­­Sommaire

By

esprit teal

by

Carmelo ROBERTO

élève de la wilde code school

Titre Professionnel

Développeur Web et Web mobile

Founder: Timothée Couchoud

E-Mail: timothee@esprit-teal.com

1. Introduction 4

1.1. français 4

1.2. english 5

2. Projet 6

2.1. L’Entreprise 6

2.2. TealFinder 6

3. Analyse du projet 7

3.1. Fonctionelles 7

3.1.1. Fonctionnalités du front : 7

3.1.2. Fonctionnalités du backoffice : 7

3.2. Techniques 9

3.2.1. La base de données : 10

3.2.2. Le backoffice : 10

3.2.3. La sécurité : 11

3.2.4. Le sondage : 11

3.2.5. L’importation du fichier CSV : 11

3.2.6. L’envoi d’un email 12

3.2.7. La traduction 12

3.3. Competénces du réferentiel 12

4. Eléments les plus significatifs 14

4.1. Les fixtures 14

4.1.1. La création des fixtures 14

4.1.2. La fixture : translation 15

4.2. L’importation du fichiers CSV 17

4.2.1. Le type du fichier 17

4.2.2. Les en-têtes 17

4.2.3. Les en-têtes et ses noms 17

4.2.4. Les en-têtes et ses valeurs 18

4.2.5. Les en-têtes et leurs positions 18

4.2.6. Les en-têtes et le nombre de tous les champs 19

4.2.7. Les champs vides 19

4.2.8. Le type de la valeur 19

4.2.9. L’adresse email 20

4.2.10. Résumé 21

4.3. Le Sondage 24

4.3.1. Le tableau : subject 25

4.3.2. Le SondageController et ses Services 27

4.3.3. Le Template Twig 28

4.4. L’envoi d’un e mail 29

4.4.1. Autowiring / Symfony Mailer 30

4.4.2. send() 31

4.5. La Sécurité 32

# Introduction

## français

Après avoir servi dans l’armée pendant 16 ans, du côté allemand et français, j’ai décidé de me réorienter vers l’informatique. J’ai pris cette décision parce-que j’aime passer mon temps libre à faire de l’informatique en général. De plus, c’était le bon moment pour prendre cette décision, que je portais depuis longtemps, parce-que ma motivation pour rester dans l’armée diminuait. A ce moment-là, je ne savais pas dans quel domaine de l’informatique je voudrais commencer. Le Pôle Emploi m’a donné l’opportunité de mieux connaître le domaine de l’informatique. J’ai donc pu effectuer un stage en tant que Technicien d’assistance en informatique chez ISTATIS, ce qui était très intéressant. Cependant, comme j’ai eu des entretiens avec plusieurs écoles du développement web, j’ai décidé de commencer en tant que développeur web pour me lancer dans le monde de l’informatique.

Les cinq mois que j’ai pu passer à la Wild Code School ont été très instructifs pour moi et ont pu élargir considérablement mes connaissances en programmation dans le domaine du développement Web. Même si cinq mois, ce n’est pas beaucoup, l’intensité des courses et la pression de plusieurs missions à la fois étaient les bons moyens pour moi d’apprendre.

J’ai investi beaucoup d’énergie dans cette formation, ce qui a par la suite eu un effet positif dans les entretiens car j’ai pu trouver un stage pendant ma formation qui a commencé juste après cela. J’ai pu approfondir mes connaissances en tant que stagiaire chez Actimage pendant quatre mois et faire partie d’une super équipe jeune et dynamique. Et où je suis toujours stagiaire.

## english

After serving in the army for 16 years, on the Germany and French, I decided to reorient myself in the IT. I made this decision because I like to spend my free time doing IT in general. Also, it was the right time to make this decision, which I had been carrying for a long time, because my motivation to stay in the army was shrinking. At this time, I didn’t know which IT area to choose. The job center gave me the opportunity to better understand what it is to work in IT. So, I was able to do an internship as an IT-Support Technician at ISTATIS, which was very interesting. However, as I have had interviews with several web development schools, I decided to start as a web developer to get started in the world of IT.

The five months I spent at the Wild Code School were very instructive for me and could greatly expand my programming knowledge for Web development. Even if five months is not much, the intensity of the course and the pressure of several missions at the same time were the right thing for me to learn.

I invested a lot of energy in this training, which later had a positive effect in the interviews because I was able to find an internship during my formation which started right after that. I was able to deepen my knowledge as an intern at Actimage for four months and be part of a great young and dynamic team. Where I am still an intern.

# Projet

## L’Entreprise

EspritTeal est un cabinet de conseil spécialisé dans l'accompagnement de dirigeants et d'entreprises dans leur processus de transformation culturelle et structurelle. Crée en 2019 et Basé à Strasbourg qui est composée de quatre membres dont le CTO est Timothée Couchoud. L’idéologie de l’entreprise trouve son origine dans le livre de « Reinventing Organizations » est divisée en trois parties qui sont :

* Culture Teal
* TealFinder
* MyInitiatives

Et où TealFinder nous a été confié en tant que projet.

## TealFinder

Dans le cadre de la formation développeur web et web mobile et du Projet 3, à la Wilde Code School, mon groupe composé des membres Emmanuelle Bouno, Cyrille Stammler et moi-même, a eu la chance de s’occuper du projet TealFinder. Le but de ce projet était d’évaluer une organisation (qui peut être bilingue) à l’aide d’un questionnaire interactif pour les utilisateurs qui, une fois répondu avec un de 5 réponses aux 20 questions estimeront la culture de l’organisation. Sa culture peut avoir un de cinq caractères :

* Evolutionnaire
* Pluraliste
* Orienté réussite
* Conformiste
* Impulsif

La signification de ces caractères se retrouve sur la première page. Qui sert de guide et d’explication de l’objectif de ce questionnaire. Le résultat des réponses est représenté par un graphique avec la quelles il est possibles de comparer ses réponses avec celles de l’ensemble de son organisation. Les organisations entre elles ne peuvent pas être comparer. La gestion des questions, des réponses, des langues des organisations et des utilisateurs doit être située dans un espace séparé. Cet espace sépare appeler backoffice à son tour est divisé en trois zones afin que les utilisateurs, les organisations et les administrateurs aient chacun leur propre espace. Les administrateurs ont tous les droits. Les organisations ont la possibilité d’inviter des personnes et donnent aux personnes la possibilité d’inviter d’autres personnes. Ce qui peut être très utile lorsqu’une organisation est plus grand et qu’il y a plusieurs managers. Les utilisateurs, à leur tour, n’ont que la possibilité de changer leur mot de passe.

# Analyse du projet

Le projet est une nouvelle création. Les contenus comme les images, les polices, les couleurs et le style du ce projet ont déjà été créés par le groupe Esprit Teal qui a également fourni un Figma. Ce qui a rendu très clair l’aspect visuel du projet. Il nous a donc suffi de remplir le côté programmation du projet. Ce qui est expliqué ici plus en détail.

## Fonctionelles

Afin d’analyser correctement le projet, la fonctionnalité du projet a été discutée et a donné les résultats suivants.

### Fonctionnalités du front :

* + La connexion
    - L’utilisateur doit se connecter pour avoir accès au site.
  + Les rôles d’utilisateurs
    - Selon le rôle d’utilisateur, il a accès à différentes parties du site.

* + Les questions
    - Tous les utilisateurs ayant accès au site ont la possibilité de répondre aux questions qui s’ajoutent au résultat global du groupe auquel il appartient.
    - L’utilisateur peut à nouveau répondre aux questions déjà répondues.
  + Les résultats
    - Affiche uniquement les questions auxquelles il a été répondu.
    - Peut toujours être comparé au groupe auquel on appartient.
  + Le menu
    - Selon le rôle d’utilisateur, le menu sera ajusté.
    - Selon la réponse d’utilisateur, le menu sera ajusté.

