PELOPS

epsi

Rapport de stage :

Développement d’un outil SIRH permettant le suivi de candidats

Staelen Rémi

4 Janvier - 18 Février

Tuteur de stage : Nahim BENBAHLOULI
Etablissement de formation : EPSI
Entreprise d’accueil : PELOPS

Remerciements

En premier lieu, je voudrais remercier Nahim BENBAHLOULI , mon maître de stage pour sa confiance, son écoute et le partage des connaissances.

Je tenais ensuite à remercier Julien VANHAUWAERT et Benjamin WATSON, qui sont d'autres stagiaires, pour leur professionnalisme et l'optimisme partagés durant ce stage.

De plus, je tiens à remercier M. MONTAGNE et Mme GROCKOWIAK du corps administratif de EPSI Lille, d'avoir répondu à nos interrogations concernant le déroulement du stage.

Enfin, je souhaite remercier profondément ma mère pour le soutien qu'elle m'a apporté durant la recherche de mon stage.

Introduction

La Recherche :

Étant en 2e année de l'école informatique EPSI, j'ai déjà eu la chance d'avoir fait un stage en entreprise lors de la fin de ma 1ere année. Ce stage m'avait permis de développer de nombreuses compétences, que ce soient des compétences dans le milieu de l'informatique, ou des compétences sociales, d'interaction et de coordination avec l'équipe.

Pour cette nouvelle année, je voulais trouver un stage qui me permettrait d'exprimer à nouveau mes connaissances acquises depuis la rentrée, mais je voulais également apprendre de nouvelles technologies, venant soutenir mon bagage de compétences. De plus, étant passionné d'intelligence artificielle, je voulais trouver un stage dans le domaine de la data science, car entraîner une intelligence artificielle veut dire utiliser une base de données contenant des centaines, des milliers (beaucoup plus pour les grosses entreprises comme Meta) d'informations.

Ayant commencé mes recherches en Octobre 2021, j'ai rapidement trouvé un stage dans une entreprise faisant de la data science pour conseiller l'achat de parts de marché au niveau des énergies vertes. Après plusieurs entretiens de motivation, de compétences et seulement une confirmation verbale, ils m'ont finalement annoncé qu'ils ne prenaient pas de stagiaire 2 semaines avant la date de début des stages.

C'est alors que je me suis tourné vers M. Nahim BENBAHLOULI, qui a eu l'amabilité d'accepter ma requête. C'est à ce moment que je suis devenu stagiaire pour Tailor.

L'entreprise

PELOPS est une société éditrice du logiciel Tailor. TAILOR est un logiciel SIRH (Système d'information ressources humaines) permettant un suivi de candidats et de collaborateurs.

Elle est née d'un constat à propos du Onboarding en entreprise, c'est -à -dire l'ensemble des pratiques concernant l'accueil et l'intégration d'un collaborateur en entreprise.

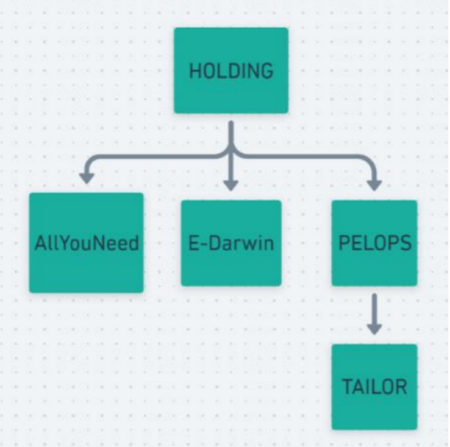
En effet, le onboarding nécessite l'apprentissage des outils de la société, l'adaptation dans une nouvelle équipe et cela prend du temps, ce pourquoi il est souvent baclé ou même inexistant dans certaines entreprises, ce qui résulte a des pertes financières.

C'est alors que M. Maxime Zanchi et M. Nahim BENBAHLOULI y ont vu une opportunité pour se lancer dans ce marché. Suite à cela, ils ont eu l'idée de créer le

logiciel du nom de TAILOR, permettant aux entreprises partenaires d'avoir un meilleur suivi de l'onboarding de ses collaborateurs et ainsi que d'améliorer l'image de l'entreprise.

L'onboarding ne se limite pas qu'à l'entrée du collaborateur dans l'entreprise. Elle se fait également tout au long de sa carrière au sein de l'entreprise et jusqu'à ce qu'il parte (dit "offboarding").

La société PELOPS n'est pas la seule société de Maxime ZANCHI. Il a également créé ALLYOUNEED et E-DARWIN. ALLYOUNEED étant une société d'accompagnement de candidats pour rejoindre des entreprises, et E-DARWIN étant une entreprise de coaching professionnel. Le diagramme ci-dessous démontre les liens entre les entreprises



Organisation des entreprises partenaires de PELOPS

La « Holding » est l'entité rattachant les 3 entreprises : AllYouNeed, E-Darwin et PELOPS. Elle est celle qui va récupérer les capitaux, et va lui permettre de faire des remontées de fonds. E-Darwin est considérée comme étant une entreprise partenaire indépendante, même si elle appartient au même dirigeant. Elle va apporter un coaching professionnel aux candidats. AllYouNeed, quant à elle, est une agence de recrutement de CDI et Freelance. L'idée centrale autour de ces entreprises est de remettre l'être humain au centre de toutes ses actions, en ayant des suivis réguliers, et des volontés transparentes.

Les Missions

Durant mon stage dans l'entreprise PELOPS, j'ai pu effectuer 3 missions principales, les 2 premières en tant que développeur back-end et front-end, c'est-à-dire la relation entre le site internet et la base de données, et l'autre en tant qu'ingénieur IA.

La première mission m'a pris 1 mois à effectuer, nous avions à disposition les prémices du POC pour le logiciel TAILOR ainsi que la base de données. Mon rôle était de récupérer des candidats à un poste, stockés dans la base de données, puis les afficher dans l'application sous forme de liste avec certaines de leur informations. Puis, lorsque l'on clique sur un profil, une popup apparaît avec des informations beaucoup plus détaillées sur son parcours, ses expériences, ses compétences, ses informations de contact et les informations concernant le statut d'avancement du contrat. Pour pouvoir faire cela, j'ai dû également modifier la base de donnée mise à disposition et ai dû ajouter certains éléments.

La 2e mission a été effectuée en 1 semaine, elle consistait à créer le formulaire permettant de créer des profils dans la base de données. Car, pour l'instant, les profils dans la base de données que nous affichions étaient des profils créés aléatoirement en lançant une commande. Cependant, il fallait avoir la possibilité de pouvoir créer des profils directement sur l'application web, que l'utilisateur pourra utiliser pour créer les profils des candidats souhaitant postuler pour une société.

La dernière mission était une mission de recherche qui s'est déroulée les 2 dernières semaines de stage. Il fallait créer ou utiliser un modèle d'intelligence artificielle permettant de récupérer des éléments d'un texte, car nous voulions l'utiliser pour mettre en valeur des compétences dans un CV. Ce qui permettrait à l'entreprise de pouvoir lier des postes et des candidats en comparant les compétences nécessaires au poste avec celles acquises par celui-ci.

Début de stage

Formation

Au début des vacances de noël, 2 semaines avant le début du stage. M. Nahim BENBAHLOULI, ainsi que Florimond Jaulin, ancien stagiaire chez PELOPS et celui qui avait commencé à développer le POC de TAILOR, nous ont fait une présentation complète de la solution. L'objectif de cette présentation était de nous former à l'infrastructure du logiciel, car nous ne partions pas de rien, tout comme nous présenter les missions que chacun allait effectuer. De plus, il fallait recenser les technologies utilisées dans TAILOR que nous ne connaissions pas, pour pouvoir débuter une formation dans ces technologies.

Au cours de ce stage, j'ai donc suivis plusieurs formations :

- “React JS” pour la partie back et front de l’application

- “Spacy” pour utiliser un modèle de machine learning

- “Jupyter” pour avoir un IDE qui facilite l’utilisation de data et d’intelligence artificielle

Suivre ces formations m’a permis de mieux me préparer pour entamer ce stage, mais m’a également permis de découvrir des technologies que je ne connaissais pas et a également participé à l'émergence de nouvelles compétences.

Quotidien et ressenti

Les 2 premières semaines se sont passées principalement en distanciel, ce qui était contraignant car nous n'étions pas dans un environnement optimal au travail. Cependant, M. Nahim BENBAHLOULI a fait en sorte que nous puissions venir en présentiel 3 jours par semaine, le lundi, mardi et jeudi. De plus, je n'étais pas le seul stagiaire a PELOPS. Julien VANHAUWAERT et Benjamin WATSON étaient aussi avec moi.

L’ambiance était très bienveillante, et nous prenions plaisir à travailler ensemble. A chaque avancement majeur dans le projet, je sauvegardais mon avancement sur GitLab, un service web permettant le versioning du projet. J’ai créé une branche où mettre mes sauvegardes pour ne pas causer de conflits avec le code de mes collègues. Puis, je signalais mon avancement sur le service web “Trello”, permettant une simplification de la gestion de projet (annexe 1).

Plan

Tout au long de ce rapport, je reprendrai les différentes étapes clés qui ont permis de réaliser et de développer une version d’essai du logiciel TAILOR. À l’issue de ce rapport, j’apporterai ma vision ainsi que mon appréciation sur mon expérience personnelle et sur ce que j’ai acquis grâce aux différentes missions abordées durant ce stage.

