

RAPPORT DE STAGE

8 Avril 2024 - 31 mai 2024



Etudiant - Stagiaire Développeur web | **Quentin Brandy**

Tuteur de Stage - Gérant FW16 , formateur MMI IUT d'angoulême | **Benjamin Rougier**

Professeur référent - Professeur de hébergement, maître de conférence | **Réda Guedira**

Remerciements

Avant de commencé ce rapport , je tiens à exprimer ma plus profonde gratitude à toutes les personnes qui ont contribué à la réussite de ce stage et à la rédaction de ce rapport.

Tout d'abord, je remercie chaleureusement mon tuteur de stage , Monsieur Benjamin Rougier directeur de l'agence Freelanceweb16 et aussi professeur remplaçant MMI au moment de mon stage à l'IUT d'Angoulême , pour sa présence , ses conseils avisés et son aide précieuse tout au long de cette expérience . Son soutien et son encadrement ont été indispensables pour mon apprentissage et mon intégration au sein de l'équipe pendant ces 2 mois.

Je souhaite également remercier mon professeur référent à l'IUT de Limoges, Monsieur Réda Guedira , pour son suivi et ses conseils tout au long de cette période. Sa disponibilité et son soutien ont été d'une grande aide pour mener à bien ce projet.

Je n'oublie pas les employés de l'agence Freelanceweb16, dont la bienveillance et la collaboration ont grandement facilité mon intégration et ont rendu cette expérience encore plus enrichissante. Merci à chacune d'entre elle pour leur accueil chaleureux et pour m'avoir partagé leur connaissances avec tant de générosité et faire découvrir tout les aspects d'une petite agence comme celle-ci.

Enfin, je remercie mes amis pour leur aide pendant l'écriture et la relecture de ce rapport.

Leurs contribution à tous a été essentielle pour la réussite de ce stage, et je leur en suis profondément reconnaissant.

Sommaire

Remerciements	2
Introduction	4
1. Présentation de l'entreprise	5
1.1 Présentation général de l'entreprise	5
1.2 Présentation de l'environnement de travail	6
2. Projets Effectué	7
2.1 Présentation du Projet	7
2.2 Recherche graphique	8
2.3 Mise en place en local du projet	9
2.4 Développement des itérations	11
2.4.1 Développement des comptes entreprises	11
2.4.2 Développement du service de connexion	13
2.4.3 Développement Des Offres	14
2.4.4 Développement Des compte utilisateurs	15
2.4.5 Développement du système des offres	16
2.5 Bilan du projet	18
3. Conclusion	19
4. Glossaire	20
5. Annexes	21

Introduction

Dans le cadre de la validation de ma deuxième année de BUT Métiers du Multimédia et de l'Internet (MMI) avec un parcours en développement web et dispositifs interactifs, j'ai effectué un stage de deux mois au sein de l'agence Fw16. Ce stage avait pour objectif principal de me permettre de mettre en pratique les compétences acquises durant ma formation en développement full stack. En intégrant cette agence, j'ai eu l'opportunité de travailler sur un projet fictif de développement web qui a mis toute mes compétences à l'épreuve, ce qui m'a permis de me confronter aux réalités du métier de développeur et d'améliorer mes compétences techniques en front-end et back-end.

Le projet abordé au cours de ce stage était de concevoir et de développer un site de recherche d'emploi similaire à des plateformes telles qu'Indeed ou HelloWork. L'objectif était de réaliser une application complète permettant aux entreprises de publier des offres d'emploi et aux utilisateurs de rechercher et postuler à ces offres. De cette mission, une problématique se dégage : **Comment développer un site de recherche d'emploi optimal et intuitif qui puisse répondre aux besoins des entreprises et des chercheurs d'emploi tout en assurant une expérience utilisateur fluide et efficace ?**

Le rapport qui suit est structuré en plusieurs étapes clés du projet, afin de résumer de manière détaillée le travail effectué durant ces deux mois. Tout d'abord, une présentation générale du projet sera effectuée pour situer le contexte et les objectifs de celui-ci. Ensuite, je décrirai la phase de recherche graphique qui a été essentielle pour définir l'interface utilisateur. Je parlerais ensuite de la mise en place en local du projet , suivie par le développement des différentes itérations nécessaires à la réalisation de l'application.

Le développement a été subdivisé en plusieurs sous-étapes : création des comptes entreprises, développement des offres d'emploi, gestion des comptes utilisateurs, et enfin, développement du système de gestion des offres. Je conclurai ce rapport par un bilan de l'application produite, en soulignant les réalisations accomplies et les améliorations possibles pour optimiser davantage l'application.

1. Présentation de l'entreprise

1.1) Présentation général de l'entreprise

J'ai eu la chance d'effectuer mon stage au sein de l'agence FW16, située à Mouthiers-sur-Boëme. FW16 est une petite agence dynamique composée de cinq membres permanents : trois alternants, une commerciale, et le dirigeant, Benjamin Rougier, qui a également été mon tuteur tout au long de ce stage. (voir la devanture de l'agence dans l'annexe n°1)

Les trois alternants de l'agence se spécialisent chacun dans des domaines spécifiques. Deux d'entre eux sont graphistes : l'une se concentre sur la création de visuels et de logos, tandis que l'autre est davantage impliquée dans la conception graphique de sites web, principalement sur InDesign. La troisième alternante est en charge de la comptabilité de l'agence, assurant ainsi une gestion financière rigoureuse.

FW16 propose une gamme complète de services dans le domaine du web, allant de la création et la refonte de sites web à la conception d'applications mobiles et de boutiques e-commerce. En plus de ces services, l'agence réalise des audits SEO, met en place des campagnes publicitaires, et développe des stratégies de marketing par email.

