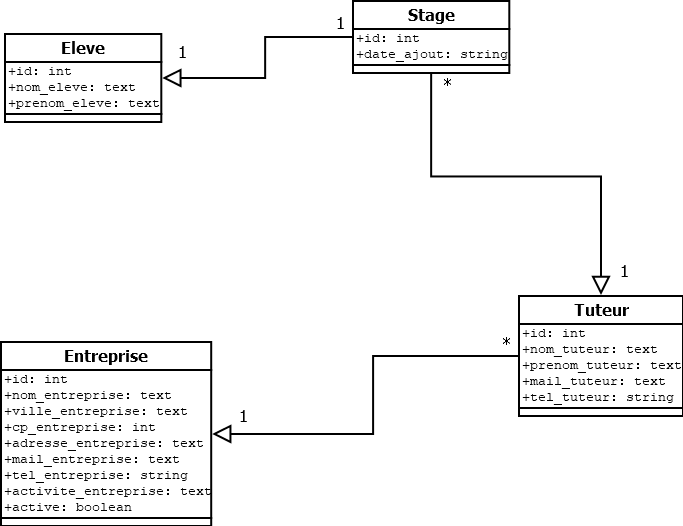
**Mission 3** : Portage d’une solution PHP vers Symfony

Compte rendu

**Sommaire**

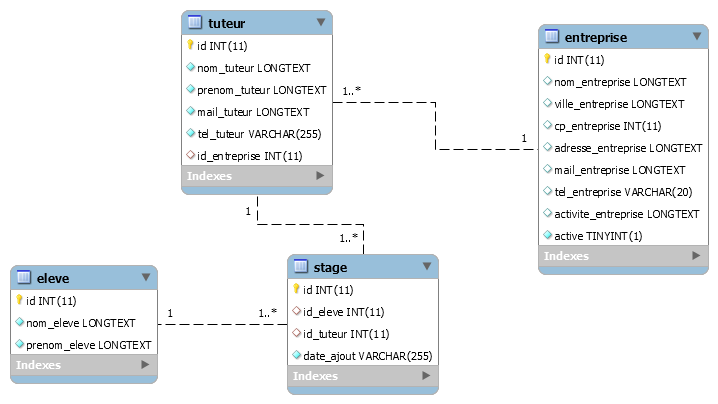
1. Rapport d’analyse
2. Diagramme UML
3. Schéma relationnel
4. Diagramme des cas d’utilisations
5. Rapport d’implémentation
6. Schéma organisationnel
7. Extraits de l’application web
8. **Rapport d’analyse**
9. **Présentation du diagramme UML**

****

Après avoir récupéré le projet que Salomé et moi (Angélique) ont réalisé, nous avons (Raphaël et moi) analysé leur base de données afin de sélectionner ce que nous allions utiliser pour la mission en Symfony. Ainsi, comme Symfony 4 ne prend pas encore en charge la gestion des sessions, nous n’avons pas gardé la table professeur, mais les entités Éleve, Entreprise, Tuteur et Stage ont été conservées. Nous expliquerons les relations entre les classes et le choix des types des attributs dans la partie suivante.

Après avoir réalisé un brouillon sur papier et effectué diverses recherches pour savoir et comprendre ce que nous réalisions, nous avons utilisé le logiciel **Dia** pour réaliser ce diagramme.

1. **Présentation du schéma relationnel**



Voici, ci-dessus, le schéma relationnel représentatif de notre base de données. Nous allons vous expliquer les liens entre les tables ainsi que le choix du type des champs.