### Fonctionnalités du backoffice :

* + Gestion du profil
    - L’utilisateur a la possibilité de changer son mot de passe.
    - Accessible pour tous les rôles.
  + Gestion des utilisateurs (groupe)
    - Répertoire tous les utilisateurs inviter.
    - Donner la possibilité d’inviter un personne grâce à un formulaire.
    - Donner la possibilité d’inviter plusieurs personne grâce à l’import d’une fichier csv.
    - Donner la possibilité de réinvite une personne.
    - Donner la possibilité de changer le nom d’une personne.
    - Donner la possibilité de changer le prénom d’une personne.
    - Donner la possibilité de changer le statut d’une personne.
    - Donner la possibilité de supprimer une personne.
    - Accessible que par les administrateurs de l’organisation.
  + Gestion des entreprises
    - Répertoire toutes les organisations inviter.
    - Donner la possibilité d’inviter une organisation grâce à un formulaire.
    - Donner la possibilité de changer le nom de l’organisation.
    - Donner la possibilité de changer la langue de l’organisation.
    - Donner la possibilité de changer l’image de l’organisation.
    - Donner la possibilité de supprimer une organisation.
    - Accessible que par les administrateurs du site.
  + Gestion des utilisateurs (site)
    - Répertoire tous les utilisateurs inviter.
    - Donner la possibilité de réinvite une personne.
    - Donner la possibilité de changer le nom d’une personne.
    - Donner la possibilité de changer le prénom d’une personne.
    - Donner la possibilité de changer le statut d’une personne.
    - Donner la possibilité de supprimer une personne.
    - Accessible que par les administrateurs du site.
  + Gestion des questions
    - Répertoire toutes les questions.
    - Donner la possibilité d’ajouter une question.
    - Donner la possibilité de changer le titre d’une question.
    - Donner la possibilité de changer la question d’une question.
    - Donner la possibilité de changer les réponses d’une question.
    - Donner la possibilité de changer la position d’une question.
    - Donner la possibilité de changer l’icône d’une question.
    - Donner la possibilité de changer la langue d’une question.
    - Donner la possibilité de supprimer une question.
    - Accessible que par les administrateurs du site.
  + Gestion des icônes
    - Répertoire toutes les icônes.
    - Donner la possibilité d’ajouter une icône.
    - Donner la possibilité de changer le nom d’une icône.
    - Donner la possibilité de changer l’icône même.
    - Donner la possibilité de supprimer une icône.
    - Accessible que par les administrateurs du site.
  + Gestion des traductions
    - Répertoire toutes les traductions.
    - Donner la possibilité d’ajouter une traduction.
    - Donner la possibilité d’ajouter un mot-clé.
    - Donner la possibilité d’ajouter une langue.
    - Donner la possibilité de changer le texte de la traduction.
    - Donner la possibilité de changer le mot-clé de la traduction.
    - Donner la possibilité de changer la langue de la traduction.
    - Donner la possibilité de mettre les fichiers de traduction a jour.
    - Donner la possibilité de supprimer une traduction.
    - Accessible que par les administrateurs du site.
  + Gestion des messages a envoyé
    - Répertoire tous les messages.
    - Donner la possibilité d’ajouter un message.
    - Donner la possibilité de changer le texte du message.
    - Donner la possibilité de changer la langue du message.
    - Donner la possibilité de supprimer une traduction.
    - Donner la possibilité de saisir nom prénom et lien dans le massage comme on le souhait.
    - Accessible que par les administrateurs du site.

## Techniques

Une fois les fonctionnalités déterminées, nous pourrions commencer à éclairer les aspects techniques. Que nous avons divisé en sept groupes en fonction des besoins du client. Ces groupes sont :

* La base de données
* Le backoffice
* La sécurité
* Le sondage
* L’importation du ficher CSV
* L’envoi d’un email
* La traduction

### La base de données :

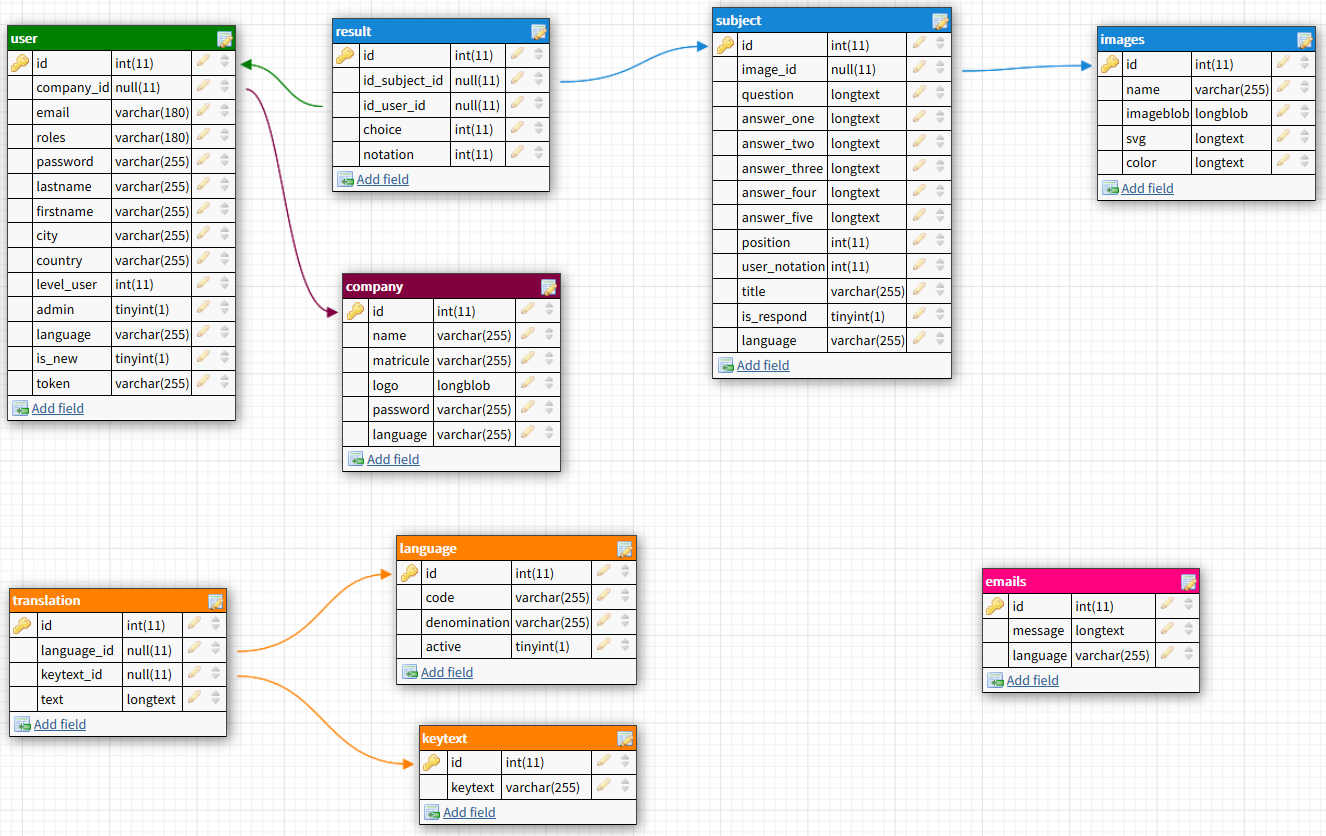
Etant donné que les conditions fixées par le client supposent le stockage des données sur une longue période. Donc nous avons besoin d’une base de données. Cette base de données figure °1 comportera neuf tableaux qui répondront aux exigences demandées. La création de cette base de données a été réalisée avec le projet de Doctrine fourni avec Symfony. Doctrine, qui est une collection de plusieurs bibliothèques PHP, est spécialisée dans l’affichage des tables d’une base de données dans un projet PHP en tant qu’un objets PHP, tout en utilisant ces tables comme objet. Cette connexion entre les éléments de la base de données et les éléments utilisés comme objets dans PHP est également appelée ORM.

Figure 1 : Base de données Projet TealFinder

### Le backoffice :

Vu que la base de données qui fournit des informations aux site, et que ces informations peuvent changer avec le temps, nous avons besoin d’un backoffice qui contrôle cette base de données. Nous y parvenons en créant un espace dans le site pour chaque table de la base de données, qui permet la manipulation de la base de données sur ces quatre opérations CRUD ce qui est censé signifier Créer, Lire, Mettre à jour, Supprimer ou en anglais Create, Read, Update, Delete. Cela se fait grâce à Doctrine qui d’une parte, avec le ServiceEntityRepository apporte les informations et avec ça les bonnes valeurs du tableau. Et de l’autre parte, c’est le EntitManagerInterface du Doctrine qui, créé, fait la mise à jour des informations ou supprime les valeurs du tableau.

### La sécurité :

La sécurité est régulée par le fichier security.yaml. Qui régule :

* + L’algorithme de cryptage du mot de passe des utilisateurs.
  + Le pare-feu et donc les routes utiliser en fonction de ce que l’utilisateur fait.
  + Les rôles hiérarchiques existant et ses héritages.
  + Si un utilisateur peut utiliser cette route en fonction de son rôle ou non.

