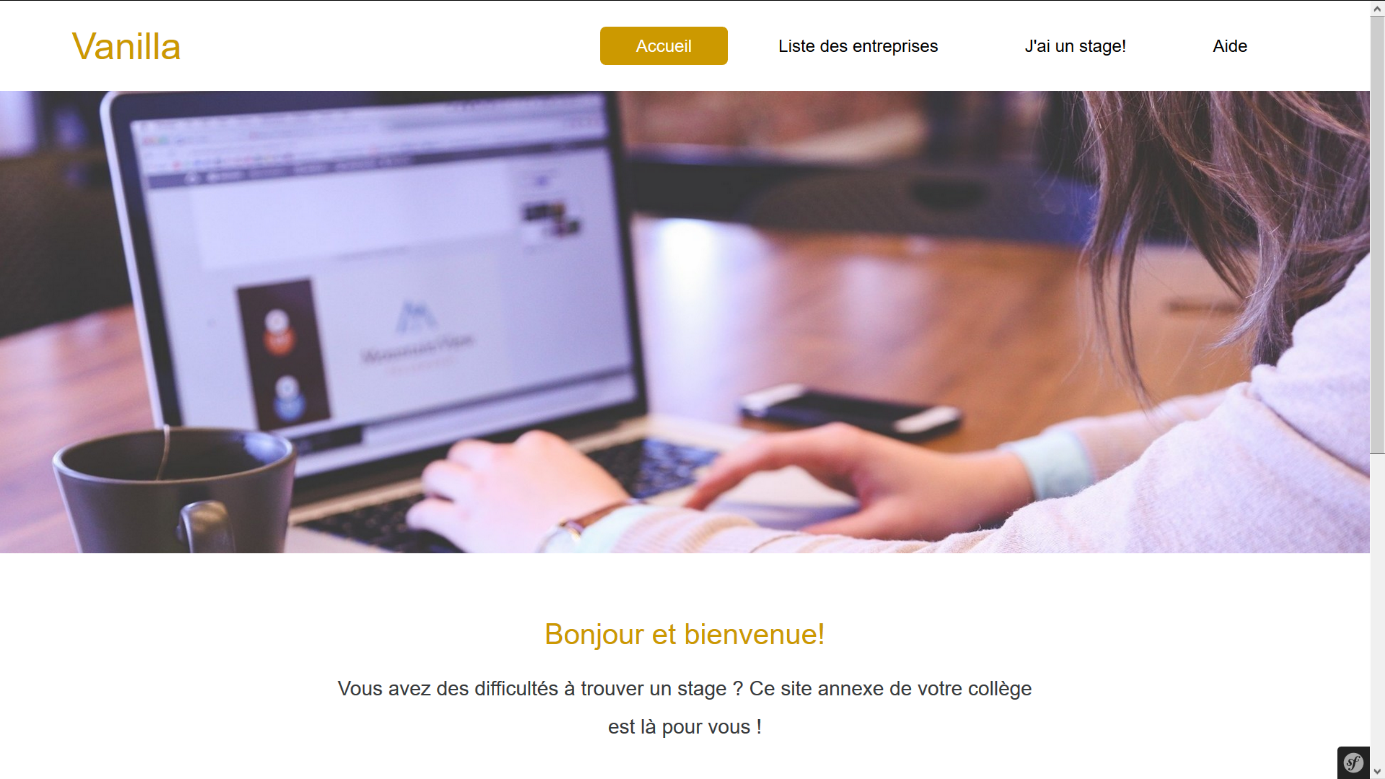
|  |
| --- |
| Documentation de l’application web : ‘Vanilla’ |



Projet professionnel -‘Vanilla’

Développer à partir d’un projet existant, refactoring avec la solution Symfony 4 et mySQL.

Mars 2018Crété​ ​Jonathan/Adelaide Donovan/Ball Philippe​ ​–​ **Lycée Léonard de Vinci**

**Cahier des charges**

Besoins essentiels :

* Reprendre un projet existant et le déporter avec le framework web Symfony 4.
* La gestion de l’authentification n’est pas nécessaire car le composant FOSUserBundle n’est pas disponible pour symfony 4.
* Réaliser une analyse complète avec un diagramme UML des entités du domaine, un schèma Relationnel et une diagramme USECASE complet
* L’application devra faciliter l’accès à la liste des entreprises locales qui accueillent, ou qui ont accueilli dans le passé, des élèves de 3eme.

Besoins additionnels :

* Concevoir un système de session à l’aide du composant ‘security’
* Accorder des droits administrateur a l’utilisateur « professeur »
* Intégration de Google Maps
* Ajout d’une messagerie interne permettant aux élèves d’être en contact avec leurs professeur durant le stage

**Description de l’application**

« Vanilla » est une application web destinée au collège les capucins, pour M.Fortin, à Melun.

Elle a pour principal objectif de faciliter l’accès à la liste des entreprises locales qui accueillent, ou qui ont accueilli dans le passé, des élèves de 3eme.

L’application comprend est adressé à deux types d’utilisateurs : Le professeur et l’éléve.

Cette application est un « refactoring » d’un application web déjà existante conçue en février 2018.

Afin d’assurer une maintenance évolutive, Vanilla a été développée avec le framework php

Symfony 4, réduisant le nombre de dépendances par rapport à Symfony 3 (au nombre de 3)

