**Dossier de projet**



**Formation Développeur web**

**&**

**Web Mobile**

**Session 2020 – Domain Loïc**

# Introduction

Dans le cadre de ma formation de développeur web et web mobile au centre AFPA de Dunkerque, j’ai effectué mon stage au sein de l’association « Ferme d’animations éducatives des rives de l’Aa » de Gravelines. Ce stage s’est déroulé sur une période de 12 semaines, du 25 mai 2020 au 14 août 2020. Durant cette période, j’ai eu l’opportunité de mettre en application ce que j’ai acquis lors de la formation, mais aussi d’apprendre de nouvelles techniques. Mon travail a été essentiellement fait à distance, en télétravail, ce qui est quelque chose de nouveau pour moi. Pendant cette période, j’ai eu l’occasion de pouvoir concevoir une application web de comptabilité pour faciliter cet aspect pour l’association.

Dans un premier temps, j’ai effectué des recherches en téléchargeant des logiciels de comptabilité gratuits ou en version d’essai pour avoir une idée plus précise pour la conception du projet. Après cela et une discussion avec mon tuteur concernant les éléments qu’il voudrait ajouter, modifier ou supprimer, j’ai écrit le cahier des charges. L’application permettra de faire une comptabilité simplifiée pour l’association, ce qui permettra au trésorier de gagner du temps et d’avoir une gestion plus facile. Il y aura aussi un accès pour la secrétaire, qui permettra de pouvoir gérer les listes de bénévoles ou d’enfants, de pouvoir imprimer ou exporter les listes, le bilan… Un accès de l’application à la page de connexion de la banque m’a aussi été demandé.

J’ai mis en place une application permettant de gérer la comptabilité de l’association avec un accès sur les écritures comptables, mais aussi la possibilité d’ajouter, modifier ou supprimer des comptes dans le plan comptable ou encore la gestion des fournisseurs, des événements… Pour les listes de bénévoles ou celle des enfants, il est aussi possible d’ajouter, modifier ou supprimer.

J’ai vraiment pris plaisir à réaliser ce projet qui m’a permis de travailler dans des conditions réelles, mais aussi le fait de concevoir un projet qui sera utilisé régulièrement par l’association m’apporte une satisfaction supplémentaire.

# Remerciements

Je tenais à remercier toutes les personnes qui ont contribué à ce stage, mais aussi l’équipe de l’AFPA de Dunkerque pour cette formation enrichissante.

Pour commencer, merci à ma formatrice Mme Martine Poix pour la formation développeur web et web mobile. Pour tous ses conseils durant ma période de formation, mais aussi pour l’aide qu’elle a pu me fournir durant mon stage.

Mr Gilles Domain, mon maître de stage, pour m’avoir fait confiance durant ses 12 semaines de stage en entreprise, ce qui m’a permis de m’accomplir totalement dans mes missions.

Enfin, je tiens à remercier toutes les personnes qui m’ont aidé, conseillé ou relu lors de la conception du projet ou de ce dossier de projet.

# Présentation de l’association

**Association « Ferme d’animations éducatives des rives de l’Aa » :**

# Liste des compétences du référentiel

|  |
| --- |
| **Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité** |
| Maquetter une application. |
| Réaliser une interface utilisateur web statique et adaptable |
| Développer une interface utilisateur web dynamique |
| Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce |
| **Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité** |
| Créer une base de données |
| Développer les composants d’accès aux données |
| Développer la partie back-end d’une application web ou web mobile |
| Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce |

# Projet : Application web de comptabilité

## Cahier des charges

La Ferme d’animations m’a confié la réalisation d’une application de comptabilité pour simplifier le travail du trésorier, mais aussi pour faciliter l’accès à certaines informations concernant l’association comme les listes de bénévoles ou des enfants comme énoncé plus haut.

Il a fallu avant tout définir les points essentiels à la réalisation du projet, c’est ce que comporte mon cahier des charges.

Avant la rédaction du cahier des charges, comme dis plus tôt, j’ai d’abord effectué des recherches pour avoir une idée plus précise du projet final. J’ai ensuite rédigé le cahier des charges avec les informations trouvées, ou celle fournie par mon tuteur avec tout ce qu’il souhaité intégrer dans l’application.

Cette application va être utilisée uniquement localement sur une seule machine, mais avec un accès pour plusieurs personnes comme le trésorier, la secrétaire, le président, etc. Pour cela, j’ai conçu une interface de connexion qui permettra de pouvoir gérer des droits utilisateurs, mais aussi pour qu’aucune autre personne qui n’a pas l’autorisation d’aller sur l’application puisse y pénétrer. Effectivement, l’ordinateur qui est utilisé dans l’association peut passer dans différentes mains et le nombre de passages lors des animations dans le bâtiment est assez conséquent. Cette connexion permettra donc d’éviter ce genre de soucis.

Ensuite, j’ai détaillé comment sera le menu. Il se compose d’un sous-menu "gestion" avec les différentes listes, un sous-menu "journaux" qui correspond aux journaux comptables (charges, recettes, banque et caisse), un sous-menu "budget" avec le budget actuelle et prévisionnel, un sous-menu "impressions" et "exporter" pour pouvoir imprimer les listes, les comptes ou exporter le bilan par exemples. Un sous-menu "comptes" donnera l’accès au lettrage, au plan comptable, etc. Et pour finir un sous-menu "outils" avec une calculatrice et un moyen pour stocker des documents importants pour l’association.

J’ai réalisé les diagrammes, ainsi que la base de données que j’ai montrée à mon tuteur et à Martine Poix. Une fois ces différentes étapes validées, j’ai pu commencer à coder.

## Logiciels utilisés

**Wampserver64 / phpMyAdmin :** C’est une plateforme de développement web permettant de faire fonctionner localement des scripts PHP. Elle comprend trois serveurs (Apache, Maria DB, MySQL). MySQL permettra la création de la base de données.

**Visual Studio Code :** C’est un éditeur de code que j’ai utilisé depuis le début de ma formation. Beaucoup d’extensions permettent de faciliter le développement.