Suite à l'évolution de la place des réseaux sociaux depuis ces dernières années et leurs évolutions constantes, FW16 c'est aussi lancé dans l'accompagnement sur les réseaux sociaux des entreprises en créant du contenu et gérer leurs présence sur les réseaux sociaux. Elle offre des services de gestion de réseaux sociaux, de campagnes publicitaires, de gestion de l'e-réputation.

1.2) Présentation de l'environnement de travail

Sachant que l'agence Freelanceweb16 (FW16) est une petite agence, elle ne comporte que trois pièces : l'agence elle-même où nous travaillons et accueillons les clients, des toilettes et une remise.(vous pouvez retrouver mon bureau dans l'annexe **n°2**). Étant à 35 heures par semaine, je travaille de 9h à 18h du lundi au vendredi. J'aurais pu choisir de faire une heure de pause le midi pour terminer à 17h, mais le fait d'avoir deux heures de pause me permet d'être plus efficace sur le long terme.

Le matin, j'arrive un peu en avance, vers 8h55, pour avoir le temps d'installer tout mon matériel : mon ordinateur portable, un câble HDMI pour connecter un deuxième écran, et lancer les applications nécessaires à mon travail. Ensuite, je consulte mon Trello pour savoir où j'en étais et planifier ma journée. Je travaille jusqu'à environ 10h45, moment où je prends une pause de 15 minutes, puis je continue jusqu'à midi. Après ma pause déjeuner, je reprends à 14h et travaille jusqu'à une autre pause vers 16h30, puis je continue jusqu'à 18h. À la fin de la journée, je mets à jour mon Trello et pousse mon travail de la journée sur GitHub.

Ma journée est également rythmée par le passage de clients, ce qui me permet de pouvoir d'observer les échanges avec les clients et l'évolution des projets. De plus, il arrive des fois que des commerciaux et des entrepreneurs viennent à l'agence pour rencontrer le gérant. Ces rencontres m'ont permis d'acquérir des connaissances intéressantes sur la gestion d'une entreprise et le travail de certains métiers.

L'agence FW16, bien que petite, offre un environnement de travail propice à l'apprentissage et au développement professionnel. La proximité avec les membres de l'agence et les clients permet une immersion complète dans le monde professionnel, m'offrant ainsi une vision globale des différents aspects du métier de développeur web, de la conception à la gestion des projets, en passant par les relations avec les clients.

2. Projet Effectué

2.1) Présentation du projet

Comme un je l'ai mentionné un petit peu durant l'introduction j'avais un projet fictif à réalisé. L'objectif était de réalisé un site de recherche d'emploi comme Indeed , HelloWork etc. Lors de ce projet, j'ai eu la liberté de choisir les langages et frameworks avec lesquels j'allais travailler. Pour la partie frontend, j'ai opté pour React. Bien que j'aie déjà pratiqué ce framework pendant plusieurs mois, je voulais approfondir ma compréhension globale afin de pouvoir le manipuler avec plus de maîtrise et d'efficacité. Pour la gestion du CSS, j'ai décidé d'utiliser le framework TailwindCSS en raison de sa facilité d'utilisation et de son exécution rapide. TailwindCSS permet de créer des designs propres et personnalisés sans avoir à écrire de longues feuilles de style, ce qui accélère considérablement le développement des sites.

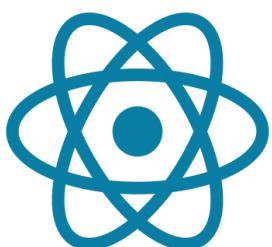


Fig 1 : Logo React



Fig 2 : Logo Tailwindcss

Pour la partie backend, j'ai décidé d'utiliser Node.js. Car j'ai toujours utilisé PHP ou le Framework Symfony pour faire du back et cela fait quelque année que PHP diminue en pourcentage d'utilisation par les développeur et pourrait diminuer encore plus à l'avenir, et ce stage représente une opportunité idéale pour explorer une autre approche en adoptant Node.js. Node.js est reconnu pour sa performance et sa capacité à gérer un grand nombre de requêtes simultanées, ce qui en fait une technologie robuste et moderne pour le développement backend. Pour la partie base de donnée ayant déjà une expérience avec MySQL, j'ai choisi de continuer à l'utiliser en raison de sa fiabilité, de sa robustesse



Fig 3 : Logo node.js



Fig 4 : Logo MySQL

2.2) Recherche Graphique

Pour mener à bien ce projet, j'ai dû commencer par créer une charte graphique pour le site. La première étape a été de choisir un nom pour l'application. Après avoir examiné les noms d'autres entreprises similaires et dressé une liste de mots associés à l'emploi, j'ai décidé de nommer l'application "**Job Odyssey**".

Le mot "**Job**" fait directement référence au travail en anglais, tandis que "**Odyssey**" évoque un voyage épique, souvent rempli de défis et de découvertes, comme le célèbre voyage d'Ulysse dans l'*Odyssée* d'Homère. En rapport avec l'application, l'*odyssey* représente l'idée d'un parcours professionnel, ponctué d'entretiens et de refus, à la recherche du job parfait.

Ensuite, il a fallu créer un logo pour accompagner le nom de l'application. N'étant pas particulièrement fort en créativité, j'ai décidé de rester simple et efficace. J'ai choisi une typographie originale pour le nom de l'application et j'ai ajouté une fusée en marche pour symboliser l'*odyssée* moderne. Adieu les bateaux, bonjour la conquête de l'espace.



Fig 5 : Logo du site "Job Odyssey"

je suis parti à la recherche des couleurs que j'allais utiliser pour le site. Pour cela, je me suis aidé du site Adobe Color en consultant les chartes graphiques tendance en UI/UX. J'ai ensuite sélectionné certaines couleurs d'une charte graphique existante et y ai ajouté quelques nuances supplémentaires pour personnaliser cette palette et l'adapter parfaitement à l'identité visuelle de "Job Odyssey".