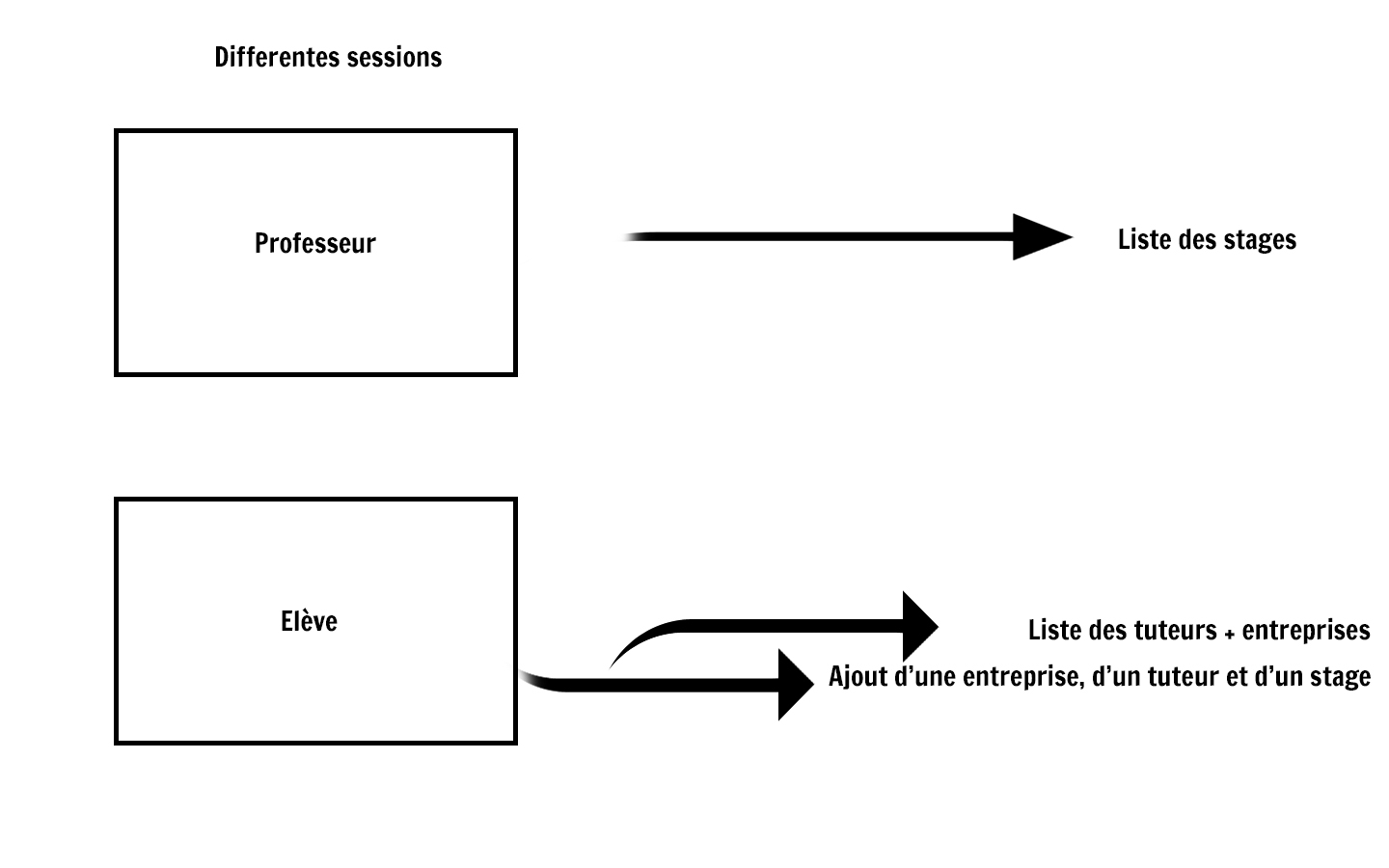
**Relations entre tables :**

* Une entreprise peut avoir un ou plusieurs tuteurs. Un tuteur appartient à une et une seule entreprise.
* Un élève n’a qu’un et un seul stage. Un stage ne concerne qu’un et un seul élève.
* Un tuteur peut être lié à un ou plusieurs stages. Un stage ne concerne qu’un et un seul tuteur.

