Partie Modèle : Classe

1. **Introduction du concept des classes en PHP**
2. **Définition des classes du projet ANT**
3. **Classes brutes**
4. **Classes et associations**
5. **Modélisation de la notion d’héritage**
6. **Modélisation des associations entre classes**
7. **Présentation des classes majeures**

**I-Introduction du concept des classes en PHP**

La programmation orienté objet a vu le jour en PHP après sa version 4. Cette approche qui existait bien avant dans plusieurs autres langages, et dont ils doivent leurs succès, vient alors apporter son grain de flexibilité à ce langage du web dynamique. Elle suscite l’intérêt des développeurs depuis la sortie de PHP5.

Elle présente plusieurs avantages dont nous citons ce qui suit :

* La possibilité de réutiliser le code dans différents projets. Les classes ainsi créées pourront avoir une nouvelle vie dans une application tierce.
* Une conception de l’algorithme plus claire et organisée. Le programmeur identifie chaque élément de son programme comme un objet ayant son contexte, ses propriétés et des actions qui lui sont propres.
* Un code modulaire. Chaque type d’objet possède de son propre contexte et ne peut agir avec d’autres suivant des interfaces bien précises. Cela permet d’isoler chaque module et d’en créer séparément de nouveaux qui viendront s’ajouter à l’application. Cette approche est particulièrement employée dans le cas de projets répartis entre plusieurs développeurs.
* Possibilité de s’adapter aux design patterns (motifs de conception) pour une meilleure structure du code.

II- **Définition des classes du projet ANT**

ANT repose sur cette approche modulaire, le modèle du projet a été conçu d’une façon à solliciter une grande abstraction entre le jeu de lignes retournées par la base de données et les manipulations souhaitables de ce jeu. Pour assurer cette abstraction, le jeu de lignes retournées est alors stocké au niveau d’une correspondance d’objet instancié par les valeurs de retour, l’état interne de l’objet ainsi définit permet dans un prochain temps son appel pour des manipulations au niveau du Contrôleur.

L’arborescence du modèle se présente comme suit :

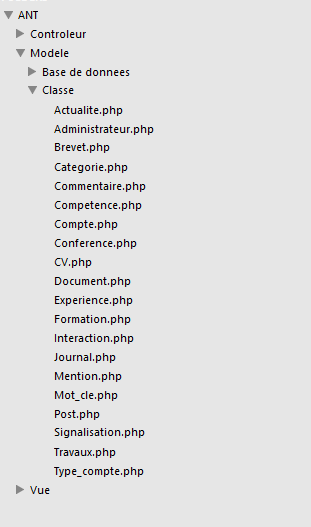


Figure : Arborescence du dossier \_ANT/ Modèle/Classe

**a. Classes brutes**

La diagramme de classe modélise les différentes classes à implémenter dans le projet, on en compte une vingtaine dans le projet ANT, représentées dans le diagramme de classe ci-dessous  :



Figure : Diagramme de Classe \_ projet ANT

Pour écrire le code des classes il faudrait alors prendre compte de plusieurs paramètres, le premier est de rédiger les classes dont leur état brutes indépendamment des associations dont lesquels elles interviennent, elle comportera en un premier temps les attributs, un constructeur, des getters et des setters. Les méthodes seront implémentées au fur et à mesure du besoin fonctionnel.

La classe Administrateur présente l’exemple simplifié du concept, elle comporte un seul attribut, ce qui génère en un premier temps un getter de cet attribut, un setter et un constructeur appelé pour instanciation de la classe de l’extérieur.