**ArgoUML :** C’est un logiciel libre de création de diagrammes UML (cas d’utilisation, diagramme des classes…).

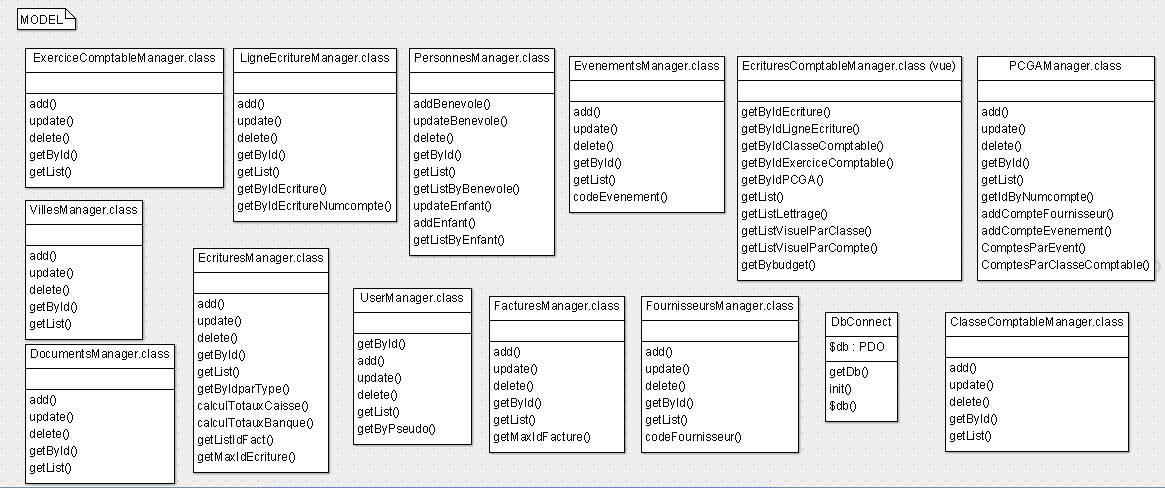
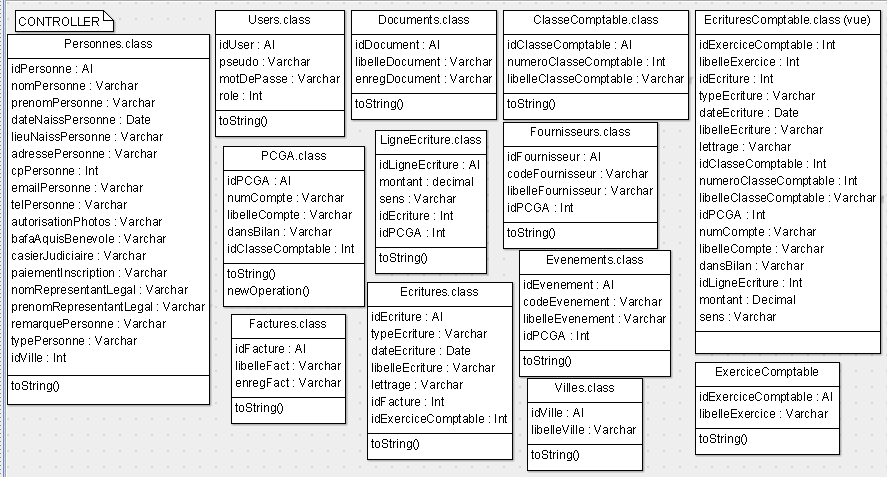
**JMerise :** C’est un logiciel dédié à la modélisation des modèles conceptuels de donnée pour Merise (méthode d'analyse, de conception et de gestion de projet informatique).

**Microsoft Word :** C’est un logiciel de traitement de texte publié par Microsoft.

**Microsoft Excel :** C’est un logiciel tableur de la suite bureautique Microsoft Office développé et distribué par Microsoft.

## Diagrammes

### Diagramme de classes

Le diagramme de classes permet de décrire clairement la structure de l’application en modélisant ses classes, ses attributs et les fonctions. Il permet d’avoir une idée assez précise sur les classes et les managers créer pour l’application.

### Diagramme de cas d’utilisation

Le diagramme de cas d’utilisation est utilisé pour donner une vision globale du comportement fonctionnel d’un système logiciel. Très utile pour la présentation au tuteur de stage.

*Insérer le diagramme de cas d’utilisation*

### Diagramme de flux

Le diagramme de flux a pour but de montrer les différentes interactions et toutes les possibilités qui s’offrent à l’utilisateur lors de sa navigation. Pour ce projet, il a permis à mon tuteur d’avoir une idée du rendu visuel de l’application (voir annexes).

Les diagrammes m’ont permis d’avoir les points de vue des personnes concerné par l’application et de corriger ce qui ne leurs convenait pas. En accord avec eux, j’ai pu commencer à travailler sur la partie front-end.

## Dictionnaire de données :

La première étape, dans la construction d’une base de données, consiste à réunir tous les documents et les divers autres supports représentatifs des données que l’on souhaite modéliser. Ensuite, on donne un nom à l’attribut, un type et une taille pour les chaînes de caractère par exemple (les id ne figure pas dans le dictionnaire de données mais lors de la création des tables).

|  |  |  |
| --- | --- | --- |
| **Nom** | **Type** | **Description** |
| libelleFact | Varchar | Libellé de la facture |
| enregFact | Varchar | Chemin de la facture (image) |
| libelleExercice | Int | Année de l’exercice |
| typeEcriture | Varchar | (charges, recettes …) |
| dateEcriture | Date | La date sur la facture … |
| libelleEcriture | Varchar | Description de l’écriture |
| lettrage | Varchar | Lettrage comptable |
| libelleVille | Varchar | Nom de la ville |
| pseudo | Varchar | Identifiant de connexion |
| motDePasse | Varchar | Mot de passe en MD5 |
| role | Int | Admin ou User |
| libelleDocument | Varchar | Description document |
| enregDocument | Varchar | Chemin du document |
| numeroClasseComptable | Int | Numéro des classes (PCG) |
| libelleClasseComptable | Varchar | Compte de tiers, financiers … |
| numCompte | Varchar | Numéro de chaque comptes |
| libelleCompte | Varchar | Description du comptes |
| dansBilan | Varchar | Apparait au bilan (oui/non) |
| codeFournisseur | Varchar | Code unique au fournisseur |
| libelleFournisseur | Varchar | Nom du fournisseur |
| codeEvenement | Varchar | Code unique à l’événement |
| libelleEvenement | Varchar | Description de l’évenement |
| nomPersonne | Varchar | Nom (bénévole, enfant) |
| prenomPersonne | Varchar | Prénom (bénévole, enfant) |
| dateNaissPersonne | Date | Date de naissance |
| lieuNaissPersonne | Varchar | Ville de naissance |
| adressePersonne | Varchar | Adresse (enfant, bénévole) |
| cpPersonne | Int | Code postal |
| emailPersonne | Varchar | E-mail de la personne |
| telPersonne | Varchar | Numéro de téléphone |
| autorisationPhotos | Varchar | Autorisation photo oui/non |
| bafaAquisBenevole | Varchar | Bafa acquis / non acquis |
| casierJudiciaire | Varchar | Casier bénévole fourni oui/non |
| paiementInscription | Varchar | Mode de paiement |
| nomRepresentantLegal | Varchar | Nom représentant enfant |
| prenomRepresentantLegal | Varchar | Prénom représentant enfant |
| remarquePersonne | Varchar | Pour plus d’informations |
| typePersonne | Varchar | bénévole, enfant |
| montant | Decimal | Somme de chaque ligne |
| sens | Varchar | Débit ou crédit |