Mise en place de l’affichage des profils candidats

Présentation

L’application TAILOR étant une application destinée aux ressources humaines, il est important d’y retrouver la liste des profils de candidats étudiés par ceux-ci. En effet, ce serait un gain de temps énorme d’avoir accès en un seul endroit à toutes les informations de candidats voulant rejoindre l’entreprise. Que ce soit les informations d'expériences, de compétences, ou de contact.

Le but à cela est d’une part faciliter le travail des ressources humaines en ayant tous les candidats en un seul endroit, et donc de gagner un temps conséquent, mais également d'avoir un suivis de leur état de recrutement, pour savoir en un coup d’oeil combien d’entretiens ils ont passé, et où ils en sont dans le recrutement. Par ailleurs, il ne faut pas seulement avoir une liste de candidats, mais également avoir la possibilité d’avoir une fiche d’informations précises sur celui-ci.

Pour cela, nous allons utiliser et modifier la base de données de TAILOR car pour afficher des candidats, il faut les récupérer quelque part et c’est bien sûr depuis une base de données, ici déjà présente.

La difficulté de la mission a été de gérer l'état d’avancement du recrutement des candidats. A ce stade du stage je n’avais encore jamais fait interagir le front-end et le back-end d’une manière aussi poussée et c'était donc une première.

Démarche

Avant de commencer à réfléchir sur la partie algorithmique de l'affichage des profils, il faut d'abord commencer par la modification de la base de données afin de pouvoir y stocker toutes les informations dont nous avons besoin. J'ai alors utilisé la maquette visuelle que M. Nahim BENBAHLOULI m'avait donné pour savoir quoi modifier car toutes les informations qu'il souhaitaient étaient sur cette maquette.

Les informations que je devais afficher étaient :

- Les informations générales du candidat sous forme d'une carte (Nom, Prénom, Age, Années d'expériences, Statut de l'avancement du recrutement sous forme d'une barre de progression)
- Une section sur les formations effectuées par le candidat et ses expériences
- Une partie sur les compétences acquises par le candidat, avec un système de notation sur 5 pour afin de savoir s'il est à l'aise avec ces compétences ou non
- Une section contact dans le but d'avoir tous les moyens disponibles pour échanger avec le candidat (Email, Téléphone, Adresse, lien Linkedin, lien Viadeo)
- Enfin une section où l'on voit plus en détail l'avancement de recrutement. (Informations de la prise de contact avec le candidat, Éléments du déroulement des entretiens, Savoir si le contrat a été signé ou non)

Utilisant un Framework pour faciliter la liaison entre le logiciel et la base de données (dont je détaillerai l'utilisation dans la prochaine partie), il fallait également modifier ce code pour pouvoir plus aisément modifier la base de données.

Ensuite, nous allons afficher les informations essentielles des candidats sous forme de cartes. Ces cartes seront automatiquement affichées pour chaque candidat dans la base de données. Lors du chargement de la page, le logiciel va aller chercher tous les candidats et leurs informations, et l'envoyer dans la partie front-end.

De plus, au niveau de la barre de progression du recrutement, Il y aura des icônes permettant de savoir où nous en sommes. Par exemple une icône au début pour signifier la prise de contact, une au centre pour les entretiens, et une à la fin pour la signature du contrat. De plus, 3 types d'entretiens seront mis en place :

- Entretien de motivation
- Entretien technique
- Entretien RH

Lorsque l'on clique sur cette carte de profil, une popup doit apparaître avec plusieurs onglets, ces onglets étant les informations des candidats cités précédemment, pour une meilleure organisation des informations.

Réalisation

Modification de la base de données

Avant de pouvoir afficher les profils, il faut d'abord que la base de données soit à jour avec ce que l'on veut récupérer. La base de données de TAILOR est faite sur "Postgresql". C'est le choix qu'a fait l'ancien stagiaire qui a débuté le projet.

Logo de marque de Postgresql

Pour mieux visualiser une base de données, il est préférable d'utiliser un dictionnaire de données. Cependant, nous n'en avons pas eu lors de notre stage. De ce fait, j'ai rédigé un dictionnaire de données pour les tables que je devais modifier (Annexe 2). Cela a été effectué avec le service web "Google sheets" car c'est un outil pratique à utiliser et les fichiers sont aisément partageables avec d'autres personnes

Logo de Google sheets

Cependant, j'ai eu beaucoup de mal à utiliser pgadmin (notamment pour visualiser les données), logiciel pour interagir avec une base de données postgresql. Je ne trouvais pas le logiciel assez intuitif. C'est pour cela que j'ai décidé d'utiliser DBeaver a la place, car je l'utilise pour chaque projet personnel, et je sais donc aisément l'utiliser.

Logo de PgAdmin

Logo de DBeaver

Après avoir fini le dictionnaire des informations à insérer dans la base de données, nous pouvons alors mettre en place cette modification. Pour cela nous allons utiliser "symfony".

Symfony est un framework PHP (langage informatique pour du back-end), c'est-à- dire un ensemble d'outils et de composants permettant de faciliter le développement et la mise en relation du back-end et du front-end. Dans le développement de TAILOR, symfony est principalement utilisé pour lier la base de données à l'application. En effet, c'est depuis symfony que la base de données est créée. Pour cela, nous allons utiliser des "Entités" dans symfony. Une entité peut être représentée comme une table dans une base de données. C'est également une

classe. J'ai effectué toute la partie programmation de mon stage sur le logiciel Webstorm car il permet une autocompletion fantastique sans devoir installer divers plugins, que je ne retrouve pas sur les autres IDE.

Logo de webstorm

Pour ajouter une colonne dans une base de données, il faut l'ajouter dans l'entité Symfony sous forme de variable.

```
private $Firstname;
```

Création d'une variable "Firstname"

Puis, il faut faire une fonction pour être capable de récupérer cette variable, et une autre capable de mettre une information dans cette variable (dans le cas ou on voudrait créer un profil)

```
public function getFirstname(): ?string
{
    return $this->Firstname;
}
```

Fonction permettant de récupérer ce qu'il y a dans la variable

```
public function setFirstname(string $Firstname): self
{
    $this->Firstname = $Firstname;

    return $this;
}
```

Fonction écrivant des informations dans la variable

Lorsque l'on a écrit les variables, nous pouvons utiliser une commande bien pratique à symfony qui va transformer toutes les entités en une base de données avec les tables correspondantes grâce a doctrine, un package symfony pouvant interagir avec des base de données. Les 2 commandes nécessaires sont :

- php bin/console make:migration (permet de préparer la transformation des entités en tables)

- php bin/console doctrine:migrations:migrate (envois les informations de modification de la base de données postgresql)

Nous avons alors notre base de données modifiée et prête à être utilisée. Cependant, pour pouvoir afficher des profils, il faut qu'ils soient dans la base de données. Nous n'avons pas encore la capacité de les insérer directement en remplissant un formulaire. C'est pourquoi j'ai utilisé un script mettant des informations aléatoires dans les différents champs des tables, pour avoir des profils à afficher. Ils ont pu être mis dans la base de données en utilisant l'api de symfony "API platforme". APi Plateforme est une API permettant d'envoyer des demande a la base de données directement depuis le code en passant par les entités.
De ce fait, nous pouvons dès à présent commencer l'affichage de ces informations.

Affichage des cartes de profils

L'affichage se fait en javascript grâce notamment au framework "React js" permettant de pouvoir créer des pages web dynamiques, c'est-à- dire que l'on a la capacité de pouvoir faire des actions sur une page web sans devoir recharger celle-ci (ex : cliquer sur un lien).

Logo de React JS

La maquette visuelle qui nous était fournie (annexe 3) m'a bien aidée à identifier les points importants et j'en ai donc fait un modèle à suivre pour l'affichage des profils. Cet affichage se fait dans le fichier "ProfileList". Dans ce fichier, la 1ere chose que l'on va faire est de demander à la base de données de nous envoyer tous les profils. Pour cela, il faut appeler l'api grâce a un lien http

```
// Recupere le resultat de la requete GET et la met dans la variable "profil"
useEffect( effect: () => {
    axios.get( url: "http://localhost:8000/api/profils").then((response : AxiosResponse<any> ) => {
        setProfil(response.data["hydra:member"]);
    });
}, deps: []);
```

Fonction permettant de récupérer des profils

Sur le code ci-dessus, on va appeler l'api pour qu'ils nous renvoient les profils sous forme d'une réponse json. Une réponse json est une réponse sous forme d'objet comme par exemple : { clé : valeur}. Puis, lorsque l'on a récupéré les informations, nous allons les enregistrer dans une variable "profil" grâce à "setProfil()".