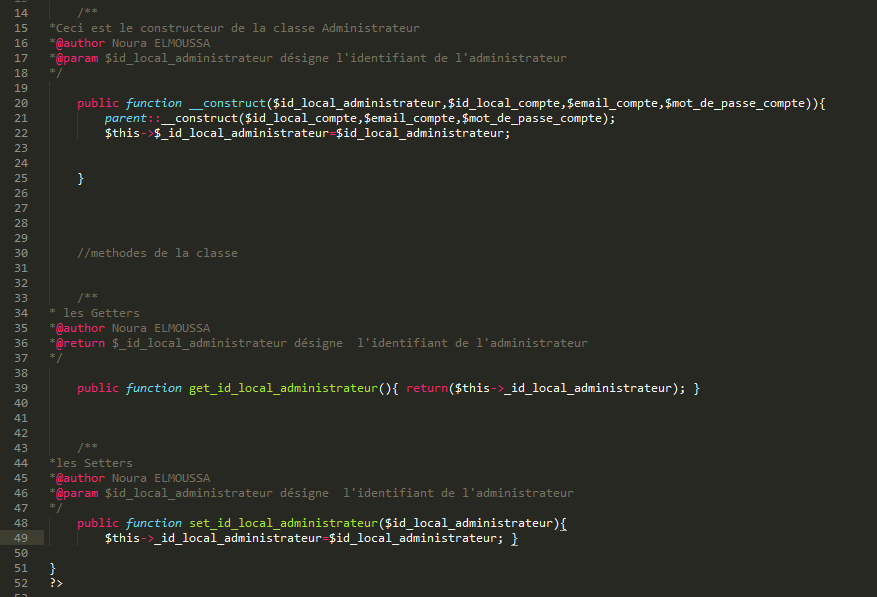


Figure : Classe Administrateur \_ANT/ Modèle/Classe

La classe présente un comportent spécial, dont la mesure où elle est étendue par héritage d’une classe Compte présentant l’ensemble des informations communes entre utilisateurs, en faisant abstraction de son profil au sein de la plate-forme, pour cela il est nécessaire de modéliser cette héritage par un *extends* lors de la définition de la classe .



**b.Classes et associations**

**i. Modélisation de la notion d’héritage**

Plusieurs classes présentent un héritage, cette notion d’héritage est reproduite au sein des classes par un *extends* vers la classe mère aussi bien qu’un *require* du fichier de la classe mère.

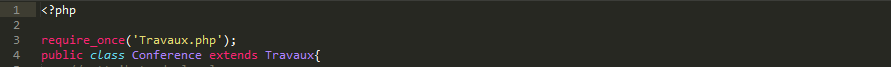


Figure : Classe Conférence \_ANT/ Modèle/Classe

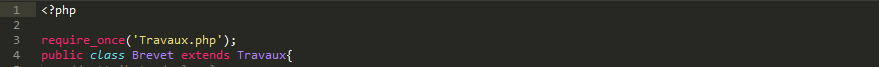


Figure : Classe Brevet \_ANT/ Modèle/Classe

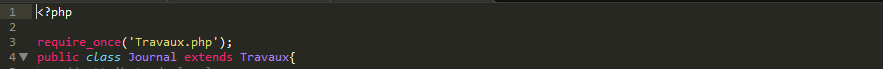


Figure : Classe Journal \_ANT/ Modèle/Classe

**ii.Modélisation des associations entre classes**

Certaines représentations font que les associations entre objets sont reproduites par des tableaux passés en attribut dans la classe. Ce choix rajoute complexité et alourdissement au code. Notre choix converge alors vers une alternative plus fluide, qui permet de modéliser les associations via des méthodes au niveau de la classe.

