

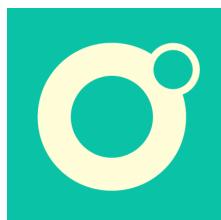
DOSSIER DE PROJET



Handi-Vision

PRÉSENTÉ PAR
ABDESELAM BIMKHOUD

TITRE PROFESSIONNEL
DÉVELOPPEUR WEB ET WEB MOBILE



ECOLE O'CLOCK
[PROMOTION QILIN 2023-2024]

SOMMAIRE

REMERCIEMENTS	3
INTRODUCTION	4
LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET	5
<i>I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité</i>	5
A. Maquetter une application	5
B. Réaliser une interface utilisateur web statique et adaptable	5
C. Développer une interface utilisateur web dynamique	5
<i>II. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité</i>	6
A. Créer une base de données	6
B. Développer les composants d'accès aux données	7
C. Développer la partie back-end d'une application web ou web mobile	7
RÉSUMÉ DU PROJET	8
CAHIER DES CHARGES	8
<i>I. Présentation du projet</i>	8
<i>II. Définition du besoin</i>	9
<i>III. Cible du projet</i>	9
<i>IV. Fonctionnalités du projet</i>	10
A. Inscription et Profils Utilisateur	10
B. Gestion des Profils Candidat	10
C. Gestion des Profils Entreprises (Recruteur)	10
D. Gestion des Offres d'Emploi	10
E. Système de Matching de Base	11
<i>V. Livraison du MVP</i>	11
<i>VI. Description des solutions</i>	11
<i>VII. Accessibilité</i>	12
<i>VIII. Fonctionnalités attendues par le client</i>	12
<i>IX. Navigateurs compatibles</i>	14
<i>X. Évolutions potentielles</i>	14
MÉTHODES DE TRAVAIL	16

CONCEPTION DU PROJET	17
<i>II. Users Stories</i>	17
<i>II. Arborescence</i>	19
<i>III. MVP</i>	19
<i>IV. Fonctionnalités Détaillées des Pages</i>	20
<i>V. Wireframes</i>	23
SPÉCIFICATIONS TECHNIQUES	24
<i>I. Technologies</i>	24
<i>II. Création de la base de données</i>	25
A. MCD	25
B. MLD	27
C. MPD	28
<i>III. Back-end</i>	29
<i>IV. Connexion base de données</i>	32
<i>V. Front-end</i>	35
<i>VI. Accessibilité</i>	36
<i>VI. Harmonisation des Processus de Travail</i>	38
RÉALISATIONS PERSONNELLES	39
<i>I. Upload de fichiers</i>	40
A. Back	40
B. Front	44
VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE	49
<i>I. Sequelize</i>	49
<i>II. JSON Web Token (JWT)</i>	49
<i>III. Cryptage du mot de passe utilisateur</i>	52
<i>IV. Protection route back avec roleCheck</i>	53
<i>V. Protection route front avec L'élément privateRoute</i>	53
PRÉSENTATION DU JEU D'ESSAI	53
PROCESSUS DE RECHERCHE ET TRADUCTION	58
CONCLUSION	59
ANNEXE	61

REMERCIEMENTS

Je tiens à exprimer ma profonde gratitude à l'entreprise Sigma-Vision, représentée par notre client Monsieur Gabriel Ribo, qui nous a donné l'opportunité de travailler sur ce projet passionnant. Leur confiance et leur soutien ont été essentiels à la réussite de notre travail.

Je souhaite également remercier chaleureusement l'école O'Clock et toute l'équipe pédagogique pour leur enseignement de qualité, leur accompagnement et les connaissances précieuses qu'ils ont partagées avec nous.

Un grand merci à mon groupe de travail pour leur collaboration, leur engagement et l'esprit d'équipe dont ils ont fait preuve tout au long de ce projet. Travailler avec vous a été à la fois un plaisir et une source d'inspiration constante.

Enfin, je voudrais remercier l'ensemble de la promotion Qilin. Notre parcours ensemble a été une aventure incroyable, et j'ai énormément apprécié chaque moment passé à apprendre à vos côtés.

Votre soutien et votre dévouement ont joué un rôle crucial dans la réalisation de ce projet, et pour cela, je vous en suis infiniment reconnaissant.

INTRODUCTION

Mon parcours professionnel a débuté dans le domaine ferroviaire, où j'ai travaillé pendant plusieurs années. Bien que cette expérience ait été enrichissante, une certaine lassitude s'est installée avec le temps, me poussant à réévaluer mes aspirations professionnelles. Depuis mon plus jeune âge, j'ai toujours été fasciné par l'informatique et la technologie. Cette passion de longue date, alliée à mon désir de renouvellement professionnel, m'a naturellement orienté vers une carrière dans le développement web.

J'ai commencé à explorer le monde de la programmation et du développement web en autodidacte. Cette phase d'auto-apprentissage a été cruciale pour moi, me permettant de confirmer mon intérêt et de renforcer mes compétences de base dans ce domaine.

Fort de cette auto-formation et désireux d'approfondir mes connaissances, j'ai choisi de m'engager dans un parcours structuré et professionnel avec l'école O'clock. Cette formation en développement web et accessibilité représente une étape décisive dans ma reconversion professionnelle. L'approche pédagogique de l'école, axée sur la pratique et l'immersion, a parfaitement répondu à mes attentes et m'a permis de développer des compétences techniques solides et adaptées aux exigences actuelles du marché.

Le projet de groupe, que je détaille dans ce dossier, a été l'un des aspects les plus enrichissants de cette formation. Non seulement il a mis en pratique mes compétences acquises en autodidacte et à O'clock, mais il a également été une occasion exceptionnelle de travailler en équipe, de relever des défis réels et de contribuer à un produit fonctionnel et innovant. Ce projet illustre parfaitement mon évolution, depuis mes débuts en autodidacte jusqu'à la réalisation d'un travail collaboratif et professionnel, essentiel pour valider les compétences nécessaires à l'obtention de mon diplôme en développement web et web mobile.

LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

I. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

A. Maquetter une application

Avant de commencer le développement de notre application, nous avons décidé de nous concentrer initialement sur l'élaboration des documents de conception. Notre première étape a été de créer des "users stories" afin de définir précisément les fonctionnalités disponibles pour les utilisateurs de notre application. Ensuite, nous avons conçu des wireframes pour les versions mobile et desktop, permettant de visualiser l'agencement de nos pages.

B. Réaliser une interface utilisateur web statique et adaptable

Ce projet, conçu principalement pour les personnes en situation de handicap, exigeait une interface utilisateur simple, accessible et facile à utiliser, indépendamment du niveau de compétence informatique de l'utilisateur.

En outre, il était essentiel que l'application soit compatible avec tous les types de supports, y compris les ordinateurs de bureau, les tablettes et les smartphones. Cette polyvalence garantit que l'application peut être utilisée dans divers contextes et situations, offrant une expérience utilisateur cohérente et efficace sur tous les appareils. Cette approche inclusive et universelle renforce l'accessibilité de l'application, assurant qu'elle est non seulement adaptée aux besoins spécifiques des personnes en situation de handicap, mais également facilement accessible à tous, quelle que soit la plateforme utilisée.

C. Développer une interface utilisateur web dynamique

Suite à l'élaboration des User Stories, nous nous sommes attelés à dynamiser l'interface pour enrichir l'expérience utilisateur. Notre objectif était de rendre les interactions avec la plateforme plus intuitives et captivantes.

Par exemple, dans Handi-Vision, cela permet aux utilisateurs (candidats et recruteurs) de naviguer facilement à travers les diverses fonctionnalités, comme la consultation des offres d'emploi, la gestion des profils, et l'utilisation du système de matching.

Le développement d'une interface utilisateur web dynamique pour Handi-Vision n'est pas seulement une question d'esthétique ou de tendance technologique, mais un

élément fondamental pour garantir une expérience utilisateur efficace, accessible, et engageante, alignée sur l'objectif d'inclusion de la plateforme.

L'aspect dynamique assure également que l'interface s'adapte de manière réactive aux différentes actions de l'utilisateur. Cela comprend l'ajustement en temps réel des éléments de l'interface en fonction des interactions, comme l'affichage des résultats de recherche ou la mise à jour des informations de profil.

Le client a spécifiquement demandé l'utilisation de ReactJS pour le développement front-end, une technologie reconnue pour sa capacité à créer des interfaces dynamiques et réactives. Cela contribue à offrir une expérience utilisateur améliorée et plus efficace.

II. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

A. Créer une base de données

La conception de la base de données de Handi-Vision a été menée selon une méthodologie structurée et collaborative, parfaitement alignée avec les exigences du client et les besoins fonctionnels du projet. Ce processus a débuté par la formulation des user stories en groupe, une étape cruciale pour comprendre et définir clairement les attentes des utilisateurs finaux de la plateforme.

En nous appuyant sur les user stories établies, nous avons développé le Modèle Conceptuel de Données (MCD). Cette étape a été fondamentale pour visualiser la structure de la base de données et les relations entre les différentes entités. Le MCD a permis d'établir une représentation claire et organisée des données, facilitant la compréhension de leur interdépendance et de leur fonctionnement au sein de la plateforme.

Suite à l'établissement du MCD, notre équipe a procédé à la création du Modèle Logique de Données (MLD), qui a servi à définir de manière plus précise les tables, les champs et les relations au sein de la base de données. Cette phase a transformé le modèle conceptuel en une structure logique, en préparation pour la mise en œuvre effective.

Le Modèle Physique de Données (MPD) a suivi, traduisant le modèle logique en un schéma concret utilisable dans notre base de données. Cette transformation du MLD en MPD a impliqué la définition des paramètres techniques précis pour le stockage et l'organisation des données, en vue de leur implémentation effective dans MySQL, le système de gestion de base de données choisi pour Handi-Vision.

En plus de ces étapes, nous avons créé un fichier **migration.sql** pour la création réelle de la base de données. Ce fichier contient toutes les instructions SQL nécessaires pour établir la structure de la base de données conformément au MPD. Nous avons également élaboré un fichier **seeding.sql**, qui contient des données de base essentielles pour le fonctionnement initial de la base de données. Ces fichiers sont essentiels pour assurer une mise en place efficace et une initialisation rapide de la base de données.

L'utilisation de MySQL, à la demande du client, a été un choix stratégique pour sa fiabilité et sa compatibilité avec les exigences du projet.

B. Développer les composants d'accès aux données

Dans notre projet Handi-Vision, le développement des composants d'accès aux données a été réalisé en utilisant Sequelize, un ORM (Object-Relational Mapping) populaire pour Node.js. Sequelize nous a permis de simplifier les interactions avec notre base de données MySQL, en offrant une abstraction de haut niveau pour l'exécution des requêtes et la gestion des données.

En utilisant Sequelize, nous avons pu bénéficier d'une approche plus structurée et simplifiée pour l'accès aux données. Cela a non seulement rendu le code plus lisible et maintenable, mais a également facilité la gestion des relations entre les données, les transactions, et la migration de la base de données. Ce choix s'aligne avec notre objectif de créer une application robuste, efficace et facile à maintenir.

C. Développer la partie back end d'une application web ou web mobile

Pour le développement du back-end de notre application Handi-Vision, nous avons opté pour Express, un framework pour Node.js. Cette approche nous a permis de structurer efficacement l'application en définissant diverses routes, chacune étant associée à des contrôleurs spécifiques.

Dans notre architecture, les routes servent de points d'entrée pour différentes fonctionnalités de l'application, guidant les requêtes des utilisateurs vers les contrôleurs appropriés. Ces contrôleurs sont ensuite responsables de la gestion de la logique métier, traitant les requêtes et renvoyant les réponses adéquates.

En plus de la configuration des routes et des contrôleurs, nous avons mis en place plusieurs middlewares pour renforcer la fonctionnalité de notre application. Parmi eux, des middlewares dédiés à l'authentification et à la validation des données se sont révélés cruciaux. Le middleware d'authentification assure la sécurité de l'application en vérifiant l'identité des utilisateurs avant de leur permettre d'accéder à certaines routes.

RÉSUMÉ DU PROJET

Le projet Handi-Vision est né de la volonté et de l'engagement de l'entreprise Sigma-Vision dont l'ambition est de faciliter l'accès à l'emploi aux personnes en situation de handicap.

C'est dans ce contexte que notre équipe a eu la responsabilité de développer la plateforme Handi-Vision.

Handi-Vision est une plateforme web innovante visant à faciliter l'emploi des personnes en situation de handicap. L'objectif principal est de créer un lien entre employeurs et travailleurs handicapés, promouvant une inclusion professionnelle plus forte. Ce projet a été réalisé en groupe dans le cadre d'un projet tutoré avec l'école O'cock.

Les fonctionnalités clés incluent l'inscription et la gestion des profils, pour les candidats et les recruteurs, la publication, la gestion et la consultation des offres d'emploi, ainsi qu'un système de matching de base alignant les compétences techniques des candidats avec celles requises par les offres. La plateforme dispose également d'un administrateur qui a la responsabilité de valider les inscriptions des candidats et des recruteurs, ainsi que les offres d'emploi publiées.

Le projet cible les entreprises engagées envers l'inclusion et les personnes handicapées en quête d'opportunités professionnelles.

Le projet adhère au RGAA pour l'accessibilité et utilise Bootstrap 5.0, ReactJS, NodeJS, Express JS, et MySQL pour le développement.

Ce projet vise à briser les barrières à l'emploi pour les personnes handicapées et à promouvoir une égalité des chances.

CAHIER DES CHARGES

I. Présentation du projet.

Le site handi-vision.fr sera accessible aux personnes en situation de cécité, de surdicécité et aux personnes malvoyantes, pour ce faire nous développons un outil pour la mise en relation des personnes à la recherche d'emploi et des entreprises. Son objectif principal est de faciliter la connexion entre les employeurs et les travailleurs en situation de handicap, en favorisant une inclusion professionnelle accrue.

Cet outil est conçu comme un système de matching visant à collecter et à mettre en correspondance les données des candidats (niveau de handicap, compétences, expérience, etc.) avec les exigences d'un poste vacant spécifique. Il s'appuie sur des

algorithmes avancés pour évaluer la compatibilité entre les candidats et les postes, optimisant ainsi le processus de recrutement.

Pour les employeurs, le site sera une plateforme centralisée pour la publication d'offres d'emploi inclusives, mettant en avant les avantages de l'embauche de travailleurs en situation de handicap.

En somme, handi-vision.fr aspire à être un catalyseur de l'inclusion professionnelle en créant un environnement où les talents des personnes en situation de handicap sont reconnus et valorisés. Cette plateforme innovante contribuera à réduire les barrières à l'emploi et à créer des opportunités équitables pour tous les individus, quel que soit leur handicap.

II. Définition du besoin.

La plateforme pourrait aider les entreprises à identifier leurs besoins en matière d'emploi pour les personnes en situation de handicap et à les mettre en correspondance avec des candidats appropriés.

III. Cible du projet.

➤ Entreprises

Les entreprises désireuses d'offrir des opportunités d'emploi aux personnes en situation de handicap constituent une partie importante de la cible. La plateforme vise à sensibiliser et à encourager ces entreprises à embaucher des candidats en situation de handicap en les aidant à identifier leurs besoins spécifiques en matière d'emploi et en mettant en correspondance ces besoins avec des candidats adaptés.