Les rôles suivants sont configurés :

* + ROLE\_USER
  + ROLE\_ADMIN (groupe)
  + ROLE\_SUPER\_ADMIN (site)

En outre, pour créer un compte utilisateur, un email sera envoyé dans lequel il y a un token nouvellement généré, avec lequel l’utilisateur peut se temporairement connecter et sera obligé de créer un nouveau mot de passe avant de pouvoir se connecter. Ce token qui est une valeur de la table dans la base de données de l’utilisateur est détruit après la première connexion. Et peut-être régénéré si le mot de passe est perdu, ce qui rend le compte inaccessible tant que le mot de passe n’a pas été recréé. La création du mot de passe doit être une chaîne d’au moins 6 caractères avec au moins une valeur numérique et au moins une valeur alphabétique.

### Le sondage :

La partie principale de la page, qui comprend la réponse aux question et l’affichage des résultant et qui aura beaucoup de dialogue avec la base de données. En effet, l’utilisateur peut à nouveau répondre à aux questions déjà répondues. De plus chaque utilisateur peut consulter son résultat à tout moment et le comparer avec son organisation. Pour rendre cela possible, Doctrine sera utilisée pour créer, mettre à jour ou lire des données. D'une part, l'utilisateur peut visualiser ses résultats pour chaque question individuellement ou les afficher dans un diagramme circulaire comment demandé par le client.

### L’importation du fichier CSV :

Afin de donner aux organisations la possibilité d’enregistrer plusieurs utilisateurs en même temps dans la base de données l’administrateur(groupe) aura la possibilité d’importer un fichier CSV. Le fichier CSV doit contenir le contenu suivant :

* + Firstname
  + Lastname
  + Email

Pour contrôler cette importation un service sera mis en place qui représentera une classe. Cette classe a pour but de sauvegarder les utilisateurs dans le tableau User de la base de données dès que les points suivants ont été vérifiés et confirmés.

* + Si le fichier n’est pas vide.
  + Si le fichier a toutes les valeurs requises
  + Si le prénom est uniquement alphabétique.
  + Si le nom est uniquement alphabétique.
  + Si l’adresse mail est une adresse email.

Cette classe est présentée en détail dans le chapitre 4 ou la procédure d'importation du fichier CSV est analysée.

### L’envoi d’un email

Pour pouvoir inviter un utilisateur, il doit être prévenu par email. Cet e-mail donne à son tour à l’utilisateur la possibilité d’accéder à une page spéciale où il peut définir son mot de passe. Pour envoyer l’e-mail le service Mailer du Symfony sera utilisé qui fournira une mise en page créer par le Template Twig. Le contenu du message et le positionnement de certaines valeurs dans ce message peuvent être configurés par l'administrateur dans le backoffice.

### La traduction

Afin d’afficher le site dans au moins deux langues, comme exiger par le client, nous allons d’abord enregistrer toutes les valeurs dans la base de données. Les valeurs enregistrées sont divisées en trois tableaux différents. Qui sont :

* + Translation
  + Keytext
  + Language

Ces tableaux sont liés entre eux et un fois actualiser ils enregistreront pour chaque langue et ses valeurs un fichier différant. Grâce à ces fichiers et le TranslatorInterface du Symfony, il est possible d'appeler la langue respective de l'utilisateur avec la fonction trans en Twig et la fonction $this->translator->trans() en PHP, si disponible. Par défaut la langue française sera affichée

## Competénces du réferentiel

* Développer une interface utilisateur web ou mobile dynamique
  + L’interface utilisateur est affiné grâce à des langage de programmation dynamique, elle est alors capable d’interagir et d’évoluer en fonction de l’utilisateur de manière la plus efficace possible.
  + L’interface est sécurisée
* Créer une base de données
  + Le schéma entité association couvre les règles de gestion sur les données
  + Le schéma respecte le formalisme du modèle entité association
  + Le schéma physique de la base de données est normalisé
* Développer des composants d’accès aux données
  + Les traitements relatifs aux manipulations des données répondent aux fonctionnalités décrites dans le dossier de conception technique
  + Connaissance du langage de requête de type SQL
* Développer la partie back-end d’une application web ou web mobile
  + Les pages web répondent aux fonctionnalités décrites dans le cahier des charges
  + L'architecture de l'application répond aux bonnes pratiques de développement d'application web
  + Le code source des composants est documenté
  + Le développement est réalisé grâce au paradigme de l’objet

# Eléments les plus significatifs

Parce que le projet Tealfinder était un projet de groupe, les décisions qui ont été prises, ont été prises ensemble, ce qui a eu une influence positive sur les fichiers créés par moi. Cela inclut :

* + Les fixtures
  + L’importation du fichiers CSV
  + L’envoi d’un email
  + La sécurité
  + Le sondage

## Les fixtures

L'une de ces influences positives a été qu'il a été décidé de créer des fixtures qui ont finalement été un énorme gain de temps car il était facile d'adapter la base de données aux besoins du client et à la logique, au course d’évolution du projet, sans perdre de données. Au final, le site peut être entièrement créée avec tout son contenu pendant un nouvelle installation sur un serveur.

### La création des fixtures

Fondamentalement, les fixtures sont des informations incorrectes (fake) qui sont stockées dans la base de données pour pouvoir tester plus facilement le site entier ou des éléments individuels et finalement gagner du temps. Parce que ces valeurs contenues dans les fixtures peuvent être de tout type, il peut également s'agir de valeurs réelles. Ce qui signifie que tout le contenu du site TealFinder peut être chargé par les fixtures. Vu que le client nous a déjà fourni tout le contenu du site avec le Wireframe sur Figma, tout le contenu de nature dynamique ont été sauvés dans les fixtures. De plus, les fixtures ont été créés pour créer du contenu fake donc ces valeurs sont incorrectes. Ces fixtures sont :

|  |  |
| --- | --- |
| Contenu dynamique | Contenu fake |
| * + SvgFixtures   + KeytextFixtures   + LanguageFixtures   + TranslationFixtures   + SubjectFixtures | * + CompanyFixtures   + CsvKeyValuesFixtures   + EmailsFixtures   + NavigationsFixtures   + ResultFixtures   + UserFixtures |

Ces fixtures ont été créés ou développés pendant toute la durée du projet selon la façon dont nous en avions besoin. Chacun de ces fixtures représente un tableau dans la base de données avec ses valeurs respectives. Ces fichiers ont été créés à l'aide du bundle DoctrineFixturesBundle qui peut être ajouté à Symfony. L'ajout du fichier se fait avec composer et la commande :

composer require --dev doctrine/doctrine-fixtures-bundle

Une fois installé, je n'avais besoin que de créer une classe pour chaque Entity, une Entity est le représentant d’un tableau dans la base de données et fait partie de la ORM Doctrine, qui hérite du classe abstraite fixture. Cet héritage m'oblige à créer une méthode avec le nom load() qui prend une ObjectManager comme paramètre.

**public** **function** load(**ObjectManager** $manager);

Avec l'aide de ce ObjectManager, il m'est alors possible de sauvegarder des valeurs dans la base de données. Cette sauvegarde, se produit avec les méthodes persist(), qui collectent les données, et flush(), qui sauvegarde finalement tout ce qui a été collecté.

    $manager->persist($company);

    $manager->flush();

Je n'ai qu'à écrire la logique qui détermine quelles données doivent être écrites dans la base de données. Au fur et à mesure de l’avancement du projet, des connexions ont été établies entre les tableaux, j'ai donc dû faire attention à l'ordre dans lequel la fixture est chargé en premier. Parce que si je veux écrire des valeurs dans une tableau qui reçoit des valeurs d'une autre tableau et que ce tableau n'existe pas encore, l'enregistrement des données sera réinitialisé, aucune des données ne sera enregistrée et j'obtiens un message d'erreur qui m'indiquant qu'une valeur est manquante. Pour cela, on peut modifier les noms des fichiers du fixture afin qu'ils soient chargés par ordre alphabétique. Parce que Symfony fait exactement cela. Ou, une autre option consiste à hériter de l'interface OrderedFixtureInterface, ce qui m'oblige à créer la méthode getOrder(), qui renvoie une valeur entière, qui à son tour spécifie la séquence du chargement du fixtures.