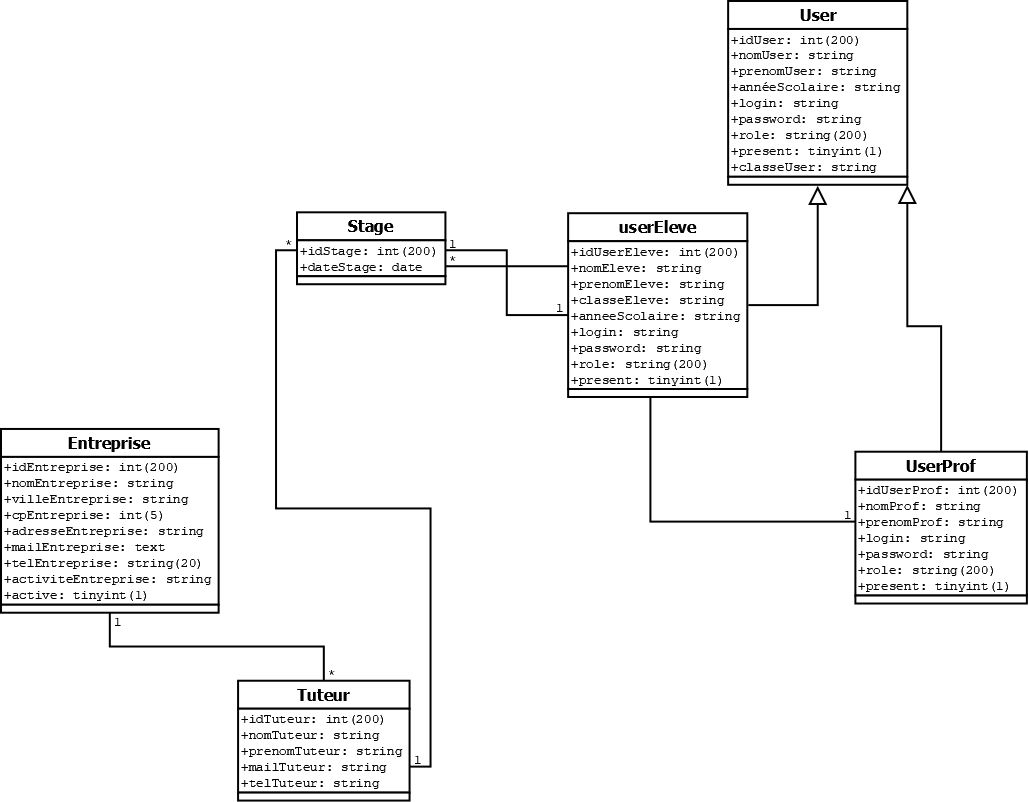
Le modèle MVC (Modèle\Vue\Contrôleur) à été utilisé afin de mieux structurer le code.

Celui-ci permet également une conception claire et efficace grâce à la séparation des données de la vue et du contrôleur, mais aussi d’un gain de temps pour une potentielle team de développeurs qui reprendraient le projet dans le futur.

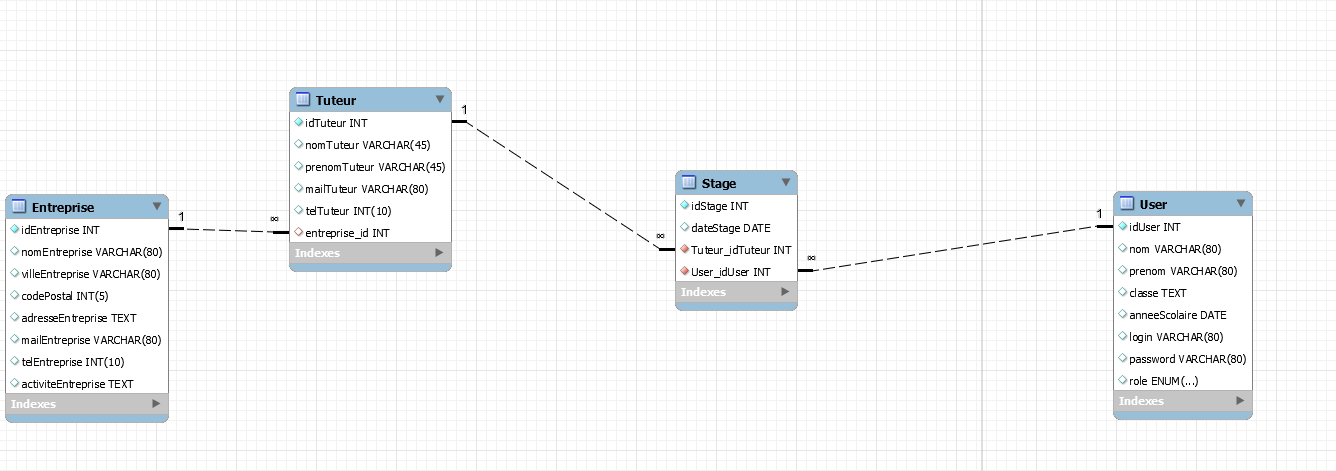
**Choix\contraintes techniques :**

* Framework coté serveur : Symfony 4 (Php)
* Framework CSS (design) coté client : Bootstrap 4
* Utilisation du logiciel Laragon pour administrer une base de données mySql.
* Template responsive issu du site freecss.com
* Respect de la norme W3C
* Apprentissage du langage de template twig

**Diagramme UML d’analyse des entités du domaine**



**Schéma Relationnel**

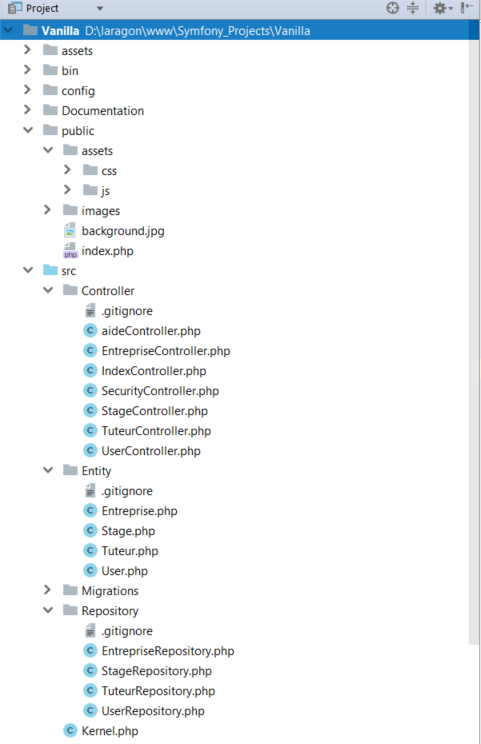


**Diagramme USECASE**



Les cas d’utilisations matérialisés par la couleur bleu ont été traités.