**Choix du type de certains champs :**

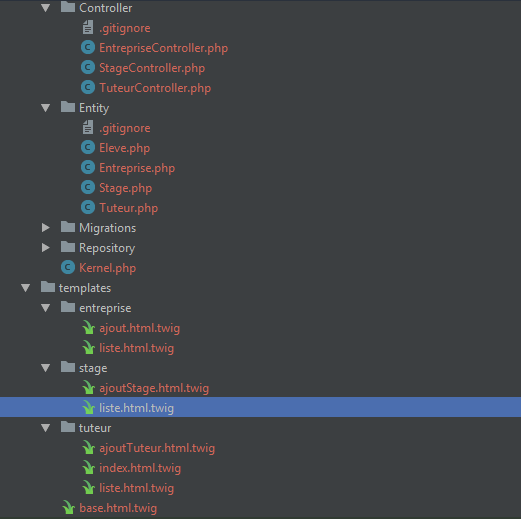
* Vous remarquerez que le champ **stage.date\_ajout** est de type « **String** » au lieu de « **datetime** ». Nous avions, au départ, commencé à manipuler la donnée en type « **datetime** ». Toutefois, ayant du mal à manipuler la donnée, et enchaînant erreur sur erreur, nous avons opté pour plus de faciliter en utilisant le type « **String** ».
* Les champs **tuteur.tel\_tuteur** & **entreprise.tel\_entreprise** sont de type « **String** ». Cela s’explique par le fait que, dans les formulaires d’ajout, lorsque l’on rentrait le numéro alors que les champs en question étaient de type « **int** », le chiffre « 0 » en début de numéro n’était pas conservé.

1. **Présentation du diagramme d’utilisation**



Bien que Symfony 4 ne prend pas en charge la gestion des sessions, nous avons réalisé le diagramme des cas d’utilisation conformément à ce qui nous était initialement demandé. Si aucune session ne peut être trouvée sur le site, la page concernant la consultation des stages existe bel et bien.

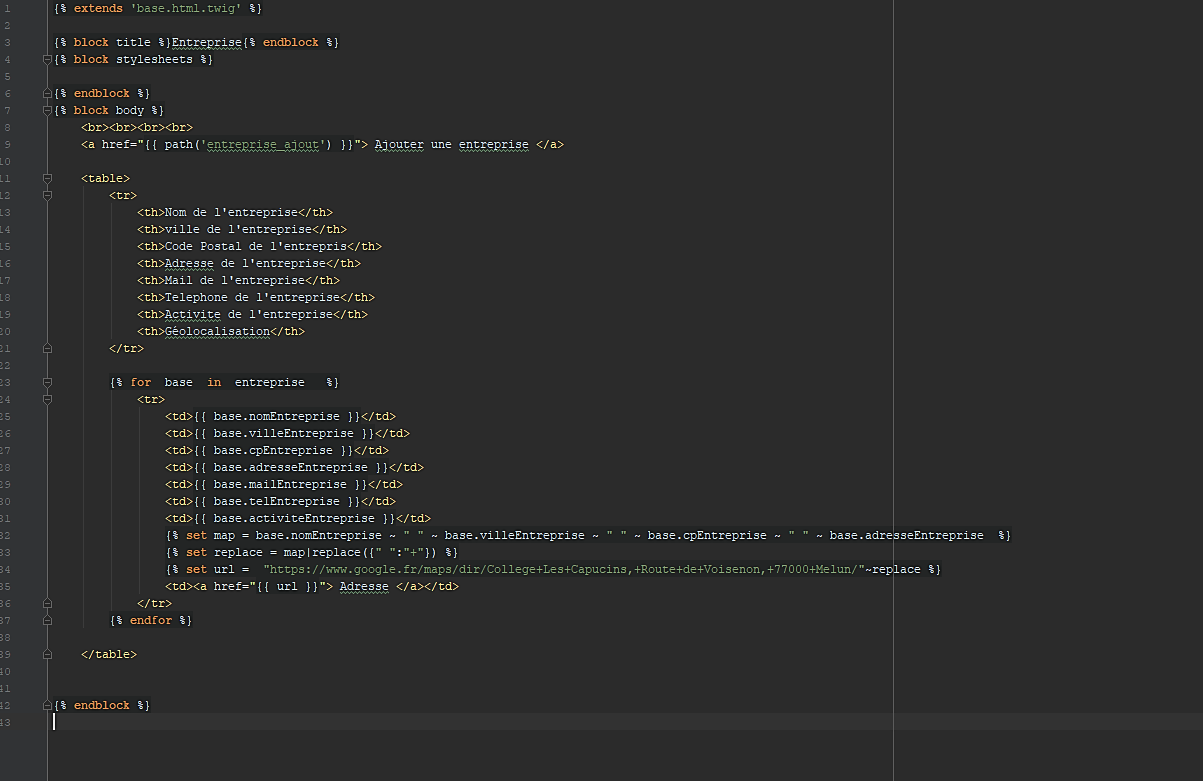
1. **Rapport d’implémentation**
2. **Schéma organisationnel de la structure du projet : l’arborescence des dossiers**