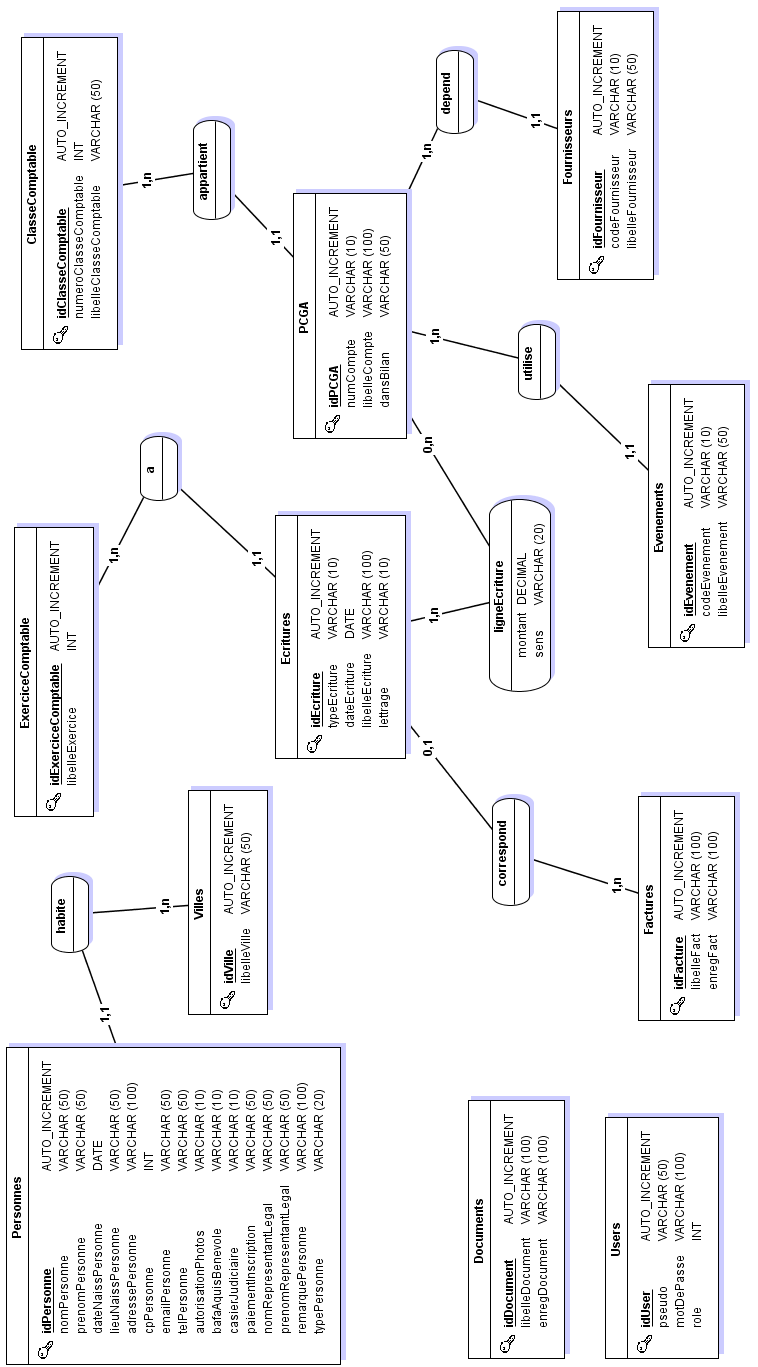
### Création d’une vue

Pour plus de facilité à gérer le lettrage par exemple, j’ai décidé de créer une vue regroupant les tables ExerciceComptable, Ecritures, ClasseComptable, PCGA et LigneEcriture.



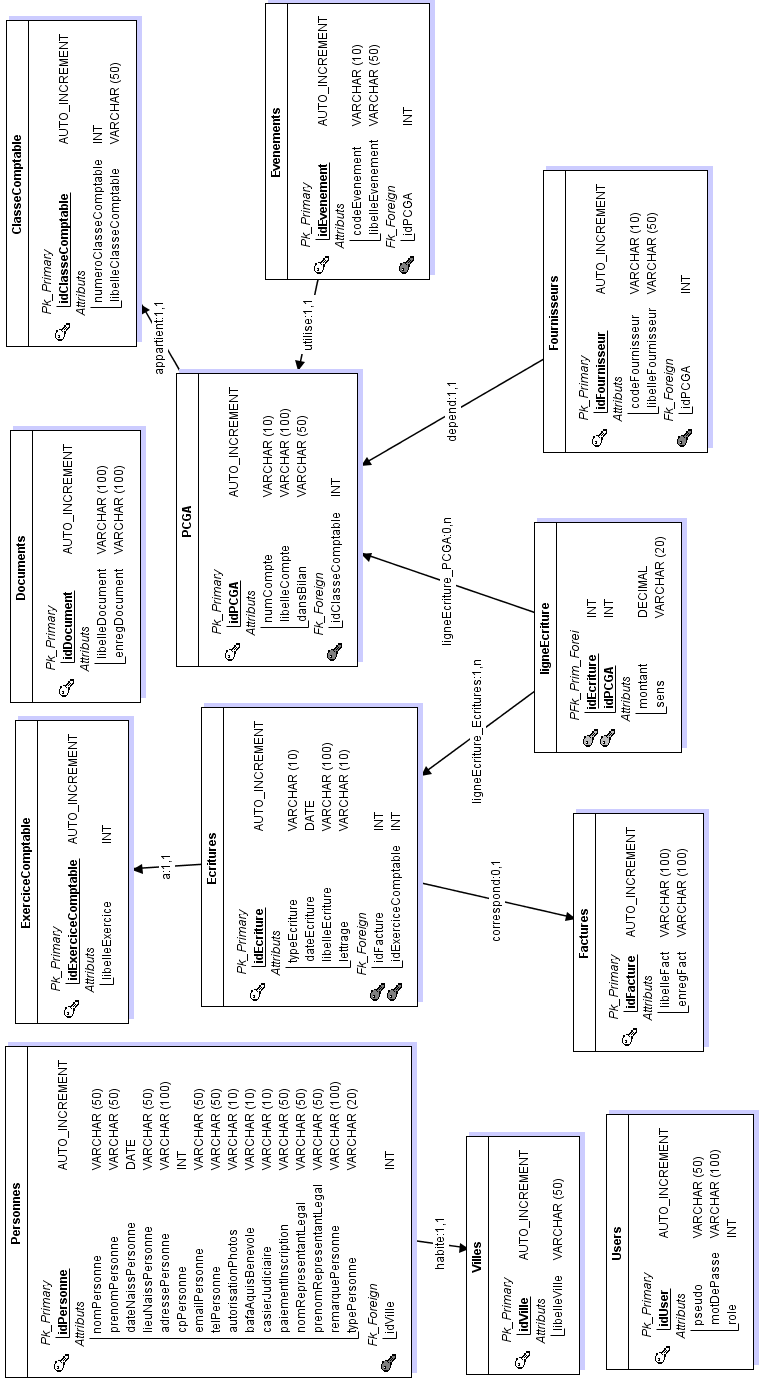
## MCD

Le **MCD** ou **Model Conceptuel de Donnée** est une représentation graphique qui permet de comprendre comment les éléments sont liés entre eux. Les cardinalités sont très importante pour la création du **MLD.**



## MLD

Le **MLD** ou **Model Logique de Donnée** se situe chronologiquement juste après le **MCD**. Les objets présentés sont désormais des tables ainsi que les liens qui les unissent.