Maintenant que nous avons récupéré les profils, comment les affichons-nous ? Pour cela, nous allons d'abord utiliser la fonction ".map()" qui permet d'effectuer une action pour chaque élément (et donc ici chaque profil). Cela va nous permettre de créer une carte pour chaque profil (annexe 4).

```
// Ajouter dans l'affichage
profils.push(
    <div className="card" key={item.Firstname} onClick={togglePopup}>
        <h3>{item.Firstname} {item.Lastname}</h3> {/*Affiche nom et prenom*/}
        <div><strong>{t("desiredJob")}</strong> : {item.Status}</div> {/*Affiche le status*/}
        <div><strong>{t("age")}</strong> : {ageSansDecimal}</div> {/*Affiche l'age*/}
        <div><strong>{t("yearsOfExperience")}</strong> : {yearExperience}</div> {/*Affiche les annees d'experiences*/}
        <div><strong>{t("recruitmentProcessus")} :</strong></div> {/*Affiche la barre de progression du recrutement*/}
        <div className="recruitmentprogress">
            <div className="progressbar">
                <ProgressBar percent={getPourcent(item)} width={2}/>
            </div>
            <div id="listIcon">
                {Icons()} {/*Affiche les icone se situant sur la barre de progression*/}
            </div>
        </div>
    </div>,
```

Code permettant l'affichage de carte de profil

Pour la barre de progression, j'ai utilisé les règles suivantes :

- S'il y a juste la prise de contact, la barre sera à 20 % et l'icône de contact sera en vert
- S'il y a entre 1 et 2 entretiens, la barre sera à 50% et l'icône d'entretien sera en orange pour signifier que c'est en cours.
- S'il y a 3 entretiens, la barre sera aussi à 50% mais l'icône est verte pour signifier que tous les entretiens ont été passés.
- S'il y a signature du contrat, la barre est à 100% et l'icône de contrat est verte.
-

Le code traduisant ces règles se trouve en Annexe 5.

Avec la liste des profils enfin finie, Il faut désormais mettre en place le système de popup permettant d'afficher des informations précises du candidat

Mise en place de popup profil

Pour mettre en place la popup, nous allons utiliser un composant de React ayant déjà été créé dans le code par un ancien stagiaire. Nous avions alors le “moule” de la popup, il ne restait qu'à l'utiliser pour y insérer les informations de profil.

Cependant, la popup a été créée de sorte à n'en utiliser qu’une seule par page, et non plusieurs car si l’on place plusieurs popup, lors du clic sur l’une d’entre elles, elle s’ouvrent toutes. de ce fait, j’ai modifié la manière d’afficher une popup.

Avant le changement, la popup avait un état “isOpen”. Si isOpen est “True”, la popup était ouverte. Si IsOpen était en “False”, elle se refermait. C’est pourquoi plusieurs popup les ouvraient en même temps.
Après le changement, chaque carte de profil a un numéro. Lorsque l’on clique sur un profil, la variable “isOpen” va récupérer ce numéro. Ensuite, la popup ayant le numéro de “isOpen” va s’ouvrir. Grâce à cela, on arrive à ouvrir une popup à la fois, comme le code ci-dessous le montre

```
// Lie la fonction togglepopup au profil (sinon toutes les popup s'affichent en meme temps)
function togglePopup() { // A chaque fois qu'on clique sur une carte de profil
    if (isOpen === 0) {  // Si la popup est fermée
        setIsOpen(item.id) // isOpen recois le numero du profil (ce qui permet de l'ouvrir)
    } else if (isOpen !== 0) { // Si la popup est deja ouverte
        setIsOpen(0) // Met isOpen a 0 ce qui referme la popup
    }
}
```

Code permettant l’assignation de numéro avec les popups

Nous avons alors les popups fonctionnelles. Il nous faut maintenant afficher tous les éléments du candidat dans ces popups.

En 1er lieu, la partie “CV” avec les formations et les expériences du candidat était relativement simple à mettre en place car il fallait juste utiliser du HTML (langage de base pour écrire dans un site web) (visuel sur l’annexe 6).

Pour la partie compétences, il faut afficher un nombre d'étoiles jaune équivalent à la note sur 5 des compétences. Le nombre restant d'étoiles étant des grises. Le code ci- dessous montre la partie du code permettant cela.

```
// Permet d'afficher le bon nombre d'etoiles en fonction du niveau
function getStars(skill, stars) {
    let starList = [];
    for (let i = 1; i <= 5; i++) { //boucle 5 fois
        if (i <= stars) { // si le numero de la boucle est plus petit que la note de la competence
            //Ajouter une etoile jaune
            starList.push(<img className="star" src={star} alt="" onClick={() => changeSkillLevel(skill.id, i)}/>)
        } else {
            //Sinon ajouter une etoile grise
            starList.push(<img className="star" src={greyStar} alt="" onClick={() => changeSkillLevel(skill.id,i)}/>)
        }
```

Code permettant d'afficher le bon nombre d'étoiles

Maintenant que nous avons affiché les étoiles, il faut pouvoir changer le niveau de la compétence en cliquant sur une d'entre elles. Pour cela, nous allons créer une fonction "ChangeSkillLevel" prenant en paramètre l'id de la compétence (permettant de l'identifier dans la base de donnée) et le numéro de l'étoile (si c'est la 1ere, la 2e, etc). Cette fonction va appeler la base de données et lui dire de modifier le niveau de compétences, toujours en utilisant l'api Plateforme précédemment mentionnée.

```
function changeSkillLevel(id, level){
    axios
        .put( url: "http://localhost:8000/api/skill_assignations/" + id,  data: { // appelle l'api pour changer le niveau
                                                                                  // de la competence
            Level: level  // Le niveau de competence que l'on a choisi
        })
    window.location.reload(false); // Recharger la page
}
```

Code modifiant le niveau d'une compétence

Cette section étant finie (visuel dans l'annexe 7), nous pouvons passer à la partie de contact. Cette partie, comme la partie CV, est essentiellement composée de code HTML, sans algorithme.

```
<div key={formation.id} className="formation-item">
    <div className="ligne"><strong>{t("school")} : </strong>{formation.School}</div>  {/* l'ecole*/}
    <div className="ligne"><strong>{t("diploma")} : </strong>{formation.Diploma}</div>{/* diplome*/}
    <div className="ligne"><strong>{t("field")} : </strong>{formation.FieldStudy}</div>{/* domaine */}
    <div className="ligne"><strong>{t("dateStart")} : </strong>{start}</div>{/* date de debut de la formation */}
    <div className="ligne"><strong>{t("dateEnd")} : </strong>{end}</div> {/* date de fin de la formation */}
    <div className="ligne"><strong>{t("description")} : </strong>{formation.Description}</div> {/* description  */}
    <div className="separateur"/> {/* Permet de faire une barre horizontale */}
</div>)
```

Code affichant les informations de contact (visuel dans l'annexe 8)

Enfin, il ne nous reste plus que la partie de l'état de recrutement du candidat. D'abord, nous avons vu que l'état d'entretien d'un candidat pouvait être en cours (icone orange) ou terminée (icone verte). Cependant, il peut également être bloqué. Cela peut arriver quand un candidat a certains problèmes personnels et ne peut plus effectuer d'entretiens temporairement. Pour cela, j'ai ajouté un simple bouton "isblocked" permettant de rapidement signaler si un profil est bloqué au niveau des entretiens ou non. Lorsque l'on coche celui-ci, le status "isBlocked" des entretiens passe à "True", et l'icône apparaît en rouge.

Carte du profil lorsque le processus des entretiens est bloqué

Dans un second temps, nous afficherons les éléments de la prise de contact (date de prise de contact et méthode de prise de contact). De même pour la signature du contrat, il est constitué de la date de signature et d'informations complémentaires.

Enfin, pour le processus de recrutement détaillé, nous avons pour chaque entretien le type (motivation technique et RH), la date de l'entretien, l'avis global sur le déroulement de l'entretien et sur le candidat, la description, et la date du prochain entretien. Pour cela, je crée 3 blocs qui ne seront pas affichés et seront au même endroit. Chaque bloc correspond à un type d'entretien. A côté il y a 3 icônes d'entretien, une pour chaque type. Si nous avons dans la base de donnée un entretien de motivation, la 1ere icône sera en vert. L'entretien technique est la deuxième icône et celui des RH est la troisième. Lorsque l'on clique sur une icône en vert, les informations de cet entretien s'affichent.

Section des entretiens

Puis, si l'on clique sur une icône d'entretien blanche, un texte apparaît pour nous dire que nous n'avons pas encore fait d'entretien.

J'ai réussi cela en créant 2 classes CSS. La 1ere étant la classe "displayed" qui permet l'affichage du bloc, et la deuxième étant "nodisplayed" qui fait disparaître le bloc en question. Lorsque l'on clique sur une des 3 icônes, cela va lancer une fonction "switchInterviews". Cette fonction cache tous les blocs sauf celui lié à l'icône cliquée.

```
function switchInterviews(divId) { // id du bloc que l'on veut afficher
    document.getElementById( elementId: "1").style.display = "none" //
    document.getElementById( elementId: "2").style.display = "none" // Efface tous les blocs
    document.getElementById( elementId: "3").style.display = "none" //
    document.getElementById(divId).style.display = "block" // Affiche celui que l'on veut
}
```

Code permettant de changer d'entretien

Avec cela, nous avons dès lors fini l'affichage des informations des candidats. Cependant, à ce stade de développement, les informations des candidats dans la base de données sont mises aléatoirement. Un début de formulaire avait déjà été réalisé par le précédent stagiaire, j'en ai donc utilisé une partie et l'ai terminé.

Mise en place du formulaire de candidat

Présentation

La mise en place d’un moyen de création de candidat est primordiale pour le bon fonctionnement de l’application. En effet, comment ajouter de nouveaux candidats dans la base de données si nous ne pouvons pas avoir un moyen d’enregistrer ses informations ? Car pour l’instant nous avons bien des informations dans la base de données, mais elles ne sont pas réellement utilisables, car étant des données aléatoires. C’est pour cela qu’il faut créer un formulaire permettant de recenser des personnes existantes.

Le but de cette mission est de faciliter la mise en ligne des informations du candidat, tout en gardant une trace de celui-ci. Pour cela, je ne vais pas le créer à partir de rien, mais je vais utiliser le début de formulaire créé par un ancien stagiaire. De plus, je n’ai plus à modifier la base de données, ce qui va me permettre de gagner du temps.