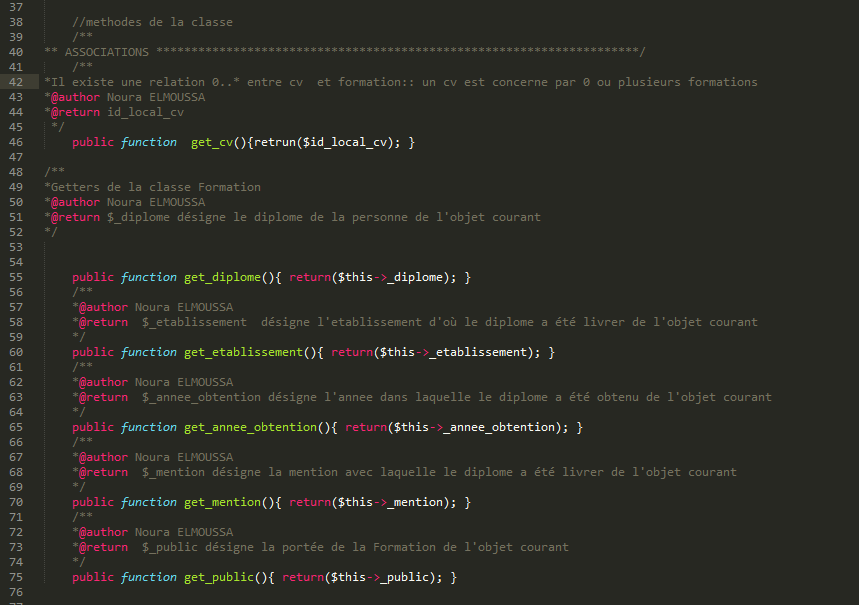


Figure : Association 0..\* entre CV et Formation \_ANT/ Modèle/Classe

**III-Présentation des classes majeures**

Certaines classes du projet présentent un intérêt majeur, les classes compte entre dans plusieurs associations qu’il est d’ordre logique et essentiel de représenter  :

**i-attributs et constructeur**

****

Figure : Attributs et Constructeur de la classe Compte \_ANT/ Modèle/Classe

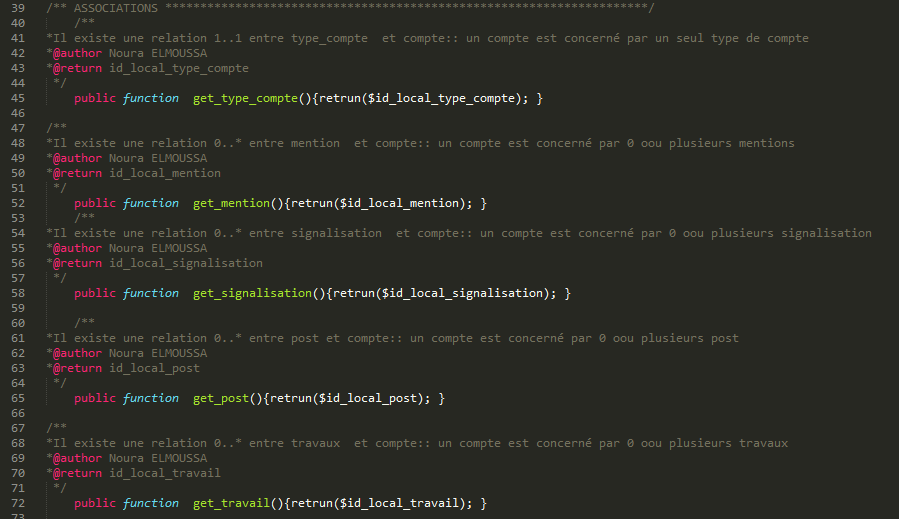
**ii-associations**

Figure : Association (1) de la classe Compte \_ANT/ Modèle/Classe

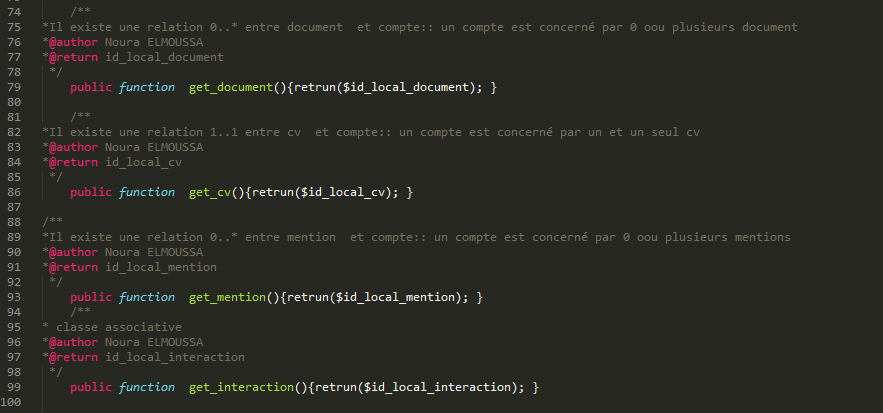


Figure : Association (2) de la classe Compte \_ANT/ Modèle/Classe

**iii-Setters et Getters**

****