## Langages et techniques utilisés

Durant mon stage et pour la réalisation du projet, j’ai utilisé différents langages de programmation :



* **HTML :** Ce langage permet l’affichage du contenu de la page web à travers les différents navigateurs (Google Chrome, Firefox, Safari, Opera, Internet Explorer…).



* **CSS :** Le CSS (Cascading Style Sheet) quant à lui va permettre de donner des règles de mise en page (couleur, typographie, taille, hauteur …) pour la page web.

****

* **JavaScript :** J’ai utilisé le JavaScript afin de gérer des événements et dynamiser certaine partie de mon site (œil de connexion par exemple).

****

* **Ajax :** L’Ajax est une architecture informatique que j’ai utilisé pour récupérer les codes fournisseurs en base de données, pour vérifier si le code ajouté n’existe pas déjà.
* **PHP :** Ce langage orienté objet est principalement utilisé pour produire des pages web dynamique via un serveur http.

## La structure

### Modèle MVC (Model/View/Controller) :

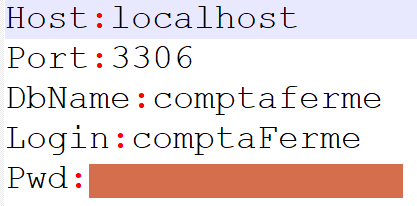
Pour le projet, j’ai utilisé une architecture **MVC (Model View Controller)**. Il permet dans un premier temps de bien organiser son code. Elle est divisée en trois parties :

* **Model :** Cette partie gère les données du site, elle a pour rôle de récupérer les informations en base de données. Il contient les données et la logique en rapport avec les données. Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. On y trouve donc entre autres les requêtes SQL.
* **View :** Elle a pour rôle l’affichage, contient la partie visible de l’interface graphique.
* **Controller** **:** Il joue le rôle d’intermédiaire entre le **Model** et la **View**. Le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le contenu à afficher à la vue. Il traite les actions de l’utilisateur, modifie les données du modèle et de la vue.

Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet. En effet, étant donné qu'il y a une séparation entre les différentes couches, il sera plus facile de modifier uniquement la partie vue ou encore uniquement le traitement de la requête dans le contrôleur.

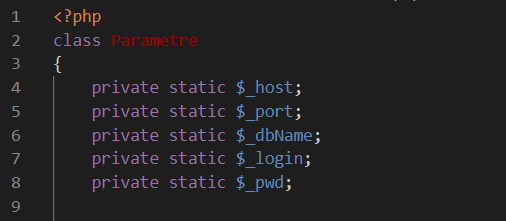
### Connexion à la base de données, structure et interactions :

Afin de connecter notre projet à la base de données nous allons utiliser un fichier à la racine de notre projet nommé « parametre.ini » avec les informations nécessaires pour la connexion à la BDD :

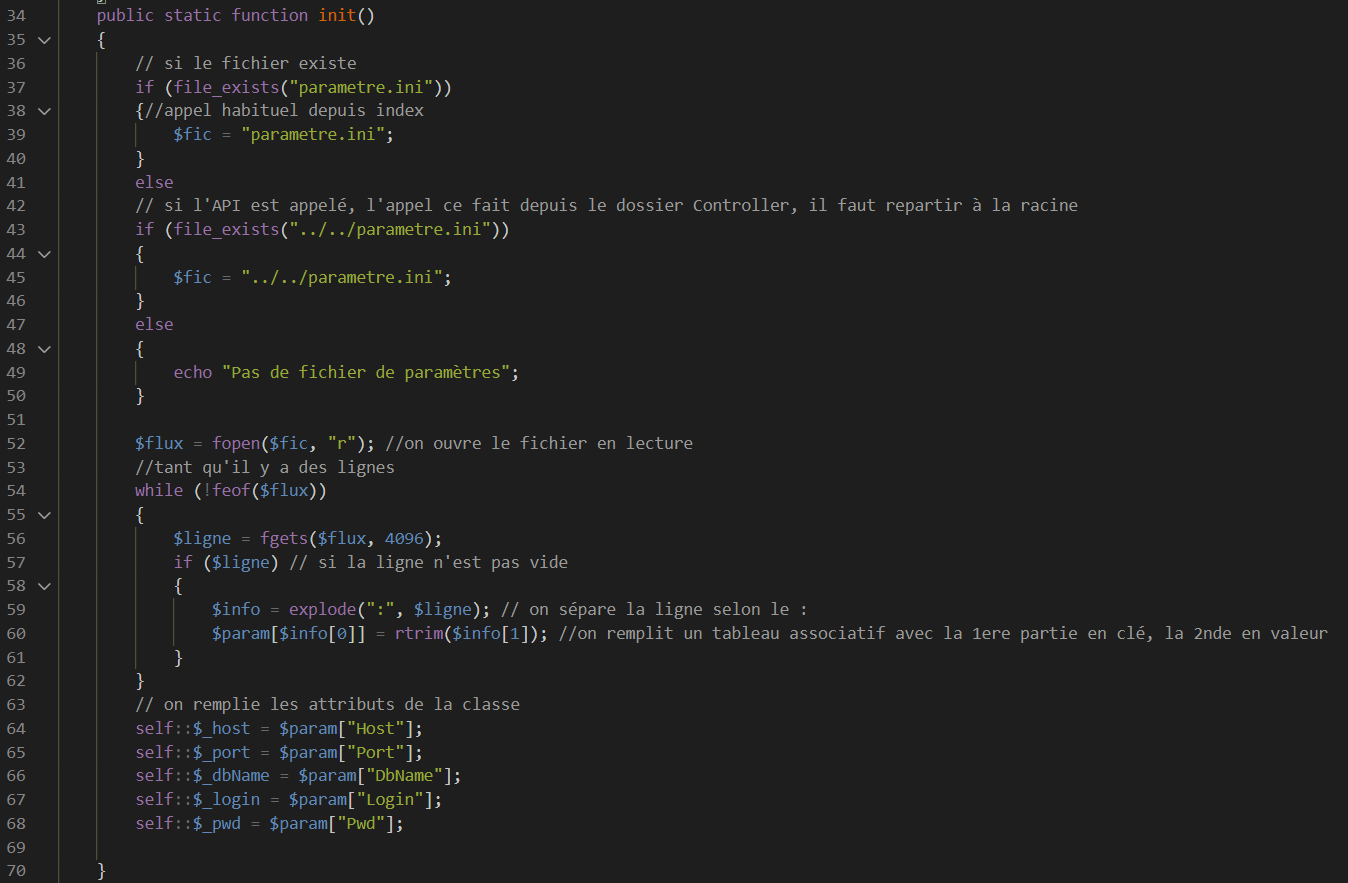


Le choix de rester en local a été décidé avec le président de l’association qui préfère que l’application soit accessible seulement sur l’ordinateur personnel de l’association.