La difficulté de cette mission a été de reprendre un début de formulaire déjà existant et de devoir le manipuler afin d’y insérer les informations que je voulais.

Démarche

Tout d'abord, afin d'effectuer les modifications sur le début de formulaire déjà existant, il faut comprendre comment celui-ci fonctionne. Car avant ce stage, j’avais un peu utilisé "React js" et donc effectuer un formulaire avec cette technologie était nouveau pour moi. Après diverses manipulations pour expérimenter la mise en place du formulaire, je compris comment cela fonctionnait.

Il fallait donc maintenant préparer le formulaire. Celui-ci doit avoir des parties similaires à l'affichage des informations des candidats. Ce qui était déjà effectué dans ce formulaire étaient les informations de base du candidat (toutes les informations de la table “profil” de la base de données) et les assignations des compétences. De ce fait je devait mettre en place les parties du formulaire pour :

- Les Formations
- Les Expériences
- L'état de recrutement

Pour faire cela, j’utilise toujours le logiciel Webstorm car c’est dans la continuité de ma première mission.

Réalisation

Formulaires “formation” et "expérience"

Ayant déjà la partie informations du profil (annexe 9), nous pouvons attaquer la partie formulaire de formations. La partie “formations” et "expériences" des candidats sont identiques. De ce fait, je n’expliquerai que la méthode pour la partie "formations". Pour celle-ci, nous allons d’abord créer un objet formation vide. Nous allons lui donner les noms des variables de l’objet formation, mais toutes ses variables seront vides. Nous allons associer ces variables au formulaire, que nous enverrons ensuite dans une requête pour remplir la base de données.

```
const addEducationField = () => {
  educationsFields.push({ // Creation d'un objects avec divers elements
    school: "",
    diploma: "",
    fieldStudy: "",
    dateStart: "",
    dateEnd: "",
    description: "",
  });
};
```

Code de l’objet formation vide

Après avoir fait cela, nous allons mettre en place les champs pour écrire nos informations. Ces champs sont composés d’un label, d’un nom (pour le reconnaître au niveau du code), d’un placeholder (le texte qui va apparaître quand rien n’est écrit dans le champs), d’une valeur (en l'occurrence ici nous allons mettre 0 car on va écrire les valeurs nous même), d’un type (“text” si c’est du texte, “date” si c’est une date, “checkbox” si c’est une case à cocher, etc).

```
<Field
    label={t("company")}
    name="company"
    placeholder={t("company")}
    value={field.company}
    onChange={handleChange}
    type="text"
/>
```

Code du champs pour remplir une information

De plus, l'application va afficher un formulaire "formation" pour chaque objet formations dans la variable "educationFields". Pour l'instant nous n'avons qu'un objet donc il n'y aura qu'un formulaire de formation, mais cela nous sera utile plus tard lors de la possibilité d'ajouter une multitude de formations.

Sur l'image ci-dessus, nous avons également une partie "onChange". "onChange" va effectuer une action a chaque vois que l'on va écrire dans ce champ. Ici, ce qu'il va effectuer est une fonction enregistrant ce qu'il y a dans le champ vers la variable vide correspondante (une de celle que l'on a mis dans la classe education).

```
const handleChange = ({ currentTarget }) => {
    const { name, value } = currentTarget; // Recupere le nom et la valeur du champ
    field[name] = value; // Assigne la valeur a la variable souhaitée
};
```

Code enregistrant la valeur d'un champ

Ayant fait cela, il ne nous restait plus qu'à mettre en place de moyen d'avoir plusieurs formations ou expériences car en effet, il se peut qu'un candidat ait été dans plusieurs écoles ou entreprises.

A cette fin, j'ai fait un bouton permettant d'ajouter une nouvelle formation ou expérience. Lorsque ce bouton est cliqué, il appelle une fonction "addEducationField()" (expliquée précédemment) qui crée un 2e objet education distinct du 1er que l'on a fabriqué précédemment. De ce fait, nous avons maintenant 2 objets "formations". Étant donné que le programme affiche un formulaire par formation, il va alors en afficher un deuxième. Et ainsi de suite pour le nombre de formations (ou d'expériences) que l'on souhaite.

```
<span
  className="badge badge-secondary button btn-success"
  onClick={() => addEducationField()}
>
  +
</span>
```

Code du bouton permettant l'appel de la fonction "addEducationField"

L'ajout de ceux-ci a été facilité par le fait que chaque champ de texte est unique, et donc si nous avons 2 champs du même nom (ex 2 champs "ecole"), le formulaire ne se trompera pas dans l'envois des réponses, il sait dire quelle information appartient à quel champ.

Formulaire de recrutement

Ce formulaire est divisé en 3 parties. Tout d'abord les parties “contact” et “contrat”, et de l’autre la partie des entretiens.

Les parties contacts et contrats sont relativement similaires.
Il y a une case à cocher pour savoir si oui ou non la prise de contact ou la signature de contrat a été effectuée, puis il y a un champ de date pour le contact et le contrat et de même pour un champ description. En plus de cela, il y a un champ pour indiquer la méthode de prise de contact.

```
<Field
    label={t("contact")}
    labelClassName="form-check-label"
    inputClassName="form-check-input"
    name="contact"
    onChange={handleChange}
    checked={field.contact}
    value={field.contact}
    type="checkbox"
/>
```

Code d’une case à cocher

Enfin, la dernière partie sera la mise en place du formulaire des entretiens. Pour cela nous allons afficher 3 cases à cocher. Chaque case sera liée à un type d’entretien (motivation, technique et rh). Lorsque l’on coche une case, un petit formulaire apparaît pour l’entretien correspondant. Nous allons créer une fonction permettant cet affichage. Par exemple, si l’on coche la case “rh”, la fonction “DisplayRh” va se lancer. La fonction créer un objet “rh” ayant comme variables le type de l’entretien (ici “rh”), la date de l'entretien, l’avis, la description et la date du prochain entretien.

```
const DisplayRh = () => {
    setIsRh(!isRh); // Motifie la variable "isRh" par son opposé (true ou false)
    if (isRh === false){
        interviewsFields.push({ // Creer un objet rh avec des champs vides
            interviewType: "rh",
            dateMeeting: "",
            review: "",
            description: "",
            dateNextInterview: "",
        });
    }else{
        interviewsFields.pop() // Supprime l'objet
    }
}
```

Code de la fonction “DisplayRh”

La manière de procéder est exactement la même pour les entretiens de motivation et techniques.

Après avoir mis en place d’affichage d’une liste de profils de candidats, la vue détaillée de ceux-ci et le formulaire permettant d’en ajouter dans une base de données, nous avions terminé l’ensemble des missions tournant autour des profils.

Récupération des compétences d’un CV grâce à l’intelligence artificielle

Présentation

Comme énoncé précédemment, TAILOR est un logiciel ayant comme but de faciliter le travail des ressources humaines d’une entreprise partenaire. C’est pour cela qu’une des idées de M. Nahim BENBAHLOULI était de créer ou d’utiliser un modèle d’intelligence artificielle de traitement de texte.

Cette intelligence artificielle aurait pour but de récupérer un cv et d’y extraire les compétences du candidat. Ce qui à long terme peut faire gagner un temps considérable car le personnel des ressources humaines n’auraient plus besoin de mettre toutes les compétences à la main.

De plus, à l'avenir, cette intelligence artificielle pourrait être utilisée pour d’autres moyens, comme par exemple remplir une fiche candidat automatiquement. Uniquement grâce aux informations contenues dans un CV.

Démarche

Lorsque j’ai commencé cette mission, je ne connaissais pas grand-chose à l'intelligence artificielle. Je ne savais donc pas exactement où commencer.
Cependant, cela fait un an que je m'intéresse à ce champ et que j’apprends de mon côté la “data science” (science des données), premier pas pour faire de l’intelligence artificielle car celle-ci repose exclusivement sur de la donnée.

C’est alors que M. Nahim BENBAHLOULI m’a envoyé des liens de quelques projets github (3) permettant de récupérer des compétences d’un CV avec l'aide de l’intelligence artificielle. Ces projets github étaient apparemment très connus, compte tenu du nombre d'étoiles (les étoiles sont sur github ce qui permet de savoir le nombre de personnes qui aiment ou suivent le projet. Cela peut être comparé aux “j’aimes” de twitter ou instagram).

Les projets d’intelligence artificielle se faisant quasi tous en langage “Python”, je peux donc regarder la structure du code des projets donnés, et comprendre la plupart des choses qui sont inscrites, car je connaissais déjà ce langage. Cependant, en regardant ceux-ci, je me suis vite rendu compte qu’ils allaient être compliqués à utiliser. De plus, l’en d’entre eux n'étaient pas vraiment un projet d’intelligence artificielle, mais plus un guide de comment bien écrire du code dans un

projet d'IA. L'un des projets était composé de centaines de fichiers, et en passant quelques heures à parcourir un bon nombre d'entre eux, je compris que ce projet était lié à un autre, et qu'il fallait probablement les lier pour que cela fonctionne. J'ai donc abandonné l'idée d'utiliser celui-ci. Enfin, le dernier lien github donnait également vers un projet d'intelligence artificielle. Cependant, également en regardant dans les fichiers du projet, il n'y avait pas de trace de récupération de mots depuis un texte. Cela était visible car aucun morceau de code ne demandait de récupérer un fichier texte ou un PDF.