Fig 6 : Couleur utilisé pour le site

Le choix des couleurs effectué, j'ai recherché une police adaptée sur Google Fonts. J'ai opté pour la police "Work Sans", car elle était non seulement esthétiquement plaisante mais aussi évocatrice de l'objectif du site grâce au mot "work" qu'elle contient.

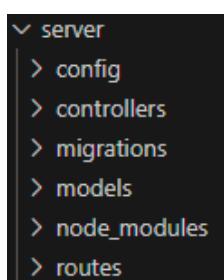
Après avoir finalisé le choix des couleurs et de la typographie, j'ai créé des styles sur Figma pour les intégrer facilement dans mes maquettes(voir annexe n°3). Une fois toutes ces étapes complétées, j'étais prêt à commencer le développement du site.

2.3 Mise en place en local du projet

Pour mon application, j'ai utilisé un modèle MVC¹ (Modèle-Vue-Contrôleur) où la vue est gérée par mon frontend et le backend s'occupe des modèles et des contrôleurs. Cette architecture permet de séparer les différentes responsabilités de l'application, rendant le code plus modulaire et facile à maintenir.

Pour la mise en place du backend avec Node.js, j'ai décidé d'utiliser l'ORM² (Object-Relational Mapping) Sequelize. Sequelize simplifie la gestion des bases de données en permettant de manipuler les données comme des objets JavaScript. Il gère automatiquement la création des modèles et des contrôleurs, facilitant ainsi l'ajout et la gestion des entités dans l'application.

Pour créer un serveur avec Node.js pour mon développement, j'ai utilisé le Framework : Express.js. Ce Framework permet de gérer les routes que reçoit l'adresse du serveur local lancé par Express. J'ai ensuite créé un dossier routes contenant des fichiers pour chaque modèle, répertoriant toutes les routes que mon serveur surveille ainsi que leurs méthodes de récupération : POST, GET, DELETE, et UPDATE. Ces routes exécutent les fonctions présentes dans mes fichiers de contrôleurs. Si il y'a une requête d'une autre méthode que celle surveillé par express alors les fonctions ne seront pas exécutés.



```
router.get('/getcompany', controllerscompany.getCompanyById);
```

Fig 8 : Exemple de route présente dans le backend

Fig 7 : Organisation des dossier de la partie backend

Pour gérer la vue de mon application, j'ai utilisé Vite.js, un outil qui permet de créer une application React prête pour le développement en local. Vite.js offre un environnement de développement rapide et efficace, idéal pour les projets React.

Pour la gestion des routes dans mon frontend, j'utilise le module react-router-dom, qui permet de réaliser du "client-side routing". Cela signifie que la navigation entre les différentes pages de l'application se fait côté client, sans recharger la page entière, ce qui améliore l'expérience utilisateur.

Ensuite, pour mes requêtes, j'ai structuré mon code en créant des dossiers pour chaque type de méthode HTTP. Chaque dossier contient des fonctions qui permettent d'appeler une route présente sur le serveur local Node.js et de récupérer la réponse. Si le serveur ne répond pas, une erreur est retournée. (voir annexe n°4 pour voir la construction d'une requête)

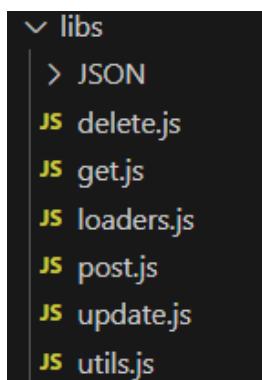


Fig 9 : Organisation des fichiers pour les requêtes en frontend

Pour la mise en place de la base de données en local, j'ai utilisé le logiciel XAMPP, qui permet de simuler un serveur Apache et de lancer, entre autres, une base de données MySQL en local. Ensuite, il m'a suffi de configurer les informations de connexion de la base dans Sequelize, et la base de données était prête. Ensuite, j'ai élaboré un schéma conceptuel pour structurer ma base de données et visualiser toutes les relations entre mes tables et leurs attributs (voir le schéma de la base de données en annexe n°5).

2.4 Développement des itérations

2.4.1 Développement des comptes entreprises

Pour commencer le développement de l'application, j'ai décidé de créer d'abord la page principale. Ensuite, je me suis concentré sur la création du formulaire d'inscription des entreprises. Pour cela, j'ai d'abord réfléchi aux informations nécessaires pour la création d'une entreprise. Afin de déterminer ces informations, j'ai consulté des sites similaires pour obtenir un maximum d'informations pertinentes et garantir des profils d'entreprises complets. près cela, j'ai utilisé le CLI³ (interface de ligne de commande) pour créer le modèle des entreprises. J'ai ensuite adapté ce modèle et exécuté une autre commande de sequelize pour créer la table dans la base de données à partir du modèle que j'ai créer.

Une fois le modèle créé en base, j'ai développé la page de création d'entreprise côté frontend, incluant toutes les informations présentes dans le modèle. J'ai pris soin de forcer l'utilisation d'un schéma d'email classique et d'un mot de passe comportant au minimum 6 caractères à l'aide de regex⁴ pour garantir que c'est bien un email valide et un mot de passe suffisamment sécurisé. J'ai également demandé aux utilisateurs de retaper le mot de passe pour s'assurer qu'ils ont correctement saisi le mot de passe souhaité.

```
const validateEmail = (email) => {
  const emailre =
    /^(([^<>(){}\.,;:\s@"]+)(\.(^<>(){}\.,;:\s@"]+)*))|((.
    +"))@((\[[0-9]{1,3}\.\[0-9]{1,3}\.\[0-9]{1,3}\.\[0-9]{1,3}
    \])|(((a-zA-Z)-0-9)+\.)+[a-zA-Z]{2,}))$/;
  return emailre.test(String(email).toLowerCase());
};
```

Fig 10 : Regex⁴ pour vérifier la forme de l'email rempli

```
const validatePassword = (password) => {
  const passwordre = /^.{6,}$/;
  return passwordre.test(password);
};
```