➤ Personnes en situation de handicap

Les personnes en situation de handicap qui recherchent activement un emploi ou une opportunité d'insertion professionnelle constituent également la cible de ce projet. La plateforme vise à les aider à trouver des opportunités qui correspondent à leurs compétences et à leurs aptitudes, tout en luttant contre les stéréotypes et les préjugés liés à l'handicap.

IV. Fonctionnalités du projet.

A. Inscription et Profils Utilisateur:

- Les candidats peuvent s'inscrire avec leur nom, prénom, adresse e-mail, mot de passe, ect(spécifier ou non leur handicap).
- Les entreprises peuvent s'inscrire avec leur nom, adresse e-mail, mot de passe, et spécifier leur secteur d'activité.

B. Gestion des Profils Candidat

- Les candidats peuvent créer un profil qui comprend : civilité, poste recherché, mobilité géographique, numéro de téléphone, expérience, compétences techniques, compétences relationnelles et qualités personnelles.
- Les candidats peuvent modifier ou supprimer leur profil à tout moment

C. Gestion des Profils Entreprises (Recruteur)

- Le recruteur peut et doit Créer un Profil entreprises qui comprend : secteur d'activité, raison sociale (pour les personnes morales), nom (pour les personnes physiques), adresse, liens vers les réseaux sociaux, numéro de téléphone principal, numéro de téléphone portable (pour les personnes physiques), e-mail, site Web, code NAF principal, effectif, statut juridique, politique de télétravail.
- Les recruteurs peuvent modifier leur profil à tout moment.

D. Gestion des Offres d'Emploi :

- Les recruteurs peuvent publier des offres d'emploi comprenant : poste, lieu du poste, type de contrat, durée de contrat, horaires, expérience requise, salaire, politique de télétravail, compétences techniques requises, compétences relationnelles et qualités personnelles recherchées.

- Les recruteurs peuvent modifier ou supprimer leurs offres d'emploi.

E. Système de Matching de Base :

- Le système de matching compare les compétences techniques des candidats avec les compétences requises pour les offres d'emploi.
- Les candidats peuvent voir les offres d'emploi correspondantes à leurs compétences.
- Les entreprises peuvent voir les candidats correspondant aux compétences requises pour leurs offres.

V. Livraison du MVP

Le MVP sera déployé sur un serveur web pour permettre aux utilisateurs de s'inscrire, de créer des profils, de publier des offres d'emploi, et d'utiliser le système de matching de base.

N'oublions pas que le MVP est une version minimale de notre application, conçue pour valider le concept et recueillir des retours d'utilisateurs. Nous pourrons ensuite revenir dessus et ajouter des fonctionnalités plus avancées en fonction des besoins et des commentaires des utilisateurs.

VI. Description des solutions.

Graphismes et aspect visuel

 Contraste de couleurs conformes au RGAA

Palette couleur

#F28500 - orange;

#1A2D41 - noir;

#032F91 - bleu foncé;

Accessible color combinations

Please don't use these color combinations; they do not meet a color contrast ratio of 4.5:1, as they do not conform with the standards of Section 508 for body text. This means that some people would have difficulty reading the text. Employing accessibility best practices improves the user experience for all users.

	White text #FFFFFF Aa	Noir text #1A2D41 Aa	Orange text #F28500 Aa	Bleu foncé text #002FB1 Aa
Bleu foncé background #002FB1	Aa			
Orange background #FF2800		Aa		
Noir background #1A2D41	Aa		Aa	
White background #FFFFFF		Aa		Aa

VII. Accessibilité.

Le développement du projet devra être conforme au Référentiel Général d'Amélioration de l'Accessibilité, (RGAA), dans sa version en vigueur au moment du projet.

VIII. Fonctionnalités attendues par le client.

Administration

L'outil doit permettre la définition de groupes-utilisateurs (par exemple, "Administrateur", "Candidat", "Entreprise") de façon à pouvoir intervenir à différent niveau sur une entité de l'outil (créer, modifier, supprimer, valider).

Page Candidat doit contenir

- Civilité, Prénom et Nom ;
- Poste recherché;
- Mobilité géographique;
- N° de téléphone ;
- E-mail ;
- Expérience;
- Compétences techniques (savoir-faire);
- Compétences relationnelles et qualités personnelles (savoir-être);

Page Entreprise doit contenir

- Secteur d'activité;
- La raison sociale / prénom et nom dans le cas d'une personne physique ;
- L'adresse;
- Un lien vers les réseaux sociaux (LinkedIn, Facebook, Twitter, Instagram,...)
- Téléphone principal (personne morale), ou fixe (personne physique) ;
- Téléphone portable (personne physique);
- E-mail ;
- Site Web ;
- Code NAF principal ;
- Effectif ;
- Statut juridique ;
- Politique de télétravail;

Page Offre d'emploi doit contenir

- Poste;
- Entreprise;
- Lieu du poste;
- Type de contrat;
- Durée de contrat;
- Horaires;
- Expérience;
- Salaire;
- Politique de télétravail;
- Compétences techniques;
- Compétences relationnelles et qualités personnelles recherchées;

Grille de handicap

Doit permettre de faire un matching entre la demande et les possibilités humaines du demandeur

Grille de matching

- Télétravail ou présentiel
- Plein temps ou temps partiels (nombre d'heure)
- Déplacement possible ou pas
- Compétence technique (logiciel ou autre) -> Permet un filtre de recherche
- Qualité humaine (liste de choix permettant une recherche)
- Poste;
- Entreprise;
- Lieu du poste;
- Type de contrat;
- Durée de contrat;

IX. Navigateurs compatibles.

Voici un tableau des Navigateurs compatible avec notre projet :

Navigateur	Bootstrap 5 (%)	React.js (%)	Node.js (%)
Google Chrome	Très Compatible	Très Compatible	N/A (Côté Serveur)
Mozilla Firefox	Très Compatible	Très Compatible	N/A (Côté Serveur)
Apple Safari	Très Compatible	Très Compatible	N/A (Côté Serveur)
Microsoft Edge	Très Compatible	Très Compatible	N/A (Côté Serveur)
Internet Explorer	Partiellement	Partiellement	N/A (Côté Serveur)
Opera	Très Compatible	Très Compatible	N/A (Côté Serveur)
Brave	Très Compatible	Très Compatible	N/A (Côté Serveur)
Android Browser	Très Compatible	Très Compatible	N/A (Côté Serveur)
iOS Safari	Très Compatible	Très Compatible	N/A (Côté Serveur)
Samsung Internet	Très Compatible	Très Compatible	N/A (Côté Serveur)
Node.js (Côté Serveur)	N/A (Côté Serveur)	N/A (Côté Serveur)	Très Compatible

X. Évolution(s) potentielle(s).

- **Élargissement de la Gamme de Services** : Élargir sa gamme de services pour inclure davantage d'outils et de ressources destinés aux personnes en situation de handicap. Par exemple, des fonctionnalités pour la formation professionnelle, l'orientation de carrière, ou encore des réseaux sociaux professionnels spécifiques.
- **Amélioration de l'Accessibilité** : Continuer à améliorer l'accessibilité de la plateforme est essentiel. Cela pourrait inclure la prise en charge de

nouvelles technologies d'assistance, des tests d'accessibilité réguliers et l'ajout de fonctionnalités conviviales pour les utilisateurs en situation de handicap.

- **Intégration de l'Intelligence Artificielle** : L'application pourrait utiliser des technologies d'intelligence artificielle pour améliorer le système de matching. Par exemple, l'IA pourrait analyser plus efficacement les profils des candidats et les offres d'emploi pour des correspondances plus précises.
- **Extension de la Couverture Géographique** : Élargir son service pour inclure des offres d'emploi et des candidats dans d'autres pays ou langues, permettant ainsi à un public plus large de bénéficier de la plateforme.
- **Programmes de Sensibilisation** : L'application pourrait mettre en place des programmes de sensibilisation auprès des entreprises pour encourager davantage l'embauche de travailleurs en situation de handicap. Cela pourrait inclure des ressources éducatives et des incitations pour les entreprises.
- **Intégration de Retours Utilisateurs** : Recueillir des retours d'utilisateurs et les intégrer dans le processus d'amélioration de la plateforme. Les commentaires des utilisateurs en situation de handicap sont essentiels pour identifier les domaines à améliorer.
- **Élargissement des Partenariats** : La plateforme pourrait établir des partenariats avec d'autres organisations ou entreprises œuvrant pour l'inclusion professionnelle des personnes en situation de handicap, ce qui pourrait renforcer son impact.
- **Soutien Mobile** : Développer une application mobile complémentaire pour permettre aux utilisateurs d'accéder à Handi-Vision depuis leurs appareils mobiles, ce qui amélioreraient encore la portabilité de la plateforme.
- **Réglementation et Conformité** : Continuer à suivre les évolutions législatives en matière d'inclusion professionnelle et d'accessibilité pour garantir que la plateforme reste en conformité avec les normes et les exigences réglementaires en constante évolution.

- **Témoignages et Succès** : Mettre en avant des témoignages et de succès d'individus en situation de handicap qui ont trouvé un emploi grâce à Handi-Vision, ce qui pourrait inspirer d'autres candidats et entreprises à rejoindre la plateforme.

MÉTHODE DE TRAVAIL

Nous utilisons une méthode de travail agile avec des sprints hebdomadaires du jeudi au jeudi pour gérer efficacement notre projet de développement web. Cette approche est particulièrement adaptée à notre structure d'équipe, qui est divisée en deux groupes distincts : l'équipe front-end et l'équipe back-end. En tant que développeur full-stack, je joue un rôle clé dans la coordination entre ces deux équipes, veillant à ce que les efforts de développement soient synchronisés et que les objectifs de chaque sprint soient atteints.

➤ Sprints Hebdomadaires (Jeudi à Jeudi)

- **Objectif** : Chaque sprint commence le jeudi et se termine le jeudi suivant. Cette périodicité permet une planification concise, une réactivité aux changements et une livraison continue de fonctionnalités.
- **Avantages** : Permet une révision et une adaptation régulières des objectifs, favorisant ainsi une amélioration continue et une flexibilité dans le développement du projet.

➤ Équipes Front-End et Back-End avec Rôle Full-Stack

- **Répartition des Rôles** : Deux équipes distinctes se concentrent sur les aspects front-end et back-end du développement.
- **Collaboration et Spécialisation** : Cette répartition permet une spécialisation des compétences tout en favorisant la collaboration inter-équipes pour une intégration fluide des différentes parties du projet.

➤ Communication et Outils

- **Utilisation de Discord** : Discord sert de plateforme principale pour la communication de l'équipe, offrant un moyen rapide et efficace de partager des informations, de résoudre des problèmes et de maintenir tout le monde informé.
- **Daily Meetings à 9h** : Des réunions quotidiennes à 9h permettent de synchroniser les efforts de l'équipe, de discuter des progrès, des obstacles et de planifier les activités de la journée.
- **Meeting client**: Points de communication réguliers avec le client pour garantir que le projet reste aligné sur ses besoins et attentes

CONCEPTION DU PROJET

I. Users Stories

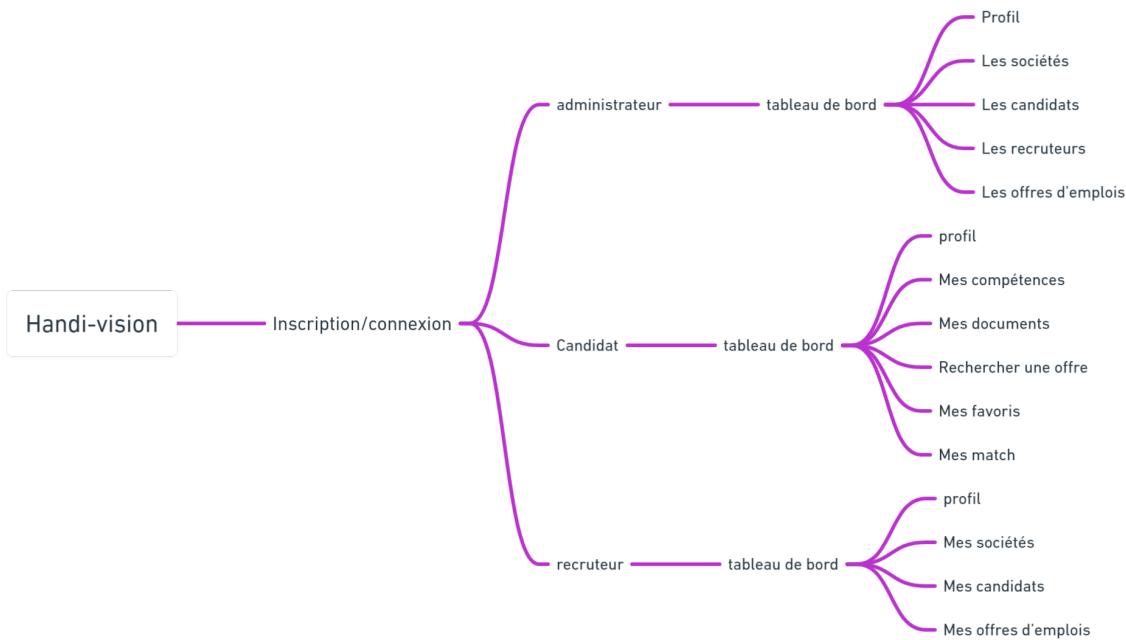
Vert : MVP

Jaune : versions futures

En Tant que	Je souhaite	Afin de
utilisateur	créer un compte	Me connecter
utilisateur	me connecter	accéder aux différents services
candidat	accéder à mon profil	ajouter/modifier mes informations
candidat	accéder à mon profil	supprimer mon profil
candidat	ajouter mes compétences	voir les emploi me correspondant
candidat	ajouter mon attestation rqth	mon compte soit valide
candidat	ajouter mon cv	pour qu'il soit visible par les recruteurs
candidat	accéder à la liste des offres d'emplois	pour y postuler

candidat	accéder à la liste des offres d'emplois qui match avec mon profil	pour y postuler
candidat	ajouter des offres d'emplois à mes favoris	les consulter plus tard
recruteur	accéder à mon profil	ajouter/modifier mes informations
recruteur	accéder à mon profil	supprimer mon profil
recruteur	enregistrer une société	pour que la société puisse être valide et que je puisse publier des offres d'emplois
recruteur	enregistrer une offre d'emploi	pour qu'elle puisse être valide et visible par les candidat
recruteur	rechercher un candidat	contacter
recruteur	Avoir accès à la liste des candidats qui match avec les critères que je recherche	contacter
administrateur	accéder à mon profil	ajouter/modifier mes informations
administrateur	valider/rejeter un candidat	que le candidat puisse avoir accès aux offre d'emplois
administrateur	valide/rejeter une entreprise	pour pouvoir publier des offres d'emplois
administrateur	valider/rejeter une offre d'emplois	pour qu'elle soit visible par les candidats

II. Arborescence



III. MVP (Produit Minimum Viable) de Handi-Vision

➤ Introduction

Notre MVP a été conçu pour permettre aux différents utilisateurs (candidats, recruteurs) d'utiliser les fonctionnalités de base d'un site de recherche d'emploi, adapté principalement aux personnes en situation de handicap. Ceci inclut l'enregistrement, la compléction de profils, et la soumission ou consultation d'offres d'emploi. L'administrateur pourra valider les profils et les offres d'emploi, avec une vérification spécifique de la RQTH pour les candidats.

➤ Fonctionnalités du MVP

- **Création de Compte:** Les utilisateurs (candidats, recruteurs) peuvent créer un compte selon leur rôle.
- **Authentification:** Les utilisateurs se connectent avec un email et un mot de passe.

