Projet SmartRecruiting

Rapport de projet

Estelle LANDI
Estelle REYMOND
Loïc SCHANEN
Rémi VARENNE

2019-2020

Sujet du projet **2**

Missions à accomplir **3**
Gestionnaire de scolarité 3
Gestion des entreprises 4

Technologies utilisées **4**
Keras 4
MySQL 4
Flask 5
Angular et Material Design Bootstrap 5
Apache HTTP Server et mod_wsgi 5

Architecture **5**

Réalisations techniques **6**
Front-end 6
Login/register 7
Persona Gestionnaire 7
Persona Entreprise/Recruteur 8
Back-end 8
Organisation de la base de données 8
Vectorisation 8
Réseau de neurones 9
Tf-idf 9
Allocation de Dirichlet Latente 9
Déploiement 9
Green IT 10

Gestion de projet **10**

Métriques logicielles **11**

Evolutions futures **11**
Évolutions back-end 11
Amélioration des modèles 11
Remplissage base de données 12

Évolutions front-end **12**
Différents niveaux de travail 12
Sécurité 12

Conclusion **13**

Bibliographie **14**

Sujet du projet

Le projet Smart Recruiting suit la logique de dynamisation et de simplification de la relation université-entreprise des Disrupt Campus.

La problématique auquelle répond le sujet est la suivante : il est aujourd'hui difficile de trouver une véritable connexion entre un étudiant et un stage ou une formation. En effet, tout le service éducatif, dont l'université, joue un rôle extrêmement important dans l'emploi en France. C'est là que l'on voit la pertinence de se rapprocher au maximum de l'écosystème pour être au plus proche des attentes de toutes les parties. Cela veut dire que le premier intérêt de ce sujet est d'être au plus près pour comprendre les différentes dynamiques, que ce soit du marché de l'embauche ou des intérêts et envie des étudiants.

Le but de ce projet est de relier plus efficacement 5 acteurs : des étudiants cherchant un stage ou une formation, des gestionnaires de scolarité, des universités, et des entreprises qui cherchent des formations et donc des élèves adaptés à leurs offres de stage.

Cette plateforme qui serait publique permettrait à l'entreprise de ne plus être dépendante des plateformes privées pour garantir, ou du moins faciliter, l'embauche de ses étudiants. Elle pourrait aussi permettre aux universités de comprendre et d'agir directement sur le marché de l'emploi en adaptant les formations ou en cherchant des entreprises qui pourrait devenir partenaires.

Voici une représentation schématique du projet qui permet de comprendre l'articulation du projet entre les différents acteurs.

Je souhaite	Trouver une formation	Trouver une offre de stage	Voir la place d'une formation au sein du marché
Qui ?	 ÉTUDIANTS RECRUTEUR	 ÉTUDIANTS	  GESTIONNAIRE SCOLARITÉ DIRECTEUR UNIVERSITÉ
Comment ?	En écrivant ce que je recherche	En écrivant ce que je recherche	
Résultat de ma demande Smart Recruiting	Nom de la formation Coordonnées de son responsable	Offres de stages disponibles Coordonnées de recruteur	Tendance de la formation d'une année à l'autre Top entreprises, top domaines Nombre d'offre Compétence les plus demandées
Qui est en lien avec la demande ?	 GESTIONNAIRE DE SCOLARITÉ	 RECRUTEUR	  RECRUTEUR ÉTUDIANTS

Tableau 1 : Représentation des interactions entre les différents personas

On peut définir les acteurs comme des personas :