Je me suis alors tourné sur la création d'une IA pour faire ce dont j'avais besoin. Cependant, n'ayant jamais fait d'intelligence artificielle auparavant. Après quelques recherches sur comment récupérer du texte grâce à une IA, j'ai trouvé un modèle d'intelligence artificielle python utilisant le système "OCR" (optical character recognition). Le système "OCR" permet au modèle d'IA de contextualiser le mot qu'il récupère. Ce qui est extrêmement important pour avoir de bons résultats. Ce modèle porte le nom de "spaCy" et est beaucoup utilisé quand il s'agit de récupérer du texte gravé à l'intelligence artificielle.

Avec l'approbation de mon tuteur de stage, j'ai entrepris une brève formation du modèle "spaCy" pour pouvoir récupérer des compétences d'un texte. Cette formation a duré trois jours et m'a permis de mieux comprendre le fonctionnement et les cas d'utilisations du modèle spaCy.

Avant de pouvoir commencer a mettre ce modèle en place, il ne me restait plus qu'une chose manquante, les données. En effet, pour faire fonctionner une IA, nous avons besoin de beaucoup de données, pour que celle-ci puisse s'entraîner beaucoup plus et donner de meilleurs résultats. Ne pas avoir assez de données peut avoir comme conséquences d'avoir des résultats faussés. Il me fallait donc un grand nombre de CV. Cependant, je ne pouvais pas les créer moi même car cela me prendrait un temps monstrueux. De plus, créer un système pour récupérer des profils linkedin en CV aurait été trop long a faire par rapport au temps qu'il me restait.

En plus de cela, il fallait que pour chaque CV, il y ait une partie a la suite de celui-ci, identifiant toutes les entités dans le CV (nom, competence, lieu, organisation, etc) en écrivant le numéro du caractère de début (ex : 10 si le 10e caractère du CV est la 1ere lettre du mot), le caractere de fin de l'entité, et le type (nom, competence, etc). J'ai alors trouvé sur le net un fichier avec 200 CV en anglais avec à la fin de chacun d'entre eux une liste des entités que j'allais finir par utiliser.

```
{'entities': [(210, 251, 'Skills'),
  (189, 200, 'College Name'),
  (165, 187, 'Degree'),
  (117, 126, 'Companies worked at'),
  (103, 116, 'Designation'),
  (42, 85, 'Email Address'),
  (13, 20, 'Location'),
  (0, 12, 'Name')]})
```

Exemple d'entités dans un CV

Ayant maintenant toutes les cartes en main, je pouvais débuter la récupération des compétences d'un CV grâce à une IA.

Réalisation

En premier lieu, avant de débuter l'utilisation de spacy, il faut choisir son environnement de travail. En effet, les programmes de data science ou d'IA ne se font pas pour la plupart dans des outils traditionnels (vsCode, Pycharm, etc), mais dans ce qui permet de générer des notebook (Programmes contenant à la fois du texte et du code). Le plus connu, et également celui que j'ai utilisé pour cette mission est "Jupyter notebook". C'est une application web permettant de créer et gérer des notebook. J'ai décidé d'utiliser celui-ci car je l'ai déjà manié dans le passé, pour des projets personnels.

Logo de Jupyter Notebook

Après avoir lancé jupyter notebook et créé un fichier, nous pouvons installer 3 composants :

- spaCy ("pip install spacy" pour l'installer), étant le modèle d'IA
- pickle ("pip install pickle"), paquet python pour ouvrir facilement des fichiers
- random (déjà installé par défaut), paquet python pour générer des nombres aléatoires.

Puis, il faut récupérer dans une variable “train_data” les CV que l’on a trouvés. Nous utilisons à cette fin “pickle” avec sa fonction “load”

```
#Recuperation des CV dans la variable "train_data"
train_data = pickle.load(open('train_data.pkl', 'rb'))
```

Code affectant les CV a une variables

Une fois les dépendances installées et les CV récupérés, nous pouvons créer notre modèle spaCy et choisir la langue avec laquelle il va travailler. Comme nous avons des échantillons de CV anglais, nous allons indiquer à spaCy que nous travaillons en anglais. Pour cela, nous allons l’indiquer dans la fonction “blank” de spaCy, pour qu’il crée un modèle d’IA utilisant l’anglais, que nous allons mettre dans une variable “nlp”.

Pour que spaCy puisse trouver des mots en lien avec un contexte, nous allons alors utiliser spaCy avec son pipeline “NER”. Une pipeline est un ensemble de tâches informatiques. Cette pipeline est une reconnaissance d’entité, permettant de d’identifier et classer des entités (lieu, organisation, compétences). Parfait pour ce que nous voulons faire.

```
# Creation d'un modele spaCy en anglais
nlp = spacy.blank('en')

# Ajoute la pipeline "NER" au modele spaCy
ner = nlp.add_pipe("ner")

# on affiche toutes les pipelines actuelles pour savoir
#    si la pipeline que l'on a ajouté est bien presente
nlp.pipe_names
```

```
["NER"]
```

Code crean le modèle spaCy et ajoutant la pipeline “NER”

Maintenant, il est temps d'entraîner notre modèle. Nous commençons donc par dire à notre modèle que pour chaque CV, dans la liste des entités du texte, le 3e élément représente le type de l’entité (le premier et deuxième élément représentant la place de la 1ere lettre et de la dernière lettre de l'entité).

```
# '_' car on a un tuple donc on devrait mettre 2 elements dans le for, mais on veut que annotation et pas le texte
for _, annotation in train_data:
    for ent in annotation['entities']:
        # On ajoute touts les elements de la partie "entites" dans la pipeline un par un
        # Pour chaque objet dans la pipeline, on indique que le label est l'element en 3e position
        ner.add_label(ent[2])
```

Code présentant les entités du texte au modèle spaCy

Ensuite, nous regardons s'il y a d'autres pipelines que "NER" dans le modèle spaCy. S'il y en a, nous les retirons du modèle. Nous faisons cela surtout en bus de gain de performance, car s'il y a d'autres pipelines, il va les faire tourner a vide et donc utiliser de la performance pour rien.

```
# On regarde si on a d'autres pipelines que 'ner'
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']

#S'il y en a d'autres alors on les enlève (pour alléger)
with nlp.disable_pipes(*other_pipes): # only train NER
```

Code retirant les pipelines en trop

Puis, nous lançons l'entraînement de l'IA. Le but est que pour chaque Cv, l'IA regarde quelles entités apparaissent à quels endroits, et les sauvegarde pour s'en "rappeler".

Nous choisissons ensuite combien de fois nous allons entraîner l'IA. A chaque fois, nous allons demander d'afficher "Starting Itération + le numéro de la boucle" pour savoir a quel nombre d'entraînements nous sommes. Nous allons ensuite mélanger les CV pour que l'endroit du CV dans la liste n'impacte en rien les décisions de l'IA.

Pour chaque CV dans la liste, l'IA va donner ses prédictions sur quel mot / entité du CV est de quel type. Puis nous lui donnons la liste des entités du CV pour qu'il sache ou est-ce qu'il a eu faux, et apprendre. Pour cela nous utilisons la fonction "update". Dans le cas où il y aurait un problème pour lire un CV, nous ordonnons au programme de continuer et de passer ce CV. (Code entier pour faire cela en Annexe 10)

Nous lançons ensuite l'entraînement de l'IA. Entraîner une IA demande du temps. Au plus nous avons de la data à lui transmettre, au plus il met de temps à s'entraîner. Dans mon cas, en ayant 200 CV à analyser, cela a pris entre 40 et 45

minutes. Cependant, dans de gros projets ayant des milliers voir des centaines de milliers de data, cela peut prendre des jours a entraîné une IA.

Le modèle spaCy étant maintenant entraîné, nous pouvons utiliser un CV pour tester les résultats. Nous avons cependant une contrainte. Les CV utilisés pour entraîner l'IA étaient en anglais. De ce fait, il faut utiliser un CV écrit en anglais. Si nous envoyons un CV écrit dans une autre langue à l'IA, cela va fausser les résultats. Nous allons alors utiliser un CV "Échantillon" (un faux CV que l'on va utiliser pour tester l'IA). Ici, notre échantillon sera le CV nommé "Alice Clark".