➤ Candidat

- **Tableau de Bord:** Après connexion, accès aux onglets "Mon profil", "Mes compétences", "Mes documents", "Rechercher une offre", "Mes favoris", "Mes matchs". Initialement, seuls "Mon profil" et "Mes documents" sont accessibles, les autres étant en attente de validation.
- **Mon Profil:** Possibilité d'ajouter une photo, modifier les informations personnelles, ou supprimer le profil.
- **Mes Documents:** Téléchargement de l'attestation RQTH et du CV pour validation par l'administrateur.
- **Rechercher une offre:** Accès à toutes les offres d'emploi, recherche spécifique, détails des annonces, ajout aux favoris, et postulation.
- **Mes Favoris:** Gestion des annonces favorites.

➤ Recruteur

- **Tableau de Bord:** Accès initial à "Mon profil" et, après validation, aux onglets "Mes sociétés", "Mes candidats", "Mes offres d'emploi".
 - **Mes Sociétés:** Création et gestion des entreprises.
 - **Mes Offres d'Emploi:** Création et modification des offres d'emploi.
 - **Candidats:** Recherche et gestion des candidats.
-
- **Administrateur**
 - **Tableau de Bord:** Gestion des profils candidats, recruteurs, entreprises, et offres d'emploi.
 - **Sociétés:** Validation, rejet, contact, ou suppression des entreprises.
 - **Candidats:** Vérification de la RQTH, validation ou rejet des candidats.

- **Recruteurs et Offres d'Emploi:** Validation ou rejet des recruteurs et offres d'emploi.

IV. Fonctionnalités Détaillées des Pages

Les pages/routes sont sécurisées et donc accessibles seulement une fois l'utilisateur connecté. Seules les pages d'inscription et de connexion sont accessibles à tous. Le candidat ou le recruteur doit d'abord créer un compte, en accédant à l'onglet "créer un compte" et en renseignant les informations requises. Une fois le compte créé, ils peuvent se connecter via la page de connexion.

- **Page Crédation de Compte :** L'utilisateur, sans accès au menu, doit compléter les champs requis en fonction de son rôle (candidat ou recruteur). Il valide son inscription via le bouton "Créer un compte "et accède à la page de connexion.
- **Page Connexion :** Après la création du compte, l'utilisateur est redirigé vers la page de connexion où il peut accéder au menu en complétant les champs email et mot de passe. Une fois connecté, il est redirigé vers une page d'accueil spécifique selon son rôle.
- **Pour le Profil Candidat :** Une fois connecté, le candidat a accès aux fonctionnalités suivantes à partir du menu :
 - **Mon Profil :** Téléchargement de photo de profil, modification des informations personnelles (nom, prénom, email, numéro de téléphone).
 - **Mes Compétences :** [version future]
 - **Mes Documents :** Possibilité de télécharger l'attestation RQTH et le CV.
 - **Rechercher une Offre :** Recherche d'offres, affichage de la liste, détails des offres, ajout/retrait des favoris, postulation.
 - **Mes Favoris :** Affichage des offres favorites, détails des offres, retrait des favoris, postulation.
 - **Mes Matchs :** [version future]
- **Pour le Profil Recruteur :** Une fois connecté, le recruteur a accès aux fonctionnalités suivantes à partir du menu :

- **Mon Profil** : Téléchargement de photo de profil, modification des informations personnelles.
- **Mes Sociétés** : Création et gestion des entreprises avec un formulaire comprenant le nom de l'entreprise, secteur d'activité, raison sociale, adresse, liens vers les réseaux sociaux, téléphones, email, site web, code NAF, effectif, statut juridique, politique de télétravail. Après validation du formulaire, attente de validation par l'administrateur pour publier des offres d'emploi.
- **Mes Candidats** : Recherche de candidats, affichage de la liste correspondante.
- **Mes Offres d'Emploi** : Création d'offres avec un formulaire détaillé (poste, lieu, type et durée de contrat, description, etc.). Les offres doivent être validées par l'administrateur pour être publiées.

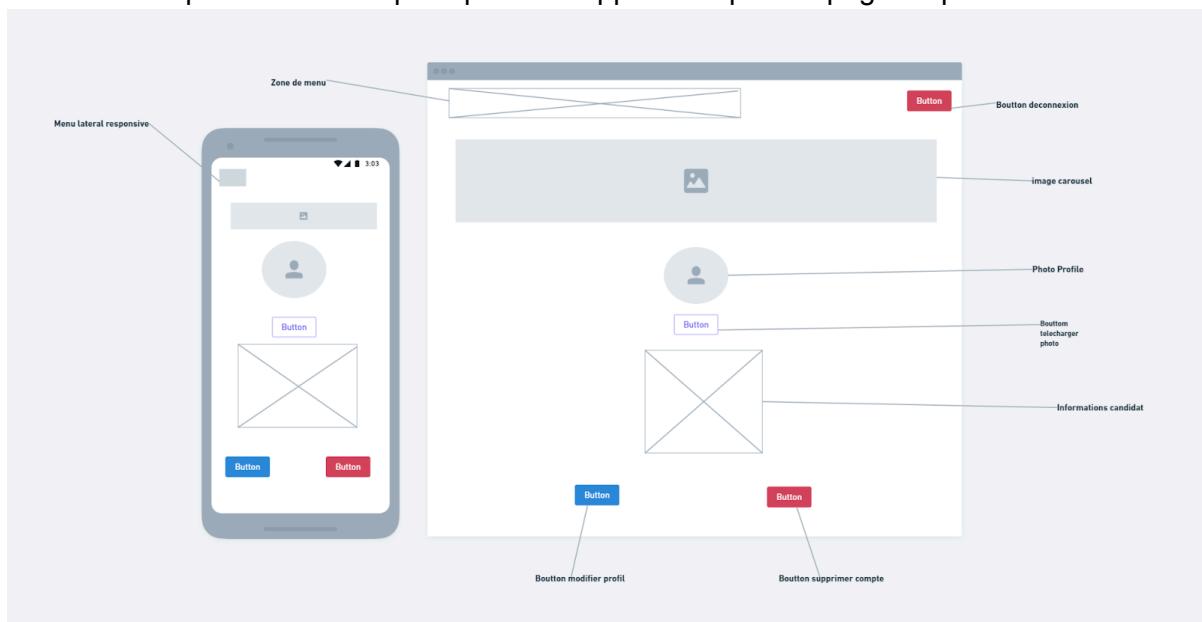
➤ **Pour le Profil Administrateur** : L'administrateur, en tant que modérateur de tous les utilisateurs du site, a accès aux fonctionnalités suivantes à partir du menu :

- **Profil** : Affichage et modification de ses informations personnelles.
- **Sociétés** : Gestion des entreprises (validation, rejet, contact, suppression).
- **Candidats** : Gestion des candidats (validation, rejet, contact, suppression).
- **Recruteurs** : Gestion des recruteurs (validation, rejet, contact, suppression).
- **Offres d'emploi** : Gestion des offres (validation, rejet, contact, suppression).

V. Wireframes

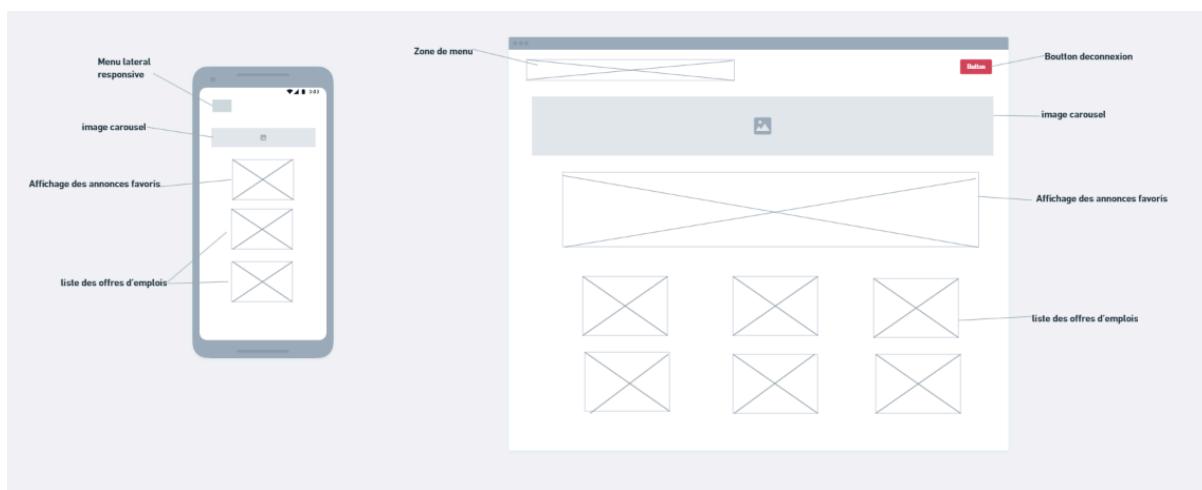
Accueil candidat

Ce wireframe présente la vue principale de l'application pour la page de profil du candidat.



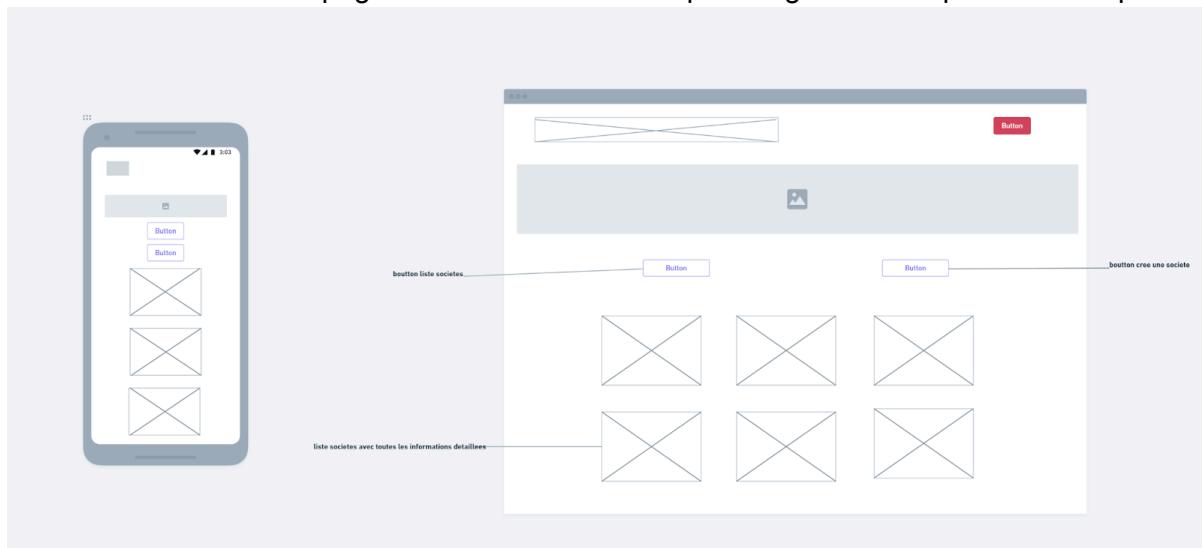
Mes offres (Candidat)

Cette vue illustre la page de gestion des offres d'emploi pour les candidats



Mes Sociétés (recruteur)

Ce wireframe décrit la page dédiée aux recruteurs pour la gestion des profils d'entreprise



SPÉCIFICATIONS TECHNIQUES

Lors de l'élaboration de notre plateforme Handi-Vision, un soin particulier a été apporté à la sélection des technologies à utiliser, ces technologies, imposées par le client, ont orienté notre approche de développement et se sont avérées être des choix judicieux. Elles nous ont permis de développer une plateforme qui non seulement répond aux attentes en termes de fonctionnalité et de performance mais qui est également en phase avec les meilleures pratiques actuelles en matière de développement web. En adhérant à ces choix techniques, nous avons pu créer une plateforme qui non seulement réalise la vision du client mais qui est également conçue pour l'avenir, en prenant en compte l'évolutivité, la sécurité, et surtout, l'accessibilité pour tous les utilisateurs.

I. Technologies

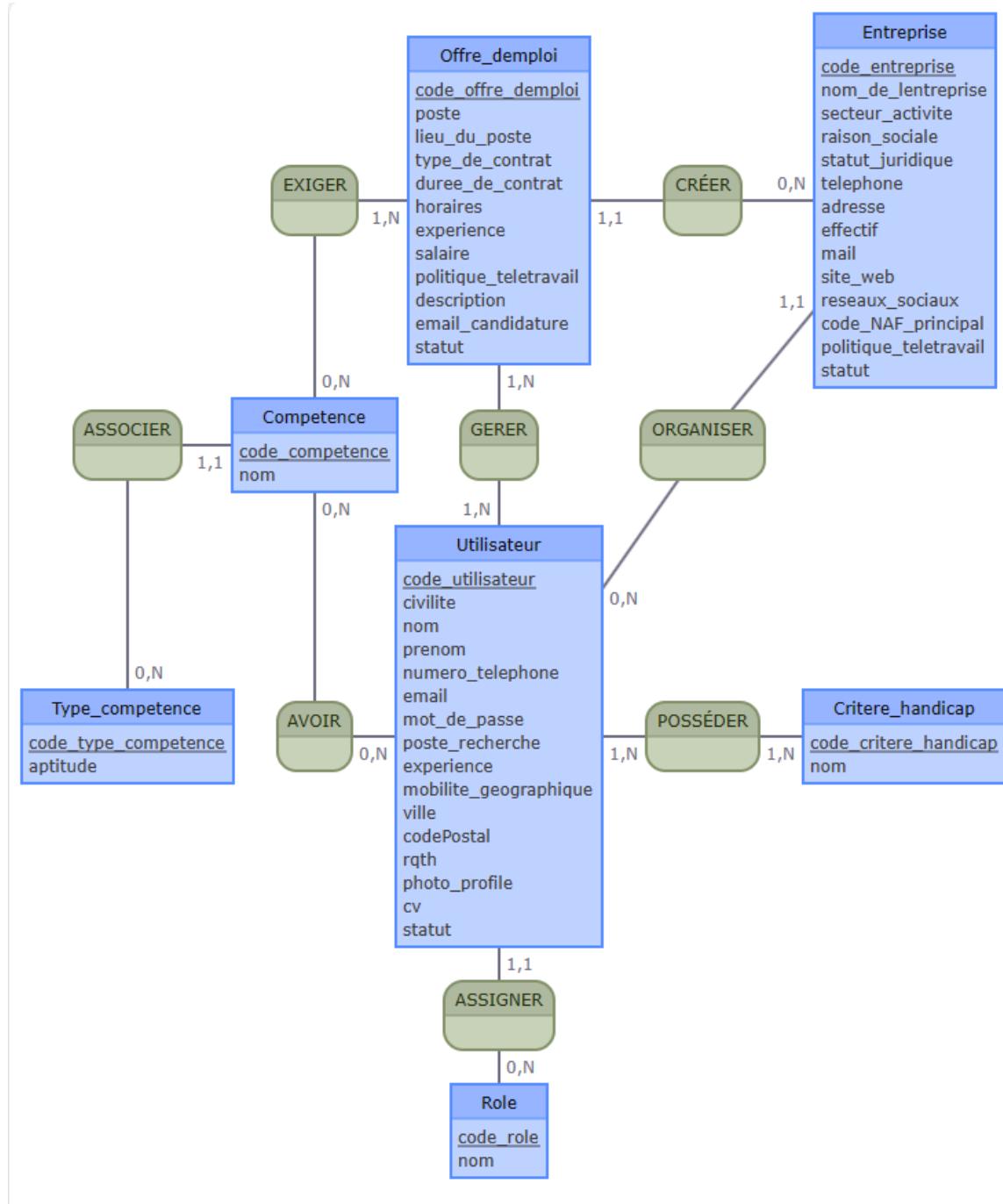
- **Front-End :**
 - React
 - Bootstrap (react-bootstrap)
 - React-router
- **Back-End :**
 - Nodejs
 - Express

- Sequelize
- **Base de Données :**
- MYSQL

II. Création de la base de données

L'élaboration de notre base de données pour Handi-Vision a été rigoureusement menée en commençant par définir les user stories qui ont déterminé les besoins et les fonctionnalités de notre système. Ces scénarios d'utilisation par les candidats et recruteurs ont directement influencé la conception du Modèle Conceptuel de Données (MCD), essentiel pour structurer notre base de données relationnelle. Le MCD, en cours de développement, schématise les entités principales, leurs attributs et les relations, formant ainsi le socle sur lequel repose la gestion des données de la plateforme.