Puis nous allons utiliser, dans le dossier Controller, un fichier nommé « Parametre.class.php » afin d’utiliser les informations de « parametre.ini » :



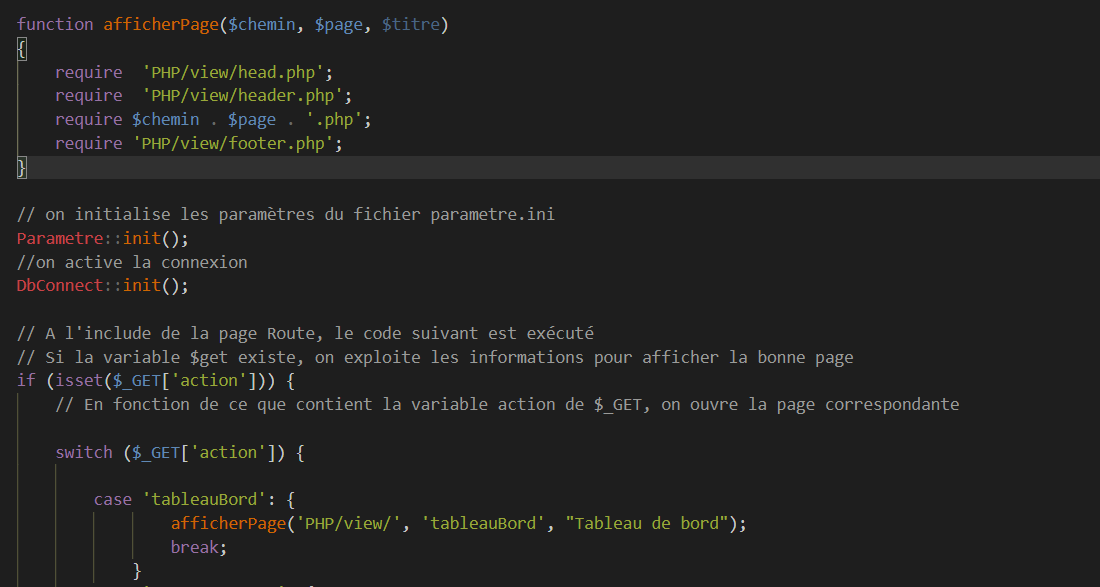
*Fichier « Parametre.class.php » : les attributs*



*Fichier « Parametre.class.php » : fonction init()*

Grâce à cette fonction « init() » on récupère les informations de « parametre.ini » en triant ligne par ligne les différents paramètres puis en les rentrant chacun dans un tableau. La fonction « Parametre::init() » va être au cœur du projet, et va être utilisée par plusieurs fichiers.

Par exemple, lorsqu’on va exécuter le projet dans un navigateur, c’est le fichier « index.php », qui se trouve lui aussi à la racine du projet, qui va se charger de gèrer les ouvertures et les interactions entre toutes les pages du projet. Il faut voir ce fichier comme le coordinateur de toutes pages. Il aura donc besoin de la fonction « Parametre::init() » pour la connexion à la base de données.



*Fichier «index.php » : fonction afficherPage, initialisation BDD*

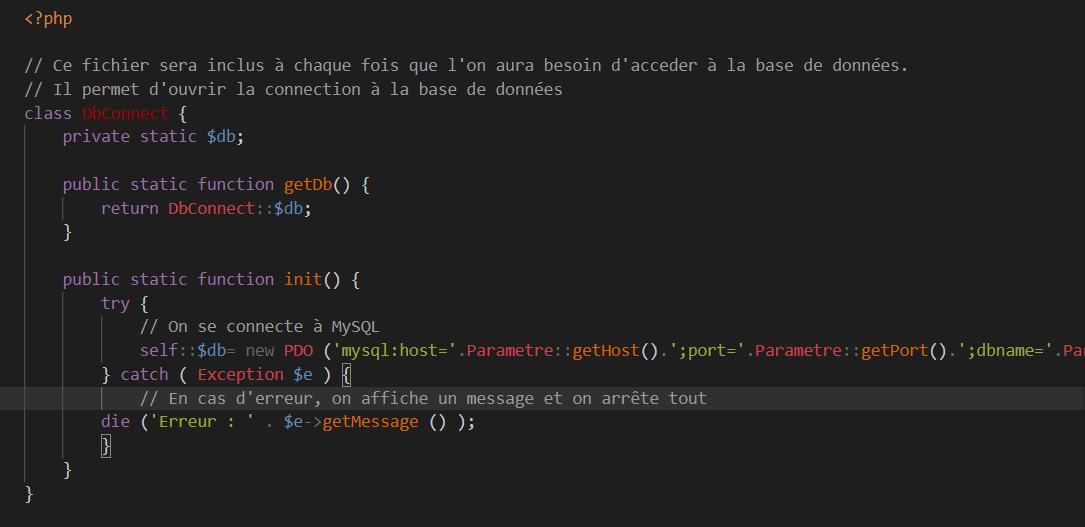
« index.php » comprend donc la fonction « afficherPage() » et un switch qui va comprendre chacune des pages (ci-dessus on peut voir le premier case du switch à savoir la page « tableauBord.php »).

Toute page chargée va donc afficher tout d’abord le Header.php ainsi que le contenu escompté (exemple « listeMenu.php ») et son Footer.php.

Le « Header.php » et le Footer.php sont donc communs à presque toutes les pages du projet. En effet, certaines pages sont un peu différentes, comme pour les pages d’impression ou le header avec le menu a été retiré.

On peut voir ci-dessus la présence de la fonction «DbConnect::init()». Il va permettre à toutes les pages d’être connectées avec la base de données. Cette fonction est réaliser dans le fichier « DbConnect.class.php ».

La classe « DbConnect.class.php » va permettre la connexion à la BDD en utlisant les informations de « parametre.ini » grâce à la classe « Parametre.class.php » :



*Fichier «dbConnect.class.php » : fichier pour la connexion BDD*

### Sécurité

Nous allons aborder un point essentiel dans un projet, celui de la sécurité. Après sa réalisation, votre projet web peut être infesté de vulnérabilité. Elles peuvent constituer une cible pour les utilisateurs malveillants qui voudraient détourner son fonctionnement initial.

* Connexion par utilisateurs avec un pseudo et mot de passe afin de pouvoir gérer les droits, mais aussi de pouvoir empêcher toutes personnes n’ayant pas l’autorisation de voir la comptabilité d’y accéder (beaucoup de passage).
* Les mots de passe sont cryptés en MD5. C’est un algorithme de hachage qui permet de protéger en cryptant les mots de passe en Base de données.
* De plus, l’application sera utilisée localement. Le président souhaite qu’elle soit installée sur l’ordinateur de l’association et accessible nulle part ailleurs.
* Dans chaque partie de ma structure MVC j’ai placé un index. PHP vide afin d’empêcher toute tentative d’accès au dossier qui le contient. Si l’utilisateur malveillant tente d’accéder au dossier il verra uniquement une page blanche.

## Réalisation

Afin de voir à quoi ressemble l'application web, je vous invite à vous rendre à la page ?? dans la partie "Annexes", vous pourrez ainsi étudier le visuel de l'application sous forme de maquettes, ainsi que les interactions entre les différentes pages sous la forme d'un Diagramme de Flux.