- **Persona “Chargé de recrutement”** : La personne dispose d’une fiche de poste et voudrait recruter via l’université. La personne dépose son offre en texte libre. La personne reçoit une liste de résultats de formation et peut choisir de transmettre ou non son offre aux gestionnaires de scolarité concernés qui la transmettront aux étudiants concernés.
- **Persona “Étudiants qui cherche un stage”** : L’étudiant veut trouver un stage et décrit ce qu’il veut faire. Il reçoit une liste de propositions de stages disponibles, une liste d’entreprises qui ont auparavant pu proposer ce type de stage et une liste de contact lui permettant de postuler aux stages ou de faire des candidatures spontanées.
- **Persona “Étudiant qui cherche une formation”** : L’étudiant veut trouver une formation. L’étudiant décrit ce qu’il veut étudier et il reçoit une liste de formations qui pourrait correspondre à sa demande.
- **Persona “gestionnaire de scolarité”** : Le gestionnaire veut connaître les tendances du marché. Il consulte un dashboard avec divers indicateurs.
- **Persona “directeur d’université”** : Le directeur veut avoir un dashboard et des indicateurs qui restent à définir mais qui pourraient être les suivants : composantes qui ont le plus de demandes / domaines où les entreprises n’arrivent pas à trouver des recrues (avec des trous dans la raquette des formations)

La plateforme Smart Recruiting a donc pour objectif de permettre à l’éducation supérieure de gagner en visibilité sur le marché de l’emploi afin de mieux orienter les politiques et les investissements futurs.

Missions accomplies

Sachant que le projet était plutôt vaste, on a dû spécifier lors d’une première réunion avec Anthony Geourjon et Gerard Pollier quels étaient les objectifs qu’on allait chercher à atteindre. Nos missions étaient tout d’abord de nous occuper du persona “gestionnaire de scolarité”, traiter le persona “chargé de recrutement” tout en s'intéressant au côté Green IT.

Gestionnaire de scolarité

Le premier sprint consistait à implémenter le persona “gestionnaire de scolarité”. Suite à des échanges entre Anthony et Nadine Chatti, la responsable

relations entreprises de Polytech Grenoble, nous avons pu établir la liste des informations pertinentes qu'il faudrait afficher sur le dashboard.

La majeure partie des demandes comprenait des statistiques ayant pour but de montrer l'évolution des intérêts des autres acteurs du sujet (étudiants voulant un stage, étudiants voulant une formation et chargé de recrutement). On peut y retrouver l'affichage du nombre d'offre où la filière / une des filières de la composante arrive première, l'affichage des domaines (ou secteurs) associées aux offres de stages et donc aux entreprises, l'affichage des compétences requises pour les formations ou pour les offres, l'affichage de tendances des domaines et compétences les plus recherchés ou encore l'affichage des entreprises les plus demandeuses et dont le nombre d'offres a le plus évolué.

Gestion des entreprises

La prochaine étape concernait en la gestion des entreprises, notamment l'inscription de chargé de recrutement de l'entreprise avec l'inscription de l'entreprise associée en base de données. Il fallait aussi permettre la gestion des offres par l'entreprise, que ce soit l'ajout, la modification ou la suppression d'offres. Le recruteur devrait aussi être capable de trouver les formations associées à son offre, lui permettant de diffuser plus rapidement son offre.
Il s'agissait de poursuivre le travail du groupe d'étudiants d'ENSIMAG/GEM en travaillant notamment la partie Front du code.

Technologies utilisées

Keras

Keras est l'une des principales API de réseaux neuronaux de haut niveau. Elle est écrite en Python et prend en charge plusieurs moteurs de calcul de réseaux neuronaux en arrière-plan.

MySQL

MySQL est un système de gestion de bases de données relationnelles. Il fait partie des logiciels de gestion de bases de données les plus utilisés au monde. C'est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place vers celui de mises à jour fréquentes et fortement sécurisées.

Flask

Flask est un framework web qui fournit des outils, des bibliothèques et des technologies qui vous permettent de construire l'application web. Les avantages de Flask sont que le framework est léger, qu'il y a peu de dépendances à mettre à jour et de bugs de sécurité à surveiller. Les inconvénients sont qu'à un moment donné, il faudra faire plus de travail par soi même ou augmenter le nombre de dépendances en ajoutant des plugins.