**Arborescence des dossiers**



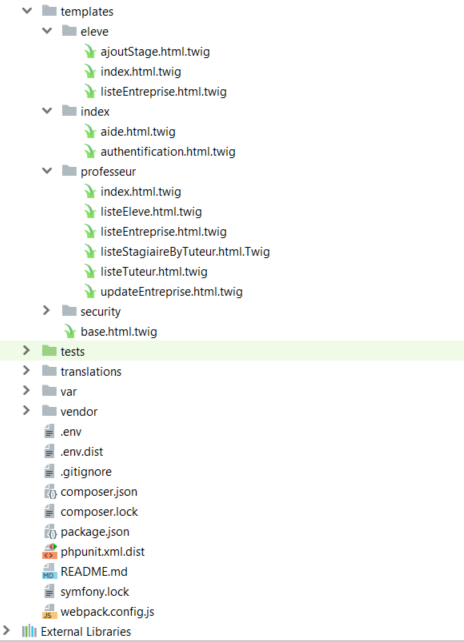
Répertoire contenant les différents fichiers de migrations BDD

Répertoire contenant les différents contrôleurs

Répertoire contenant les différentes entités

Composants web en accès public (css/js/images)

Fichiers de configuration



.gitignore : Fichier énumérant les fichiers qui doivent être ignorés par Git lors d’un commit

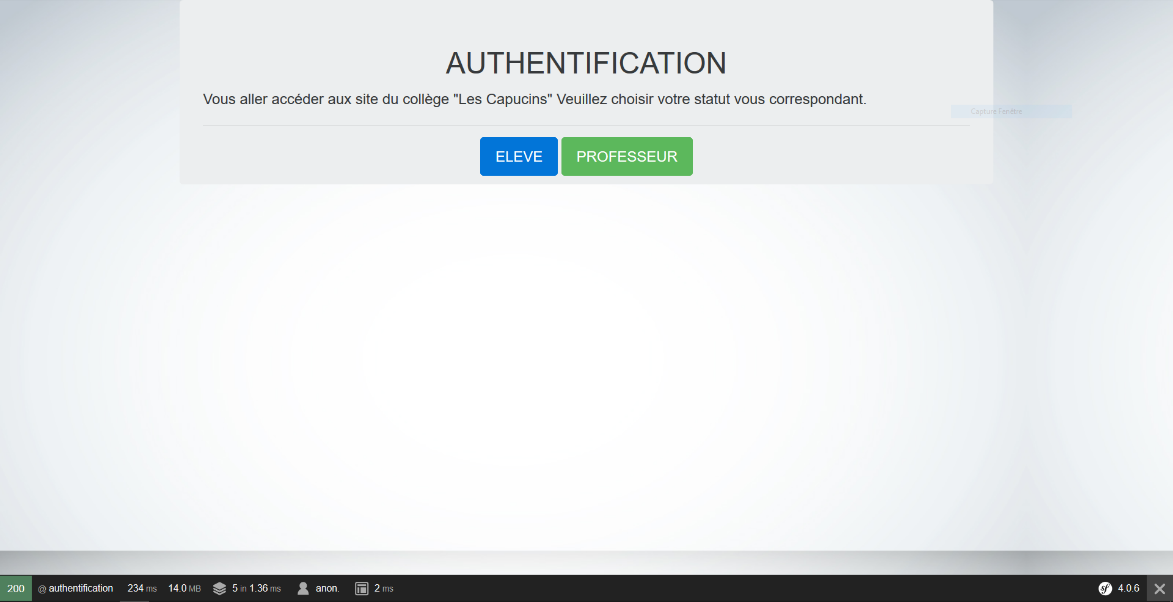
/vendor : Répertoire contenant les composants tiers

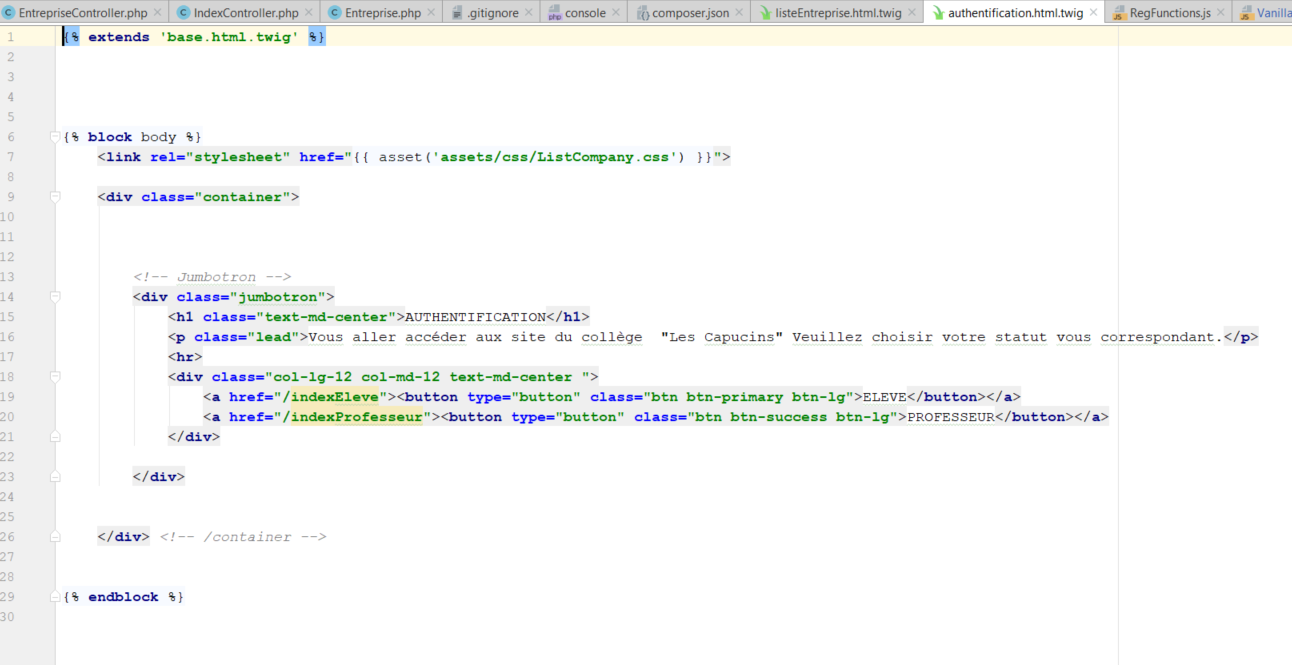
Répertoire contenant les différentes vues :