Toute l’application est codée avec le modèle de boîte FlexBox (pour Flexible Box) afin de rendre le design de l'application responsive.

*Ce modèle est différent du modèle de boîte standard car il fournit une façon efficace de disposer, aligner et distribuer l'espace entre les éléments d'un container, même lorsque leurs dimensions sont inconnues et/ou dynamiques (redimensionnables).*

Ainsi l’application sera totalement responsive, c’est-à-dire qu’elle sera consultable et adaptée pour des écrans de tailles très différentes.

L’application est également responsive par le fait que les tailles des éléments de l’application sont configurées avec les unités (vh/vw/em ou %) et non pas en pixel (px).

Les couleurs et le design de l’application ont été choisi lors d’une discussion avec les personne qui utiliseront l’application ou concerner par celle-ci. Les couleurs sont un rappel à celle du logo de l’association, que j’ai donc aussi utilisé pour le design de mon dossier de projet.

Comme énoncé plus haut, des listes avec la création d’un CRUD ont été réalisé. Cependant ma partie centrale du projet est la création et la gestion des écritures composées de ligne d’écriture et de l’enregistrement d’images pour les factures.

Pour réaliser cette partie l’enregistrement des lignes et des écritures ont été faites à l’aide du CRUD, du JavaScript, de certaines requêtes Ajax et du php pour uploader les images.

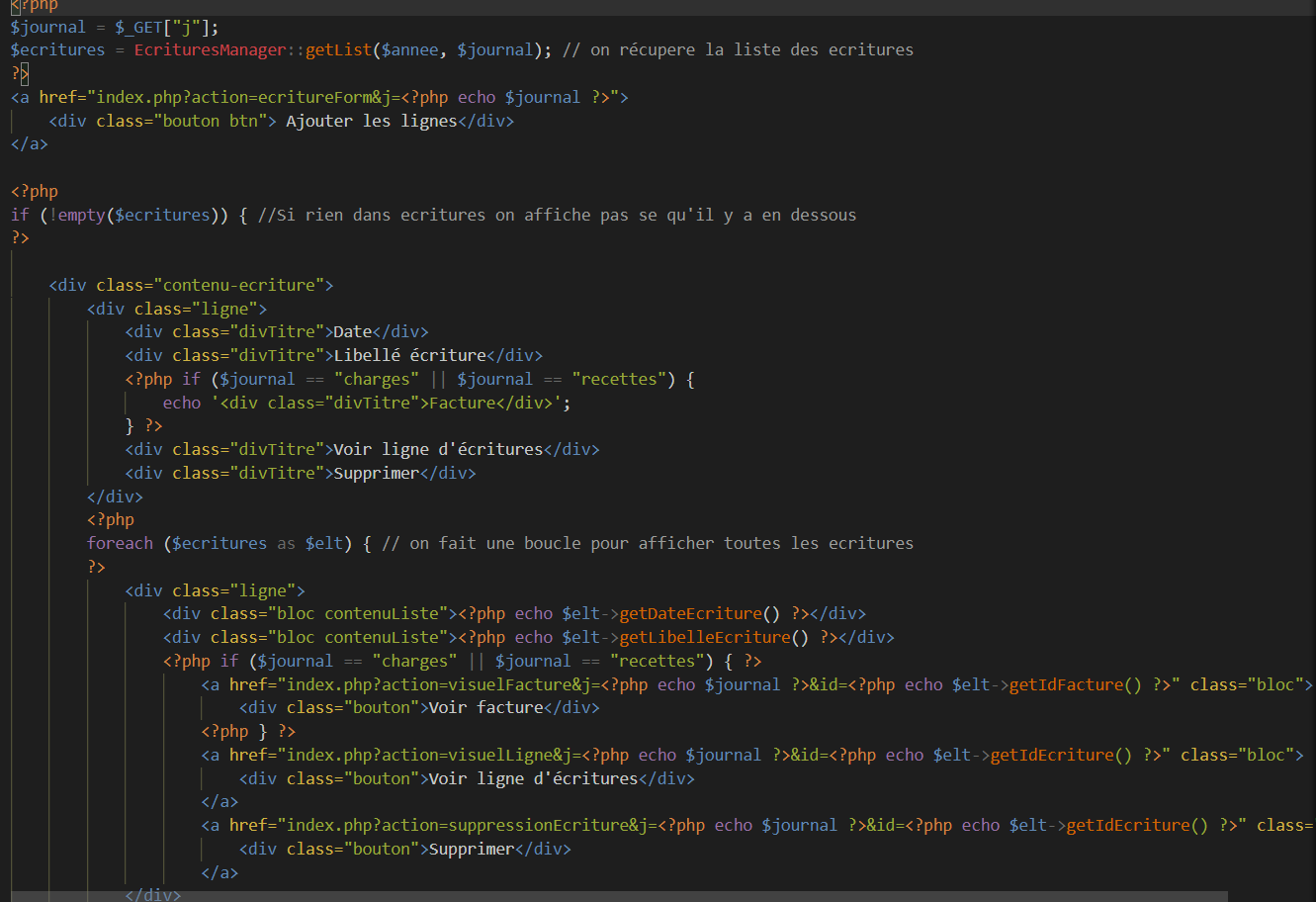
Le CRUD (pour create, read, update, delete) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage d'informations en base de données. Plus généralement, il désigne les opérations permettant la gestion d'une collection d'éléments.

Afin de pouvoir interagir avec la base de données, c’est-à-dire créer, récupérer, modifier ou supprimer des données nous avons besoin de créer des requêtes SQL dans notre code PHP.

Pour cela nous allons créer dans le dossier « Model » les « Managers » des différentes classes que nous avons déjà créé dans le dossier « Controller ».

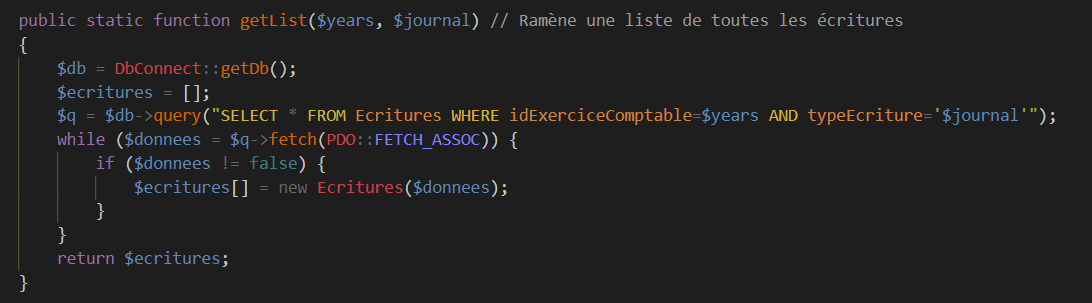
Ainsi, si on prend par exemple la classe : « Ecritures.class », son manager sera «EcrituresManager.class » et contiendra le CRUD en lien avec la classe Ecritures. On place aussi dans le manager les fonctions en rapport avec la table Ecriture (dans ce cas) qui pourront être utile par la suite comme « getMaxIdEcriture() » par exemple, qui sert ici à récupérer le dernier idEcriture pour pouvoir enregistrer les lignes d’écritures qui on l’idEcriture comme Foreign Key (clé étrangère).