Cependant, ce CV est en pdf. Nous allons alors transformer ce pdf en texte et le nettoyer (lorsque l'on passe un fichier pdf en texte, il y a des irrégularité dans les espaces, les remises a la ligne, et il peut y avoir également des "•" ou des tirets) pour avoirs le moins de choses qui pourrait altérer les décisions de l'IA

```
fname = 'Alice Clark CV.pdf'
doc = fitz.open(fname) # Ouvrir le fichier
text = ""
for page in doc:
    text = text + str(page.get_text()) # Transforme le pdf en texte
tx = " ".join(text.split('\n')) # supprime les nouvelles lignes pour avoir un block de texte
tx = " ".join(text.split('•')) # supprime les gros poinrts pour nettoyer le texte
tx = " ".join(text.split('-')) # supprime les tirets poinrts pour nettoyer le texte
print(tx)
```

Code transformant un pdf en texte et le nettoyant

Enfin, nous envoyons ce texte au modèle spaCy, puis nous lui demandons d'afficher ses résultats, avec d'un côté le label de l'entité, et de l'autre le texte qui correspond

```
doc = nlp_model(tx)
for ent in doc.ents:
    print(f'{ent.label_} - {ent.text}') # affiche les elements avec ce qu'ils signifient (ex : "Microsoft - Compagnie")
```

Code affichant les résultats de l'IA

Voici les résultats obtenus avec ce CV échantillon :

```
NAME                          - Alice Clark
LOCATION                      - Delhi
DESIGNATION                   - Stream Analytics
DESIGNATION                   - Software Engineer
COMPANIES WORKED AT           - Microsoft –
DEGREE                        - Indian Institute of Technology – Mumbai
SKILLS                        - Machine Learning, Natural Language Processing,
and Big Data Handling    ADDITIONAL INFORMATION  Professional Skills  • Excelle
nt analytical, problem solving, communication, knowledge transfer and interpers
onal
```

Comme on peut le voir, nous avons reçu de bons résultats même si le nombre de CV pour entraîner le modèle n'était pas énorme. Nous n’avons pas obtenu un résultat parfait car il manque certaines compétences ou encore les anciennes entreprises de la personne. Nous pouvons améliorer les résultats de 2 manières :

- Ajouter plus de CV au fichier (200 n’est pas assez)
- Augmenter le nombre d'entraînements (Ce qui va faire passer le temps d'entraînement à plusieurs heures voir plusieurs jours)

Enfin, nous aurions pu aller plus loin, comme attribuer des compétences à un métier, et avec les compétences tirés d’un CV, afficher un % d’affinité avec de métier.

Évaluation des réalisations et des compétences mobilisées

Adéquation du travail

Lors de ce stage, toutes fin de semaine, nous faisions un point avec M. Nahim BENBAHLOULI sur l'avancement du projet. Il nous demandait de parler de ce qu'on avait produit, de noter sur 5 la satisfaction que nous avions par rapport au travail fourni et de lister nos objectifs de la semaine suivante. Ces discussions étaient extrêmement positives pour notre productivité, car cela pourrait nous permettre de donner un avis reculé sur le travail fourni, pour essayer de s'améliorer la semaine suivante.

De plus, les aides fournies étaient d'une très grande aide. Les maquettes visuelles m'ont permises de pouvoir fournir un travail qui était en accord avec la vision de son créateur, et les liens des githubs pour l'intelligence artificielle m'ont aidé à m'aiguiller pour une piste de recherche.

Compétences mises en oeuvre

Au cours de ce stage, j'ai pu démontrer certaines de mes compétences et en développer d'autres.

Lorsque l'on travaillait en groupe, il m'arrivait de prendre naturellement certaines décisions et de les partager avec les autres stagiaires ou encore de les aider lorsqu'ils avaient des difficultés. Cela a énormément développé ma gestion d'équipe ou encore la communication.

Mes connaissances en React m'ont permis d'être rapidement efficace au début de ce stage, en n'ayant besoin d'une piqûre de rappel au lieu d'une formation à part entière. Ce qui m'a fait gagner un temps considérable pour le développement de ce projet.

De plus, les recherches effectuées pour l'utilisation d'un modèle d'intelligence artificielle m'a fait prendre conscience de la largeur du sujet, en plus de la complexité de la chose. Cependant, avec des recherches pertinentes, il est possible de trouver des solutions à des problèmes en apparence insurmontables, qui plus est si nous ne maîtrisons pas le sujet.

Dans le futur, pour devenir ingénieur en intelligence artificielle, je vais devoir accentuer mes recherches sur comment sont créés les modèles d'intelligence

artificielle que l’on utilise. Faire des recherches sur les réseaux de neurones ou les arbres décisionnels par exemple.

Cependant, j’ai réalisé un objectif que je me suis fixé durant le stage de première année : améliorer ma communication avec les autres. En effet, l'année dernière je n’arrivais pas à prendre une équipe en main, je ne faisais qu'acquiescer des décisions prises par les autres. Cette année, j’ai pu prendre la parole et surtout des décisions pour l'équipe ou encore aider à résoudre des problèmes en donnant des conseils afin que les autres puissent s'améliorer.

Conclusion

En conclusion, ce stage a été une chance de bien des manières.

En premier lieu, nous avons été accueillis dans l'entreprise comme si nous n'étions pas des stagiaires mais plutôt comme des employés travaillant depuis un certain temps dans l'entreprise. De plus, ils nous ont fait confiance sur nos décisions pour l'avancement de l'application et les ont respectées. Nous sentions que le personnel nous faisait confiance et qu'il était à l'écoute de nos propositions. Cela s'est sûrement ressenti dans notre productivité et dans les résultats que nous avons fournis.

De plus, après avoir fini les 2 premières missions de mon stage, M. Nahim BENBAHLOULI, connaissant mon intérêt envers l'IA, m'a proposé de faire une mission en rapport avec l'intelligence artificielle. Effectuer une mission en entreprise en lien avec notre projet professionnel est une expérience formidable. J'ai pu de ce fait développer des compétences qui me serviront au niveau professionnel et j'en suis reconnaissant.

Ensuite, grâce aux missions réalisées, j'ai pu, pour la première fois, utiliser 2 frameworks en même temps (React JS et Symfony). Je les ai toujours utilisés dans les différents projets diffèrent (soit l'un, soit l'autre), mais jamais dans le même projet. Grâce à cela j'ai pu découvrir comment cohabitent 2 frameworks et comment ils se complétaient. A l'avenir, j'utiliserai plus souvent ce genre de système.

Les stages étant finis, mon prochain objectif est de trouver une alternance dans une entreprise avec une équipe de data scientists dans le Big Data (Le Big Data est l'ensemble des données, nécessitant plusieurs machines pour les traiter). J'aimerai, au cours de mon année d'alternance, développer mes compétences en data science (la data science est un terme employé pour désigner le fait d'utiliser des données acquises, pour répondre à des besoins d'une entreprise), pour ensuite m'orienter dans le machine learning en entreprise, car travailler dans le machine learning est mon but professionnel. Dans le futur, la quasi-totalité des entreprises seront amenées à utiliser des modèles d'intelligence artificielle pour résoudre des problèmes qui n'auraient pas pu être résolus sans intelligence artificielle. C'est une des raisons pour laquelle j'aimerai travailler dans ce secteur.

Glossaire

API : Ensemble normalisé de méthodes servant de façade par laquelle un logiciel offre des services à d'autres logiciels.

Backend : PArtie du code s'occupant de la liaison à la base de données

Base de données :
Stocke des données sous forme de tables, pouvant être reliées entre elles.

Candidat : Personne voulant rejoindre un poste

Collaborateur :
Candidat à un poste ayant été embauché

Data science :
C'est la discipline permettant à une entreprise d'explorer et d'analyser les données brutes pour les transformer en informations précieuses permettant de resoudre des problemes de l'entreprise

Dictionnaire de données :
Un dictionnaire de données est une liste où est regroupée l'entièreté des données. On y inscrit également toutes les informations de chaque élément.

Frontend : Partie du code s'occupant du visuel

Offboarding : Désigne la période précédant le départ, souhaité ou non, d'un salarié

Onboarding : Consiste à familiariser vos nouvelles recrues avec tous les éléments nécessaires pour accomplir ce pour quoi elles ont été embauchées

PELOPS : Logiciel créé par TAILOR

TAILOR : Entreprise dans laquelle j'ai effectué mon stage

Bibliographie et webographie

- Apprendre SpaCy :
    - https://spacy.io/usage
    - https://www.youtube.com/watch?v=dIUTsFT2MeQ&t=9460s

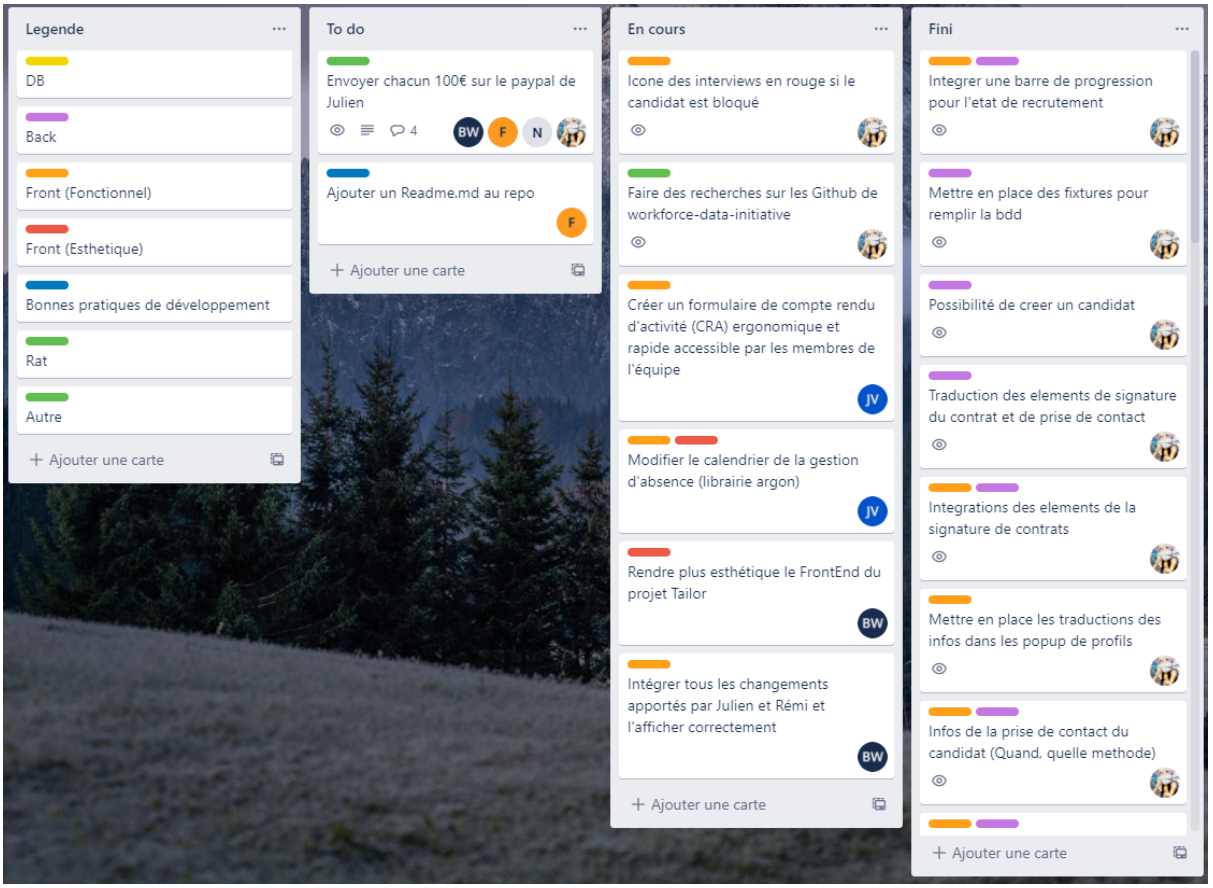
- Apprendre le React js
    - https://www.udemy.com