**public** **function** getOrder()

    {

        return 8;

    }

### La fixture : translation

Ci-dessous, je voudrais montrer un exemple complet d’une fixture qui représente le fichier TransnaltionFixtures.php. Cette fixture charge un fichier CSV avec lequel il est possible de lire les valeurs correctes du tableux language et du tableau keytexte de la base de données et d'écrire ces valeurs dans le tableau translation de la base de données. Le fichier CSV est une copie brute(dump) du tableau translation dans la base de données, s'il est nécessaire de changer ce tableau, il est suffi d'adapter le fichier TransnaltionFixtures.php et on peut continuer à travailler sans perte de données.

**class** TranslationFixtures **extends** Fixture **implements** OrderedFixtureInterface

{

**private** $languageRepository;

**private** $keytextRepository;

**private** $container;

**public** **function** \_\_construct(

**LanguageRepository** $languageRepository,

**KeytextRepository** $keytextRepository,

**ContainerInterface** $container = null

    ) {

        $this->languageRepository = $languageRepository;

        $this->keytextRepository = $keytextRepository;

        $this->container = $container;

    }

**public** **function** load(**ObjectManager** $manager)

    {

        $csvDirectory = $this->container->getParameter('csvs\_directory');

        $fileTranslation = $csvDirectory . '/translation.csv';

        $handle = fopen($fileTranslation, "r");

        $fileTranslation = fgetcsv($handle, 10000, ",");

        while ($fileTranslation) {

            $translation = new Translation();

            $translation->setKeytext($this->keytextRepository-> findOneById($fileTranslation[2]))

                ->setLanguage($this->languageRepository-> findOneById($fileTranslation[1]))

                ->setText($fileTranslation[3]);

            $manager->persist($translation);

            $fileTranslation = fgetcsv($handle, 10000, ",");

        }

        $manager->flush();

    }

**public** **function** getOrder()

    {

        return 6;

    }

}

Même s'il était difficile au début de comprendre exactement comment fonctionnent les fixtures, j'ai pu apprendre à les aimer. Et constatez que les fixtures sont un nécessité absolu dans tous les développements, car ils peuvent gagner beaucoup de temps et éviter les problèmes.

## L’importation du fichiers CSV

Une autre partie du projet consistait à donner aux organisations la possibilité d'enregistrer plusieurs personnes dans la base de données en même temps à l'aide d'un fichier CSV et de les inviter par e-mail. Cette tâche m'a été confiée et je voudrais expliquer plus en détail dans ce chapitre ce qui se passe lors de cette importation du fichier CSV. Avant de commencer à créer le service chargé d'importer le fichier et d'écrire les valeurs dans la base de données, j'ai réfléchi attentivement et j'ai pu identifier à l'avance les problèmes suivants, que je dois prendre en compte lors de l'importation du fichier. Que j'aimerais déclarer un par un et expliquer comment j'ai traité ces problèmes.

### Le type du fichier

* public function checkFileType(): bool

Le fichier téléchargé doit être vérifié pour son type MIME. MIME qui est une norme de référence pour identifier les types de fichiers et est actuellement divisé en dix types. Y compris le type de texte dont j'ai besoin pour contrôler mon fichier CSV. Puisque je contrôlerai cela sur son type MIME text/csv avec la méthode mime\_content\_type() du la bibliothéque PHP. Tant que le type de fichier téléchargé correspond au type text/csv, tout va bien.

if (mime\_content\_type($this->file) == "text/csv") {

      return true;

### Les en-têtes

* public function checkKeyValuesEmpty(): bool

Parce que le fichier CSV est un tableau qui montre le nom, le prénom et l'adresse e-mail de l'utilisateur respectif, j'ai besoin d'un en-tête dans le fichier CSV qui me dit quelle colonne représente quoi. Ce qui signifie que je dois vérifier cet en-tête. Ce que je fais en convenant avec le client qu'aucun des en-têtes ne peut être inférieur à 3 caractères et avoir seulement les valeurs a-z ou A-Z. Je peux donc contrôler cet en-tête.

if (preg\_match('/[ [a-zA-Z]{3,};[a-zA-Z]{3,};[a-zA-Z]{3,}/', $this->data)) {

      return true;

### Les en-têtes et ses noms

* public function checkKeyValuesName(): bool

Afin de donner à l'organisation la possibilité d'importer ses fichiers CSV aussi facilement que possible, l'administrateur du site TealFinder peut déterminer à quoi doivent ressembler les valeurs dans l'en-tête. Cela se passe dans le backoffice figure2 où l'administrateur du site a la possibilité de saisir les valeurs souhaitées et chacune d'elles à l'une des trois valeurs firstname, lastname, email (tAsValue)figure2. Ces valeurs, qui sont stockées dans la base de données, sont appelées lors d'une importation de fichier et transmises à la variable $csvKeyValuesNames en tant que table dans le constructeur de la classe. Je peux donc comparer les valeurs d'en-tête enregistrées dans la base de données avec celles du fichier. Et ce en comparant les valeurs de la deuxième colonne (Nom)figure2 avec celles des valeurs du fichier. Si les trois valeurs du fichier sont présentes, c'est ok.

    foreach ($this->fields as $field) {

      if (!in\_array(strtolower($field), $this->csvKeyValuesName)) {

        return false;

      }

}

### Les en-têtes et ses valeurs

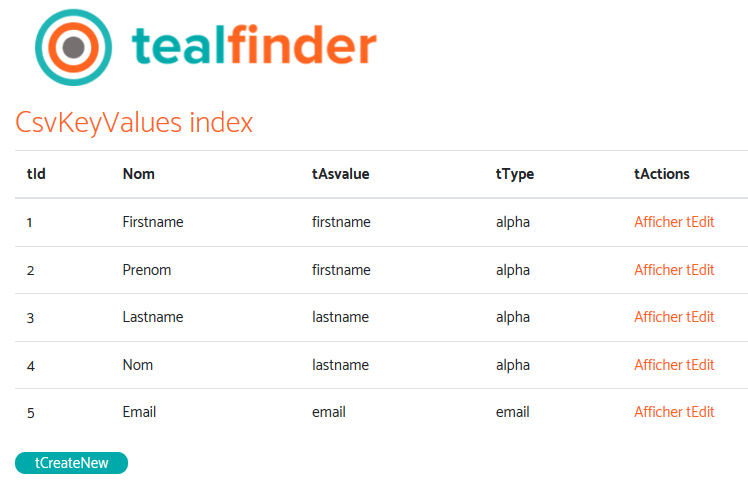


Figure 2: Backoffice Administrateur site | /csvkeyvalues

* public function checkKeyValuesValue(): bool

Cependant, je dois également vérifier si chacune des trois valeurs qui j'ai besoin, c'est-à-dire firstname, lastname et email, sont également disponibles. Comment avant je vais donner les valeurs de la base de données à la variable $csvKeyValuesValue dans le constructeur de la classe. Cette requête de base de données est effectuée avec une méthode de Doctrine que j'ai créée moi-même et qui contient le mot clé distinct pour obtenir une seule des valeurs existantes dans la colonne (tAsvalue)figure2.

        return $this->createQueryBuilder('c')

            ->select('c.asValue')

            ->distinct()

            ->getQuery()

            ->getResult();

De plus, chaque valeur de l'en-tête du fichier est comparée aux valeurs (Nom)figure2 de la base de données dans une boucle. Si les valeurs correspondent, la valeur (tAsvalue)figure2 est enregistrée dans un tableau $asValue. Cela m'amène au fait que si les valeurs du tableau $csvKeyValuesValue correspondent à celles du tableau $asValue, tout est OK.

$results = array\_diff($this->csvKeyValuesValue, $asValues);

### Les en-têtes et leurs positions

* public function setKeyValues(): bool

Afin de sauvegarder plus tard les données que je reçois du fichier CSV au bon endroit dans la base de données, je dois trouver quelle valeur est dans quelle colonne du fichier. Pour ce faire, je vais boucler les valeurs d'en-tête du fichier et comparés chaque valeur avec celles des valeurs du tableaux $csvKeyValuesName. S'il y a une correspondance, un nouveau tableau $csvKeyValuesPosition est créée. La valeur clé de ce tableau est la valeur dans (tAsvalue)figure2 et la valeur elle-même est l'index de boucle.

    foreach ($this->fields as $key => $field) {

      if (in\_array(strtolower($field), $this->csvKeyValuesName)) {

        $tmp = $this->csvKeyValuesRepository->findOneByName($field);

        $this->csvKeyValuePositions[strtolower($tmp->getAsValue())] = $key;

      }

    }

    return true;

  }

### Les en-têtes et le nombre de tous les champs

* public function checkFieldQuantity(): bool

Ici, il est seulement vérifié si le nombre de colonnes, dans notre cas il y en a trois, multiplié par le nombre de lignes est plus grand ou plus petit que tous les champs. Si ce n'est pas le cas, c'est ok.