Angular et Material Design Bootstrap

Angular est un framework développé par Google basé sur TypeScript. Il est grandement orienté autour des composants et des services. Ce framework nous a permis de développer la partie front-end du projet. Nous nous sommes basés sur un template de dashboard de MDBootstrap (Material Design Bootsrap). MDBootstrap est très populaire pour développer des applications web réactives.

Apache HTTP Server et mod_wsgi

mod_wsgi est un module Apache qui permet d'héberger une application WSGI Python.

Architecture

Sachant que nous sommes partis d'un projet existant, nous nous sommes basés sur l'architecture qui avait été choisie par les anciens groupes. Il est développé en plusieurs parties :

- **une partie Front-End**, développée en TypeScript, utilisant le framework Angular. Elle est chargée de faire le lien entre l'utilisateur et le back-end. C'est ce qui permet de rendre vie aux données brut qu'on récupère du back-end.
- **une partie Back-End** qui est développé en Python et utilise le framework Flask. Elle a pour but de proposer des fonctionnalités se faisant hors ligne comme le chargement des données, les entraînements des modèles ou le traitement des datasets. De plus elle permet de répondre aux requêtes du client quand celui-ci demande une action en lien avec les données (comme demander une corrélation entre un texte tapé et les offres se trouvant en base de données). Ces fonctionnalités se faisant en ligne passe par une API.
- **une partie base de données** qui est conçue en une base MySQL. Elle sauvegarde les offres, les entreprises, les différents utilisateurs, les formations, les composantes, …

Ci-dessous se trouve une représentation schématique de l'organisation des parties lors d'un appel pour chercher une formation.

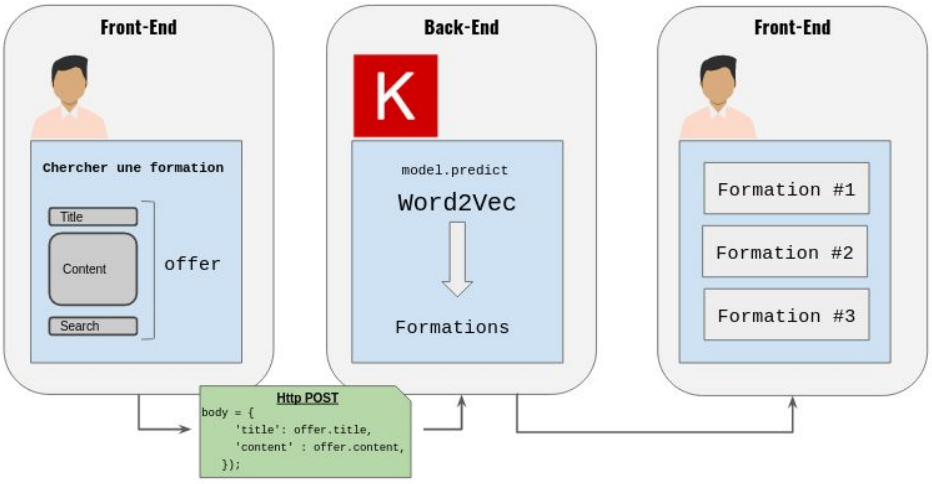

Figure 1 : Communication entre les différentes parties de l'architecture

Lorsqu'un étudiant ou un recruteur cherche une formation, il va taper un texte et lancer la recherche pour connaître les formations les plus adaptées à sa demande. Les traitements sur le serveur back-end vont se faire suivant ce protocole :

1. Réception de l'offre à travers l'API, sous le format de fichier JSON
2. Vectorisation de l'offre avec le modèle Word2Vec qui a été précédemment entraîné.
3. Calcul de la formation de sortie, que l'on peut aussi appelée classe, avec le réseau de neurones, avec en entrée l'offre vectorisée.