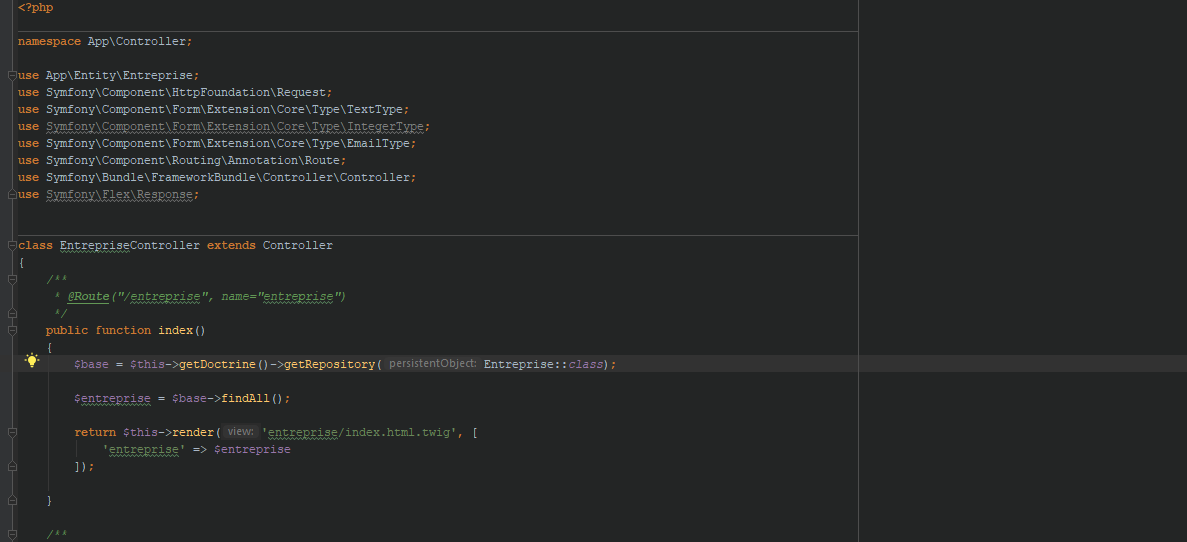
****

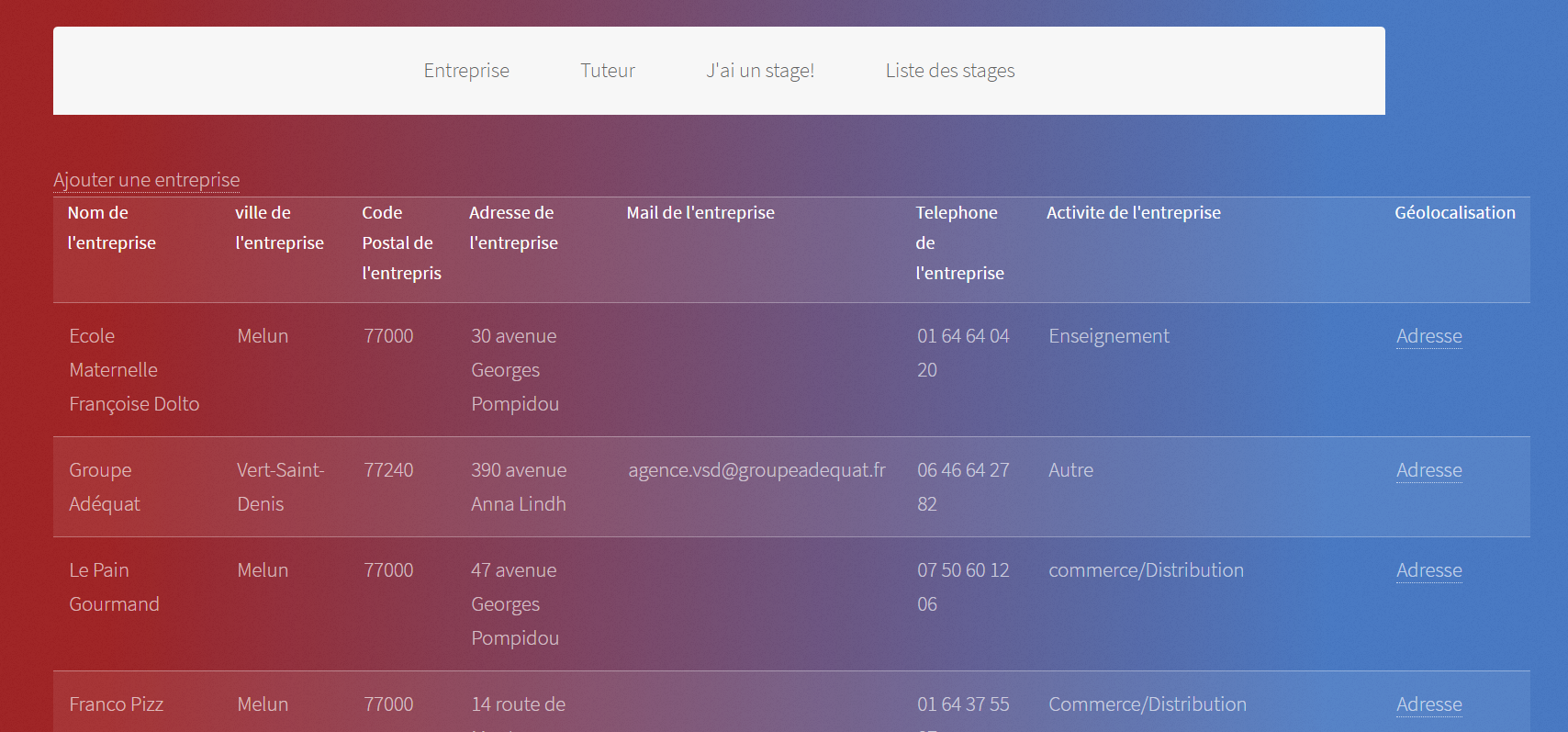
Voici l’arborescence de notre projet. Nous avons 4 entités qui nous permettent de structurer les tables dans la base de données. Les controllers présents permettront de manipuler ces entités et donc les tables liées à celles-ci. Il y a plusieurs vues, notamment vues nommées « liste » et « ajout », pour permettre d’afficher les tables et d’ajouter des données dans celles-ci.