* templates/eleve : Vues liées à « l’espace élève »
* templates/index : Vues dites communes tant aux élèves que aux professeurs
* templates/professeur : Vues associés à l’espace professeur

**Extraits d’implémentations**

Vue d’authentification :

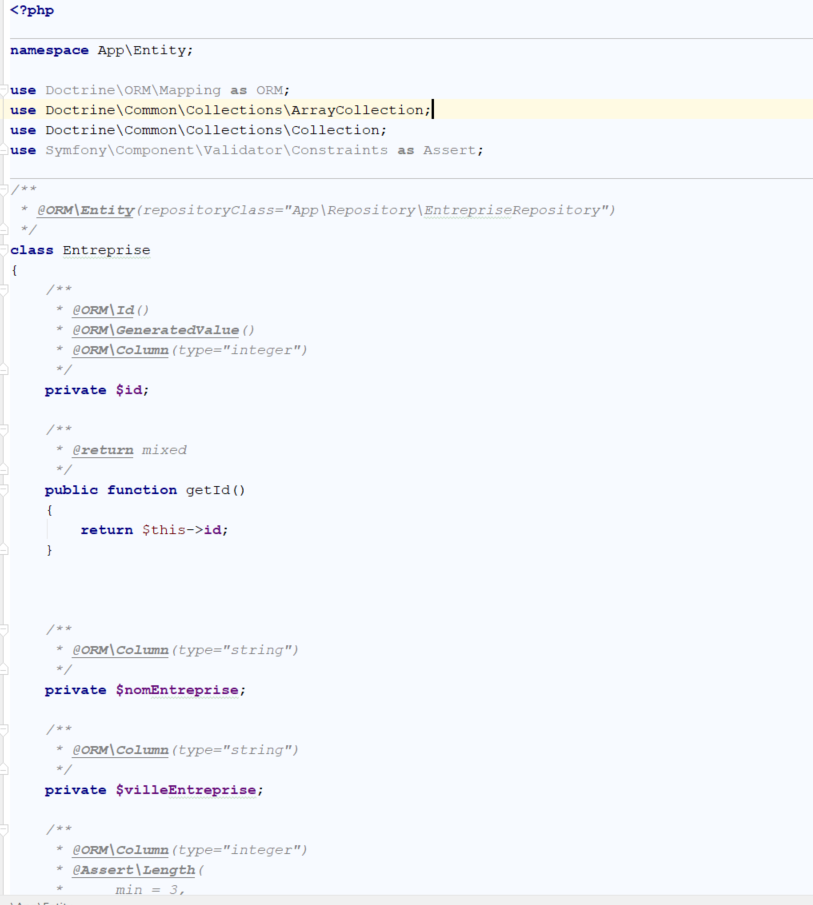




Deux boutons liens.

Cas d’utilisation : Lecture des entreprises d’accueil pour l’élève :

1 -> Extrait de l’entité Entreprise :



2-> Contrôleur EntrepriseController.php  :



Appel de la méthode héritée (render) en lui passant le nom de la vue ‘listeEntreprise.html.twig’

3-> Extrait de la vue ‘*listeEntreprise.html.twig’* :

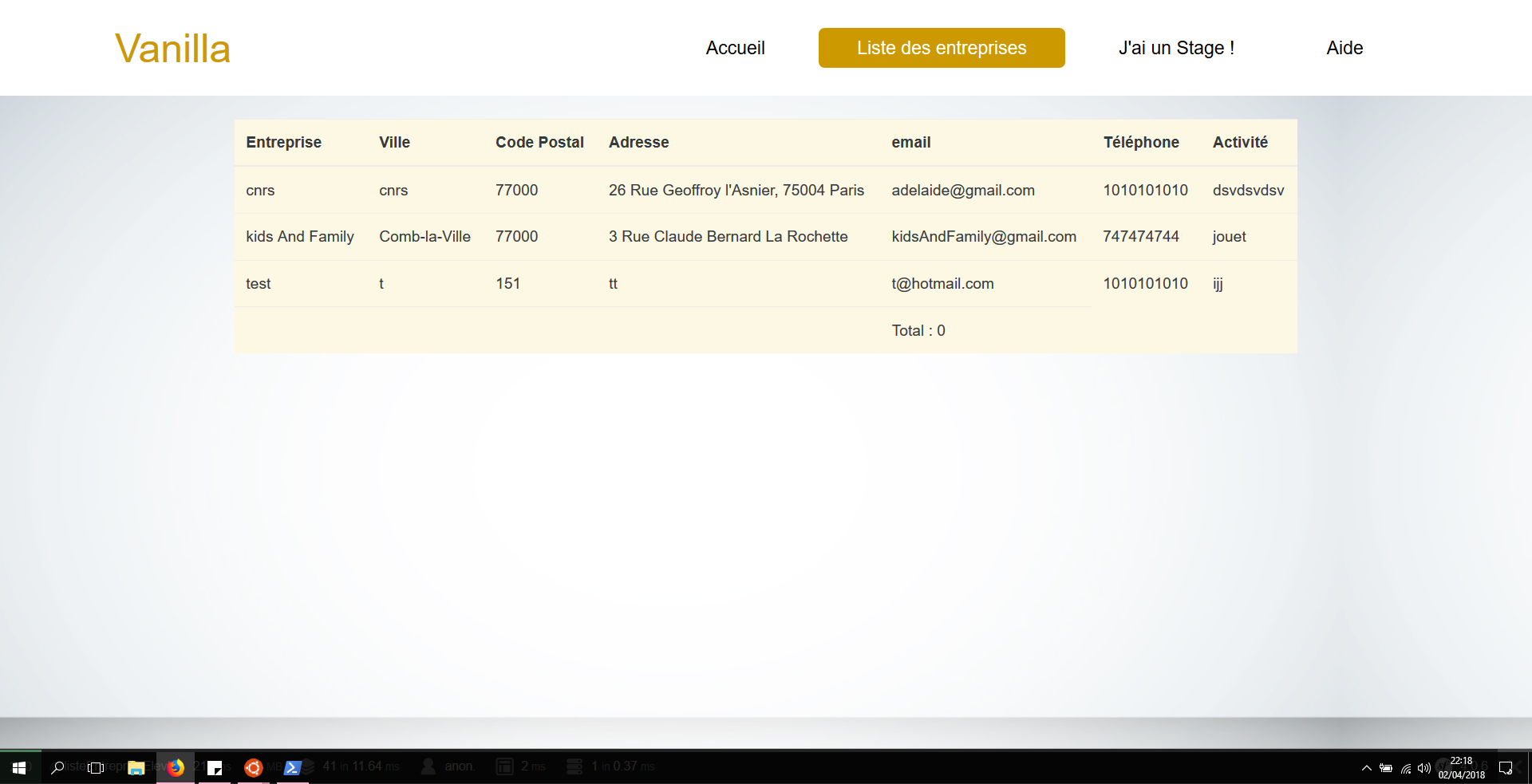


On remarque que les classes propres à Bootstrap 4 sont utilisées :