A. MCD



Le MCD de la plateforme Handi-Vision est conçu pour soutenir efficacement la mise en relation entre les candidats en situation de handicap et les entreprises proposant des offres d'emploi. Les entités et leurs relations sont pensées pour refléter les besoins et fonctionnalités décrits dans les user stories, assurant ainsi une plateforme à la fois performante et adaptée aux utilisateurs.

- **Entités Principales :**

- **Utilisateur** : Représente à la fois les candidats et les recruteurs avec des informations personnelles et professionnelles détaillées, ainsi que des données spécifiques telles que le type de handicap via l'entité 'Critere_handicap'.
- **Offre_d'emploi** : Contient tous les détails relatifs aux postes à pourvoir, y compris le type de contrat, la durée, l'expérience requise et d'autres critères essentiels.
- **Entreprise** : Stocke les informations sur les entreprises, y compris leur secteur d'activité, leurs coordonnées et leur engagement envers le télétravail.
- **Competence et Type_competence** : Ces entités définissent les compétences des utilisateurs, avec une distinction entre les compétences générales et les aptitudes spécifiques.
- **Role** : Identifie les différents rôles assignés aux utilisateurs au sein de la plateforme, permettant de définir leurs permissions et interactions possibles avec le système.

B. MLD

Nous avons à partir de ce MCD, établi le MLD puis le MPD (Annexe page 65):

- Le MLD est une traduction technique du MCD qui prépare la structure pour l'implémentation dans un système de gestion de base de données. Il détaille les tables, les clés (primaires et étrangères) et les relations entre elles.

Les clés primaires sont déterminées pour chaque table, souvent basées sur les identifiants uniques de chaque entité dans le MCD. Ces clés primaires sont essentielles pour assurer l'unicité de chaque enregistrement dans une table.

Bien sûr, la conception d'un Modèle Logique de Données (MLD) est une étape cruciale dans la modélisation des bases de données. Elle implique la transformation des entités, des attributs et des relations du Modèle Conceptuel de Données (MCD) en un schéma qui peut être mis en œuvre dans un système de gestion de base de données (SGBD). Voici une description enrichie incluant les cardinalités et les tables de liaison :

Le MLD transforme les entités du MCD en tables de la base de données. Chaque entité devient une table et chaque attribut de l'entité devient une colonne de cette table. Les clés primaires sont déterminées pour chaque table, souvent basées sur les identifiants uniques de chaque entité dans le MCD. Ces clés primaires sont essentielles pour assurer l'unicité de chaque enregistrement dans une table.

Les relations entre les entités sont exprimées dans le MLD par des clés étrangères. Une clé étrangère est un attribut ou un ensemble d'attributs qui permet de référencer de manière unique une ligne d'une autre table. Les cardinalités des relations du MCD, qui spécifient les règles de correspondance entre les instances des entités (comme 1:N, N:1, ou N:N), déterminent comment ces clés étrangères sont utilisées.

- Pour les relations de cardinalité **1:N**, la clé primaire de l'entité du côté "1" est ajoutée comme clé étrangère à l'entité du côté "N".
- Pour les relations de cardinalité **1:1**, la clé primaire de l'une des tables est souvent utilisée comme clé étrangère dans l'autre table, ou vice versa, selon la direction de la dépendance.
- Les relations de cardinalité **N:N** sont un peu plus complexes car elles ne peuvent pas être représentées directement dans une base de données relationnelle. Elles nécessitent la création de tables de liaison (aussi appelées tables associatives ou tables de jointure) qui contiennent les clés primaires des deux entités en relation comme clés étrangères. Ces tables de liaison permettent de gérer les relations multiples entre les entités et de conserver la normalisation de la base de données.

C. MPD

Nous avons ensuite déterminé le mpd

À ce stade, nos tables sont quasiment prêtes pour la phase de création. C'est là qu'intervient le Modèle Physique des Données (MPD), qui va affiner notre conception pour la transformer en un script SQL spécifique. Ce modèle est ajusté pour correspondre aux exigences d'un Système de Gestion de Base de Données (DBMS) déterminé et est le produit dérivé direct du Modèle Logique de Données (MLD).

Dans le MPD, nous allons préciser(annexe) :

- **Le type de données** pour chaque colonne, sélectionné en fonction des types de données supportés par le DBMS cible et des besoins de stockage pour chaque information.
- **Les contraintes de table** qui vont au-delà des clés primaires et étrangères pour inclure, par exemple, des contraintes d'unicité, des

contraintes de vérification ou des valeurs par défaut, afin d'assurer l'intégrité et la validité des données.

Ce modèle représente le blueprint final qui sera utilisé pour générer le script de création de notre base de données. C'est une étape déterminante qui assure que la base de données, une fois créée, réponde précisément à nos attentes en termes de structure, de contraintes et de performances.

III. Back end

Le Backend, construit avec Express.js, c'est une API qui peut être utilisée par un client web. Nous avons utilisé Le pattern MVC, ou Modèle-Vue-Contrôleur qui vise à séparer les préoccupations au sein d'une application pour faciliter la maintenance, l'évolutivité et la réutilisabilité.

- Routes back-end

Le fichier ci-dessous est le fichier router.js, qui définit les endpoints de l'API et associe chaque route à une fonction de contrôleur spécifique chargée de gérer les requêtes entrantes.

```

● ● ●

import express from 'express';
import mainController from './controllers/mainController.js';
import authController from './controllers/authController.js';
import usersController from './controllers/usersController.js';
import updateController from './controllers/updateController.js';
import companiesController from './controllers/companiesController.js';
import jobOfferController from './controllers/jobOfferController.js';
import adminController from './controllers/adminController.js';
import { roleCheck } from './middleware/roleCheck.js';
import { jwtGuard } from './middleware/jwtGuard.js';
import { upload } from './middleware/storageFiles.js';
import { uploadProfilePicture } from './middleware/storageProfilePhotos.js';
import { uploadcv } from './middleware/storagecv.js';
import uploadController from './controllers/uploadController.js';
import competencesController from './controllers/competencesController.js';
import formJobOfferController from './controllers/formJobOfferController.js';
import matchController from './controllers/matchController.js';

const router = express.Router();

router.post('/api/user/register', usersController.setUser);
router.post('/api/users/login', authController.login);

router.get('/api/users', jwtGuard, roleCheck([3, 4]), usersController.getUsers);
router.get('/api/me', jwtGuard, roleCheck([2, 3, 4]), authController.me);
router.patch('/api/me/update', jwtGuard, roleCheck([2, 3]), updateController.updateUser);
router.delete('/api/me/deleteprofile', jwtGuard, roleCheck([2, 3]), usersController.deleteProfile);
router.post('/api/company/register', jwtGuard, roleCheck([3, 4]), companiesController.setCompany);
router.get('/api/companies', jwtGuard, roleCheck([4]), companiesController.getCompanies);
router.get('/api/companies/me', jwtGuard, roleCheck([3, 4]), companiesController.getUserCompanies);
router.post('/api/user/joboffer', jwtGuard, roleCheck([3]), jobOfferController.setJobOffer);
router.get('/api/joboffers', jwtGuard, roleCheck([2, 3, 4]), jobOfferController.getJobOffers);
router.get('/api/joboffers/me', jwtGuard, roleCheck([3, 4]), jobOfferController.getUserJobOffers);
router.get('/api/admin/getusers', jwtGuard, roleCheck([4]), adminController.getUsers);
router.patch('/api/admin/updateuser', jwtGuard, roleCheck([4]), adminController.updateUser);
router.delete('/api/admin/deleteuser', jwtGuard, roleCheck([4]), adminController.deleteUser);
router.get('/api/admin/getcompanies', jwtGuard, roleCheck([4]), adminController.getCompanies);
router.patch('/api/admin/updatecompany', jwtGuard, roleCheck([4]), adminController.updateCompany);
router.delete('/api/admin/deletecompany', jwtGuard, roleCheck([4]), adminController.deleteCompany);

router.get('/api/admin/getusers/:coderole', jwtGuard, roleCheck([4]), adminController.getUsersByRole);
router.get('/api/admin/getjoboffers', jwtGuard, roleCheck([4]), adminController.getJobOffers);

router.get('/api/admin/getusers/:coderole/:statut', jwtGuard, roleCheck([4]),
adminController.getUsersByRoleAndStatut);
router.get('/api/admin/joboffers/:statut', jwtGuard, roleCheck([4]), adminController.getJobOffersByStatut);
router.patch('/api/admin/updatejoboffer', jwtGuard, roleCheck([4]), adminController.updateJobOffer);

router.patch('/api/candidat/updatejoboffer', jwtGuard, roleCheck([2]), jobOfferController.updateFavJobOffer);
router.post('/api/candidat/favjoboffer', jwtGuard, roleCheck([2]), jobOfferController.addFavJobOffer);
router.get('/api/candidat/getfavjoboffer', jwtGuard, roleCheck([2]), jobOfferController.getFavJobOffer);
router.delete('/api/candidat/removefavjoboffer', jwtGuard, roleCheck([2]), jobOfferController.removeFavJobOffer);
router.get('/api/detailsjoboffer/:offerId', jwtGuard, roleCheck([2]), jobOfferController.getDetailsJobOffer);

router.get('/api/admin/getcompanies/:statut', jwtGuard, roleCheck([4]), adminController.getCompaniesByStatut);
router.get('/api/admin/getcompanybyjoboffer/:codecompany', jwtGuard, roleCheck([4]),
adminController.getCompanyByJobOffer);

//// upload et recuperation de fichier
router.post('/api/uploadFile', jwtGuard, roleCheck([2]), upload.single('rqth'), uploadController.uploadFile);
router.get('/downloadFile/:candidatId', jwtGuard, roleCheck([2, 3, 4]), uploadController.getFile);

router.post('/api/upload/profile/photo', jwtGuard, roleCheck([2, 3]), uploadProfilePicture.single('photo'),
uploadController.uploadProfilePicture);
router.get('/download/profile/photo/:userId', jwtGuard, roleCheck([2, 4]), uploadController.getprofilePhoto);

router.post('/api/upload/cv', jwtGuard, roleCheck([2]), uploadcv.single('cv'), uploadController.uploadcv);
router.get('/download/cv/:userId', jwtGuard, roleCheck([2, 4]), uploadController.getUserCv);

router.get('/api/getJobbCompetences', jwtGuard, roleCheck([2]), competencesController.getCompetences);
router.get('/api/getCompetences/me', jwtGuard, roleCheck([2]), competencesController.getUserCompetence);

///////////
router.post('/api/recruteur/formjoboffer', jwtGuard, roleCheck([3]), formJobOfferController.formJobOffer);
router.get('/api/recruteur/getjoboffer', jwtGuard, roleCheck([3]), matchController.getJobOffer)
router.get('/api/recruteur/getcandidate', jwtGuard, roleCheck([3]), matchController.getCandidate)

export default router;

```

➤ **Importations :**

- Les contrôleurs sont importés en haut du fichier. Chaque contrôleur est un module qui contient des fonctions pour gérer des actions spécifiques, comme l'authentification des utilisateurs, la gestion des entreprises, ou des offres d'emploi.
- Des middlewares sont également importés pour sécuriser les routes, vérifier les rôles des utilisateurs, gérer les sessions, et traiter les fichiers uploadés.

➤ **Définition des Routes :**

- Chaque route est définie par une méthode HTTP (**get**, **post**, **patch**, **delete**) qui correspond à l'action CRUD (Create, Read, Update, Delete) qu'elle représente.
- Les routes prennent un chemin (par exemple, `/api/user/register`) et une ou plusieurs fonctions de rappel qui déterminent comment la requête doit être traitée.

➤ **Middlewares :**

- **jwtGuard** est un middleware d'authentification qui vérifie si l'utilisateur est connecté et a un token JWT valide.
- **roleCheck** est un middleware de contrôle d'accès qui vérifie si l'utilisateur a le rôle approprié pour accéder à la route.
- **upload** et ses variantes (**uploadProfilePicture**, **uploadcv**) sont des middlewares pour la gestion des téléchargements de fichiers, de photos de profil et de CVs.

➤ **Contrôleurs et Actions :**

- Chaque route est associée à une action de contrôleur spécifique. Par exemple, la route `router.post('/api/user/register',`

`usersController.setUser)` indique que lorsque la route `/api/user/register` reçoit une requête POST, la fonction `setUser` du `usersController` est appelée pour traiter cette requête.

- Les contrôleurs contiennent la logique pour répondre aux requêtes, comme la création d'un nouvel utilisateur, la connexion, la mise à jour de profils, etc.

➤ **Routes Protégées :**

- Les routes utilisant `jwtGuard` et `roleCheck` sont protégées, ce qui signifie que seuls les utilisateurs authentifiés avec le bon rôle peuvent y accéder. Par exemple, `router.get('/api/users', jwtGuard, roleCheck([3, 4]), usersController.getUsers)` est une route protégée qui nécessite que l'utilisateur soit authentifié en tant que recruteur (3) ou administrateur (4) pour obtenir la liste des candidats.

➤ **Gestion des Fichiers :**

- Les routes pour l'upload et le téléchargement de fichiers permettent aux utilisateurs de soumettre et de récupérer des documents comme des justificatifs de qualification (rqth), des photos de profil et des CVs.

IV. Connexion base de données

Dans le cadre du projet Handi-Vision, nous avons décidé d'adopter Sequelize comme interface de programmation pour notre base de données. Sequelize est un ORM reconnu qui offre une abstraction efficace des bases de données relationnelles, en nous permettant de manipuler les données sous forme d'objets et de classes au lieu d'instructions SQL brutes. Cela rend le développement plus rapide, plus sûr et plus maintenable.

La configuration de la connexion est définie à travers Sequelize, tirant parti des variables d'environnement pour sécuriser nos paramètres de connexion. Ce choix architectural favorise un code plus propre et plus sécurisé, puisque les détails de connexion sensibles, tels que le nom de la base de données, l'utilisateur et le mot de

pas, ne sont pas codés en dur dans notre base de code mais stockés séparément dans un environnement contrôlé.