Fig 11 : Regex⁴ pour vérifier la robustesse du mot de passe

Ensuite, j'ai ajouté des contraintes pour l'ajout du logo et de la bannière, en spécifiant un format particulier et une taille maximale autorisée de 2 Mo. De même, pour les réseaux sociaux de l'entreprise, j'ai utilisé une regex⁴ pour s'assurer que le format soit une URL valide. Si ce n'est pas le cas, une erreur est déclenchée et le texte saisi n'est pas pris en compte lors de la création du compte. Les informations remplies sont stockées dans une variable useState⁵ appelée formData, qui contient le nom de l'input et la valeur de l'input, mise à jour à chaque modification. Les images sont stockées dans d'autres variables. Une fois le formulaire validé, je l'envoie au backend avec un fetch vers mon serveur Node.js. Ensuite, j'exécute une fonction qui vérifie si l'email n'est pas déjà utilisé, stocke les images dans un dossier spécifique, sécurise le mot de passe avec bcrypt, et stocke les valeurs dans la base de données avec Sequelize.

Une fois le compte créé, je me suis chargé de la partie profil, avec l'affichage des informations saisies et la possibilité de les modifier. Pour cette partie, je n'étais pas inquiet jusqu'à la modification de l'icône de profil et de la bannière du compte. Je me suis demandé comment permettre la modification de l'image en cliquant dessus. Après réflexion, j'ai pensé à mettre le bouton pour ajouter un fichier en position absolute et en opacity 0 pour le faire "disparaître", et cela a fonctionné.

Un autre problème est survenu lorsque j'ai eu l'idée de faire apparaître le nouveau logo à la place de l'ancien avant la confirmation de l'utilisateur, pour qu'il puisse voir le rendu. Je n'arrivais pas à réafficher le logo par-dessus. Après de nombreuses recherches et essais infructueux, j'ai finalement réalisé ce code pour le changement du logo :

```
const handleLogoChange = (e) => {
    const file = e.target.files[0];
    if (file && file.size <= 2 * 1024 * 1024) {
        setIsLogo(URL.createObjectURL(file));
        setLogFile(file);
        setConfirmLogo(true);
        setAcceptedFile(true);
        console.log(file);
    } else {
        setConfirmLogo(false);
    }
};
```

Fig 12 : Fonction exécuté lors de la modification du logo

Quand je reçois le logo, je le récupère dans une variable file. Je vérifie si la taille du fichier ne dépasse pas 2 Mo. Si c'est le cas, je crée une URL pour l'image contenue dans file que je stocke dans la variable setIsLogo, qui est la variable qui stocke la source de l'image affichée. Je stocke également le fichier dans une variable logoFile pour l'envoyer par la suite si l'utilisateur valide le formulaire, et je passe acceptedFile à true pour confirmer que le fichier respecte les restrictions. Si le logo n'est pas conforme, je définis confirmLogo à false pour afficher ensuite une erreur.

Pour finir, j'ai créé une page permettant de rechercher les entreprises par leur nom et leur domaine d'activité, afin de voir les entreprises et les offres qu'elles proposent.

2.4.2 Développement de la partie de connexion

Maintenant que j'ai un compte entreprise, je peux me connecter à ce compte. J'ai donc créé un espace de connexion simple avec un formulaire contenant un email et un mot de passe. Ensuite, les informations sont envoyées au back-end où je vérifie si l'email existe dans la base de données. Si ce n'est pas le cas, je renvoie une erreur : "email invalide". Si l'email est valide, je récupère le mot de passe associé et je le compare avec celui fourni, grâce à bcrypt.

En cas d'erreur de mot de passe, je renvoie une erreur : "mot de passe erroné". Si le mot de passe est correct, je crée un token de connexion avec le module JWT⁶ (JSON Web Token) que je renvoie en réponse. Ce token est ensuite stocké en frontend dans le localStorage⁷ avec un autre token représentant la durée de validité du premier token.

Ensuite, pour chaque requête nécessitant une authentification utilisateur, le frontend vérifie l'existence et la validité du token avant d'envoyer la requête au backend. Si l'une des conditions (existence ou validité du token) n'est pas remplie, l'utilisateur est redirigé vers la page de connexion. Sinon, le token est envoyé dans le header de la requête pour vérification.

Le backend reçoit la requête et vérifie le token avant d'exécuter les actions demandées. Si le token n'est pas reconnu ou n'est pas valide (par exemple, s'il a été modifié ou expiré), une erreur est renvoyée et la requête n'est pas traitée plus avant. Si le token est valide, les informations stockées dans le token sont ajoutées à la requête sous la forme d'un tableau 'user', contenant les détails de l'utilisateur. Cela permet aux autres fonctions de connaître l'identité de l'utilisateur et de gérer les permissions en conséquence.

```
const authenticateJWT = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  if (!authHeader) return res.sendStatus(401).json({ message: 'error' });
  const token = authHeader.split(' ')[1];
  jwt.verify(token, secretKey, (err, user) => {
    if (err) return res.sendStatus(403).json({ message: 'error' });
    req.user = user;
    next();
  });
};
```

Fig 13 : Fonction de vérification du token JWT⁶

2.4.3 Développement Des Offres

Maintenant que j'ai un compte d'entreprise et que je peux me connecter, j'ai décidé de m'attaquer à la création des offres d'emploi. Pour créer les offres, il faut passer par le compte d'entreprise. J'ai donc suivi le même schéma pour les entreprises, en recherchant sur d'autres sites les éléments indispensables pour une offre, et j'ai utilisé le CLI³ pour créer le modèle des offres.

Une fois que j'ai pu créer les offres, j'ai développé deux types de cartes : une pour les afficher sur les pages des entreprises et sur la page principale, où elles sont triées par ordre décroissant de la date de création, et une autre pour la page la plus importante et la plus difficile à réaliser de l'application : la page de filtrage des offres.