« Container » : Le container est un élément de disposition le plus fondamental et est requis afin d’utiliser le système de grille de Bootstrap v4.0.0

« for i in listeEntreprises » : Boucle permettant l’affichage des lignes du tableau.

Visualisation :

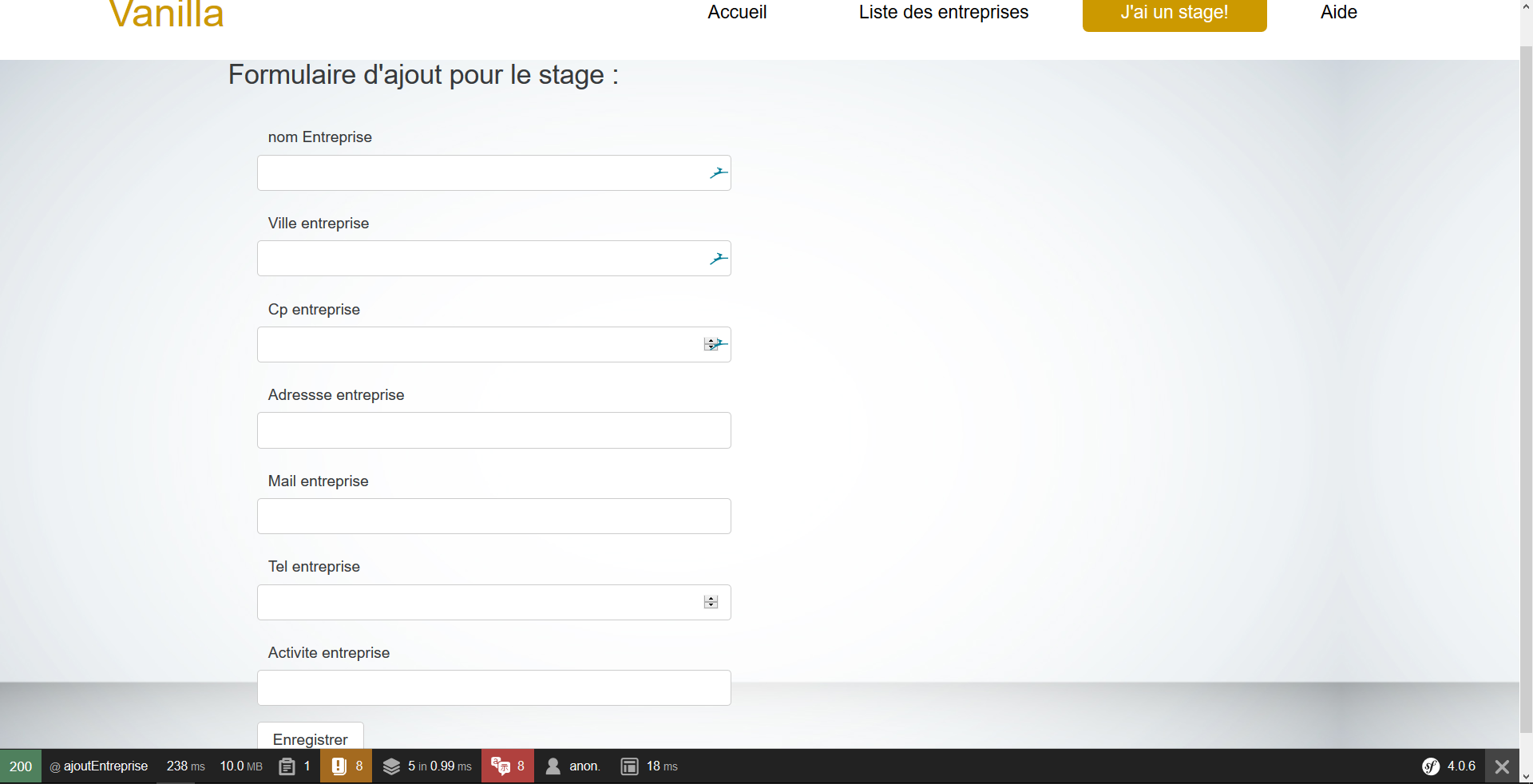


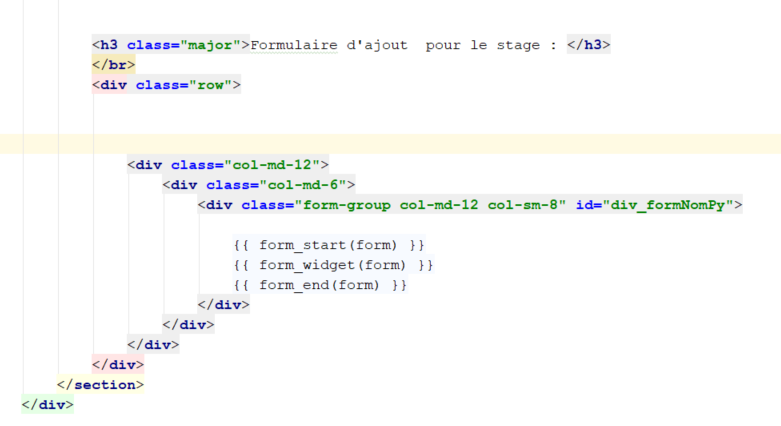
Cas d’utilisation : Ajout d’un stage

Extrait entité Stage :



Extrait du formulaire de déclaration d’une entreprise de stage élève :





Formulaire explications :

Dans sa version courante, un formulaire est lié à un objet métier (une classe Entity).