Le fichier ci-dessous sert de point d'entrée pour la connexion à la base de données MySQL, préparant le terrain pour une interaction robuste et sécurisée à travers Sequelize, et garantissant que notre application Handi-Vision communique efficacement avec la base de données dès son initialisation.



```
import { Sequelize } from 'sequelize';
import dotenv from 'dotenv';

dotenv.config();

const sequelize = new Sequelize(
  process.env.DB_DATABASE,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    host: process.env.DB_HOST,
    dialect: 'mysql',
  }
);

const isDbConnected = async () => {
  try {
    await sequelize.authenticate();
    console.log('Connection has been established successfully.');
  } catch (error) {
    console.error('Unable to connect to the database:', error);
  }
}

isDbConnected();

export default sequelize;
```

Le modèle ci-dessous fournit une structure de données pour les offres d'emploi dans une base de données MySQL, en utilisant Sequelize comme ORM pour la modélisation.

```

● ● ●

import { DataTypes } from 'sequelize';
import sequelize from '../database/database.js';
import Company from './company.js';

const JobOffer = sequelize.define('Offer', {
    poste: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    lieu_du_poste: {
        type: DataTypes.STRING,
    },
    type_de_contrat: {
        type: DataTypes.STRING,
    },
    duree_de_contrat: {
        type: DataTypes.STRING,
    },
    horaires: {
        type: DataTypes.STRING,
    },
    experience: {
        type: DataTypes.INTEGER,
    },
    salaire: {
        type: DataTypes.DECIMAL(10, 2),
    },
    politique_teletravail: {
        type: DataTypes.TEXT,
    },
    code_entreprise: {
        type: DataTypes.INTEGER,
        allowNull: false,
    },
    statut: {
        type: DataTypes.INTEGER,
        allowNull: false,
        defaultValue: 1 // Définissez la valeur par défaut à 1
    },
    description: {
        type: DataTypes.STRING,
    },
    email_candidature: {
        type: DataTypes.STRING,
    },
},
{
    tableName: 'Offre_demploi',
    timestamps: false,
    createdAt: false,
    updatedAt: false,
});
// Définir la relation entre OffreEmploi et Entreprise
JobOffer.belongsTo(Company, { foreignKey: 'code_entreprise' });
CompanyhasMany(JobOffer, { foreignKey: 'code_entreprise' });

export default JobOffer;

```

Ce modèle Sequelize sert de représentation abstraite des offres d'emploi, permettant de manipuler les données liées aux offres d'emploi en utilisant des objets JavaScript,

ce qui simplifie l'interaction avec la base de données et améliore la qualité du code en séparant la logique de gestion des données de la logique métier de l'application.

V. Front End

Le développement du front-end a été réalisé en utilisant le framework React, connu pour sa capacité à créer des interfaces utilisateur dynamiques et réactives. React facilite la définition de vues spécifiques pour chaque état de l'application. Lorsque les données changent, React s'occupe de mettre à jour de manière efficiente seulement les composants nécessaires, assurant ainsi une gestion optimale des ressources et une expérience utilisateur fluide.

Le code ci-dessous est le point d'entrée principal de notre application front-end développée avec React. Ce fichier joue un rôle crucial dans le routage et la structure générale de votre application.

```
const container = document.getElementById('app');
const root = createRoot(container);
root.render(  
  
    <BrowserRouter>
        <Routes>
            <Route path="/" element={<Connexion />} />
            <Route path="/DashboardAdmin" element={<PrivateRouteAdmin> <HeadAdminCaroussel /> </PrivateRouteAdmin>} />
            <Route path="/DashboardRecruteur" element={<PrivateRouteRecruteur> <HeadCaroussel />} />
            <Route path="/DashboardCandidat" element={<PrivateRouteCandidat> <CarousselCandidat />} />
            <Route path="/details-offre/:offreId" element={<DetailsOffre />} />
            <Route path="/Inscription" element={<Inscription />} />
            <Route path="/not" element={<NotFound />} />
            <Route path="/forbidden" element={<Forbidden />} />
            <Route path="/delete" element={<SupprimerElementAdmin />} />
            <Route path="/deleteRecruteur" element={<SupprimerElementRecruteur />} />
        </Routes>
        <ScrollButon />
        <PiedDePage />
    </BrowserRouter>
);
```

ce code permet :

- **Initialisation de l'application React** : Le code commence par obtenir une référence à un élément DOM (**container**) dans votre fichier HTML. La `createRoot(container)` crée une racine React pour cette application.
- **Rendu de l'application** : La méthode `root.render()` est utilisée pour rendre l'application React. Elle contient votre routeur et les composants associés.
- **Utilisation de BrowserRouter** : Ce composant est utilisé pour activer la navigation dans votre application. Il écoute les changements d'URL dans le navigateur et synchronise l'interface utilisateur avec ces URL.
- **Configuration des routes avec Routes et Route** :
 - Chaque **Route** définit un chemin (**path**) et le composant React (**element**) à afficher quand l'URL correspond à ce chemin.
 - Par exemple, `<Route path="/" element={<Connexion />} />` affichera le composant **Connexion** lorsque l'utilisateur navigue vers la racine de l'application (/).
 - Des routes telles que `/DashboardAdmin`, `/DashboardRecruteur`, etc., affichent différents composants en fonction de l'URL.
- **Routes Protégées** : Les composants `PrivateRouteAdmin`, `PrivateRouteRecruteur`, `PrivateRouteCandidat` sont des wrappers qui protègent l'accès à certaines routes. Ils s'assurent que l'utilisateur a les permissions nécessaires pour accéder à ces routes.
- **Autres Routes** : Il y a des routes pour des fonctionnalités telles que l'inscription, l'affichage de détails d'offre, la gestion des erreurs (`NotFound`, `Forbidden`), et la suppression d'éléments.

VI. Accessibilité

L'accessibilité est une priorité fondamentale dans la conception de handi-vision.fr, une plateforme dédiée principalement aux personnes en situation de handicap. Nous avons travaillé pour rendre le site facilement navigable pour tous. Cette démarche implique une conception intuitive, une compatibilité totale avec les lecteurs d'écran, et une navigation aisée via le clavier et les commandes vocales. Toutefois, nous sommes conscients de la nécessité d'améliorer continuellement l'accessibilité de handi-vision.

```

● ● ●
◇
<div>
  <Navbar
    role="navigation"
    aria-label="Menu de navigation principale"
    bg="dark"
    variant="dark"
    expand="lg"
    className="border-bottom col-12">
    <Navbar.Toggle
      aria-controls="basic-navbar-nav"
      onClick={handleNavToggle}
      aria-expanded={expanded}
      aria-label="Ouvrir le menu pour naviguer" />
    <Navbar.Collapse
      id="basic-navbar-nav"
      className={expanded ? "show" : ""}>
      <Nav
        className="d-flex justify-content-between col-12">
        <div>
          /* Les boutons ont maintenant des étiquettes plus descriptives pour les lecteurs
          d'écran */
        <Button
          variant="primary"
          className={`m-1 ${selectedButton === "Profil" ? "active" : ""}`}
          block="true" aria-label="Consulter mon Profil"
          onClick={() => handleButtonClick("Profil")}>
          Mon Profil
        </Button>

        <Button
          variant="primary"
          className={`m-1 ${selectedButton === "FormCompetenceCandidat" ? "active" : ""}`}
          block="true" aria-label="Mettre à jour mes compétences professionnelles"
          onClick={() => handleButtonClick("FormCompetenceCandidat")}>
          Mes Compétences
        </Button>
      </div>
    </Nav>
  </Navbar.Collapse>
</div>

```

voici la partie du code de la barre de navigation du tableau de bord du profil candidats qui est rendu accessible grâce à:

- **Rôles ARIA et Labels:** utilisation des attributs ARIA (Accessible Rich Internet Applications) tels que `role=navigation` et `aria-label` pour aider les lecteurs d'écran à mieux interpréter le contenu.
- **Boutons avec étiquettes descriptives:** Chaque bouton du menu a une étiquette `aria-label` descriptive. Cela aide les utilisateurs de lecteurs d'écran à comprendre la fonction de chaque bouton, rendant la navigation plus intuitive.
- **Navigation au clavier :** le code tient compte de la navigation au clavier, comme le montre l'utilisation de `onClick` pour gérer les interactions. Cela est essentiel pour les utilisateurs qui ne peuvent pas utiliser une souris.

- **Conception Responsive et Accessible** : L'utilisation de Navbar.Toggle et Navbar.Collapse de Bootstrap indique une attention portée à la conception responsive. Une navigation accessible doit être aussi efficace sur les appareils mobiles que sur les ordinateurs de bureau.
- **Contraste des Couleurs** : thème sombre (bg="dark") avec un contraste élevé, ce qui est bénéfique pour les utilisateurs ayant des problèmes de vision.
- Feedback Visuel pour la Sélection : L'utilisation de la classe active pour indiquer le bouton sélectionné offre un retour visuel immédiat, aidant tous les utilisateurs à comprendre quelle section du menu est active.

VII. Harmonisation des Processus de Travail

Afin de nous assurer d'un processus de développement harmonisé, efficace et conforme aux standards de l'industrie. Nous avons mis en place des règles d'Harmonisation des Processus de Travail afin de maintenir la qualité et la cohérence du code tout en facilitant la collaboration au sein de notre équipe.

➤ Organisation des Fichiers et Arborescence :

- Structure de dossiers adaptée en front et en back
- models/views/controllers suivant le pattern MVC pour le serveur express
- Organisation des fichiers selon le modèle components/views pour le front en React.

➤ Nommage des Fichiers :

- Être attentif au nommage des fichiers. Utilisation du CamelCase avec une majuscule initiale pour les classes (ex : User), et avec une minuscule initiale pour les fonctions (ex : connectUser) et les variables (ex : connectedUser).

➤ Indentation du Code :

- Bonne indentation du code pour faciliter sa lecture et sa maintenance.

➤ Commentaires :

- Commenter chaque classe, fonction et bloc complexe en français pour une meilleure compréhension et maintenabilité.

➤ **Bonnes Pratiques de GIT :**

- Aucun push directement sur la branche Main.
- Faire des commits propres et fréquents, avec des descriptions concises
- Pratiquer les Pull Requests et les Code Reviews pour assurer la qualité du code.

➤ **Utilisation des Branches :**

- Création de nouvelles branches pour les corrections majeures, le développement de nouvelles fonctionnalités, ou le code expérimental.
- S'assurer que la branche master contient uniquement des commits fusionnés depuis des branches secondaires.
- Faire fréquemment des rebases avec la branche Dev pour rester synchronisé avec l'équipe.

➤ **Autres Bonnes Pratiques :**

- Avant de modifier un fichier que j'ai déplacé ou renommé, je fais un commit.
- S'abstenir de commiter des fichiers de configuration.
- Utiliser la commande reset avec prudence pour éviter la perte de données.

RÉALISATIONS PERSONNELLES

Dans le cadre de notre projet, notre équipe était divisée en deux groupes distincts : l'équipe front-end et l'équipe back-end. Ma responsabilité unique était de servir de liaison entre ces deux équipes. Ce rôle m'a offert l'opportunité unique de travailler à la fois sur le front-end et le back-end du projet.

J'ai joué un rôle clé dans la coordination des tâches et dans l'alignement des objectifs des deux équipes, assurant ainsi que le projet avance de manière cohérente et harmonieuse.

Ce rôle m'a permis de développer une compréhension approfondie des divers aspects du projet et d'améliorer mes compétences en tant que développeur full-stack, soulignant l'importance de la collaboration et de la communication dans le développement web.

I. Upload de fichiers :

A. Back

Dans le cadre de mon projet HandiVision, j'ai développé une fonctionnalité clé pour améliorer l'expérience utilisateur : le téléchargement de fichiers. Cette fonctionnalité permet aux utilisateurs de télécharger des documents importants tels que leur CV ou une photo de profil ou l'attestation rqth obligatoire pour que le profil candidat soit valide. Voici comment j'ai procédé :

```
import multer from 'multer'

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'C:/Users/home/Desktop/handi-vision/handi-vision/back/app/uploads/rqth'); // Le dossier où les
fichiers seront stockés
  },
  filename: function (req, file, cb) {
    // Génère un nom de fichier unique, par exemple, en ajoutant une horodatage
    cb(null, Date.now() + '-' + file.originalname);
  },
});

export const upload = multer({ storage: storage });
```

➤ Configuration de Multer :

- J'ai commencé par importer le paquet Multer, un middleware Node.js pour gérer le téléchargement de fichiers.
- Ensuite, j'ai configuré le stockage Multer pour définir le dossier de destination des fichiers téléchargés et pour générer des noms de fichiers uniques. Ceci assure que les fichiers ne sont pas écrasés et facilement identifiables.

```
router.post('/api/uploadFile', jwtGuard, roleCheck([2]), upload.single('rqth'), uploadController.uploadFile);
```

➤ Route API pour le Téléchargement :

- J'ai créé une route API '`/api/uploadFile`' qui utilise plusieurs middlewares pour la sécurité et la fonctionnalité :
- **jwtGuard** pour l'authentification.
- **roleCheck([2])** pour s'assurer que seuls les utilisateurs autorisés peuvent télécharger l'attestation rqth.
- **upload.single('rqth')** est une instruction pour Multer de traiter un seul fichier nommé 'rqth' dans la requête POST vers `/api/uploadFile`. C'est une partie cruciale de votre gestion des téléchargements de fichiers, assurant que le fichier est correctement reçu, traité et stocké sur le serveur.

```

● ● ●

uploadFile: async function (req, res) {
    if (req.file) {
        const fileName = req.file.filename;

        const userId = req.user.id;

        try {
            // Mettez à jour le champ rqth pour l'utilisateur spécifié
            const updatedUser = await User.update(
                { rqth: fileName },
                {
                    where: { id: userId },
                }
            );

            console.log('Champ "rqth" mis à jour pour l\'utilisateur.');
            res.status(200).send('Fichier téléchargé et utilisateur mis à jour');

        } catch (error) {
            console.error('Erreur lors de la mise à jour du champ "rqth" :', error);
            res.status(500).send('Erreur lors de la mise à jour de l\'utilisateur');
        }
    } else {
        console.log('Aucun fichier téléchargé.');
        res.status(400).send('Aucun fichier téléchargé');
    }
},

```

➤ **Méthode uploadFile :**

- Cette méthode gère le téléchargement du fichier.
- Elle commence par vérifier si un fichier a été téléchargé et, si c'est le cas, extrait son nom du fichier téléchargé (`req.file.filename`) ainsi que l'ID de l'utilisateur (`req.user.id`) de la requête. L'ID est typiquement défini par un middleware d'authentification qui authentifie l'utilisateur avant d'exécuter cette méthode.
- Ensuite, elle utilise une méthode `update` sur le modèle `User` qui interagit avec la base de données pour mettre à jour l'enregistrement de l'utilisateur correspondant. Elle définit le champ `rqth` avec le nouveau nom de fichier, qui est le nom du fichier sur le disque après le traitement par Multer.
- Si la mise à jour réussit, elle envoie une réponse de statut 200 avec un message indiquant que le fichier a été téléchargé et que l'utilisateur a été mis à jour.
- Si une erreur se produit pendant la mise à jour, elle envoie une réponse de statut 500 avec un message indiquant qu'une erreur s'est produite pendant la mise à jour.
- Si aucun fichier n'est présent dans la requête, elle loggue un message indiquant qu'aucun fichier n'a été téléchargé et envoie une réponse de statut 400, ce qui est une réponse standard pour une requête mal formée.
- L'image est stockée sur le disque du serveur dans le chemin spécifié par la configuration de stockage de Multer. ici: "C:/Users/home/Desktop/handi-vision/handi-vision/back/app/uploads/" selon le dossier configuré dans `multer.diskStorage`.

```
router.get('/downloadFile/:candidatId', jwtGuard, roleCheck([2, 4]), uploadController.getFile);
```

➤ **Route GET pour le Téléchargement de Fichier :**

- '/downloadFile/:candidatId' est la route qui attend une requête GET pour télécharger un fichier. Le :candidatId dans l'URL est un paramètre dynamique qui sera remplacé par l'identifiant du candidat lors d'une requête réelle.
- jwtGuard est un middleware qui s'assure que la requête est authentifiée.
- roleCheck([2, 4]) est un middleware qui vérifie si l'utilisateur authentifié à l'un des rôles autorisés (2, ou 4) pour accéder à cette route.
- uploadController.getFile est le contrôleur qui sera appelé pour traiter la requête.

```

● ● ●

getFile: async function (req, res) {
    const candidatId = req.params.candidatId;

    const user = await User.findByPk(candidatId);

    const completpath = "C:/Users/home/Desktop/handi-vision/handi-vision/back/app/uploads/rqth"
    const filePath = path.join(completpath, user.rqth); // Chemin du fichier sur le serveur
    console.log("Voici le fichier: ")
    console.log(filePath)

    // Vérifiez si le fichier existe
    if (!fs.existsSync(filePath)) {
        return res.status(404).send('Fichier non trouvé');
    }

    // Renvoyez le fichier en réponse
    res.download(filePath, user.rqth, (err) => {
        if (err) {
            console.error("Erreur lors de l'envoi du fichier :", err);
            return res.status(500).send('Erreur lors de l\'envoi du fichier');
        }
    });
},
}

```

➤ Méthode getFile :

- Cette méthode est définie de manière asynchrone, indiquant qu'elle effectuera des opérations asynchrones (comme interroger la base de données).