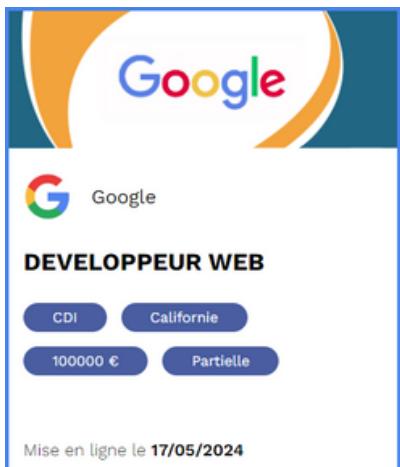


Fig 14 : Carte d'un emploi présent sur le site



Fig 15 : Carte d'un emploi dans la page de recherche des offres d'emplois

Avant de me lancer dans la page de filtrage, j'ai d'abord réfléchi aux critères que j'allais filtrer. J'ai donc décidé de filtrer les offres selon la possibilité de télétravail, le domaine d'activité, le salaire, le type de contrat et les mots-clés saisis dans la barre de recherche. Maintenant que j'avais identifié tous les critères de filtrage, j'ai d'abord créé la page sans les interactions de filtre. J'ai rencontré des difficultés pour déterminer comment filtrer les offres par salaire. Devais-je utiliser un input avec une valeur d'entrée et un bouton pour choisir les offres au-dessus ou en-dessous, ou bien un slider? Mais comment implémenter un slider? J'ai donc opté pour le slider et j'ai exploré la librairie "Material UI" pour comprendre son fonctionnement et le personnaliser à ma guise. Une fois le CSS complété, il ne me restait plus qu'à établir les liaisons avec la base de données.

Cependant, effectuer une requête lorsqu'un filtre est activé est une chose, mais il fallait aussi pouvoir cumuler les filtres. Pour cumuler les filtres, j'ai d'abord envisagé de créer une fonction complexe avec des conditions 'if' et 'else'. L'idée était de vérifier si le filtre était inclus dans la requête, puis de rechercher les offres correspondantes à ce filtre. Ensuite, si un autre filtre était également actif, je devais appliquer ce filtre en plus du premier, et ainsi de suite. Cependant, cette approche aurait rendu la fonction très lourde et très longue, et j'étais convaincu qu'il existait une solution plus concise.

Après de longues heures de recherche, j'ai décidé de me tourner vers l'ORM² Sequelize pour voir s'il était possible de cumuler les conditions de filtrage de manière plus efficace. Heureusement, j'ai découvert qu'il était effectivement possible de le faire en utilisant un objet qui contient le nom de la valeur à filtrer dans la base de données ainsi que la valeur du filtre. Grâce à cette approche, que vous pouvez retrouver dans l'annexe **n°6**, j'ai réussi à rendre cette page de filtrage très efficace et performante et vous pouvez aussi la retrouver dans l'annexe **n°7**.

Après avoir finalisé la page de filtrage, il ne me restait plus qu'à créer une page pour les offres et à la rendre fonctionnelle au clic avec les cartes créées, ce qui ne m'a posé aucun problème.

2.4.4 Développement Des Comptes utilisateurs

Pour le développement des comptes utilisateurs, j'ai repris le principe déjà utilisé pour les formulaires des entreprises et des offres. La grande différence réside dans l'ajout de certaines informations propres à un utilisateur, comme les diplômes qu'il possède ou ses expériences passées dans d'autres entreprises. Cependant, demander autant d'informations dès l'inscription pourrait décourager l'utilisateur. J'ai donc décidé de rendre certains champs, comme les diplômes, facultatifs et complétables dans l'espace profil de l'utilisateur. De plus, j'ai carrément décidé de passer les expériences directement dans l'espace profil, car cela représentait sinon un volume trop important pour une simple inscription.

Une fois un compte utilisateur créé, toutes les informations peuvent être modifiées depuis le profil. Le point complexe avec les diplômes est qu'on ne peut pas entrer n'importe quoi. J'ai donc créé un grand fichier JSON qui répertorie la majorité des diplômes disponibles. J'aurais pu développer un système plus complexe, mais malheureusement, je n'avais pas assez de temps pour cela. Ce sera donc un point à améliorer ultérieurement.

J'ai maintenant implémenté la possibilité pour un utilisateur de choisir un diplôme, mais un autre problème est apparu : il fallait pouvoir ajouter plusieurs diplômes et les supprimer tant que l'ajout n'était pas validé. J'ai consacré beaucoup de temps à permettre à l'utilisateur d'ajouter autant de sélecteurs de diplômes qu'il le souhaite et à conserver la valeur de chaque sélecteur pour les envoyer ensuite vers la base de données.

Lorsque l'utilisateur valide les modifications, j'envoie les diplômes au backend sous forme de JSON pour les stocker dans la base de données. Cependant, après avoir fait cela, je me suis rendu compte d'un autre problème : la présence potentielle de doublons. J'ai donc dû mettre en place un mécanisme de vérification pour m'assurer qu'il n'y ait pas de doublons dans les valeurs avant de les transformer en JSON.

Une fois la gestion des diplômes complétée, il ne me restait plus qu'à réaliser une page utilisateur, qui sera utilisée lorsque les utilisateurs pourront postuler aux offres d'emploi. Cette page doit afficher les informations pertinentes et permettre aux utilisateurs de gérer leurs candidatures de manière efficace

Fig 16 : Visuelle de la modification des diplômes sur le site

2.4.5 Développement du système d'offre d'emploi

Nous voici à la dernière partie majeur du projet : la mise en place de la fonctionnalité permettant de postuler à une offre d'emploi. Pour cela, j'ai d'abord rendu fonctionnelle la possibilité de postuler à une annonce. J'ai créé une table offreuser (détailée dans l'annexe n°5), qui contient l'ID de l'utilisateur, l'ID de l'entreprise et le statut de la demande.