Pour la présentation des écritures, il a fallu tout d’abord faire une liste des écritures en fonction de l’exercice comptable (de l’année) et du journal, car les écritures ont été séparées à l’affichage dans différents journaux (charges, recettes, banque et caisse). Pour pouvoir les différencier, un « typeEcriture » été donc nécessaire.



*Fichier «ecrituresListe.php » : affichage des écritures comptables*

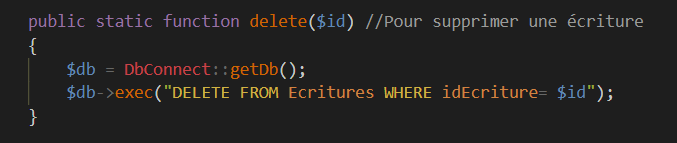
Comme on peut le remarquer, en début de fichier, on récupère toutes les écritures en fonction de l’année (qui est dans une variable session) et du journal (récupérer dans le $\_GET, soit dans l’url de la page) à l’aide du « getList » qui vient du manager. Ensuite, on crée une boucle pour afficher toutes les écritures.



*Fichier «ecrituresManager.class.php » : fonction « getList »*

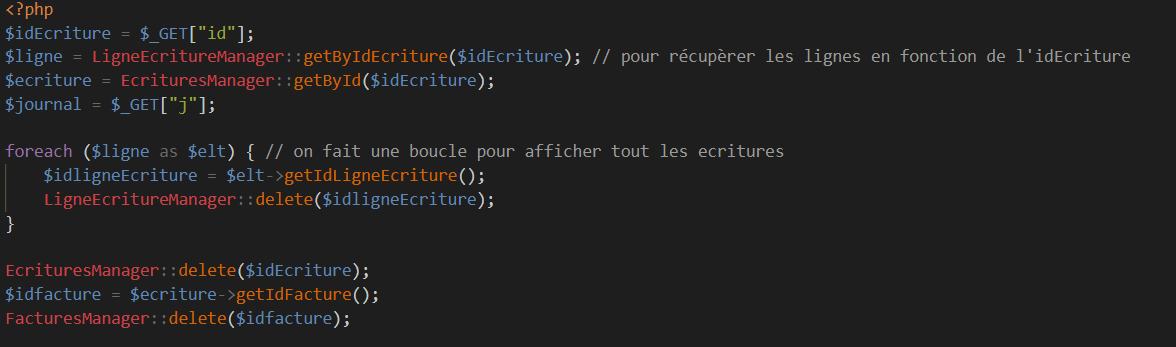
Cette fonction ce connecte à la base de données, puis envoie une requête et met le résultat dans un tableau associatif et retourne le résultat.

Lorsque l’on fait une boucle pour afficher toutes les écritures retourner par cette fonction, on affiche un bouton supprimer sur chaque écritures qui, lors du clique, amènera sur une page de confirmation, puis sur la page d’action qui supprimera les lignes, l’écriture et la facture en fonction de l’idEcriture. La fonction qui est utilisé est la fonction « delete($id) » qui vient de chaque manager.



*Fichier «ecrituresManager.class.php » : fonction « delete($id)»*

Cette fonction est la même pour la facture et les lignes d’écriture mais avec leur id respectif (idFacture et idLigneEcriture).



*Fichier «suppressionAction.php » : lancement suppression des écritures*

Pour la suppression en base de données (tout comme l’ajout), il est très important de prendre en compte l’ordre des enregistrements en fonction des clés étrangères. Ce code permet donc de supprimer avant tout les lignes d’écriture (car idEcriture est utilisé comme Foreign key dans la table LigneEcriture), puis l’écriture et enfin la facture (car idFacture est utilisé comme Foreign key dans la table Ecritures).

Il est donc aussi possible d’ajouter des écritures. Pour cela, après le clique sur le bouton ajouter, on arrive sur un formulaire d’ajout. Ce formulaire est légèrement différent en fonction du journal sélectionné préalablement. Le code est exécuté sur la même page pour tous les journaux mais un if($journal == ‘Le nom du journal’) est utilisé pour pouvoir afficher le bon formulaire en fonction du journal.



*Fichier «ecrituresForm.php » : if en fonction du journal*

Sur l’extrait de code suivant, on peut voir que l’on récupère quatre listes (la liste des évènements, des fournisseurs, des classes comptables et tous les comptes du plan comptable). Ensuite, on récupère le journal pour l’affichage du bon formulaire et les informations de l’exercice comptable par rapport à son id pour la date.

Enfin, on peut voir que l’on affiche les titres du formulaire en ligne et les informations correspondante seront à saisir en dessous dans des inputs ou des selects (liste déroulante). Ce code est utile pour l’affichage de chaque formulaire d’ajout d’écriture.



*Fichier «ecrituresForm.php » : récupération des données et affichage titres*

Pour détaillé le code d’ajout d’écriture, on s’attardera sur le formulaire des charges. Le principe est le même pour les autres journaux mais utilisé différemment.

Tout d’abord, il est important de savoir quand comptabilité l’enregistrement d’une écriture doit se faire lorsque les montants inscrit au débit soit égaux à ceux du crédit.

Ici pour le journal de charges, on doit avoir les comptes de charges (classe 6) au débit et le fournisseur au crédit. Lors de l’enregistrement d’une facture d’achat, les montants peuvent être inscrits dans différents comptes de charges mais le total est aussi inscrit au crédit dans le compte du fournisseur correspondant à la facture.

Le formulaire se présente de la façon suivante :

* Pour le débit : Date / code de l’événement / numéro de compte / libellé du compte / libellé de l’écriture / débit / crédit / « + » (pour ajouter des lignes)
* Pour le crédit : il se présente de la même façon juste en dessous mais avec le code fournisseur à la place du code d’événement et un « enregistrer facture » à la place du « + ».