### Les champs vides

* public function checkEmptyField(): bool

Ici aussi, il est seulement vérifié si le champ est vide. Si ce n'est pas le cas et que le champ compte plus de 2 chartes, c'est ok.

      if (!preg\_match('/\S{3,}/', $field)) {

return true;

### Le type de la valeur

* public function checkCsvKeyValueTypes(): bool

De plus, il faut vérifier si le fichier CSV et ses valeurs sont les mêmes que les types d'en-tête. Ce que je veux expliquer, c'est que par exemple lastname ou firstname est un type alphabétique. Ces types peuvent donc ne pas avoir de caractères spéciaux ou de chiffres. Cependant, l’email est un type qui peut comprendre certains caractères spéciaux et possède sa propre signature, qui devrait ressembler à ceci :

[a-z0-9\_.-]+@[a-z0-9-]+[.][a-z]+

Cela doit être vérifié, donc le firstname et le lastname sont vérifiés avec la méthode ctype\_alpha() s'ils sont alphabétiques. Et email avec la méthode filter\_var($varibale, $filter) et le filtre FILTER\_VALIDATE\_EMAIL vérifié s'il s'agit d'un type e-mail. Cela se fait par une switch condition dans laquelle le champ respectif du fichier est vérifié en fonction de son type qui a été déterminé dans la méthode setKeyValeus() (tType)figure2 a côté de la variable pour la position.

 switch ($this->csvKeyValueTypes[$key]) {

        case 'alpha':

          if (!ctype\_alpha($field)) {

            return false;

          }

          break;

        case 'email':

          if (!filter\_var($field, FILTER\_VALIDATE\_EMAIL)) {

            return false;

          }

          break;

      }

    }

    return true;

### L’adresse email

Un autre problème est un utilisateur qui existe déjà ne peut être écrit dans la base de données. À cet effet, tous l’adresse e-mail définie comme le type unique de la table User dans la base de données, sont comparé avec ces du fichier CSV et ensuite vérifiée si elle existe déjà. S'il n'y a pas de correspondence, c'est ok.

 if ($this->userRepository->findOneByEmail(

$this->fields[$this->csvKeyValuePositions['email']]))

{return false;}

### Résumé

Les fonctions déclarer avant ne sont qu'une partie du service que j'avais créé et ne devraient représenter que les problèmes avec lesquels j'ai dû travailler. Je pense que sans plus d'informations, on perde rapidement la trace. J'essaie donc de formuler une image globale de la classe CsvFileImport figure3 qui c’occupe de l’importation du ficher CSV, et de son fonctionnement et représenter cette fonctionnement visuellementfigure4.

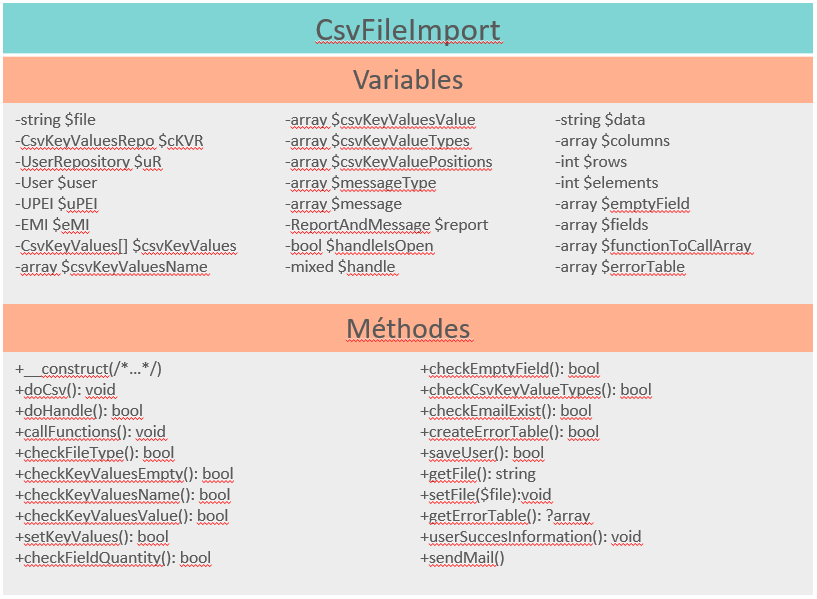


Figure 3: La classe CsvFileImport

Nous avons donc un fichier CSV téléchargé par l'utilisateur. Ce fichier est du type CSV et est transmis au variable $handle avec la méthode fopen () et sa premier ligne sera transféré au variable $data avec la méthode fgetcsv(). Les deux méthodes font partie de la bibliothèque PHP. Cette transmission dès la première ligne du fichier CSV se produit dans le constructeur de classe CSVFileImport.

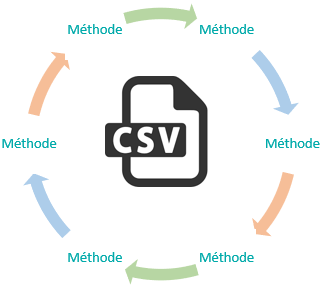


Figure 4: L’idéologie de la classe

$this->handle = fopen($file, 'r');

$this->data = fgetcsv($this->handle, 1000)[0];

L'idée est de vérifier ce fichier ligne par ligne et de passer un par un par toutes les méthodes pour vérifier le contenu. On comprend que cela pourrait causer des problèmes s'il y a une erreur dans le fichier et que le contrôle continue. Ou le contrôle s'arrête au milieu du fichier et termine la boucle. Les deux cas seraient difficiles à gérer. Afin d'éviter une erreur, et ne pas arrêter la vérification du fichier, et que chaque ligne puisse être sauvegardée si elle est correcte, j'ai divisé le contrôle en deux phasesfigure5. Les deux phases partagent les méthodes décrites précédemment. La première phase est utilisée pour contrôler l'en-tête. Si une des méthodes détecte un problème, la contrôle est annulé et la classe renvoie un message d'erreur avec les informations manquantes, incomplètes ou incorrectes à l’utilisateur. Dans ce cas, il est nécessaire d’arrêter l’importation car les informations nécessaires pour continuer à traiter le fichier CSV sont incomplètes ou incorrectes.

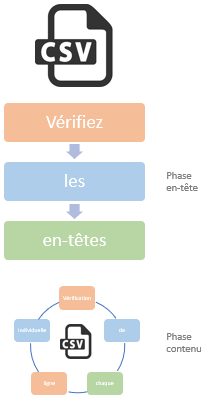


Figure 5: Les deux phases de la classe CsvFileImport

Les messages d'erreur sont traités par un autre service que j'ai créé. Ce service est utilisé pour collecter toutes les erreurs et de les afficher correctement à la fin du traitement du fichier CSV. Comme c'est un service différent, je ne veux pas aller plus loin. Néanmoins, je voudrais dire que chaque méthode qui contrôle le fichier CSV est connectée à ce service. Et en cas d'erreur, cette erreur est transmise à ce service.

Si aucune erreur ne se produit dans la première phase qui contrôle l'en-tête, le fichier CSV sera transmis à la deuxième phase. Cette phase vérifiera le contenu de chaque ligne individuelle du fichier CSV. Cela se fait dans une boucle qui,

    while ($this->doHandle()) {

      $this->rows++;

      $this->fields = explode(';', $this->data);

      $this->elements += count($this->fields);

*// Call functions.*

      $this->callFunctions();

    }

appelle toutes les méthodes sont appelées ligne par ligne. Les méthodes qui contrôlent vers la méthode qui stockent des valeurs individuelles de la ligne jusqu'à les méthodes qui envoient de l'e-mail et qui vont faire la notification à l'utilisateur.

**public** **function** callFunctions(): **void**

  {

    $counter = count($this->functionToCallArray);

    while ($counter--) {

*// Do all checks.*

      $functionToCall = $this->functionToCallArray[$counter];

      $this->report->reportRecord($functionToCall, $this->$functionToCall());

    }

  }

S’il arrive qu'une ligne ne soit pas correcte, elle sera enregistrée avec le service qui enregistre les erreurs et transmise ultérieurement à l'utilisateur en tant qu'information figure6 après avoir parcouru toutes les lignes du fichier CSV.