Ensuite, j'ai dû concevoir un moyen pour que les entreprises puissent consulter les candidatures des utilisateurs. Pour cela, j'ai créé une page utilisateur permettant aux entreprises d'accéder aux expériences, diplômes, etc., des utilisateurs. Afin de gérer un grand nombre de candidatures, j'ai eu l'idée de créer une page intermédiaire présentant des cartes résumant les informations clés des utilisateurs, telles que le nom, le prénom, la photo de profil, le CV, les diplômes, un lien vers la page détaillée de l'utilisateur, et des boutons pour accepter ou refuser la demande.

Toutes les candidatures sont affichées par ordre décroissant de date de postulation, afin que l'entreprise voie en premier les demandes les plus récentes. Si une demande est acceptée par l'entreprise, l'email de l'utilisateur lui est fourni pour qu'ils puissent entamer des discussions sur la suite du processus de recrutement.



Fig 17 : Carte de candidature sur le site

Du côté utilisateur, il est maintenant possible de voir les offres auxquelles on a postulé dans un onglet du profil. Cet onglet permet à l'utilisateur de suivre l'avancement de ses demandes. J'ai mis en place trois statuts pour les demandes : "refusé", "en attente" et "accepté".

Si une demande est refusée, un bouton redirige l'utilisateur vers la page des offres. Si une demande est acceptée, une alerte informe l'utilisateur que son email de contact a été transmis au recruteur, qui le contactera pour la suite du processus de recrutement.

Dans le temps imparti, c'est ce que j'ai pu accomplir. Mon objectif pour le reste de mon stage est de créer un système de chat permettant à l'utilisateur et à l'entreprise de communiquer directement depuis le site. Cela permettra de faciliter les échanges et de rendre le processus de recrutement plus fluide et interactif.

2.5 Bilan du projet

Après avoir rendu fonctionnelle la demande d'offre, j'ai achevé la création d'un MVP (Minimum Viable Product) pour mon site de recherche d'emploi. Pour finaliser mon projet, j'ai décidé de me concentrer sur l'optimisation du site.

Premièrement, j'ai limité la taille maximale des fichiers uploadés pour éviter de surcharger le serveur et améliorer les performances. De plus, j'ai restreint les données renvoyées par le backend au strict minimum nécessaire, ce qui a permis de réduire le temps de traitement et la taille des requêtes. Cette optimisation a également renforcé la sécurité en évitant de transmettre des données non désirées ou potentiellement dangereuses, comme les mots de passe, même s'ils sont cryptés.

Ensuite, j'ai utilisé le module 'react-helmet' pour configurer dynamiquement les balises '<head>' de chaque page. Cela m'a permis de personnaliser le titre et la métadescription pour chaque route, améliorant ainsi le SEO et l'expérience utilisateur. J'ai également défini quelles pages devaient être indexées par les moteurs de recherche et lesquelles ne devaient pas l'être, en fonction de leur contenu et de leur pertinence.

Avec ce MVP optimisé en main, j'ai décidé de l'héberger afin de quitter l'état de développement et de passer en production. L'hébergement de ce type de projet nécessite une recherche approfondie, car je n'avais jamais hébergé un projet avec un serveur Node.js auparavant. J'ai donc décidé de réaliser cette tâche sur mon temps libre afin de ne pas impacter mon temps de développement dédié à l'amélioration de certains aspects du site.

Après cinq heures de recherche et d'essais infructueux, j'ai enfin réussi à héberger le projet, et tout fonctionne bien au moment où j'écris ces lignes. Mon projet est désormais accessible à l'adresse : jobodyssey.quentinbrandy.fr.

3. Conclusion

Cette conclusion se divisera en deux parties : tout d'abord, un bilan professionnel de mon stage, suivi d'un bilan personnel.

Bilan professionnel :

Tout au long de ce rapport de stage, je pense avoir globalement répondu à la problématique principale, qui était : **Comment développer un site de recherche d'emploi optimal et intuitif qui puisse répondre aux besoins des entreprises et des chercheurs d'emploi tout en assurant une expérience utilisateur fluide et efficace ?**

En deux mois, j'ai pu réaliser un MVP (Minimum Viable Product) fonctionnel, où les utilisateurs peuvent rechercher et postuler à des offres, et où les entreprises peuvent visualiser et gérer les candidatures. J'ai réussi à implémenter les fonctionnalités essentielles, notamment la gestion des comptes utilisateurs et entreprises, ainsi que le système de suivi des candidatures.

Le travail réalisé a permis de mettre en place une base solide pour le site. Toutefois, plusieurs points restent à améliorer pour que le site soit pleinement abouti. Par exemple, une amélioration graphique du site est nécessaire car le style actuel est très simple, dû aux seulement deux mois à ma disposition, et pourrait bénéficier de grandes améliorations pour le rendre plus moderne. L'accessibilité du site doit également être revue sur certaines parties du site mobile pour garantir une meilleure lisibilité et une navigation fluide. Il y a bien sûr d'autres points, comme la sécurité du site et son optimisation, que je mentionne juste ici.

Bilan personnel :

D'un point de vue personnel, ce stage m'a apporté énormément de choses. Il m'a conforté encore plus dans l'idée de vouloir faire du développement web mon métier à l'avenir.

Ce stage m'a également beaucoup appris sur la vie dans une petite entreprise. J'ai pu observer de près la gestion quotidienne de l'entreprise, comprendre les défis et les avantages de travailler dans une structure à taille humaine, et apprécier la polyvalence nécessaire pour s'adapter à différentes tâches.

Il m'a aussi appris à mieux gérer mon temps et à organiser mon travail de manière efficace. J'ai appris à fixer des objectifs clairs, à prioriser les tâches et à faire face aux imprévus. Cette expérience m'a permis de renforcer ma capacité à travailler de manière autonome tout en restant ouvert à la collaboration et aux conseils des autres. J'ai notamment, pendant mon stage, rencontré d'autres stagiaires en développement web qui ont eu le même projet que moi à réaliser, ce qui nous a permis d'échanger sur nos méthodes, la direction que nous allions prendre pour le site, et nous pouvions donc aussi nous entraider et nous donner des conseils pour améliorer l'application.