- Chercher des solutions à des problèmes communs :
    - https://stackoverflow.com

Annexe

Annexe 1 - Image du Trello

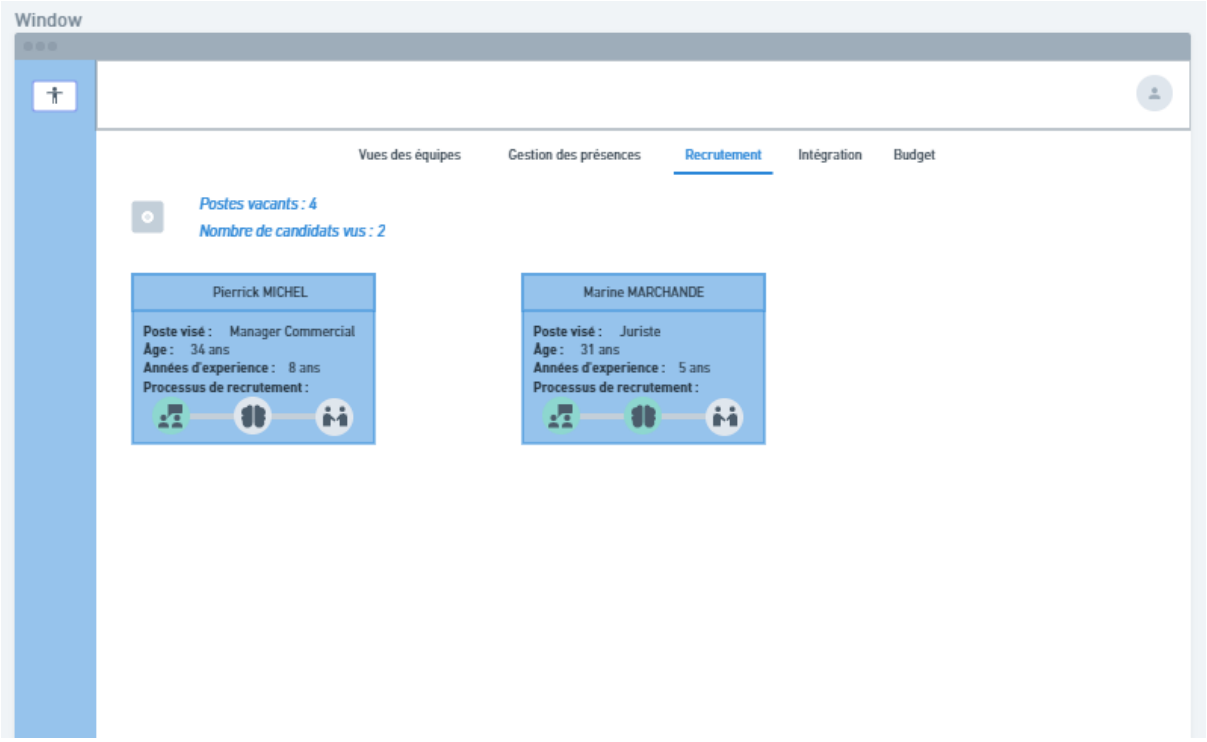
—---



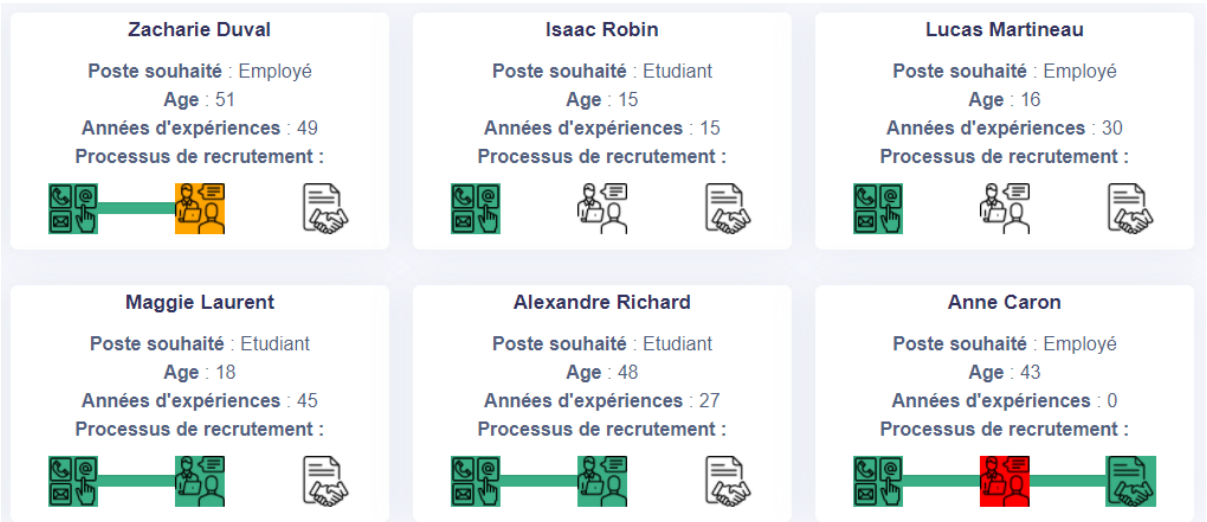
Annexe 2 - Dictionnaire des tables à modifier

Profil								
	id	id du profil	int	11				
	utilisateur_id	id de la personne ayant ajouté le profil	int	11				
	lastname	nom de famille	varchar	60				
	firstname	prenom	varchar	60				
	datebirth	date de naissance	date					
	status	status professionnel	varchar	60				
	address	adresse postale	varchar	60				
	phone	Numero de telephone	varchar	60				
	email	Adresse email	varchar	60				
	linkedin	Lien linkedin	varchar	255				
	viadeo	Lien viadeo	varchar	255				
	biography	biographie	varchar	255				
	is_searching	en train de chercher un travail ?	boolean	1				
	is_executive	en train de travailler ?	boolean	1				
	is_handicap	est handicapé ?	boolean	1				
	is_working_visa	a-t-il un visa de travail ?	boolean	1				
	is_working_visa_ok	Visa de travail valide ?	boolean	1				
	tjm	Taux journalier moyen	int	11				
	actual_remuneration	Remuneration actuelle	int	11				
	remuneration_wanted	Remuneration voulue	int	11				
Recruitment								
	id	Id du recrutement	int	11				
	id_profil	Id du profile dont le recrutement est lié	int	11				
	contact	Y a til eu prise de contact ?	boolean	1				
	contact_date	Date de la prise de contact	date	255				
	contact_method	Methode de la prise de contact	varchar	255				
	contact_description	Description de la prise de contact	varchar					
	contract	Y a til eu signature de contrat ?	boolean	1				
	contract_date	Date de la signature du contrat	date					
	contract_description	Description du contrat	varchar	255				
	blocked_state	Le recrutement est il bloqué ?	boolean	1				
Interview								
	id	Identifiant de l'entretien	int	11				
	id_recrutement	identifiant du recrutement	int	11				
	interview_type	type de l'entretien	varchar	60				
	date_meeting	date de l'entretien	date					
	review	avis	varchar	255				
	description	description de ce qu'il s'est dit	varchar	255				
	date_next_interview	date du prochain entretien	date					

Annexe 3 - Maquette visuelle de la mission



Annexe 4 - Affichage des cartes de profil



Annexe 5 - Code de la barre de progression du recrutement.