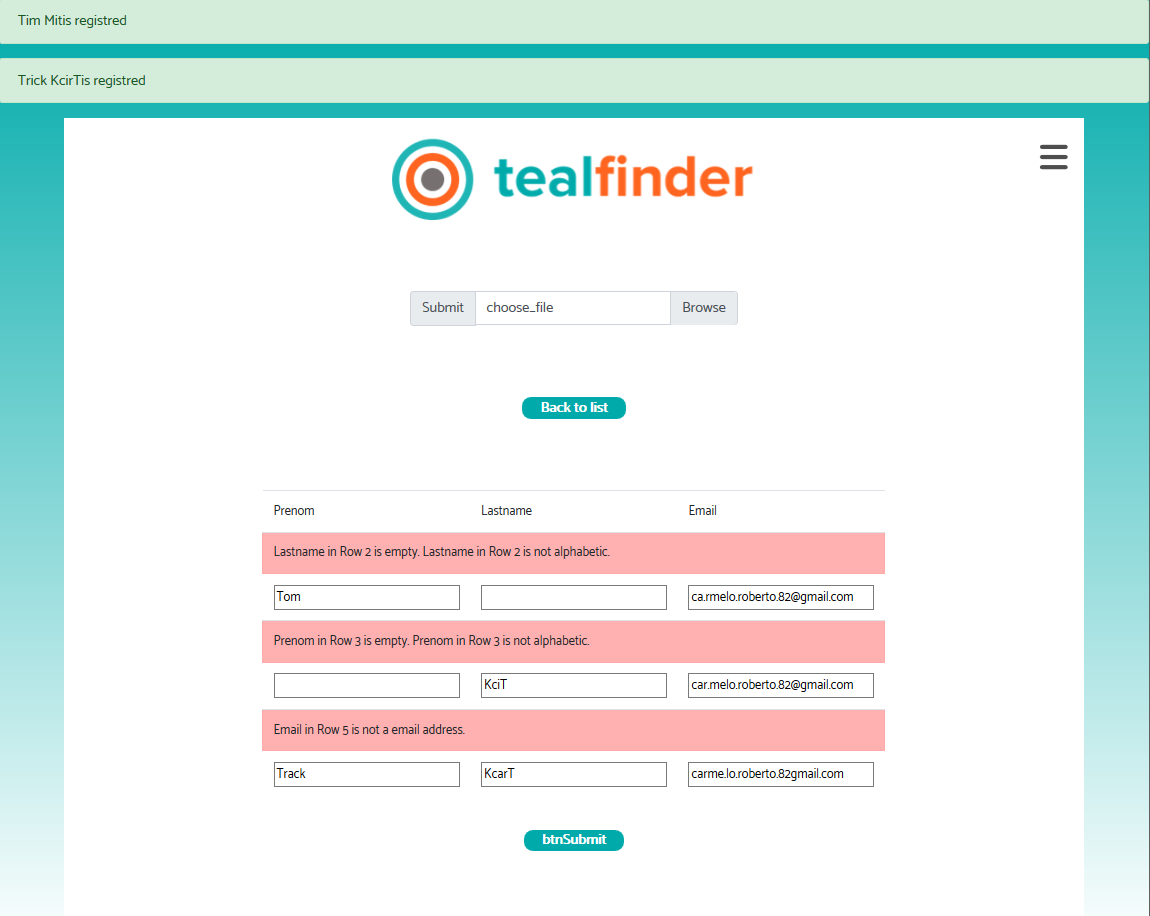


Figure 6 : Importation du fichier CSV avec erreurs

À ce moment, toutes les lignes qui n'ont pas d'erreur sont enregistrées dans la base de données et l'utilisateur de la ligne enregistrée a été invité avec un message par e-mail. L'administrateur du groupe qui a importé le fichier CSV peut corriger les champs et les réimporter dans le cas où une ou plusieurs lignes n'ont pas été enregistrées dans la base de données. Afin d'éviter d'autres erreurs après la correction et de pouvoir tester à nouveau tous les champs, le tableau qui a été corrigée par l'administrateur de groupe est enregistrée en tant que fichier CSV et la classe CsvFileImport avec la méthode doCsv() est appelée à nouveau. Dans le cas où le fichier a été importé complètement et qu'il n'y a pas eu d'erreurs, l'utilisateur est renvoyé à la page de l'administration du groupe et reçoit un message confirmation.

## Le Sondage

Une des parties principales du site TealFinder et la raison de ce projet qui est l'estimation de la culture de l'organisation à l'aide d'un questionnaire de 20 questions avec 5 réponses chacune. Le nombre de réponses reste statique, car ceux-ci sont basés sur le livre Reinventing Organizations et divisés en 5 groupes culturels :

* Evolutionnaire
* Pluraliste
* Orienté réussite
* Conformiste
* Impulsif le nombre de questions peut varier.

Cependant, il est possible que le nombre de questions puisse changer. Seulement une réponse sera possible pour chaque question et chaque questionnaire peut exister en plusieurs langues. De plus chaque utilisateur a la possibilité de noter chaque question.

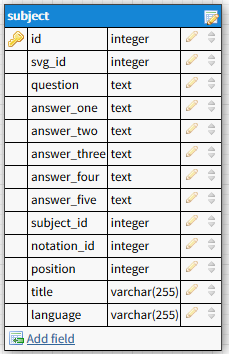


Figure 7 : Le question numéro 1 avec la possibilité de choisir une réponse et de noter la question soit même.

### Le tableau : subject

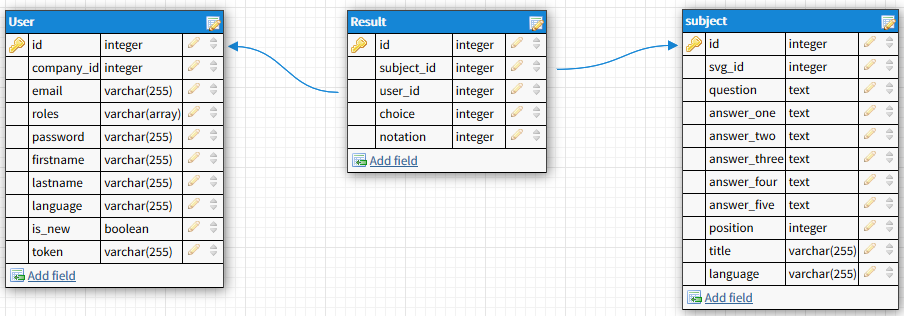
Parce que les informations (questions, réponses, résultats, etc.) doivent être conservées sur une durée de longe terme. Ces valeurs doivent être enregistrées dans une base de données. A cet effet, un tableau a été créée qui a le nom subject figure8 et qui peut contenir ces valeurs.

Figure 8 : Tableau subject de la base de données TealFinder. Un pour tous et tous pour un



Avec ce tableau, il est désormais possible d'enregistrer toutes les réponses de tous les utilisateurs. Cependant, cela serait très improductif car toutes les valeurs de ce tableau seraient répétées à chaque réponse de chaque utilisateur, bien que seule deux valeurs changent. Pour cette raison, une relation plusieurs à plusieurs (ManyToMany) a été créée. Qui stocke les réponses de tous les utilisateurs pour toutes les questions. Cette relation est établie dans Doctrine via deux relations plusieurs à un (ManyToOne) qui pointent vers deux tableaufigure9, qui ont besoin d’une relation ManyToMany. Dans notre exemple, cela se traduit par une relation entre le tableau Result qui point vers les tableaux User et Subject. Par conséquent tous les utilisateurs peuvent avoir une relation avec toutes les questions et toutes les questions peuvent avoir une relation avec tous les utilisateurs.

Figure 9 : ManyToMany connexion entre l'utilisateur au sujet via le résultat



Afin d'exprimer cette relation dans Symfony en utilisant la ORM Doctrine, la classe finale ManyToOne, qui se retrouve dans le namespace Doctrine\ORM\Mapping, doit être appelée, qui a besoin d’être fourni avec un nom du tableau, le quelle qui va être ciblé, pour créer la relation. Cela s'exprime dans l'annotation de Entity. Voici la relation entre les tableaux Result et User.