Ce projet m'a captivé au point que je prévois de continuer à développer l'application même après la fin de mon stage. L'idée de voir mon travail évoluer et s'améliorer au fil du temps me motive énormément.

En résumé, ce stage a été une expérience enrichissante tant sur le plan professionnel que personnel. Il a confirmé mon choix de carrière, élargi mes compétences techniques et personnelles, et m'a offert une vision précieuse du monde de l'entreprise. Vous pouvez retrouver mon application sur jobodyssey.quentinbrandy.fr.

4. Glossaire

MVC (Model-View-Controller) : Modèle d'architecture logicielle qui sépare une application en trois composants principaux : le modèle (Model), la vue (View) et le contrôleur (Controller). Le

1. modèle gère les données et la logique métier, la vue est responsable de l'affichage, et le contrôleur traite les entrées de l'utilisateur et les envoie au modèle.

ORM (Object-Relational Mapping) : Technique de programmation utilisée pour convertir des données entre des systèmes de types incompatibles dans des bases de données relationnelles

2. et des langages de programmation orientés objet. Un ORM permet de manipuler la base de données à l'aide d'objets, facilitant ainsi les opérations CRUD (Create, Read, Update, Delete).

CLI (Command Line Interface) : Interface utilisateur qui permet aux utilisateurs d'interagir avec le logiciel ou le système d'exploitation en tapant des commandes textuelles. Les CLI sont

3. souvent utilisées pour les tâches d'administration système, la programmation et le développement logiciel.

Regex (Expressions régulières) : Outil de recherche et de manipulation de texte qui utilise des motifs (patterns) pour identifier des chaînes de caractères spécifiques dans un texte. Les

4. regex sont puissantes pour valider les formats de données, extraire des informations et effectuer des remplacements de texte.

useState : Dans React, permet d'ajouter un état local à une fonction composant. Il renvoie une paire : la valeur de l'état actuel et une fonction qui permet de le mettre à jour. C'est essentiel

5. pour gérer les interactions utilisateur et le rendu dynamique dans les applications React.

JWT (JSON Web Token) : Un standard ouvert utilisé pour créer des tokens d'accès qui permettent de sécuriser les échanges de données entre parties. Les JWT sont souvent utilisés

6. pour l'authentification et l'autorisation dans les applications web, assurant que les requêtes proviennent de sources authentifiées.

LocalStorage : Fonctionnalité du navigateur web qui permet de stocker des données de manière persistante sur l'ordinateur de l'utilisateur.

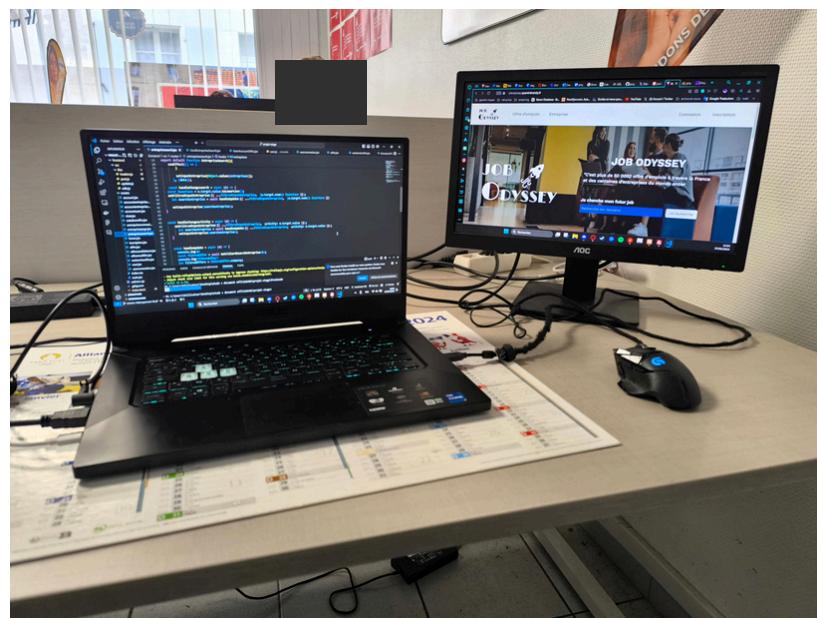
- 7.