1. **Divers extraits de codes et de vues**

**Ex1 :** La liste des entreprises

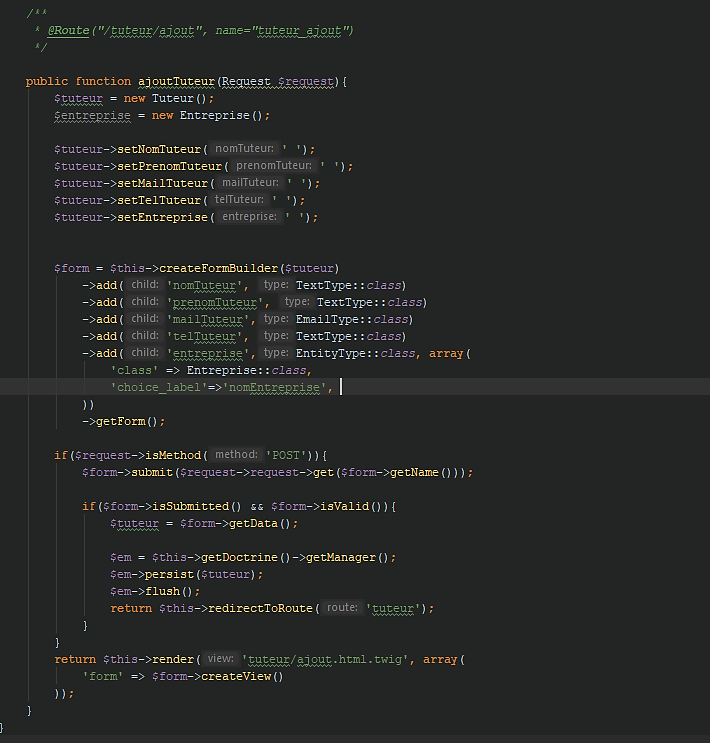
****

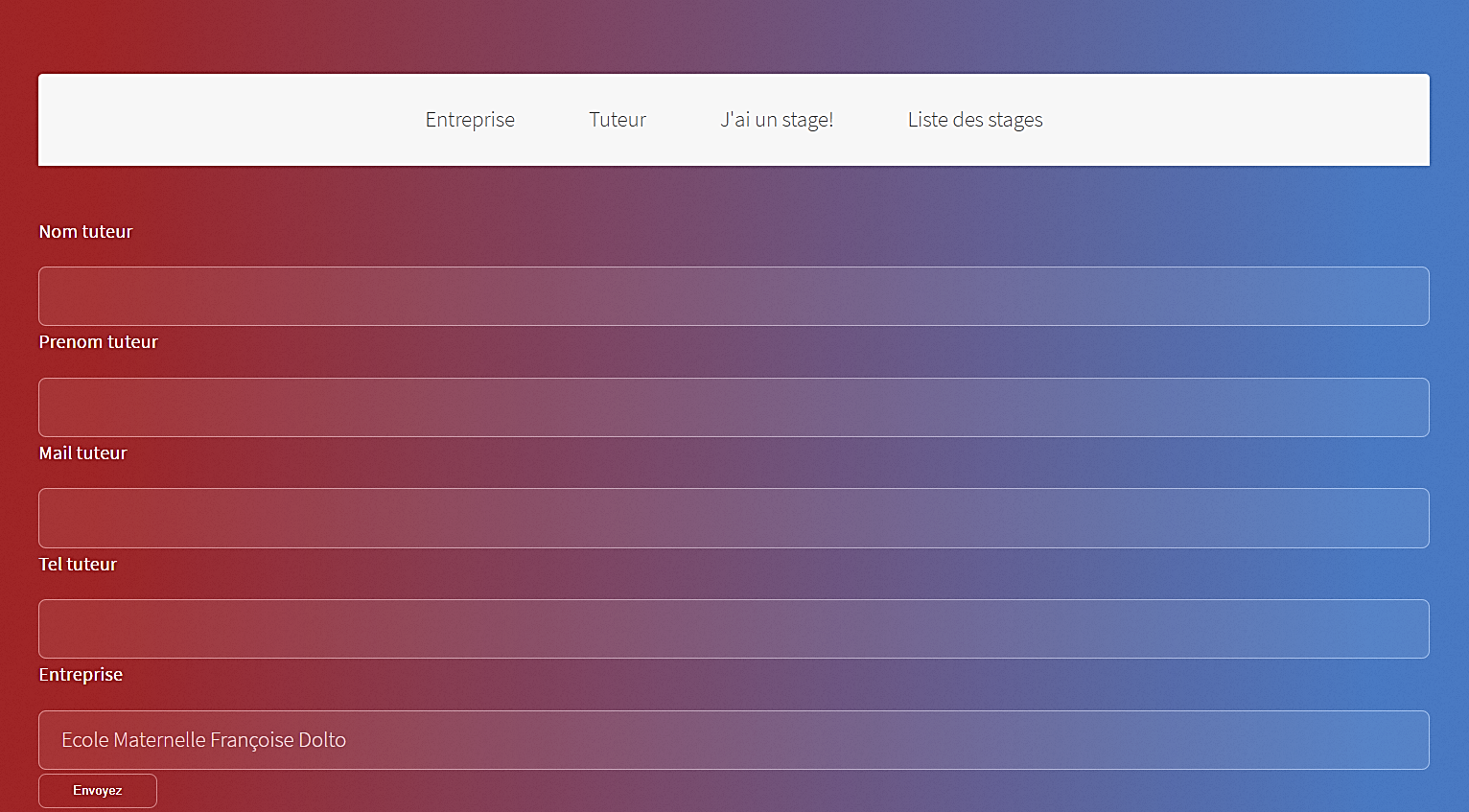
****

****

De haut en bas se trouvent le controller manipulant les attributs de l’entité Entreprise, qui envoie ces mêmes données à la vue, ce qui permetter d’afficher la liste des entreprises, et la vue par navigateur que cela rend.