Réalisations techniques

Front-end

Le framework angular est orienté composants et services. La structure du front-end est simple : on a 3 layouts : login/register, gestionnaire, entreprises. Pour chaque layout, on dispose d'un composant par parties, par pages. Toutes les requêtes qu'on effectue vers le back-end se trouve dans ce qu'on appelle des services. Ces services nous permettent de faire le lien entre la partie fonctionnelle et visuelle et la base de données.

Login/register

Concernant l'authentification sur l'application, nous nous sommes basés sur un tutoriel que nous avons trouvé en ligne et que nous avons adapté pour notre application. Nous avons donc une page de connexion et une page d'inscription ou les utilisateurs vont rentrer les informations essentielles à la création de leur compte puis s'y connecter. Nous avons également rajouté une page pour créer une entreprise lors de l'inscription si cette dernière n'est pas déjà créée (comme Pstage le fait). Dans la page d'inscription, selon le persona, différentes informations sont demandées afin de remplir la base de donnée correctement.

Afin de gérer au mieux la connexion et l'inscription, nous avons utilisé un service d'authentification. Ce service permet de faire appel à l'API backend afin d'enregistrer et créer les différents utilisateurs selon les personas. On y récupère un token de connexion JWT qui va permettre à l'utilisateur de rester connecter malgré les changements de page et les rafraîchissements. De plus, nous avons rajouté des gardes d'authentification afin de garantir que les utilisateurs d'un persona donné n'ont accès qu'aux pages dédiées à ces personas et pas aux autres.

Persona Gestionnaire

Un travail en amont a été effectué auprès de Nadine Chatti pour se rendre compte des besoins réels des gestionnaire de scolarité. Suite aux informations rapportées, nous avons pu élaborer les maquettes pour le dashboard. Il fallait miser sur la simplicité et s'assurer que les informations que les gestionnaires de scolarité souhaitent avoir soient disponibles en quelques clics.

Nous avons définis 4 onglets de navigation :

- Onglet Tendance : Évolution entre l'année N et l'année N-1 (évolution des compétences les plus demandées, les domaines les plus recruteurs, les formations les plus demandées).
- Onglet Compétences : Sous forme de diagramme à bulle, on retrouve le détail des compétences qui ressortent le plus dans les offres postées par les recruteurs.
- Onglet Entreprise : Entreprises ayant postées le plus d'offres, domaines des entreprises les plus demandeuses.
- Onglet Offre : Nombre total d'offres, classement des écoles ayant le plus d'offres associées, nombre d'offres par année d'études.

Tous tableaux, diagrammes circulaires, diagrammes en barres sont complétés suite à des requêtes http faites au back-end qui renvoie les informations nécessaires.

Persona Entreprise/Recruteur

Pour ce persona, la demande est simple. Le recruteur doit pouvoir poster une offre, chercher des formations associés à une offre et gérer ses offres (modifier, supprimer).

Nous avons donc réalisé un second dashboard ayant la même structure et le même design que pour le persona “gestionnaire de scolarité”. C’est à dire, sur la gauche, on a une barre de navigation avec “poster une offre, chercher une formation, mes offres”.

Dans poster une offre, le recruteur va rentrer une offre et cette offre va se stocker dans notre base de donnée. Il peut ensuite trouver la formation associée à l’offre qu’il vient de poster. Pour cela, on utilise le machine learning implémenté par les anciens groupe et améliorer par le groupe de GEM/Ensimag.

On a aussi gérer la modification et la suppression des offres.

Back-end

Organisation de la base de données

Une fois les nouveaux objectifs déterminés, nous avons dû revoir la base de données. En effet, au départ la base de données était seulement composée des tables nécessaires pour faire fonctionner le deep learning.
Notre Product Owner a rencontré une gestionnaire de scolarité pour définir les nouvelles données nécessaires pour le persona Gestionnaire, et nous nous sommes mis d’accord sur les données liées au persona Étudiant. Nous avons donc rajouté plusieurs tables pour nos besoins. Nous avons, dans un premier temps, créé les tables pour les différents utilisateurs puis les tables pour les écoles, les formations et les entreprises.