*/\*\**

*\* @ORM\ManyToOne(targetEntity="App\Entity\User", inversedBy="userResult",  \* fetch="EAGER")*

*\*/*

**private** $user;

De plus, plusieurs paramètres peuvent encore être transmis en fonction de la situation. Cela se fait automatiquement par le ORM Doctrine lors de la configuration de Entity, mais doit être ajusté en fonction de la situation. Le même chose se prdouit pour la méthode qui ajout ou suppression de valeurs dans le tableau de Result en utilisent des Entitys User ou Subject. Les méthodes qui rendent cela possible sont créées automatiquement pendant la configuration de la Entity respective et peuvent être modifiées selon les besoins.

### Le SondageController et ses Services

Lorsque on utilise Symfony 5, la programmation orienter objet est fortement imposer. Ce qui n'est pas obliger, mais très utile. Et encore plus pratique à utiliser avec l'architecture MVC ce qui signifié Model, View, Controller en anglais, fournie par Symfony, qui peut être divisée en trois type qui sont :

* + Controller (Controller)
  + Service (Model)
  + Template (View)

Cela peut être considéré comme une boucle dans laquelle le Controller attend que l'utilisateur saisisse des informations et transmet ces informations au Service qui traite les informations et les renvoie au Template qui génère une vue avec les informations traitées. Puis attende que l'utilisateur entre à nouveau une information. Ces informations peuvent être, par exemple, le changement de la page ou la fermeture du navigateur. Cette structure décrite par moi n'est pas donnée par Symfony car elle a que le Controller et le Template d'origine. Avec lequel le Controller peut être vu comme une intermédiafigure11 qui communique entre le Service et le Template. La raison pour laquelle ce n’est pas une boucle directe vient de ce que le Controller créé par Symfony propose déjà une communication avec le Template. On peut donc écrirai la logique même dans le Controller, mais ce serait un mauvais style de programmation. Pour cette raison, toutes les tâches liées au traitement logique sont écrites dans un Service a parte. Tout le reste dans le Controller, comme la confirmation d'un formulaire ou la suppression d'une valeur dans la base de données. Cela fait partie de la programmation SOLID que j'ai apprise lors de ma formation.

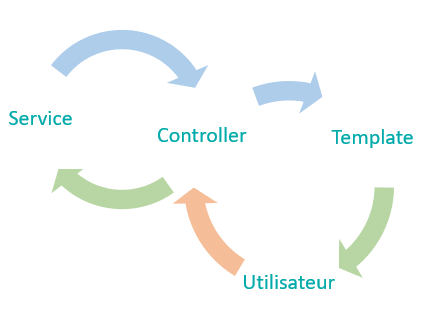


Figure 11 : Modèle MVC

* + Responsabilité unique (Single responsibility principle)
  + Ouvert/fermé (Open/closed principle)
  + Substitution de Liskov (Liskov substitution principle)
  + Ségrégation des interfaces (Interface segregation principle)
  + Inversion des dépendances (Dependency inversion principle)

J'ai donc essayé de diviser ma le sondage en quatre Services. Qui communiquent tous avec le Controller qui, à la fin, transmettent les données au Template. Donc la méthode sondage($id) de la classe SondageController a pour but d'attendre les informations de l'utilisateur, qui peuvent être la confirmation des réponses ou la sélection des questions. Si cette information est une réponse à un des questions, la méthode sondage() du Service Sondage sera appelé. Cela permet de savoir s'il existe déjà une réponse à cette question ou pas. Selon les données, la base de données sera mise à jour ou une nouvelle valeur sera écrit dans le tableau Result. Cette échanger avec la base de données est réaliser avec un autre Service, qui s’appelle InsertReponse avec ses méthode insert() et update(). Indépendamment de cela, une question avec cinq réponses est toujours affichée, ce qui signifie que les questions dans la langue de l'utilisateur doivent être affichées dans le bon ordre. Pour cela, les Services CheckLanguage et SondagePositionCounter sont appelés. Une fois tous les Services ont fait leurs travaillesfigure12 les informations traitées sont transmises au Template afin de créer une vue de la page web.

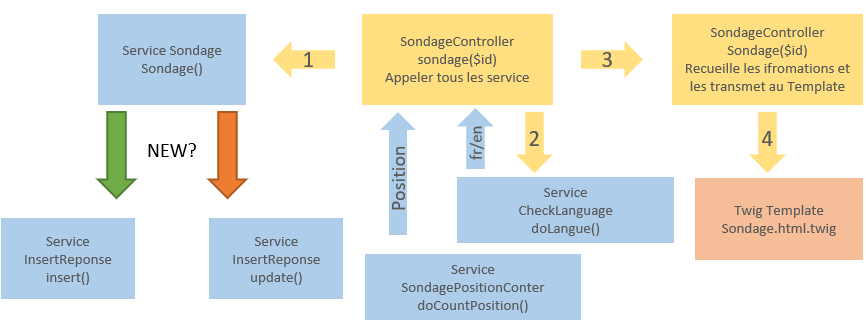


Figure 12 : Exemple MVC en cas de Sondage

### Le Template Twig

Twig est un Template moderne pour PHP, avec lequel il est possible de créer dynamiquement des pages html. Cela serait également possible avec PHP lui-même, mais cela rend le code html très illisible. Twig est la meilleure solution pour cela car sa syntaxe est plus facile à lire et ne rendent pas le code html inutilement difficile, comme ce le cas avec PHP. À la fin, Twig lui-même est du code PHP optimisé. Il est également possible d'hériter d'autres fichiers html.tiwg

{% extends "MAIN/BASE.html.twig" %}

Aussi d'ajouter d'autres fichiers html.twig

  {% include "MAIN/\_list.html.twig" %}

Cela permet d'utiliser du code déjà écrit sans avoir à le réécrire. De plus, Twig offre également le boucles et les conditions et la possibilité de la création des variables. Ce qui facilite la création de contenu dynamique. Dans l'exemple de notre Sondage, cela se traduit par la prédiction de la page suivante et la page précédent. En fonction de la page qui suive le lien est créé différemment.

    <div id="sondageArrowRight">

      {% if subject.id>=end%}

        <a href="{{path('result\_show',{'idUser':app.user.id})}}">

        {% else %}

          <a href="{{path('sondage',{'id': after.id})}}">

          {% endif %}

          {{images.site.arrow.svg|raw}}

        </a>

      </a>

    </div>

Il est également possible d'écrire des filtres et des fonctions par vous-même. Ces filtres et fonctions sont créés dans une propre classe qui est écrite en PHP. Les possibilités ne sont donc limitées qu'à la créativité de l'utilisateur. J'utilise donc des fonctions Twig pour créer dynamiquement les headers d’un tableaufigure13 afin que les noms des objets d'une classe puissent être affichés. Ou comme dans l'exemple du Sondage, je change les couleurs du Svgfigure7 des réponses selon qu'elles sont sélectionnées ou non.

{{change\_svg\_color(images.answer.one,1)|raw}}

Symfony propose également une plus grande sélection de filtres Twig. L'un de ces filtres est le filtre trans.

{{('t'~property|capitalize)|trans}}

Qui est utilisé sur toute la page pour pouvoir traduire le contenu en plusieurs langues. En raison de sa syntaxe simple, il est plus pratique à utiliser dans Twig que dans PHP ou il existe la même fonction, mais elle est plus complexe à utiliser. Twig est un outil très puissant et lorsqu'il est utilisé correctement, il peut éviter beaucoup de code inutile.

## L’envoi d’un e mail

L'envoi des mails est utilisé dans notre application lorsque le chef d'une organisation ou son adjoint souhaite (Admin groupe) inviter ses employésfigure13. Cela est nécessaire car l'utilisateur ne peut pas créer un compte sur l’ensemble du site. L’Administration de groupe peut envoyer cette invitation individuellement pour chaque utilisateur ou il peut envoyer une invitation à tous ceux qui ne l'ont pas encore activée leur compte. De plus, il est possible qu'un utilisateur activé puisse recevoir un e-mail s'il a oublié son mot de passe. Lorsque l'e-mail est envoyé, les informations de connexion sont transmises à l'utilisateur, ce qui lui donne la possibilité d'utiliser l’ensemble du site. Pour ce faire le bundle de Mailer fourni par Symfony est utilisé.