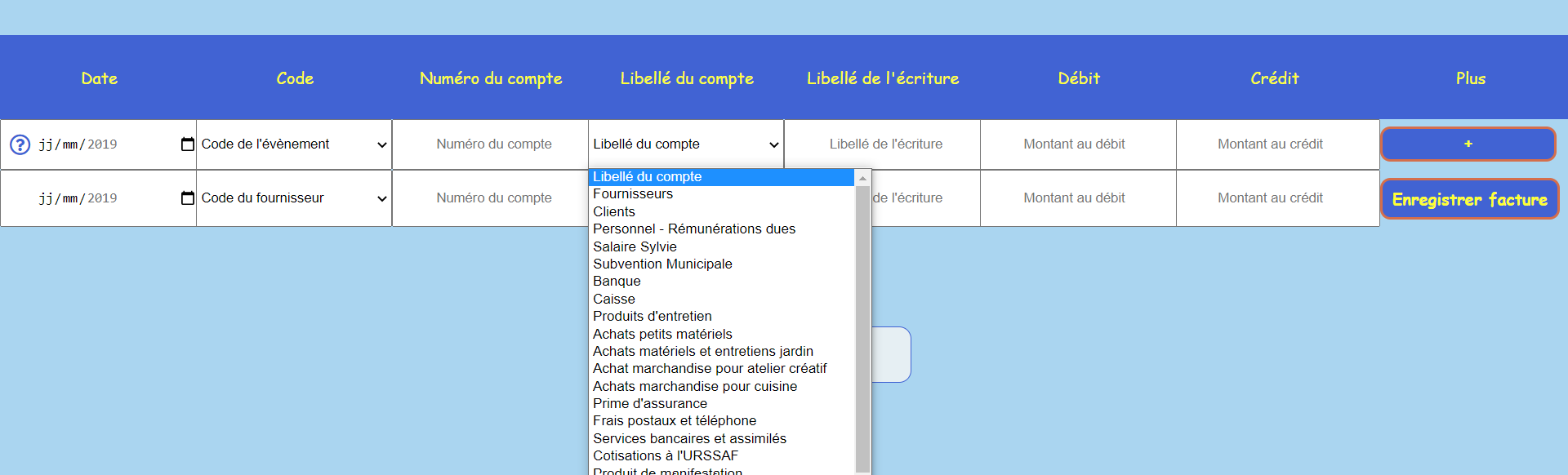


*Fichier «ecrituresForm.php » : Visuel du formulaire d’ajout de lignes*

Le bouton « + » va servir à ajouter des lignes si l’on a besoin de mettre des montants dans des comptes de charges différents. Pour cela, le clonage de ligne a été utilisé, ce qui ma permit de cloner la première ligne et l’ajouter entre les deux lignes de départ.

C’est une technique que j’ai utilisé pour la première fois, j’ai donc dût procéder à des recherches (que je détaillerai plus loin dans ce dossier). A chaque fois que ce bouton est cliqué, une nouvelle ligne identique à la première s’ajoute en dessous.

Maintenant, nous allons nous intéresser à l’utilisation des codes qui fonctionne de la même manière pour chaque ligne mais avec des données différentes (l’un avec les données concernant l’événement et l’autre celles concernant les fournisseurs). Si l’on choisit un code, les libellés de compte changes en fonction de ce code. Par exemple si l’on choisit le code événement « MER » nous auront tous les libellés concernant ce code dans la liste déroulante.

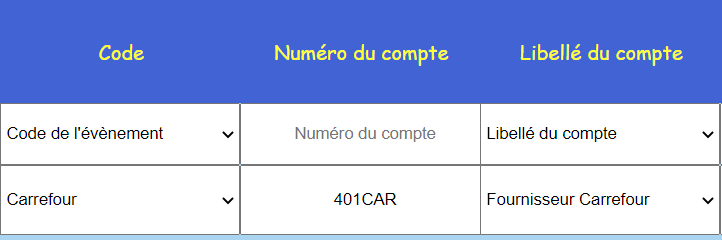


*Fichier «ecrituresForm.php » : Visuel des libellés sans le code d’événement*



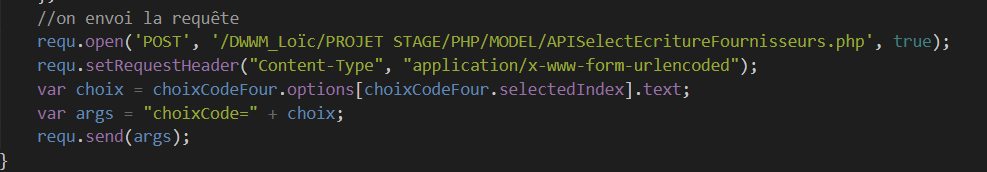
*Fichier «ecrituresForm.php » : Visuel des libellés avec le code d’événement*

Pour le fournisseur, un seul fournisseur est nécessaire donc au moment du choix du code fournisseur le numéro de compte et le libellé doit se remplir :

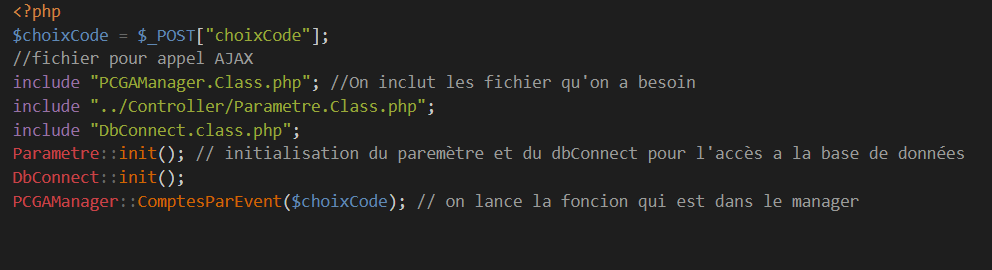


*Fichier «ecrituresForm.php» : Visuel des libellés avec le code Fournisseur*

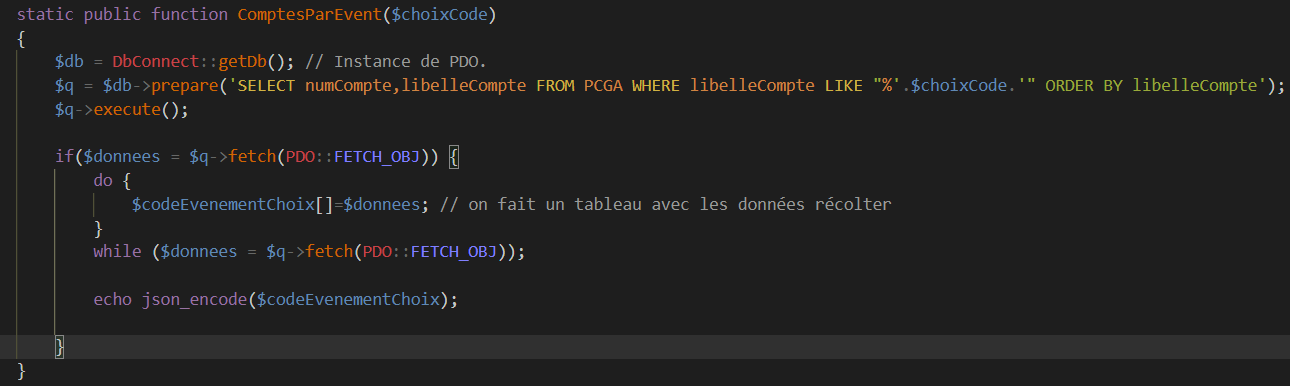
Voici le code qui permet de faire cela (requête Ajax et du JavaScript) :



*Fichier «scriptAjaxEcritureFournisseur.js» : Envoi de la requête*



*Fichier «APISelectEcritureFournisseurs.php» : Appel Ajax*



*Fichier «FournisseursManager.class.php» : fonction utilisé pour l’appel Ajax*

Sur l’extrait de code suivant, la réponse obtenu est utilisé pour remplir la liste déroulante des libellés de comptes. On commence par vider le select et on le rempli avec la réponse obtenu par la requête.