Vous pouvez retrouver le schéma de notre nouvelle base dans le rapport technique.

Nous avons essayé d'anticiper pour le futur, c’est-à-dire rajouter dans la base, les données pour que le persona Étudiant puisse être réalisé facilement sans à avoir à reconstruire la base de données.

Vectorisation

Le modèle gensim Word2Vec qui est un simple réseau neuronal à deux couches, a été pris pour vectoriser le corpus de texte. Word2Vec est une technique d’intégration de mots qui transforme un corpus de texte en des vecteurs.

Réseau de neurones

Le modèle de vectorisation qui a été créé juste avant est ajouté comme couche au modèle de réseau neuronal créé avec Keras. Toutes les dimensions du modèle Word2Vec sont ajoutées comme neurones au réseau neuronal convolutionnel.

Tf-idf

Le “term frequency-inverse document frequency” (tf-idf) est l’une des techniques les plus utilisées dans le domaine de l’exploration d’information textuelle. Elle permet d’identifier dans quelles proportions certains mots du corpus peuvent être évalués par rapport au reste du corpus. Cette technique de référencement est très utile pour déterminer les mots-clés qui pourront être idéalement utilisés pour l’entrainement du modèle.

Allocation de Dirichlet Latente

La Latent Dirichlet Allocation (LDA) est un modèle statistique qui est largement appliqué dans le domaine du traitement du langage naturel, ou natural language processing (NLP). Le score de cohérence permet d’évaluer la qualité des thèmes appris. Il calcule principalement la co-occurrence globale en prenant les mots deux-à-deux dans un même thème. Plus le score de cohérence est élevé, plus il y a de chance que les mots d’un même thème se suivent.

Déploiement

Une fois que nous avons eu une version stable et montrable de notre application. Il nous a fallu mettre en production l’application. Déployer notre application va permettre de lui donner de la visibilité.
Nous avons donc dû déployer l’application sur une machine virtuelle mise à notre disposition.

Nous avons choisi d’utiliser le serveur HTTP Apache2 pour mettre en place notre site. Nous avons également dû utiliser le module mod_wsgi pour mettre en environnement de production notre application Flask. En effet, Apache va se servir du fichier .wsgi, un script qui va permettre le lancement de l’application Flask qui est notre back-end.
Nous avons également dut configurer serveur HTTP Apache pour qu’il puisse faire fonctionner autant l’application web que l’application Flask.

Le tutoriel correspondant au déploiement est disponible dans wiki du projet doc du GitLab.

Green IT

Disrupt Campus est très soucieux de l'impact qu'a leurs applications sur l'environnement. C'est pourquoi nous nous sommes sensibilisés au Green IT et effectués des recherches pour rendre Smart Recruiting plus "green".
En premier lieu, nous avons effectué de nombreuses recherches sur les bonnes pratiques. A l'heure actuelle, il y a très peu de travaux dont l'objectif est d'avoir un code éco-responsable. Nous avons trouvé un nombre important de recommandation mais cela ne veut pas dire qu'elles sont actuellement appliquées. Cela est d'ailleurs bien dommage car la consommation d'énergie due aux services informatiques est encore actuellement sous-évaluée.

GreenIT est d'autant plus intéressant et important pour les projets utilisant des méthodes d'intelligence artificielle car cette partie là est vraiment omis dans les consommations énergétiques. Les recherches sont encore moins présentes pour cette partie là de l'informatique. C'est pourtant un domaine extrêmement consommateur même s'il permet en même temps de proposer des solutions ayant pour but de réduire les consommations d'énergie.

Nous avons déposé toutes nos recherches et références sur le repo Git, dans le wiki nommé GreenIT du dossier Doc.