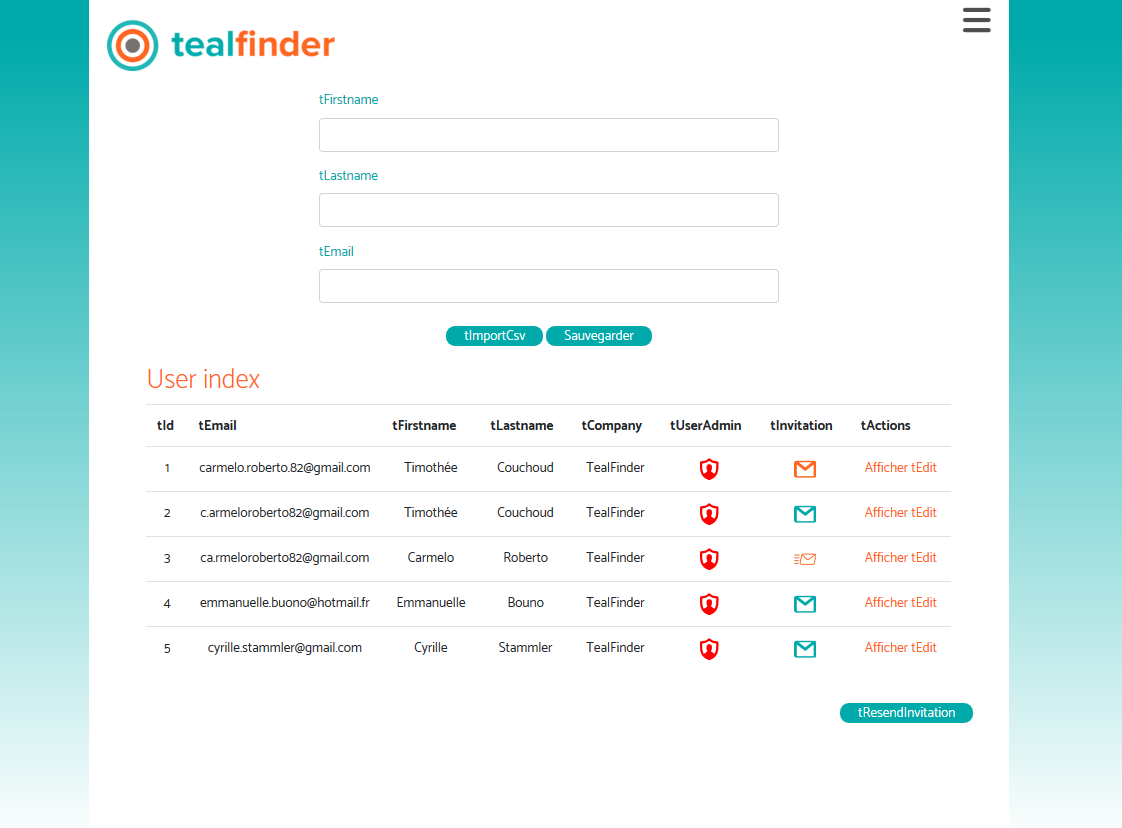


Figure 13 : Backoffice Admin Groupe avec la possibilité d’inviter ses employées

### Autowiring / Symfony Mailer

Symfony Mailer est un bundle de Symfony et peut être téléchargé avec la commande :

    composer require symfony/mailer

Un 3rd party transport bundle peut être téléchargé en fonction de l'utilisation afin d'utiliser les différents fournisseurs (Gmail, MailGun, etc). Dans mon cas, j'ai utilisé Gmail,qui envoie les e-mails pour moi, et qui j’ai téléchargé avec la commande :

    composer require symfony/google-mailer

Une fois téléchargé, on doit étendre le fichier .env avec les données de connexion du fournisseur de messagerie(Gmail), et selon les besoins, le fichier mailer.yml doit être modifié. Après cette configuration, qui n'a pas pris beaucoup de temps, j'ai maintenant eu l'occasion d'utiliser Symfony Mailer. Je le fais en créant un Service SendMailer qui peut être utilisé pour envoyer les e-mails. Cependant, pour pouvoir désormais utiliser le MailerInterface de Symfony, et avec lui la possibilité d’envoyer l’email, je dois faire une dépendance d’injection du MailerInterface et de mon Service SendMailer. Cette dépendance d'injection est rendue très facile dans Symfony. Vu qu'un Controller est appelé à chaque appel d'une page, un Container est créé avec cet appel dans Symfony. Ce Container figure14 contient, à parte de Services déjà fourni par Symfony, tous les Services que j'ai déclarés dans la classe du mon Controller. Cette déclaration se produit soit dans le constructeur de mon Controller, soit dans le paramètre de ma méthode appelée par mon Controller. Ce Controller et son méthode est appelé par la page qui l’utilisateur as demander. Un Controller peut avoir plusieurs méthodes, et chaque méthode dans la Controller peut avoir une page.

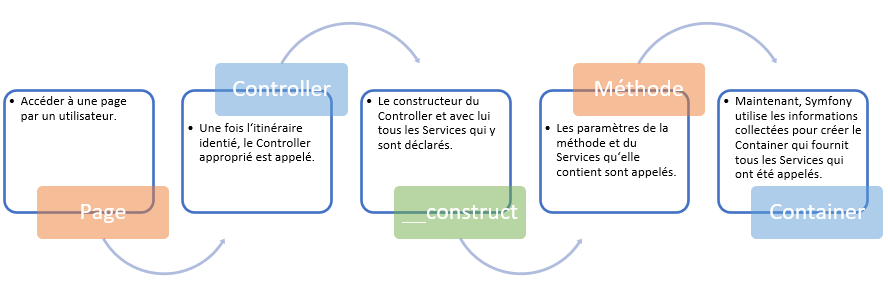


Figure 14 : Symfony et le Service Container

Si un service qui représente une classe PHP a d'autres services qui n'ont pas été déclarés précédemment, ils seront également déclarés. Comme c'est le cas pour mon Service SendMailer qui inclut le Service MailerInterface qui n'a pas encore été déclaré. Cet appel et la connexion de tous les Services qui seront disponibles ultérieurement dans le Container est nomé Autowiring(dépendance d’injection) dans Symfony. Le comportement de Symfony peut être configuré dans le service.yml si, par exemple, on ne veut pas être chargé un Service, selon des besoins.

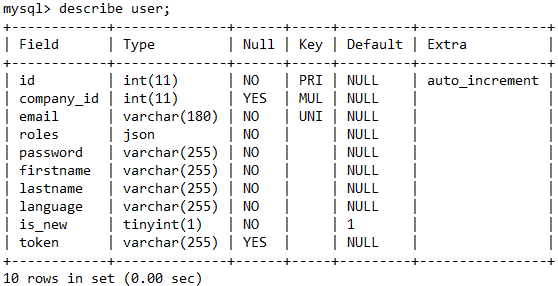
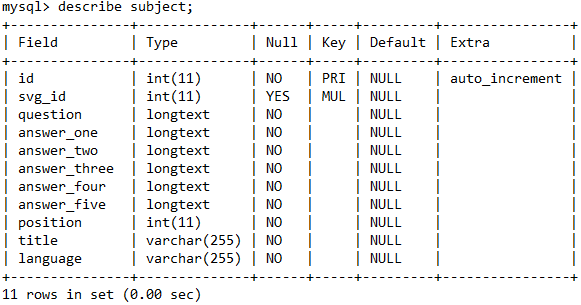
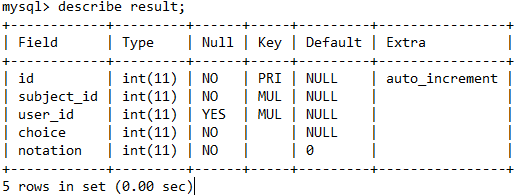
### send()

L'envoi de l'e-mail lui-même est moins compliqué. Ici, une méthode peut être créée pour chaque besoin (inviter, réinviter, demande de mot de passe), qui prend les informations qui doivent être envoyées en compte, puis les envoie avec la méthode send(RawMessage $message, Envelope $envelope = null), qui est mise à disposition par le MailerInterface. Avec le bundle Twig, qui peut être téléchargé avec composer,

    composer require symfony/twig-bundle

offre la possibilité d'envoyer l'e-mail au format HTML. Cela se fait en utilisant la classe TemplateEmail(). Qui fournit la méthode context() et prendre comme paramètres un tableau. Ce tableau contenant le contenu du message peut être organisé dans un Template html.twig. À la fin, le Template et son contenu seront envoyés au destinataire respectif.

## La Sécurité



Many

To

One

Many

To

One

* + La connexion
  + La connexion
  + La connexion