De plus, il intègre les propriétés importantes suivantes :

Gestion des requêtes HTTP : prise en charge du traitement des demandes et des téléchargements de fichiers;

Protection CSRF: Prise en charge de la protection contre les attaques CSRF (Cross-Site-Request-Forgery);

Templating: Intégration d’une couche de présentation qui permet de réutiliser des fragments HTML lors du rendu d’un formulaire;

Traduction: Prise en charge de la traduction des messages d’erreur, des étiquettes de champs et d’autres chaînes;

Validation: Intégration d’une bibliothèque de validation pour générer des messages d’erreur pour les données soumises.

Extrait de *StageController.php* :



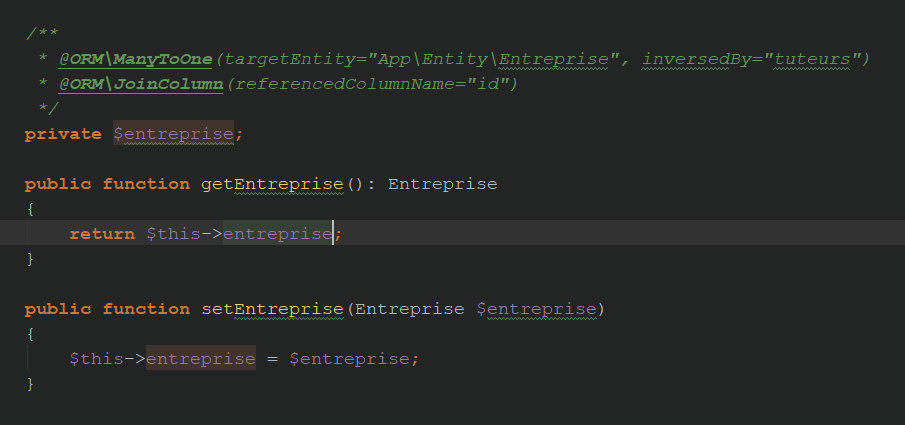
Explications + précises :

Extrait de *indexController.php* :



Les méthodes d’action suivantes substituent un système de session afin de rediriger l’utilisateur selon la route choisi, soit indexEleve pour les vues associées à élève et indexProfesseur pour les vue associée à professeur.

A partir du diagramme UML, on voit qu’une Entreprise peut avoir plusieurs tuteurs mais un tuteur ne peut être associée qu’a une seule entreprise. Cette relation se décrit Comme suit « Many To One » que l’on indique à Doctrine.



Cette relation pour fonctionner est fait dans les 2 sens cet a dire « ManyToOne » pour Tuteur et inversement « OneToMany » pour entreprise,

L’attribut « inversedBy » désigne l’attribut d’instance de l’entité inverse de la relation soit l’entité Entreprise.

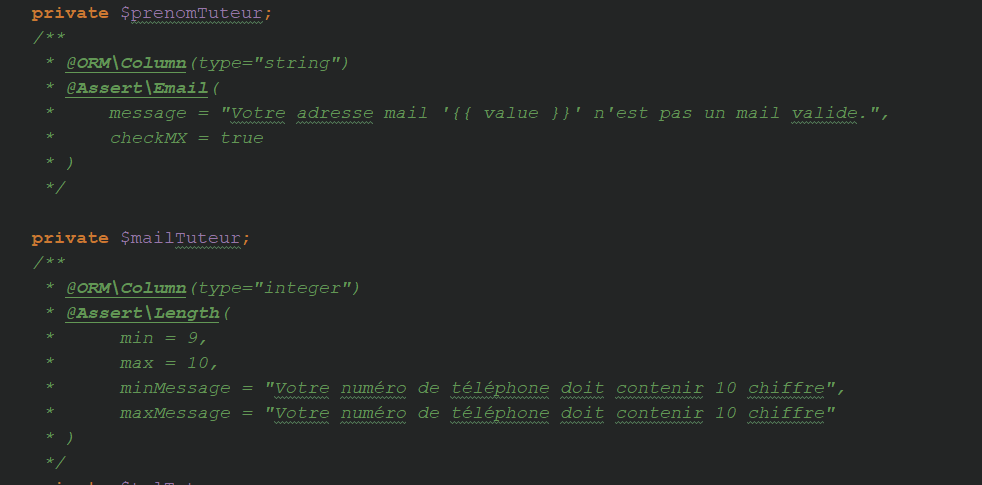
Voilà un exemple de méthode d’action désignant une opération de base (READ).



La méthode d’action listeTuteur prend en paramètre un identifiant en l’occurrence celle de l’entreprise, ce paramètre permet a la méthode de récupérer la liste de tuteur par entreprise

D’où le « findBy(array(‘entreprise’ -> $id) » et renvoie cette liste sous forme d’un tableau appelée listeTuteurByEntreprise.

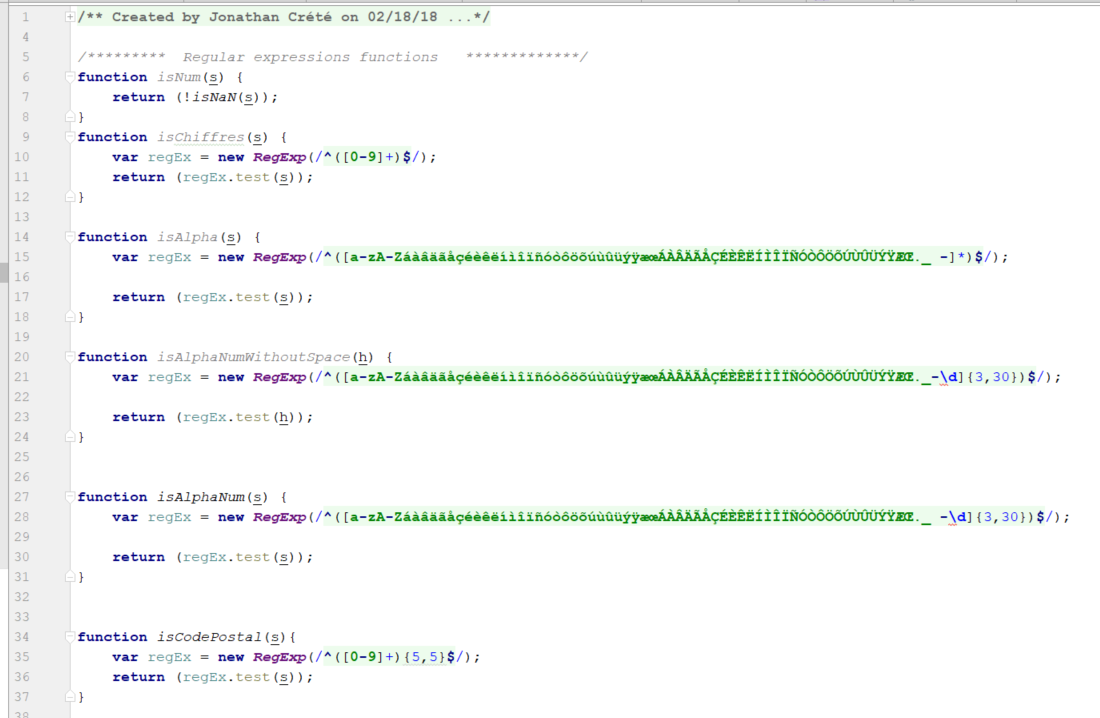
La validation est une tâche très courante dans les applications Web. Les données saisies dans les formulaires doivent être validées. Les données doivent également être validées avant d'être écrites dans une base de données ou transmises à un service Web.