- Elle commence par extraire **candidatId** des paramètres de la requête. Cet ID est utilisé pour identifier le candidat concerné.
- Elle utilise ensuite l'ID pour récupérer les informations de l'utilisateur de la base de données (**User.findByPk(candidatId)**).
- La méthode construit le chemin complet (**filePath**) où le fichier est supposé être stocké sur le serveur, en combinant le chemin de base (**completPath**) avec le nom de fichier stocké dans la base de données (**user.rqth**).

➤ **Vérification de l'Existence du Fichier :**

- Avant d'essayer de télécharger le fichier, elle vérifie si le fichier existe réellement sur le serveur (**fs.existsSync(filePath)**).

➤ **Envoi du Fichier :**

- Si le fichier existe, elle utilise **res.download(filePath, user.rqth, (err) => {...})** pour envoyer le fichier au client. **res.download** déclenche le téléchargement du fichier par le navigateur du client.
- En cas d'erreur pendant l'envoi (comme une interruption de la connexion), elle logue l'erreur et envoie une réponse d'erreur 500.

➤ **Gestion de l'Erreur de Fichier Non Trouvé :**

- Si le fichier n'existe pas, la méthode envoie une réponse 404 indiquant que le fichier n'a pas été trouvé.

B. Front

Le composant React **DocumentsCandidat** gère le téléchargement de documents par les candidats, notamment le fichier RQTH. Voici quelques parties du code



```

const [selectedFile, setSelectedFile] = useState(null);
const [selectedCV, setSelectedCV] = useState(null);
const [fileName, setFileName] = useState("");
const [fileNameCV, setFileNameCV] = useState("");

```

Ces lignes de code utilisent le Hook `useState` de React pour créer une variable d'état

➤ `> const [selectedFile, setSelectedFile] = useState(null);`

- `selectedFile` est la variable qui va stocker l'état actuel. Au départ, cette variable est initialisée à `null`, ce qui signifie qu'aucun fichier n'est sélectionné au moment de l'initialisation du composant.
- `setSelectedFile` est une fonction qui permet de modifier l'état de `selectedFile`. Par exemple, lorsque l'utilisateur sélectionne un fichier, vous pouvez appeler `setSelectedFile(fichier)` pour mettre à jour la variable `selectedFile` avec le fichier sélectionné.

➤ `> const [fileName, setFileName] = useState("");`

Cette ligne suit le même principe que la première, mais elle est utilisée pour gérer le nom du fichier sélectionné.

- `fileName` est la variable d'état qui stocke le nom du fichier actuellement sélectionné. Elle est initialisée à une chaîne de caractères vide "", indiquant qu'aucun nom de fichier n'est sélectionné ou connu au départ.
- `setFileName` est la fonction qui nous permet de mettre à jour le nom du fichier stocké dans `fileName`.



```
const handleFileChange = (e) => {
  // Récupère le fichier à partir de l'événement
  const file = e.target.files[0];

  if (file) {
    setSelectedFile(file);
    setFileName(file.name);
  }
};
```

La fonction `handleFileChange` est conçue et appelée lorsque l'utilisateur sélectionne le fichier dans l'input de type file. Elle met à jour les états `selectedFile` et `fileName`

avec les informations du fichier sélectionné, en utilisant les fonctions fournies par les Hooks **useState**.



```
const handleFileSubmit = async (e) => {
  e.preventDefault();
  const formData = new FormData();
  formData.append('rqth', selectedFile);

  if (selectedFile) {

    for (let i = 0; i < 100; i += 10)
      setTimeout(() => setUploadProgress(i), 1000);

    try {

      const token = localStorage.getItem('token');
      let fetchOptions = {
        // Méthode de la requête
        method: 'POST',
        headers: {
          'Authorization': `Bearer ${token}`,
        },
        body: formData,
      };
      const response = await fetch(`${apiUrl}/api/uploadFile`, fetchOptions);
      if (!response.ok) {

        console.error(`Erreur lors de la requête : ${response.status} ${response.statusText}`);
      } else {
        setIsFileSubmitted(true);
      }

    } catch (error) {
      console.log(`Erreur Fetch: ${error}`);
    }
  } else {
    console.log('Aucun fichier sélectionné.');
  }
};
```

La fonction **handleFileSubmit** est une fonction asynchrone conçue pour gérer la soumission du fichier téléchargé, en envoyant les données via une requête HTTP.

➤ Empêcher le comportement par défaut du formulaire :

- **e.preventDefault();** Cette ligne empêche le comportement par défaut du navigateur lors de la soumission d'un formulaire,

➤ Préparation des données à envoyer :

- **const formData = new FormData();** Crée un nouvel objet **FormData**, qui est utilisé pour construire un ensemble de paires clé-valeur représentant les champs du formulaire et leurs valeurs, prêt à être envoyé via une requête HTTP.
- **formData.append('rqth', selectedFile);** Ajoute le fichier sélectionné par l'utilisateur à l'objet **FormData** sous le nom 'rqth'. **selectedFile** est la variable d'état qui contient le fichier sélectionné.

➤ Vérification de la sélection d'un fichier :

- La condition `if (selectedFile) { ... }` vérifie si un fichier a été sélectionné. Si c'est le cas, le bloc de code à l'intérieur de cette condition est exécuté.

➤ **Envoi du fichier au serveur :**

- La requête HTTP est préparée avec un objet `fetchOptions` contenant la méthode **POST**, les en-têtes y compris un token d'authentification récupéré du localStorage, et le corps de la requête `formData`.
- La requête est envoyée au serveur en utilisant `fetch`. L'URL du serveur (``${ apiUrl }/api/uploadFile``) est utilisée pour déterminer où envoyer la requête.
- La réponse du serveur est traitée. Si la requête est réussie, le statut de soumission du fichier (`setIsFileSubmitted`) est mis à jour. Si la requête échoue, une erreur est enregistrée dans la console.

➤ **Gestion des erreurs :**

- Les erreurs éventuelles survenant lors de l'envoi de la requête sont capturées et enregistrées dans la console.

➤ **Gestion de l'absence de fichier sélectionné :**

- Si aucun fichier n'a été sélectionné (`else`), un message est enregistré dans la console pour informer qu'aucun fichier n'a été sélectionné.



```

<div className="m-3">
  <div className="row justify-content-center">
    <div className="col-md-6">
      <div className="card shadow">
        <h3 className="text-dark mb-3 text-center fw-bold">Attestation RQTH</h3>
        <div className="card-body text-center">
          <form onSubmit={handleFileSubmit} encType="multipart/form-data">
            <div className="mb-3">
              <input className="form-control" type="file" accept=".pdf" name="rqth" onChange={handleFileChange} />
              <p className="mt-4">{fileName}</p>
            </div>
            <button className="btn btn-primary" type="submit">Envoyer</button>
          </form>
          {isFileSubmitted && <div className="alert alert-success mt-3" role="alert">Fichier envoyé avec succès!</div>}
        </div>
      </div>
    </div>
  </div>
</div>

```

Ce segment de code représente une section du **return** dans un composant React. Cette partie du code définit ce que le composant va afficher ou "rendre" sur l'interface utilisateur. Des éléments JSX sont combinés avec syntaxe du HTML avec du JavaScript pour créer une expérience utilisateur interactive et dynamique.

Le code illustre un composant React conçu pour gérer l'upload et l'envoi de documents, en particulier pour une attestation RQTH. Il utilise les éléments de la bibliothèque Bootstrap pour la mise en forme et la présentation visuelle, offrant ainsi une interface esthétique et réactive adaptée à différents types d'écrans et de dispositifs.

➤ **Formulaire de Téléchargement (Attestation RQTH) :**

- Le formulaire est défini avec un événement **onSubmit** qui déclenche **handleFileSubmit** lors de la soumission.
- Un champ de sélection de fichier (**<input type="file">**) permet à l'utilisateur de choisir un fichier PDF à télécharger. Cet élément déclenche **handleFileChange** lorsqu'un fichier est sélectionné.
- Le nom du fichier sélectionné est affiché sous le champ de sélection.
- Un bouton permet de soumettre le formulaire.

➤ **Message de Succès (Attestation RQTH) :**

- Après la soumission réussie du fichier, un message de succès est affiché si **isFileSubmitted** est **true**.

➤ **Gestion des Fichiers :**

- **handleFileChange** gère le changement de l'état lors de la sélection d'un fichier, en mettant à jour les états **selectedFile** et **fileName**.
- **handleFileSubmit** gère la soumission du fichier. Elle crée un objet **FormData**, y ajoute le fichier, puis envoie une requête HTTP POST avec le fichier au serveur. Elle gère également la progression du téléchargement et les réponses du serveur.

VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

I. Sequelize

Lors de la sélection des technologies pour notre projet, nous avons initialement considéré l'utilisation de requêtes SQL brutes. Cependant, une évaluation approfondie et des recherches ultérieures nous ont orientés vers l'adoption d'un ORM, spécifiquement Sequelize, pour plusieurs raisons importantes liées principalement à la sécurité et à la maintenance.

- **Prévention des Injections SQL** : Sequelize s'est avéré être un choix avantageux principalement en raison de sa capacité à renforcer la sécurité des applications web. Un atout majeur de Sequelize est sa contribution à la prévention des injections SQL, une vulnérabilité courante dans le développement web. Grâce à l'utilisation de méthodes et de modèles prédéfinis pour les requêtes, Sequelize garantit que les entrées sont correctement échappées et formatées. Cela minimise le risque d'exposition à des injections SQL malveillantes, un facteur crucial dans la protection des données et la sécurisation des applications.
- **Facilité de Maintenance et Mises à Jour des Schémas de Base de Données** : En plus de la sécurité, la décision d'utiliser Sequelize a été fortement influencée par les avantages en termes de maintenance et de gestion des schémas de base de données. Les migrations, qui constituent une fonctionnalité clé de Sequelize, facilitent la réalisation de modifications structurées et contrôlées du schéma de la base de données. Cette approche structure les changements de manière organisée, réduisant considérablement les risques d'erreurs manuelles et améliorant ainsi la gestion globale et la sécurité de la base de données.

II. JSON Web Token (JWT)

Dans le cadre de notre projet, nous avons accordé une attention particulière à la sécurité, notamment en ce qui concerne l'authentification des utilisateurs. Pour cela, nous avons implémenté un système d'authentification basé sur JSON Web Token (JWT), une méthode standard et sécurisée pour transmettre des informations entre le client et le serveur de manière sécurisée.



```

import jwt from 'jsonwebtoken';
import User from '../models/user.js';

export const jwtGuard = async (req, res, next) => {

  const bearer = req.headers.authorization;

  if (bearer === undefined) {
    res.status(401).send("token is missing");
    return;
  }

  if (!bearer.includes('Bearer')) {
    res.status(401).send("bad token");
    return;
  }

  const token = bearer.split(' ');
  try {
    const payload = jwt.verify(token[1], process.env.SECRET_KEY);
    const user = await User.findById(payload.userId);
    req.user = user;
    console.log(user)
    next();
  } catch (e) {
    res.status(401).send(e.message);
  }

}

```

Ce morceau de code est un middleware en Node.js, utilisant Express et la bibliothèque **jsonwebtoken** pour l'authentification JWT (JSON Web Token). Ce middleware, nommé **jwtGuard**, a pour objectif de vérifier l'authenticité et la validité du token JWT fourni dans les requêtes HTTP. Voici une explication détaillée de son fonctionnement :

➤ Importation des Dépendances :

- **jwt** est importé de **jsonwebtoken**, une bibliothèque utilisée pour la vérification des JWT.
- **User** est importé de **../models/user.js**, pour interagir avec la table des utilisateurs dans la base de données.

➤ Définition du Middleware **jwtGuard** :

- **export const jwtGuard = async (req, res, next) => { ... }** : **jwtGuard** est un middleware asynchrone prenant trois arguments - **req** (l'objet de la requête), **res** (l'objet de la réponse) et **next** (une fonction callback pour passer au middleware suivant).

➤ Extraction et Vérification du Token :

- **const bearer = req.headers.authorization;** : Le middleware extrait la valeur de l'en-tête **Authorization** de la requête.
- Il vérifie d'abord si l'en-tête **Authorization** est présent. S'il est absent, le serveur renvoie une réponse 401 (non autorisé) avec le message "token is missing".
- Ensuite, il vérifie si l'en-tête contient le mot-clé **Bearer**. S'il ne le contient pas, le serveur renvoie une réponse 401 avec le message "bad token".

➤ **Décodage et Validation du Token :**

- **const token = bearer.split(' ');** : Le token est extrait en scindant la chaîne **bearer** et en récupérant la deuxième partie (le token JWT lui-même).
- **const payload = jwt.verify(token[1], process.env.SECRET_KEY);** : Le token JWT est vérifié en utilisant la clé secrète stockée dans **process.env.SECRET_KEY**. Si la vérification échoue (par exemple, si le token est expiré ou invalide), une exception est levée.

➤ **Récupération de l'Utilisateur et Passage au Prochain Middleware :**

- Si le token est valide, le middleware extrait l'identifiant de l'utilisateur (**userId**) du payload du token.
- **const user = await User.findById(payload.userId);** : Il recherche ensuite l'utilisateur correspondant dans la base de données.

```

● ● ●

login: async function (req, res) {
  const { email, mot_de_passe } = req.body;
  const secretKey = process.env.SECRET_KEY

  try {
    const user = await User.authenticate(email, mot_de_passe);

    if (user) {
      console.log('Connexion réussie');

      const token = jwt.sign(
        {
          userId: user.id, // Utilisez un identifiant unique de l'utilisateur ici
          /* role: user.code_role, // Inclue le rôle de l'utilisateur */

          // Autres données pertinentes à inclure
        },
        secretKey, // Remplacez par clé secrète
        {
          expiresIn: '10h', // Durée de validité du token (10 heure dans cet exemple)
        }
      );
    }

    const { id, code_role, statut } = user

    // avant de transmettre le user au front lui retire des informations
    res.status(200).json({ message: 'Connexion réussie', token, id, code_role, statut });
  } else {
    console.log('Identifiants incorrects');
    res.status(401).json({ message: 'Identifiants incorrects' }); // Réponse JSON en cas d'échec
  }
} catch (error) {
  console.error(`Erreur lors de l'authentification : ${error}`);
  res.status(500).json({ message: 'Erreur du serveur' }); // Réponse JSON en cas d'erreur serveur
},
}

```

Ce code implémente une méthode de connexion sécurisée, utilisant les tokens JWT pour authentifier les utilisateurs et gérer l'accès aux ressources de l'application.