Gestion de projet

Notre projet étant la suite d'un projet précédent, mené par une équipe d'étudiants en double cursus entre l'école de management de Grenoble en Big Data et ENSIMAG. Nous avons eu l'opportunité de les rencontrer durant une réunion peu de temps après le début de notre projet. Ils ont pu nous expliquer où ils en étaient, nous faire passer leur code et nous donner les différents axes d'améliorations du projet. Il faut savoir que ces axes correspondent davantage aux parties du projet sur lesquelles ils étaient en train de travailler. De plus, ils sont restés disponibles pour échanger sur le projet durant tout le long du notre.

L'équipe Disrupt Campus avec Anthony notamment à notre écoute en cherchant de nouvelles données nous permettant d'avancer. L'objectif de ce projet était de créer une première version démontrable afin de prouver que le concept était viable avec pour objectif de convaincre davantage d'université pour mettre plus de groupes de projet sur ce sujet.

Étant en petite équipe, nous avons aisément pu travailler en pratique une grande partie de l'approche agile. On a suivi des itérations, ou sprints détaillés au début du projet avec des adaptations hebdomadaires en fonction des attentes du client, qui

dans notre cas était les membres de Disrupt Campus. On a ensuite l'ensemble des réunions définies pour un projet Scrum, à savoir les planifications de Sprint, les revues de sprint, les rétrospectives de Sprint et les mêlées quotidiennes. On a également fait un "product planning" ou poker planning permettant par donner un ordre d'idée du coût en terme de temps donnée par la complexité considérée pour les tâches demandées.

Métriques logicielles

Bien que toutes ces mesures restent assez compliquées à mesurer et analyser dans le contexte où nous avons fini notre projet, nous pouvons néanmoins faire ressortir quelques valeurs pour avoir une idée de l'effort fourni durant ce projet.

Tout d'abord, nous avons déjà chacun travaillé 184 heures sur les heures définies par l'emploi du temps. Cela représente donc 736 heures de travail pour le groupe soit un peu plus de 4 mois. Il faut également penser au fait que l'on considère pas le travail qui a pu être fait à domicile, que ce soit le soir mais également le week-end.

Notre projet est séparé en deux projets git : un pour le front et un pour le back.

Côté Front, nous avons 122 fichiers et 18 907 lignes de code créées dont 70% du code est du JSON, 17% est du TypeScript et 9% est de l'HTML. En réalité, nous avons seulement travaillé sur les fichiers en TypeScript, HTML, JavaScript et SCSS. Si on considère seulement ces fichiers, nous avons 62% de Typescript, 31% de HTML, 7% de SCSS. Nous avons créé 22 branches, fait 112 commits et fait 17 merge request.

Côté Back, nous avons 857 fichiers et 113 593 lignes de codes créées dont 82% du code est du Java, 8% est du TypeScript, 6% est de l'HTML et 3% du Python. C'est peu représentatif de notre travail car nous avons presque exclusivement travaillé sur du code Python et csv pour ce qui était de traiter des datasets. Sur ce dépôt, nous avons créé 10 branches et fait 115 commits.

Evolutions futures

Évolutions back-end

Amélioration des modèles

Sachant que notre mission n'était pas vraiment d'améliorer les modèles, il serait intéressant de travailler la dessus.
En premier lieu, il faudrait réduire le nombre de catégories de formations. Il y a actuellement 685 formes de formation pour 16 000 offres. De plus, ces offres sont mals réparties : certaines formations ont très peu d'offres associées. Cela signifie que

même si on entraîne un modèle avec ces données, il y a très peu de chances que ce modèle soit plus précis à cause du manque de données sur certaines formations.
Il serait possible de tester plus en profondeur les couches cachées du modèle de réseau neuronal. Pour le moment, il n'y a qu'une seule couche utilisée mais on obtiendrait certainement une meilleure précision avec un modèle plus adapté.

Remplissage base de données

Un des problème que nous avons rencontré, était le manque de données pour remplir l'ensemble des champs de chaque table. En effet, les offres utilisées pour faire le deep learning ne sont pas renseignées avec l'ensemble des informations dont nous avons besoin pour faire nos statistiques. Par exemple, plusieurs statistiques demandées pour le persona Gestionnaire sont en relation avec le domaine de l'offre ou de la formation. Malheureusement, ces informations n'est actuellement pas disponible.