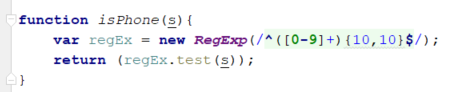
Il y a plusieurs type de contrainte afin d’éviter une erreur d’insertion, exemple « @Assert\Email » permet une restriction au niveau de la syntaxe de l’insertion « @, . » etc..

**Fonctions javaScript**

Extraits du fichier RegFunctions.js :



Exemple :



1

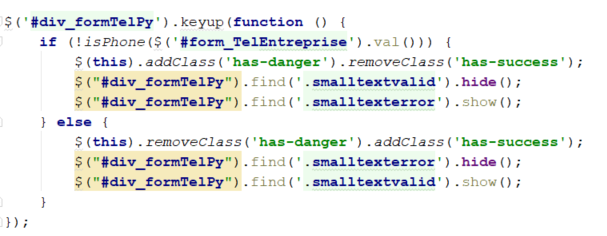
2

Manipulation de la chaine de caractère du champ « téléphone »

1 -> Instanciation de l’objet RegExp, seul les chiffres de 0 à 9 sont autorisés et la chaine de caractère doit avoir une taille minimum de 10.

2 –> La méthode test() renvoie true si le test est réussi ou false si le test échoue

Extrait du fichier Vanilla.js (Utilisation librairie jQuery)



3

3 🡪 (jQuery) Si la valeur du champ form\_TelEntreprise est différente du format attendue alors on ajoute une classe ‘has-danger’ à l’élément parent de form\_TelEntreprise ( div\_formTelPy) pour signifier une erreur de saisie et on affiche du texte en dessous du champ.

Sinon, on ajoute une classe de succès signifiant la bonne saisie de l’utilisateur.

**Suivi & Problèmes rencontrés**

**Suivi :**

- Création des méthodes d’action redirigeant vers des vue différent (substituions d’authentification)

- UPDATE modification d’entreprise pour les professeurs

- ADD ajout d’une entreprise, d’un tuteur, d’un stage

- Modification de structure du projet pour les besoins futurs

- Création des liaisons entre entité

- READ liste élève pour les professeurs

- READ liste tuteurs par entreprise pour les professeurs

- READ liste des élèves ayant un stage selon le tuteur

- Lien vers google map selon l'adresse de l'entreprise

- Intégration des fonctionnalité Bootstrap pour tous les formulaires

- Validateur sur les attributs par entité (pour les formulaire)