6. Annexes



Annexe numéro 1

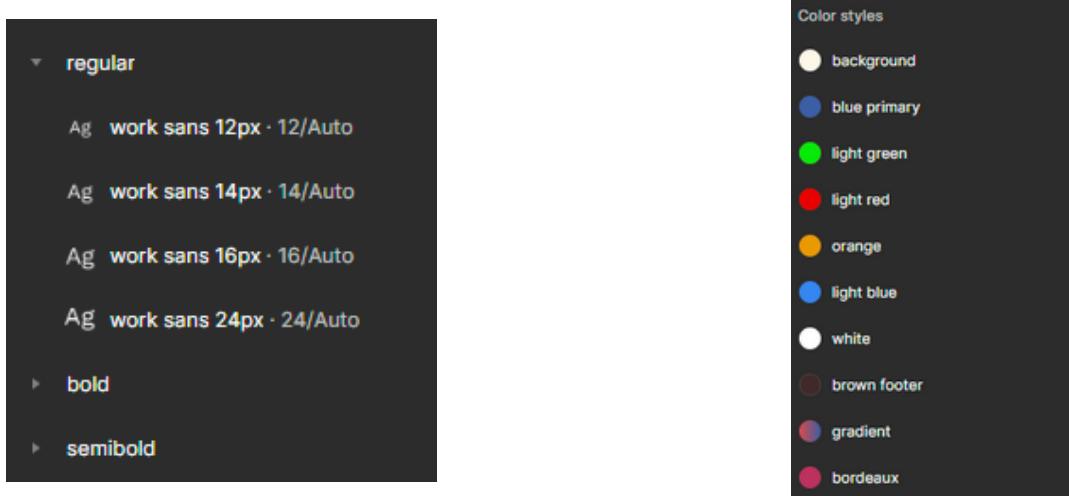
Devanture de l'agence fw16



Annexe numéro 2

Poste de travail

6. Annexes



Annexe numéro 3

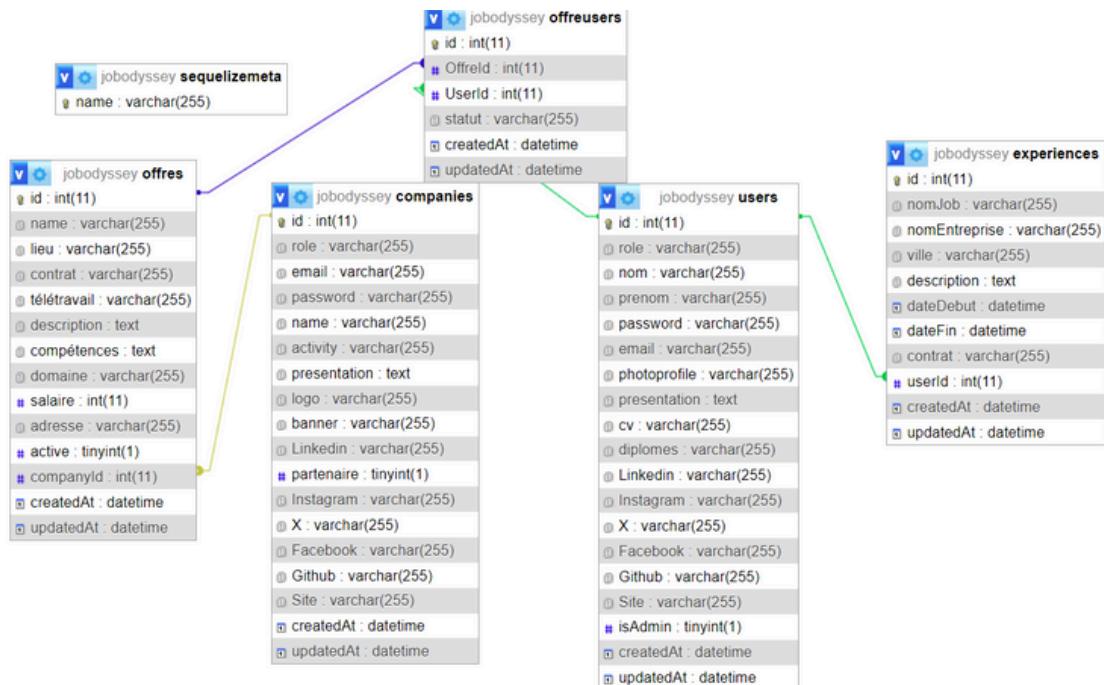
Style et police utilisé pour mon projet stocker dans les styles local.

```
export async function getOffer(offreid) {
  try {
    const response = await fetch(
      `http://localhost:3000/api/getoffer?offreid=${offreid}`,
      {
        headers: {
          "Content-Type": "application/json",
          Accept: "application/json",
        },
        method: "GET",
      }
    );
    const data = await response.json();
    return data;
  } catch (error) {
    console.error("Error getting offer", error.message);
    throw error;
  }
}
```

Annexe numéro 4

Exemple d'une requête utilisé dans le projet

6. Annexes



Annexe numéro 5

Schéma de la base donnés final du projet

```
const getFiltredOffer = async (req, res) => {
  let searchConditions = {};
  // Série de if else qui vérifie si les options sont remplies de l'offre est renseigné
  if (req.body.search && req.body.search !== "") {
    searchConditions.name = { [Op.like]: `%${req.body.search}%` };
  }
  if (req.body.domaine && req.body.domaine.length > 0 && req.body.domaine !== "1") {
    searchConditions.domaine = { [Op.like]: req.body.domaine };
  }
  if (req.body.salaire) {
    searchConditions.salaire = { [Op.between]: [req.body.salaire[0], req.body.salaire[1]] };
  }
  if (req.body.contrat && req.body.contrat.length > 0) {
    searchConditions.contrat = { [Op.in]: req.body.contrat };
  }
  if (req.body.télétravail && req.body.télétravail.length > 0) {
    searchConditions.télétravail = { [Op.in]: req.body.télétravail };
  }
  // Effectue la requête avec toutes les conditions dans l'ordre décroissant pour prendre les plus récentes d'abord
  const offres = await Offre.findAll({
    order: [['createdAt', 'DESC']],
    include: [
      {
        model: Company,
        as: 'company',
        attributes: ['name', 'logo']
      }
    ],
    where: searchConditions,
  });
  return res.status(201).json({ offres });
};
```

Annexe numéro 6

Fonctions de recherche des offres en fonctions des filtres dans la base de données

6. Annexes

Recherche tes offres :

recherche ton job

secteur d'emploi

INGÉNIEUR EN ÉNERGIES RENOUVELABLES

CDI Paris, France 50000€ télétravail Partielle

♥

DATA SCIENTIST

CDI Lyon, France 60000€ télétravail Partielle

♥

DÉVELOPPEUR LOGICIEL FULL STACK

CDI Paris, France 55000€ télétravail Interdit

♥

INGÉNIEUR EN ÉNERGIES RENOUVELABLES

CDI Paris, France 50000€ télétravail Partielle

♥

INGÉNIEUR EN ÉNERGIES RENOUVELABLES

CDI Paris, France 50000€ télétravail Partielle

♥

Annexe numéro 7

Page de filtre fonctionnel