- Si l'utilisateur est authentifié avec succès, un token JWT est créé en utilisant **jwt.sign()**.
- Ce token contient un **payload** avec des informations comme **userId** et peut inclure d'autres données comme le rôle de l'utilisateur.
- Le token est signé avec la **secretKey** et a une durée de validité (**expiresIn**), ici fixée à 10 heures.
- Le serveur renvoie une réponse au client avec le token JWT et d'autres informations de l'utilisateur (**id**, **code_role**, **statut**).
- Le token est utilisé pour authentifier les requêtes suivantes de l'utilisateur. En effet, pour chaque requête, le client devra fournir ce token, souvent dans l'en-tête de la requête, pour que le serveur puisse vérifier et autoriser l'accès aux ressources protégées.

III. Cryptage du mot de passe utilisateur

Quand un utilisateur s'inscrit et choisit un mot de passe, au lieu de stocker ce mot de passe en clair dans la base de données, l'application l'envoie à travers Bcrypt.

Bcrypt convertit ce mot de passe en une chaîne de caractères hachés. Quand un utilisateur essaie de se connecter, l'application récupère le mot de passe qu'il fournit, le passe à travers le même processus de hachage avec Bcrypt, puis je compare le résultat avec le hachage stocké dans la base de données. Si les deux correspondent, cela signifie que le mot de passe est correct.

Avec le hachage de mot de passe avec Bcrypt, nous nous assurons que même en cas de fuite de données, les mots de passe des utilisateurs restent sécurisés et indéchiffrables. C'est une pratique standard dans le développement d'applications sécurisées, et je m'assure de l'appliquer dans tous mes projets pour protéger les informations sensibles des utilisateurs.

IV. Protection route back avec roleCheck

Pour garantir que les routes de l'API soient accessibles uniquement aux utilisateurs disposant des rôles appropriés, nous avons implémenté un middleware qui contrôle le rôle de chaque utilisateur. Cette vérification de rôle est essentielle pour décider si un utilisateur a le droit d'accéder à une route spécifique ou non. Grâce à ce middleware, l'accès aux différentes routes de l'API est rigoureusement régulé, assurant ainsi que seules les personnes autorisées puissent y accéder.

V. Protection route front avec L'élément privateRoute

Les routes côté front sont également protégées pour n'être accessibles qu'aux utilisateurs en fonction de leur rôle, ce qui empêche ceux qui n'ont pas le bon rôle d'accéder aux autres routes depuis l'URL. Ce composant récupère le token et le rôle de l'utilisateur depuis le local storage. Si l'utilisateur est authentifié et qu'il a le bon rôle, le composant autorise l'accès à la route en rendant le contenu enfant (children). Sinon, l'utilisateur est redirigé vers une page indiquant un accès interdit (/forbidden).

PRÉSENTATION DU JEU D'ESSAI

Dans ce jeu d'essai je vais vous présenter l'expérience utilisateur du candidat de la connexion jusqu'à la consultation du détail des offres d'emploi en passant par l'ajout aux favoris.

Ce jeu d'essai va nous permettre de vérifier le bon fonctionnement des requêtes. Pour réaliser ce jeu d'essai, je vais me servir du logiciel Postman qui permet de tester, et de documents les requêtes vers une API.

Le candidat commence par saisir son adresse email et son mot de passe.

Lorsque ces informations d'identification sont validées par l'API, s'en suit la génération d'un token JWT (JSON Web Token). Ce token est essentiel car il agit comme un passeport numérique qui authentifie l'utilisateur pour les requêtes ultérieures.

The screenshot shows two requests made using the Postman application.

Request 1: Login (POST http://localhost:3000/api/users/login)

- Body:** JSON (Pretty)


```

1 {
2   ...
3   "email": "ab@gmail.com",
4   ...
5   "mot_de_passe": "*****"
6 }
```
- Response:** Status: 200 OK | Time: 78 ms | Size: 616 B


```

1 {
2   "message": "Connexion réussie",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjE1LCJpYXQiOjE3MDYyOTE3NTAsImV4cCI6MTcwNjMyNzc1MH0.jq9agyUS5LNCq7J2BkMRR4bQhw5PJP0kDCJgKD_E7tI",
4   "id": 15,
5   "code_role": 2,
6   "statut": 3
7 }
```

Request 2: Job Offers (GET http://localhost:3000/api/joboffers)

- Authorization:** Bearer Token (Type: Bearer Token, Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOjE1LCJpYXQiOjE3MDYyOTE3NTAsImV4cCI6MTcwNjMyNzc1MH0.jq9agyUS5LNCq7J2BkMRR4bQhw5PJP0kDCJgKD_E7tI)
- Response:** Status: 200 OK | Time: 23 ms | Size: 25.85 KB


```

1 [
2   {
3     "id": 3,
4     "poste": "Développeur fullstack",
5     "lieu_du_poste": "Paris",
6     "type_de_contrat": "CDI",
7     "duzee_de_contrat": "",
8     "horaires": "9h-17h",
9     "experience": 3,
10    "salaire": "35000.00",
11    "politique_télétravail": "2j/semaine",
12    "code_entreprise": 4,
13    "statut": 3,
14    "description": "Description du poste\nAlphalives, basée dans le 13ème arrondissement de Paris est une agence Web de conseil et de développement Internet spécialisée depuis 22 ans dans la création de projets Web et applications mobiles.\n\nNotre agence de développement (8 personnes) recherche dans le cadre d'une augmentation d'activité, des personnes ayant de bonnes compétences et expériences dans le développement web PHP.\n\nVous aurez pour"
15  }
```

Après une authentification réussie et la récupération du token JWT, l'étape suivante dans notre jeu d'essais consiste à tester l'accès aux données protégées, en l'occurrence la liste des offres d'emploi. Pour ce faire, j'ai configuré une requête GET dans Postman, ciblant l'endpoint **/api/joboffers**.

Dans la section 'Authorization' de Postman, j'ai sélectionné le type 'Bearer Token' et inclus le token JWT récupéré précédemment. Ceci est essentiel car cela permet d'authentifier la requête en tant qu'utilisateur connecté. L'inclusion du token garantit que l'accès à l'API est sécurisé et que les données ne sont disponibles qu'aux utilisateurs disposant des droits nécessaires.

La requête a abouti avec succès, comme en témoigne le code de réponse HTTP **200 OK**, indiquant que la communication avec l'API a été établie et que les données ont été reçues. La taille de la réponse et le temps de réponse rapide indiquent également une performance adéquate de l'API.

La réponse JSON affichée dans le corps de Postman montre un extrait de la liste des offres d'emploi, y compris les détails tels que l'identifiant de l'offre, le poste, le lieu de travail, le type de contrat, et autres informations pertinentes. Cela démontre que l'API transmet les données de manière structurée et cohérente, prêtes à être présentées dans l'interface utilisateur de l'application.

The screenshot shows a user interface for job listings. At the top, there are three dropdown menus: 'Tout les Postes', 'Tout les Contrats', and 'Toutes les Villes'. Below these are six job card components arranged in two rows of three:

- Developpeur fullstack**
Type de contrat : CDI
Ville : Paris
Télétravail : 2j/semaine
[Plus d'infos](#)
[Postuler](#)
- Developpeur backend**
Type de contrat : CDI
Ville : Paris
Télétravail : 4j/semaine
[Plus d'infos](#)
[Postuler](#)
- Developpeur frontend**
Type de contrat : CDI
Ville : Paris
Télétravail : 2j/semaine
[Plus d'infos](#)
[Postuler](#)

- Developpeur javascript**
Type de contrat : CDI
Ville : Paris
Télétravail : 2j/semaine
[Plus d'infos](#)
[Postuler](#)
- Boucher**
Type de contrat : cdi
Ville : paris
Télétravail : non
[Plus d'infos](#)
[Postuler](#)
- Developpeur**
Type de contrat : cdi
Ville : paris
Télétravail : 3j/semaine
[Plus d'infos](#)
[Postuler](#)

L'étape suivante de notre jeu d'essais consiste à s'assurer que le candidat peut retrouver ses offres d'emploi favorites. Pour cela, une requête GET a été envoyée à l'endpoint **/api/candidat/getfavjoboffer** en utilisant l'outil Postman. Cette requête est authentifiée par le même token JWT obtenu lors de la connexion, ajouté dans la section 'Authorization' de la requête sous la forme d'un 'Bearer Token'.

Le succès de la requête est confirmé par la réponse HTTP **200 OK**, qui indique que la liste des offres favorites a été récupérée sans erreur.

La réponse en format JSON montre clairement l'ensemble des données relatives aux offres favorites, y compris des détails tels que l'ID de l'offre, le poste, le lieu du poste, et d'autres attributs pertinents. Ce format structuré est prêt à être utilisé dans l'application pour afficher les offres aux candidats de manière intuitive et accessible.

The screenshot shows a Postman request configuration for a GET method at `http://localhost:3000/api/candidat/getfavjoboffer`. The 'Authorization' tab is selected, showing a dropdown set to 'Bearer Token' with a note: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)'. Below it, a text input field contains a JWT token: `eyJhbGciOiJIUzI1NltsInR5cCI6IkpxVCJ9.eyJ...`. The 'Body' tab is selected, displaying a JSON response with several fields like 'experience', 'salaire', and 'description'. The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 54 ms, Size: 10.72 KB, with a 'Save as example' button.

```
    "experience": 3,
    "salaire": "35000.00",
    "politique_teletravail": "2j/semaine",
    "code_entreprise": 4,
    "statut": 3,
    "description": "Description du poste\nAlphalives, bas\u00e9e dans le 13\u00e8me arrondissement de Paris est une agence Web de conseil et de d\u00e9veloppement Internet sp\u00e9cialis\u00e9e depuis 22 ans dans la cr\u00e9ation de projets Web et applications mobiles.\n\nNotre agence de d\u00e9veloppement (8 personnes) recherche dans le cadre d'une augmentation d'activit\u00e9, des personnes ayant de bonnes comp\u00e9tences et exp\u00e9riences dans le d\u00e9veloppement web PHP.\n\nVous aurez pour missions de :\n\nd\u00e9velopper des sites web sur mesure en PHP/MySQL avec le framework Laravel, ainsi que jQuery et Bootstrap.\n\nd\u00e9velopper des applications mobiles hybrides (front-end et/ou back-end) sur les diff\u00e9rentes plateformes (capacitorjs / ionic / nextjs / angular).\n\r\u00e9aliser des sp\u00e9cifications techniques\n\ntester les d\u00e9veloppements\n\nd\u00e9ployer les sites et applications\n\nProfil recherch\u00e9 :\n\nDipl\u00f4m\u00e9(e) d'un Bac + 2 ou Bac + 5 en informatique, avoir fait de l'alternance est un plus.\nExp\u00e9rience sur PHP/MySQL, Laravel\nComp\u00e9tences sur jQuery, HTML, CSS, Bootstrap\nPro-actif (ve), rigoureux(se) et passionn\u00e9(e) pour les nouvelles technologies et les m\u00e9tiers du d\u00e9veloppement logiciel et SaaS.",
    "email_candidature": "recrutement@anof.fr"
}, {
    "id": 1234567890
}
```

Ci-dessous le visuel de la liste des offres d'emploi favorite du candidat



Mes favoris

Developpeur fullstack Type de contrat : CDI Ville : Télétravail : 2j/semaine Plus d'infos Retirer des favoris	Developpeur backend Type de contrat : CDI Ville : Télétravail : 4j/semaine Plus d'infos Retirer des favoris	Developpeur frontend Type de contrat : CDI Ville : Télétravail : 2j/semaine Plus d'infos Retirer des favoris
Boucher Type de contrat : cdi Ville : Télétravail : non Plus d'infos Retirer des favoris	Developpeur Type de contrat : cdi Ville : Télétravail : 3j/semaine Plus d'infos Retirer des favoris	Developpeur Type de contrat : cdi Ville : Télétravail : 100% teletravail Plus d'infos Retirer des favoris

Le candidat a la possibilité de cliquer sur le bouton plus d'info pour avoir les détails de l'offre d'emploi.

Détails de l'offre

Developpeur backend

Type de contrat : CDI
Ville : Paris
Salaire : 35000.00
Télétravail : 4j/semaine

Description : Rattaché-e au Responsable développement du groupe, vous intégrerez une équipe de 5 personnes et travaillerez sur un tout nouveau pan du système d'information visant à fluidifier les échanges de données entre les différentes usines et le groupe. Cette partie applicative reposera sur une architecture back-end orientées services s'appuyant soit sur Node.js soit sur du Python avec FastApi. Mais le plus gros du travail sera centré sur le requêtage des données au travers de multiples bases de données relationnelles, le SQL sera donc votre compagnon le plus fidèle dans votre quotidien. Diplômé-e en informatique, vous justifiez d'une expérience post diplôme de 2 ans grand minimum au cours de laquelle vous avez acquis de solides compétences en requêtage de bases de données SQL et si possible un peu de pratique en développement back-end web quel que soit le langage utilisé du moment que vous maîtrisez bien les concepts de la programmation orienté objet et le fonctionnement d'une API REST. Les responsabilités Conception technique Développement de fonctionnalités Mise en place de tests

[Retour en arrière](#) [Postuler ↗](#)

Processus de recherche et traduction

➤ Processus

Pour effectuer mes recherches, par exemple pour sequelize j'ai adopté le processus de recherche suivant :

- Requête en anglais : Pour commencer, j'effectue une requête en anglais sur Google avec des termes simples liés à Sequelize par ex: "Sequelize tutorial"
- Sélection des sources :
 - La documentation officielle de Sequelize
 - acteurs reconnus dans le domaine de Sequelize
 - Les sites web ou blogs axés sur le développement et la programmation
 - Les questions pertinentes sur Stack Overflow

➤ Ressource en anglais :

L'article suivant, tiré de FreeCodeCamp, fournit un guide détaillé sur la configuration de Multer, étape par étape, afin de rendre ce processus aussi fluide que possible. Voici une traduction d'une partie de ce guide pratique pour intégrer le middleware Multer dans une application Express.js.

In this step, you'll create a separate file named `upload.js` to set up the Multer middleware. Multer is a middleware for handling multipart/form-data requests, specifically designed for file uploads in Node.js.

First, import the `multer` module using `require('multer')`. This ensures that you have access to the Multer functionality.

Next, define the storage configuration for uploaded files using `multer.diskStorage()`. This configuration determines where the uploaded files will be stored on the server. It takes an object with two functions: `destination` and `filename`.

The `destination` function specifies the directory where the uploaded files will be saved. In this example, we set it to '`uploads`', which means the files will be stored in a folder named "uploads" in the root directory of your project. You can customize the destination path based on your requirements.

The `filename` function determines the name of the uploaded file. In this example, we use `Date.now()` to generate a unique timestamp for each uploaded file, which helps prevent filename clashes.