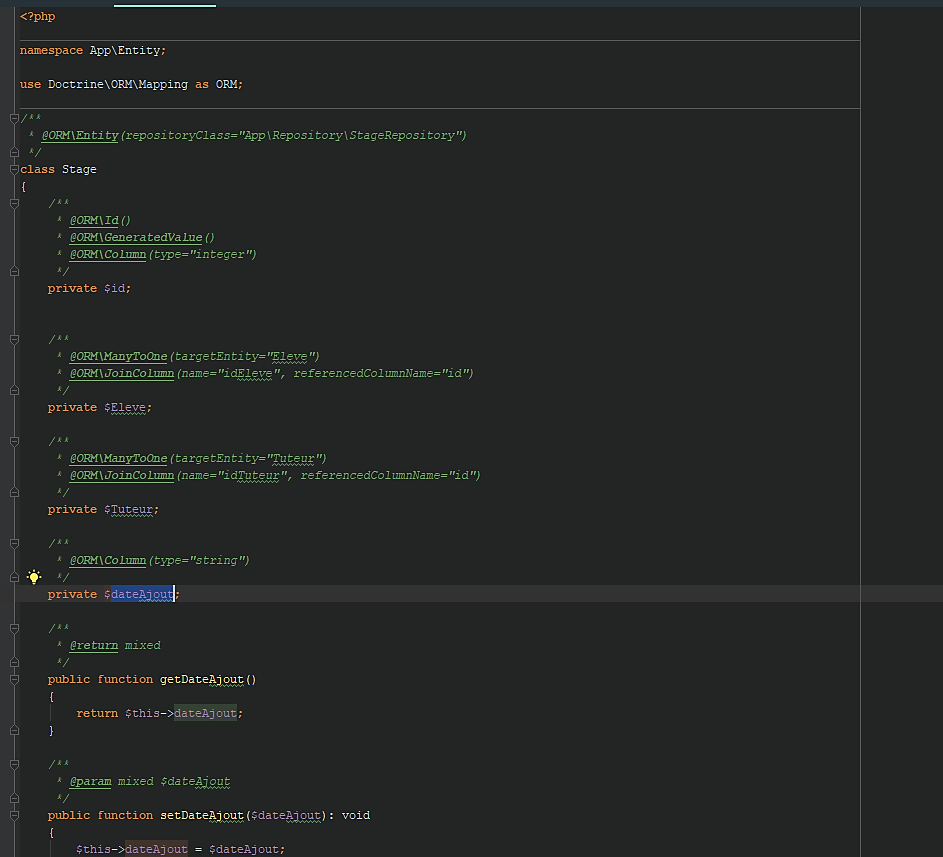
**Ex2** : L’ajout d’un tuteur



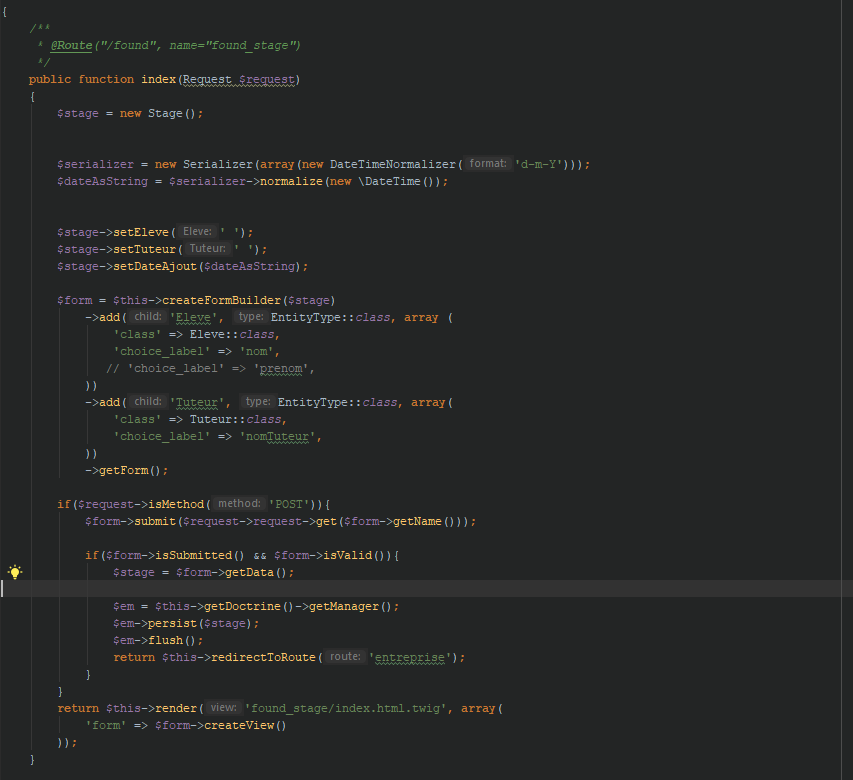


Extrait du controller « TuteurController », qui montre une fonction permettant l’ajout d’un tuteur dans la base. A la ligne où se situe le 5ème « add », on utilise le type « EntityType » pour faire comprendre que l’on va faire appel à une autre entité pour procéder à l’ajout.

**Ex3** : Entité « Stage », avec liaison ManyToOne



Voici le controller de l’entité Stage. Comme il est possible de le voir, il existe deux liaisons ManyToOne avec les entités Eleve et Tuteur, où l’on explicite à quel attribut le champ sera lié. Le champ « dateAjout » a été ajouté or liaison entre entités, conformément à ce qui était attendu de l’application.



Comme il est possible de le voir, alors que l’on initie trois setters, il n’y a que deux « add » au lieu de trois. En réalité, ce n’est pas l’utilisateur qui entrera de lui-même la date. Elle sera récupérée par le controller grâce à la méthode DateTime(). L’utilisation de Serializer indique que l’on va convertir unobjet au format donné en argument. Normalizer va ensuite passer l’objet renvoyé par la méthode DateTime() au format String : String car envoyé en argument à la méthode setDateAjout() qui attend une donnée de type String.