Pour le futur, il faudrait arriver à déterminer le domaine de l'offre en utilisant la description des offres avec du machine learning. Cela nous permettrait de remplir correctement nos tables. Il faudrait également que pour les formations associées à un domaine soient renseignées pour pouvoir réaliser les différents niveau de visualisation (UGA, composante, filière) des statistiques demandées pour le persona Gestionnaire.

Évolutions front-end

Différents niveaux de travail

Pour le persona "gestionnaire de scolarité", on peut penser à gérer les différents niveaux de vue : niveau universitaire (ex : UGA), niveau composante (ex : Polytech), niveau formation (ex: Info).
Un gestionnaire voudra par exemple voir les statistiques de sa composante, la place de la composante au sein de l'Université. Mais également, le gestionnaire voudra voir la place de chaque filière au sein de la composante.

Sécurité

Étant donné que le but de ce projet était d'avoir une première version livrable, l'aspect sécurité de l'application n'était pas une priorité.

Nous n'avons tout d'abord pas d'administrateur général de l'application pouvant gérer les différents users connectés. Ainsi n'importe qui peut s'inscrire en se déclarant par exemple gestionnaire de scolarité de Polytech sans qu'il n'y ait de validation derrière, il n'y a aucune vérification lors de la création d'une entreprise. Avoir une gestion des mails afin de gérer la fonctionnalité de mot de passe oublié

aurait aussi été une bonne chose.

Pour finir, l'application que nous avons développée met en forme des données qui peuvent être sensibles, il serait donc essentiel dans le future de s'assurer que l'accès à ses données est bien protégé.

Conclusion

Smart recruiting est un projet ambitieux qui a beaucoup de possibilités d'évolutions. Nous avons été ravie de prendre part à son évolution. Ce projet fut l'occasion de travailler sur un sujet d'actualité et on espère qu'il aboutira pour pouvoir faire repenser le mode de recrutement des élèves.

Travailler sur ce projet a permis à chacun de découvrir de nouvelles technologies et de mettre à profit ses connaissances. Ce projet étant bien défini, nous a permis de travailler autant sur le back-end que sur le front-end. On a pu avoir une vue globale de comment marche l'application Web. De plus, le fait de bien cadrer les besoins de chaque personas nous a permis de gagner du temps dans l'implémentation, de savoir où on allait et de bien définir les tâches à réaliser.

Nous avons réussi à avoir un travail d'équipe de qualité. Chacun avait un rôle défini tout en restant disponible pour aider les autres. Une bonne entente entre nous nous a permis de ne pas rester bloqué sur des éléments et de prendre du plaisir à venir travailler le projet.

Bibliographie

https://disrupt-campus.univ-grenoble-alpes.fr

https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92

https://fr.ryte.com/wiki/TF*IDF

https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158

https://fr.wikipedia.org/wiki/MySQL

https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html

https://pymbook.readthedocs.io/en/latest/flask.html

https://agiliste.fr/introduction-methodes-agiles/

Glossaire

Datasets : Jeux de données

Deep learning ou **Apprentissage profond** Ensemble de méthodes d'apprentissage automatique tentant de modéliser avec niveau d'abstraction des données.

Green IT : Concept qui cherche à diminuer l'empreinte écologique, économique, et sociale des technologies de l'information.

Langage naturel : Correspond au parlé humain (≠ langage formel)

Modèles : architecture de données permettant d'utiliser une technologie d'intelligence artificielle sur une base de données d'une même façon plusieurs fois de suite

Product Owner : Personne qui sert de lien entre le client et l'équipe de développement

JWT ou **Json Web Token** : méthode sécurisée d'échange d'informations.

WSGI ou **Web Server Gateway Interface** : interface entre les serveurs Web et les frameworks web Python.