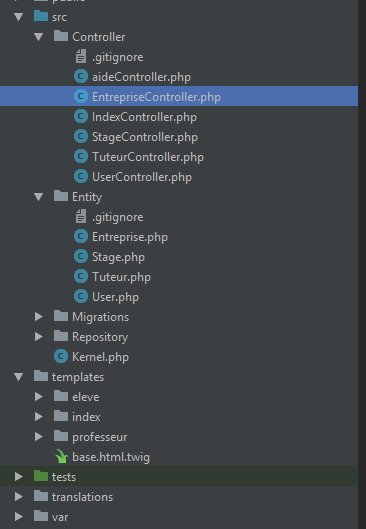
- Changement du background du Template

- Création de l'entité et Controller user

- Création des différentes vues associé aux méthodes d’action précédemment listé

**Problèmes rencontrés et éventuelle correction :**

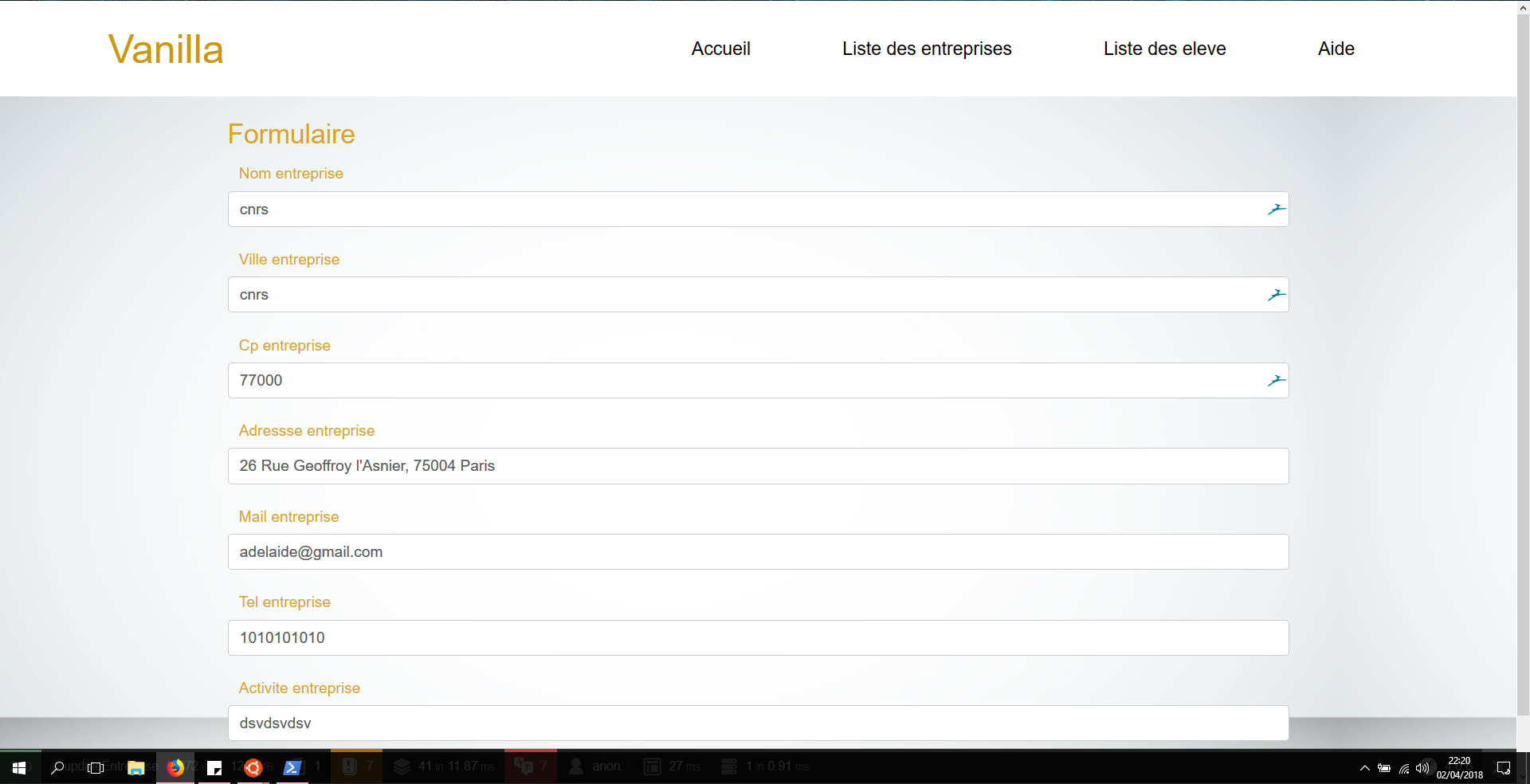
* Problème d’implémentation du composant security (01/04/18)
* Un correctif a été sur la structure du projet (les différents Controller ainsi que les entité) afin de correspondre aux améliorations futures du projet.
* Une entité pour un Controller contenant des méthodes d’action qui renvoie sur plusieurs vue twig.
* Un dossier pour les vues concernant les élevé, un dossier pour les vues concernant les professeurs, un dossier pour les vue général élève et professeur (authentification, aide).
  + Ces dossiers seront remplacés après que nous aurons intégré un système de session dans Symfony.



**Hébergement**

Ajout future de l’application web sur le service d’hébergement Hostinger.

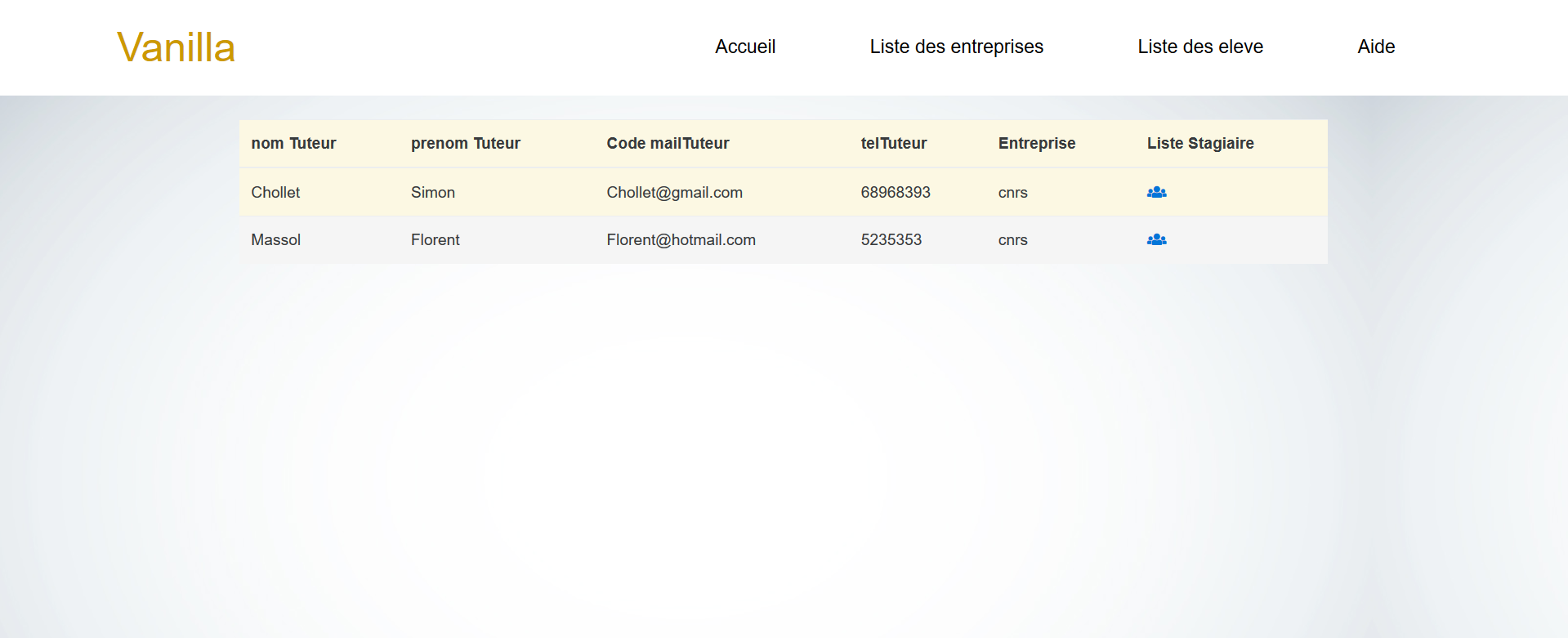
**Images**



*Formulaire update d'une entreprise (professeur)*



*liste eleve ayant un stage trié par tuteur (professeur)*



*liste tuteur par entreprise (professeur)*