.

### 6. Cas 6 : Calculer le service d’un enseignant

**Le déroulement :**

1. Prérequis : Le DDE est connecté.
2. Le DDE accède à la page des services des enseignants.

2. Une requête SQL est envoyée puis traité vers le serveur.

3. La nouvelle interface affiche un tableau où les lignes sont représentées par des enseignants, les colonnes par le type du cours (CM,TD,TP,Projet) et les cases par des services.

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

Contrôleur :

* services.php : Contrôleur de notre page de services des enseignants

Vue :

* services.php : Vue HTML de notre page de services des enseignants

**Analyse et Implémentation :**

Le DDE accède à la page des charges des enseignants par un bouton situé sur la barre de navigation. Lors du chargement de la page, une requête SQL est envoyée vers le serveur. Elle retourne le service des enseignants. Pour cela nous avons fait une jointure avec les tables : Enseignant, et ContenuModule. Une fois le résultat obtenu, nous l’affichons sous la forme d’un tableau.

**Problèmes / Bilan :**

Ce cas d’utilisation est très simple et similaire au cas 4. Nous avons encore sécurisé ce cas en ajoutant une protection qui consiste à rediriger l’utilisateur s’il tente d’accéder à cette page alors que celui-ci n’est pas un DDE.

### 7. Cas 7 : Exporter sous format CSV, iCal, ...

**Le déroulement :**

1. L’utilisateur accède à la page d’exportation des plannings.

2. Il choisit une promo et un format sous lequel il veut exporter.

3. Les requêtes SQL du cas 1 sont exécutées

4. Insertion des résultats ligne par ligne dans un fichier CSV

5. L’utilisateur télécharge le fichier CSV exporté

Pour cela, nous avons utilisé les fichiers suivants :

Modèle :

* Get\_promo.php : retourne la liste des promos

Contrôleur :

* export.php : Contrôleur de notre page d’exportation de plannings
* exportcsv.php : Code PHP exportant le planning sous le format CSV

Vue :

* export.php : Vue HTML de notre page d’exportation de plannings

**Analyse et Implémentation :**

L’utilisateur accède à la page d’exportation de plannings puis sélectionne une promo (grâce à la fonction get\_promo) et un format sous lequel il veut exporter. Pour ce cas, nous proposons à l’utilisateur d’exporter seulement en CSV. Ensuite, nous utilisons exactement les mêmes requêtes SQL exécutées par le cas 1. Au lieu d’afficher les résultats des requêtes dans une vue HTML, nous l’insérons ligne par ligne dans un fichier CSV grâce à une fonction annexe dans exportcsv.php. Une fois l’insertion terminé, l’utilisateur télécharge le planning en format CSV.

**Problèmes / Bilan :**

Ce cas est similaire au cas 1 à l’exception des vues qui diffèrent. Nous aurions pu exporter le planning en format ICAL mais ceci n’a pas été fait à cause du manque de temps.

### 8. Cas 8 : Exploiter les traces (pas implémenté)

**Analyse:**

Ce cas d’utilisation optionnel n’a pas été implémenté par manque de temps. Nous aurions pu faire un fichier log qui retrace toutes les modifications concernant les plannings (placement/déplacement) avant ou après avoir effectué un commit. Nous notifierions avec le login du RA, l’heure ainsi que le type de modification. Nous avions pensé à créer une Classe DéplacementPavé qui indique l’ancienne et la nouvelle semaine du pavé déplacé. En effet, il est plus simple d’annuler le déplacement d’un pavé en échangeant son ancienne semaine avec sa nouvelle semaine.

## Recette

Nous avons développé l’application « PlanningparSemaine » qui couvre presque toutes les fonctionnalités attendues par le cahier des charges à l’exception du dernier cas.

Notre application permet à un utilisateur de consulter les plannings d’une promotion et à un responsable d’année d’éditer la promotion dont il est en charge.

Nous avons testé le bon fonctionnement de notre application. Lors de la phase du test nous avions aussi testé les scénarios anormaux qui pouvaient se passer (comme une saisie incorrecte du login ou mot de passe, accès à une page non autorisé…). Donc tous les cas normaux et anormaux ont été testés, et à chaque fois que l’utilisateur essayait de faire une action interdite ou erronée, on lui affichait un message d’erreur.

Après l’implémentation de chaque cas d’utilisation, nous testions si les fonctionnalités offertes correspondaient bien à celles attendues dans le cahier des charges. Dans le cas où ceci n’était pas respecté nous rajoutions des contraintes ou des filtres permettant de résoudre le problème.

## Bilan

Nous avons eu toutefois quelques soucis lors du passage d’une machine Windows à une machine Ubuntu pendant la phase d’implémentation de certains cas à cause de certaine incompatibilité aux niveaux des fonctions ou due à la casse.

Grâce à une bonne analyse, nous avons pu faire une implémentation rapide et efficace. Toutefois, le respect du cahier des charges ainsi que toutes les contraintes techniques nous a rendus plus perplexe et plus minutieux lors de la phase des tests et des validations.

Conclusion

Ce projet nous a permis de mettre en pratique les différentes étapes de modélisation vues en cours qui nous permettent de faire l’analyse et la conception du système, de même nous avons appris à combiner le PHP à toutes les compétences que nous avions déjà acquises en Web (SQL, JavaScript, XPath) et d’autres que nous avons appris à prendre en main (AJAX, JSON).

La séparation des couches du problème sous la forme Modèle-Vue-Contrôleur nous aidé à avoir une meilleur organisation et structuration des fichiers constituants le système, et ce de plusieurs façons : en étudiant d'abord le système de manière globale, afin de dégager les différentes itérations des cas d’utilisations, puis en étudiant en détail chacune de ces itérations, nous sommes parvenus à produire le système.