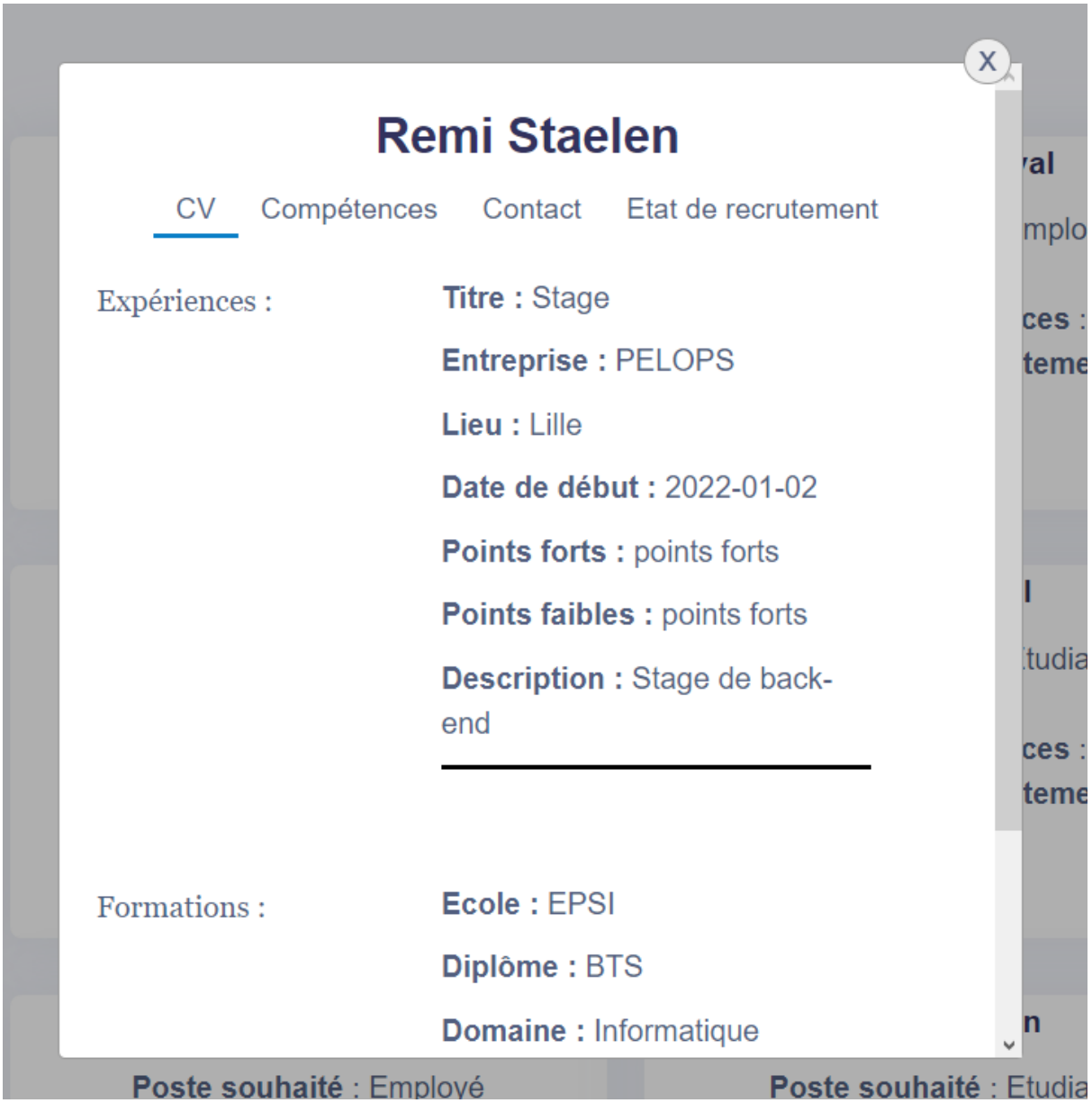
```
// Mettre le fond des images en vert si l'etat de recrutement est poussé :
function Icons() {
    let listIcon = []
    if (item.recruitment.Contact) { //Si on a le contact
        //icone de contact verte
        listIcon.push(<div><img id="contactIcon" className="enabled" src={iconCommunicate} alt=""/></div>)
    } else {
        // Sinon icone de contact rouge
        listIcon.push(<div><img id="contactIcon" src={iconCommunicate} alt=""/></div>)
    }

    if (item.recruitment.BlockedState){ //Si le recrutement est bloqué
        // Icone d'entretien rouge
        listIcon.push(<div><img id="interviewIcon" className="blockedImage" src={iconInterview} alt=""/></div>)
    }else{
        if (item.recruitment.interviews.length === 0) { // S'il n'y a pas d'entretien'
            // Icone d'entretien blanche
            listIcon.push(<div><img id="interviewIcon" src={iconInterview} alt=""/></div>)
        } else if (3 > item.recruitment.interviews.length > 0){ // S;il y a plus d'un entretien
            // Icone d'entretien verte
            listIcon.push(<div><img id="interviewIcon" className="inProgress" src={iconInterview} alt=""/></div>)
        }else{
            // ISinon icone d'entretien orange
            listIcon.push(<div><img id="interviewIcon" className="enabled" src={iconInterview} alt=""/></div>)
        }
    }

    if (item.recruitment.Contract) { //S'il y a un contrat
        // Icone de contrat en vert
        listIcon.push(<div><i mg id="contractIcon" className="enabled" src={iconContract} alt=""/></div>)
    } else {
        // Sinon icone de contrat en blanc
        listIcon.push(<div><img id="contractIcon" src={iconContract} alt=""/></div>)
    }
    return (
        listIcon
    )
}

// Recupere le pourcentage de la barre de progression en fonction de l'avancement du processus de recrutement
function getPourcent(item){
    let pourcent = 0
    if (item.recruitment.Contact && !item.recruitment.Contract){ //Si on a le contact
        pourcent = 10
        if (item.recruitment.interviews.length > 0){ //Si on a plus d'un entretien
            pourcent = 50
        }
    }else if (item.recruitment.Contract){ // Si on a le contrat
        pourcent = 100
    }
    return pourcent
}
```

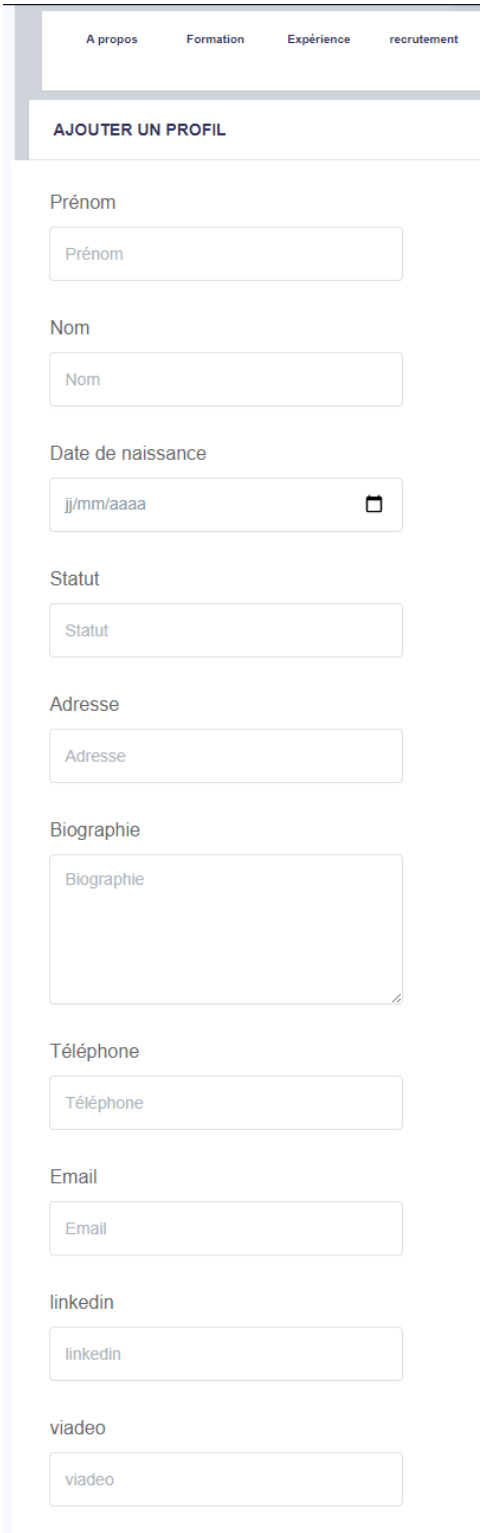
Annexe 6 - Affichage de l’onglet “CV” du candidat



Annexe 7 - Affichage de la section "Compétences"

Annexe 8 - Affichage de la section “Contact”

Annexe 9 - Formulaire du profil



Recherche un emp□i

Ca□e

Handica□

Possède un visa de trav□l

Visa de travail en cours de valic□

Taux journalier moyen

0

Rémunération actuelle

0

Rémunération souhaitée

0

Compétences +

Soumettre

Annexe 10 - Entraîner l'IA avec un jeu de données

```
def train_model(train_data):

    # '_' car on a un tuple donc on devrait mettre 2 elements dans le for, mais on veut que annotation et pas le texte
    for _, annotation in train_data:
        for ent in annotation['entities']:
            # On ajoute touts les elements de la partie "entites" dans la pipeline un par un
            # Pour chaque objet dans la pipeline, on indique que le label est l'element en 3e position
            ner.add_label(ent[2])

    # On regarde si on a d'autres pipelines que 'ner'
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']

    #S'il y en a d'autres alors on les enlève (pour alléger)
    with nlp.disable_pipes(*other_pipes): # only train NER

        # Lancement
        optimizer = nlp.begin_training()

        for itn in range(20):
            print("Starting iteration " + str(itn))
            random.shuffle(train_data) # Melange la data
            losses = {}
            index = 0
            for text, annotations in train_data:
                try :
                    nlp.update(
                    [text],  # Le texte
                    [annotations],  # Les annotations
                    drop=0.2,  # dropout - Fait en sorte que ce soit du de memoriser la data
                    sgd = optimizer,
                    losses=losses)
                except Exception as e:
                    pass

            print(losses)
```