We append the original name of the file using `file.originalname` to maintain some context about the uploaded file. You can modify this function to generate filenames based on your specific needs.

After setting up the storage configuration, you create an instance of Multer by calling `multer({ storage })`, passing in the `storage` configuration object. This creates the Multer middleware that you can use in your Express application to handle file uploads.

Finally, you export the Multer instance using `module.exports` so that it can be imported and used in other parts of your application, such as in the Express route for handling file uploads.

By setting up Multer middleware in this way, you have configured the storage destination and filename for uploaded files, allowing Multer to handle file uploads seamlessly in your application.

Traduction:

Dans cette étape, vous allez créer un fichier séparé nommé `upload.js` pour configurer le middleware Multer. Multer est un middleware conçu pour gérer les requêtes multipart/form-data, spécifiquement pour l'upload de fichiers en Node.js.

Tout d'abord, importez le module multer en utilisant `require('multer')`. Cela garantit que vous avez accès aux fonctionnalités de Multer.

Ensuite, définissez la configuration de stockage pour les fichiers téléchargés en utilisant **multer.diskStorage()**. Cette configuration détermine où les fichiers téléchargés seront stockés sur le serveur. Elle prend un objet avec deux fonctions : **destination** et **filename**.

La fonction **destination** spécifie le répertoire où les fichiers téléchargés seront enregistrés. Dans cet exemple, nous la définissons sur '**uploads/**', ce qui signifie que les fichiers seront stockés dans un dossier nommé "uploads" dans le répertoire racine de votre projet. Vous pouvez personnaliser le chemin de destination en fonction de vos besoins.

La fonction **filename** détermine le nom du fichier téléchargé. Dans cet exemple, nous utilisons **Date.now()** pour générer un horodatage unique pour chaque fichier téléchargé, ce qui aide à prévenir les conflits de noms de fichiers.

Nous ajoutons le nom original du fichier en utilisant **file.originalname** pour conserver un contexte concernant le fichier téléchargé. Vous pouvez modifier cette fonction pour générer des noms de fichiers basés sur vos besoins spécifiques.

Après avoir configuré le stockage, vous créez une instance de Multer en appelant **multer({ storage })**, en passant l'objet de configuration de stockage. Cela crée le middleware Multer que vous pouvez utiliser dans votre application Express pour gérer les téléchargements de fichiers.

Enfin, vous exportez l'instance Multer en utilisant **module.exports** afin qu'elle puisse être importée et utilisée dans d'autres parties de votre application, comme dans la route Express pour gérer les téléchargements de fichiers.

En configurant le middleware Multer de cette façon, vous avez configuré la destination de stockage et le nom des fichiers téléchargés, permettant à Multer de gérer les téléchargements de fichiers de manière transparente dans votre application.

CONCLUSION

En guise de conclusion à ce dossier de projet, je tiens tout d'abord à exprimer ma profonde gratitude envers l'entreprise Sigma Vision qui nous a accordé l'opportunité inestimable de travailler sur ce projet ambitieux. Cette collaboration a été un vecteur d'apprentissage et d'évolution professionnelle considérable pour moi.

Je souhaite également adresser mes sincères remerciements à chaque membre de l'équipe projet. Ensemble, nous avons fait preuve de synergie et de complémentarité, ce qui a grandement contribué à la réussite de notre mission. L'esprit d'équipe et le partage de connaissances ont été des facteurs clés qui ont enrichi cette expérience.

Ce projet a été d'une valeur incommensurable pour l'application pratique des notions apprises en cours. Il m'a permis de mettre en œuvre les compétences théoriques dans un contexte réel, me confrontant à des problématiques concrètes et diversifiées. J'ai pu ainsi consolider mes acquis et développer une compréhension plus profonde des différentes facettes du développement web.

Les défis rencontrés et les solutions apportées m'ont permis de réaliser des progrès significatifs. J'ai acquis une meilleure maîtrise des outils et des techniques modernes de développement, et j'ai pu affiner mon approche de résolution de problèmes. Ce projet a été, sans aucun doute, un catalyseur de ma croissance personnelle et professionnelle.

En résumé, cette expérience a été enrichissante à tous égards. Elle a consolidé ma passion pour le développement web et a renforcé ma détermination à poursuivre dans cette voie, armé d'un savoir-faire plus aiguisé et d'une motivation renouvelée.

ANNEXES

Dictionnaire de données

Table Role

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique pour chaque rôle.
nom	VARCHAR(255)	NOT NULL	Nom du rôle.

Table Utilisateur

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique pour chaque utilisateur.
civilite	VARCHAR(20)	NOT NULL	Titre de civilité de l'utilisateur.
nom	VARCHAR(60)	NOT NULL	Nom de famille de l'utilisateur.
prenom	VARCHAR(110)	NOT NULL	Prénom de l'utilisateur.
numero_telephone	VARCHAR(25)	NOT NULL	Numéro de téléphone de l'utilisateur.
email	VARCHAR(255)	UNIQUE, NOT NULL	Adresse e-mail de l'utilisateur.
mot_de_passe	VARCHAR(255)	NOT NULL	Mot de passe de l'utilisateur.
poste_recherche	VARCHAR(255)		Poste actuellement recherché par l'utilisateur.
experience	VARCHAR(50)		Expérience professionnelle de l'utilisateur.
mobilite_geographique	INT		Indicateur de la mobilité géographique.

rue	VARCHAR(255)		Rue de l'adresse de l'utilisateur.
ville	VARCHAR(255)		Ville de résidence de l'utilisateur.
codePostal	INT		Code postal de l'utilisateur.
rqth	VARCHAR(255)		Reconnaissance de la qualité de travailleur handicapé.
code_role	INT	FOREIGN KEY REFERENCES Role(id)	Clé étrangère vers la table Role .
statut	INT		Statut de l'utilisateur dans le système.
photo_profile	VARCHAR(255)		Chemin d'accès à la photo de profil.
cv	VARCHAR(255)		Chemin d'accès au CV de l'utilisateur.

Table Entreprise

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique de l'entreprise.
nom_de_lentreprise	VARCHAR(255)	NOT NULL	Nom de l'entreprise.
secteur_activite	VARCHAR(255)	NOT NULL	Secteur d'activité de l'entreprise.
raison_sociale	VARCHAR(255)	NOT NULL	Raison sociale de l'entreprise.
statut_juridique	VARCHAR(255)		Statut juridique de l'entreprise.
telephone	VARCHAR(25)	NOT NULL, UNIQUE	Numéro de téléphone de l'entreprise.
adresse	TEXT	NOT NULL	Adresse de l'entreprise.
effectif	INT		Nombre d'employés de l'entreprise.
mail	VARCHAR(255)	UNIQUE	Adresse email de l'entreprise.
site_web	TEXT		Site web de l'entreprise.
reseaux_sociaux	TEXT		Profils de réseaux sociaux de l'entreprise.

code_NAF_principal	VARCHAR(255)	NOT NULL	Code d'activité principale (NAF) de l'entreprise.
politique_teletravail	TEXT		Politique de télétravail de l'entreprise.
code_utilisateur	INT	FOREIGN KEY REFERENCES Utilisateur(id)	Clé étrangère vers la table Utilisateur.
statut	INT		Statut de l'entreprise dans le système.

Table Offre_demploi

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique de l'offre d'emploi.
poste	VARCHAR(255)	NOT NULL	Intitulé du poste offert.
lieu_du_poste	VARCHAR(255)		Lieu où le poste est basé.
type_de_contrat	VARCHAR(50)		Type de contrat proposé (CDI, CDD, etc.).
duree_de_contrat	VARCHAR(50)		Durée du contrat si applicable.
horaires	VARCHAR(50)		Horaires de travail pour le poste.
experience	INT		Expérience requise pour le poste en années.
salaire	DECIMAL(10, 2)		Salaire proposé pour le poste.
politique_teletravail	TEXT		Politique de télétravail applicable au poste.
code_utilisateur	INT	FOREIGN KEY REFERENCES Utilisateur(id)	Clé étrangère vers la table Utilisateur.
code_entreprise	INT	FOREIGN KEY REFERENCES Entreprise(id)	Clé étrangère vers la table Entreprise.
statut	INT		Statut de l'offre d'emploi dans le système.
descripition	VARCHAR(2000)		Description détaillée de l'offre d'emploi.

email_candidature	VARCHAR(255)	FOREIGN KEY REFERENCES Entreprise(id)	Email pour envoyer les candidatures.
-------------------	--------------	--	--------------------------------------

Table Type_Competence

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique du type de compétence.
aptitude	VARCHAR(255)	NOT NULL	Description de l'aptitude ou du type de compétence.

Table Critere_handicap

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique pour chaque critère handicap.
nom	VARCHAR(70)	NOT NULL	Nom du critère de handicap.

Table Competence

Nom du champ	Type de données	Contraintes	Description
id	INT	AUTO_INCREMENT, PRIMARY KEY	Identifiant unique de la compétence.
nom	VARCHAR(255)	NOT NULL	Nom de la compétence.
code_type_competence	INT	FOREIGN KEY REFERENCES Type_Competence(id)	Clé étrangère vers la table Type_Competence .

MLD :

Role (id, nom)

Utilisateur (id, civilite, nom, prenom, numero_telephone, email, mot_de_passe, poste_recherche, experience, mobilite_geographique, rue, ville, codePostal, rqth, code_role, statut, photo_profile, cv, #code_role)

Entreprise (id, nom_de_lentreprise, secteur_activite, raison_sociale, statut_juridique, telephone, adresse, effectif, mail, site_web, reseaux_sociaux, code_NAF_principal, politique_teletravail, code_utilisateur, statut, #code_utilisateur)

Offre_demploi (id, poste, lieu_du_poste, type_de_contrat, duree_de_contrat, horaires, experience, salaire, politique_teletravail, code_utilisateur, code_entreprise, statut, descrption, email_candidature, # code_utilisateur, #code_entreprise)

Type_Compentence (id, aptitude)

Competence (id, nom, code_type_competence, #code_type_competence)

Critere_handicap (id, nom)

Competence_Offre_demploi (id, code_competence, code_offre_demploi, #code_competence, #code_offre_demploi)

Utilisateur_Critere_handicap (id, code_utilisateur, code_Critere_handicap, #code_utilisateur, #code_Critere_handicap)

Utilisateur_Competence (id, code_utilisateur, code_competence, #code_utilisateur, #code_competence)

Utilisateur_Offre_demploi (id, code_utilisateur, code_offre_demploi, #code_utilisateur, #code_offre_demploi)

FICHIER MIGRATION

```
COMMIT;
START TRANSACTION;
SET foreign_key_checks = 0;
DROP TABLE IF EXISTS Role;
DROP TABLE IF EXISTS Utilisateur;
DROP TABLE IF EXISTS Competence;
DROP TABLE IF EXISTS Entreprise;
DROP TABLE IF EXISTS Utilisateur_Competence;
DROP TABLE IF EXISTS Utilisateur_Criterie_handicap;
DROP TABLE IF EXISTS Competence_Offre_demploi;
DROP TABLE IF EXISTS Type_Competence;
DROP TABLE IF EXISTS Criterie_handicap;
DROP TABLE IF EXISTS Offre_d_emploi;
DROP TABLE IF EXISTS Utilisateur_Offre_demploi;
SET foreign_key_checks = 1;

CREATE TABLE
IF NOT EXISTS Role (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL
);

CREATE TABLE
IF NOT EXISTS Utilisateur (
    id INT AUTO_INCREMENT PRIMARY KEY,
    civilite VARCHAR(20) NOT NULL,
    nom VARCHAR(60) NOT NULL,
    prenom VARCHAR(110) NOT NULL,
    numero_telephone VARCHAR(25) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    mot_de_passe VARCHAR(255) NOT NULL,
    poste_recherche VARCHAR(255),
    experience VARCHAR(50),
    mobilité_geographique INT,
    rue VARCHAR(255),
    ville VARCHAR(255),
    codePostal INT,
    rqtth VARCHAR(255),
    code_role INT,
    statut INT,
    photo_profile VARCHAR(255),
    cv VARCHAR(255),
    FOREIGN KEY (code_role) REFERENCES Role(id)
);

CREATE TABLE
IF NOT EXISTS Entreprise (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom_de_lentreprise VARCHAR(255) NOT NULL,
    secteur_activite VARCHAR(255) NOT NULL,
    raison_sociale VARCHAR(255) NOT NULL,
    statut_juridique VARCHAR(255),
    telephone VARCHAR(25) NOT NULL UNIQUE,
    adresse TEXT NOT NULL,
    effectif INT,
    mail VARCHAR(255) UNIQUE,
    site_web TEXT,
    reseaux_sociaux TEXT,
    code_NAF_principal VARCHAR(255) NOT NULL,
    politique_télétravail TEXT,
    code_utilisateur INT,
    statut INT,
    FOREIGN KEY (code_utilisateur) REFERENCES Utilisateur(id)
);

CREATE TABLE
IF NOT EXISTS Offre_demploi (
    id INT AUTO_INCREMENT PRIMARY KEY,
    poste VARCHAR(255) NOT NULL,
    lieu_de_poste VARCHAR(255),
    type_de_contrat VARCHAR(50),
    duree_de_contrat VARCHAR(50),
    horaires VARCHAR(50),
    experience INT,
    salaire DECIMAL(10, 2),
    politique_télétravail TEXT,
    code_utilisateur INT,
    code_entreprise INT,
    statut INT,
    description VARCHAR(2000),
    email_candidature varchar(255) FOREIGN KEY (code_entreprise) REFERENCES Entreprise(id)
);
```

```

CREATE TABLE
    IF NOT EXISTS Competence (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nom VARCHAR(255),
        code_type_competence INT,
        FOREIGN KEY (code_type_competence) REFERENCES Type_Competence(id)
    );

CREATE TABLE
    IF NOT EXISTS Critere_handicap (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nom VARCHAR(70) NOT NULL
    );

CREATE TABLE
    IF NOT EXISTS Competence_Offre_demploi (
        id INT AUTO_INCREMENT PRIMARY KEY,
        code_competence INT,
        code_offre_demploi INT,
        FOREIGN KEY (code_competence) REFERENCES Competence(id),
        FOREIGN KEY (code_offre_demploi) REFERENCES Offre_demploi(id)
    );

CREATE TABLE
    IF NOT EXISTS Utilisateur_Critere_handicap (
        id INT AUTO_INCREMENT PRIMARY KEY,
        code_utilisateur INT,
        code_critere_handicap INT,
        FOREIGN KEY (code_utilisateur) REFERENCES Utilisateur(id),
        FOREIGN KEY (code_critere_handicap) REFERENCES critere_handicap(id)
    );

CREATE TABLE
    IF NOT EXISTS Utilisateur_Competence (
        id INT AUTO_INCREMENT PRIMARY KEY,
        code_utilisateur INT,
        code_competence INT,
        FOREIGN KEY (code_utilisateur) REFERENCES Utilisateur(id),
        FOREIGN KEY (code_competence) REFERENCES Competence(id)
    );

CREATE TABLE
    IF NOT EXISTS Utilisateur_Offre_demploi (
        id INT AUTO_INCREMENT PRIMARY KEY,
        code_utilisateur INT,
        code_offre_demploi INT,
        FOREIGN KEY (code_utilisateur) REFERENCES Utilisateur(id),
        FOREIGN KEY (code_offre_demploi) REFERENCES Offre_demploi(id)
    );

COMMIT;

```