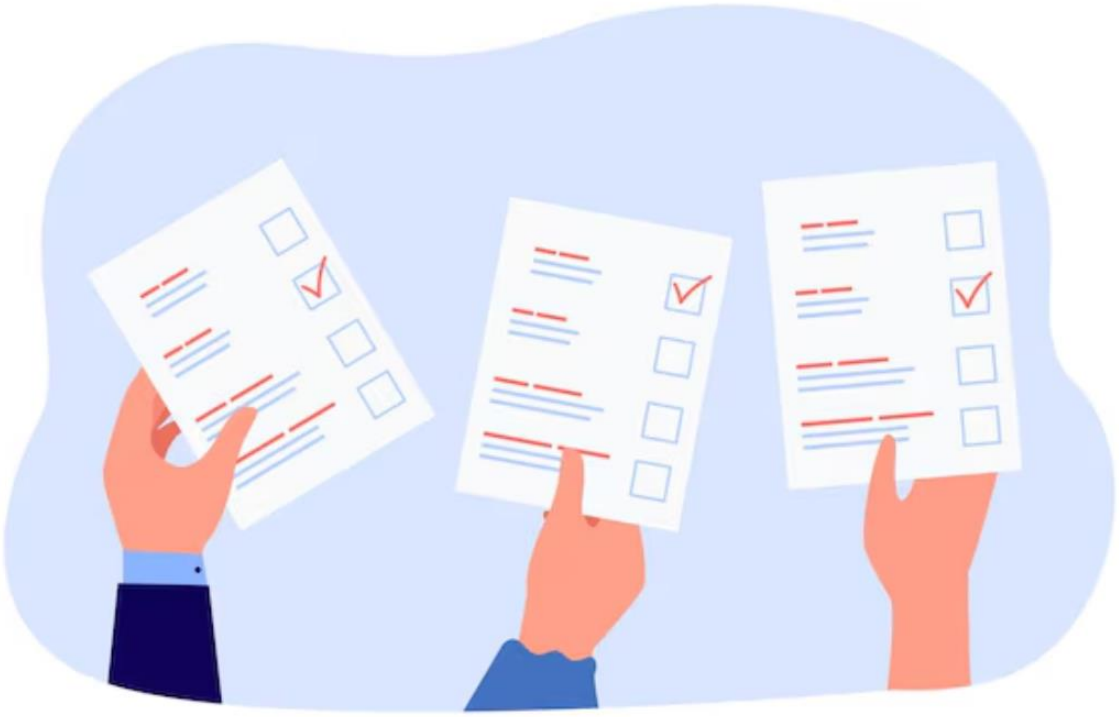
Année de réalisation
2025-2026

CAHIER DES CHARGES

JOBTRACKER

APPLICATION DE SUIVIE ET GESTION DES CANDIDATURES



Entreprse: LJ Conception
Contact: jeanlucas84@outlook.com / 06.61.03.05.94

Table des matières :

1. Informations de contexte :

1.1. Présentation du projet

JobTracker est une application pour suivre ses candidatures lors d'une recherche d'emploi. Elle centralise l'ensemble du processus de candidature dans un seul espace avec plusieurs outils de suivi et de gestion tels que :

- Des statistiques et des graphiques pour analyser les données des candidatures.
- Un suivi du statut des candidatures (en attente, entretien, acceptée, rejetée).
- Une personnalisation de l'affichage des candidatures avec des filtres et un système de recherche.
- Une automatisation des tâches manuelles (export CSV/PDF, templates d'email intégrée et rappels de suivi des candidatures).

1.2 Contexte

De nos jours, le marché de l'emploi est de plus en plus concurrentiel, les candidats doivent souvent multiplier les candidatures pour augmenter leurs chances d'avoir une réponse et de décrocher un entretien.

Cette situation soulève plusieurs problématique concernant le processus de suivi des candidatures comme :

- **Des perte d'informations :** les candidats utilisent plusieurs méthodes pour prospecter (email, réseaux sociaux, en physique) ce qui peut augmenter le risque d'oublier de suivre une candidature.
- **Un manque de visibilité :** sans un outil centralisé, il peut être compliqué d'avoir une vue d'ensemble sur l'état d'avancement de ses candidatures (rappels, relance...).
- **Un manque d'organisation :** la gestion manuelle des candidatures peut vite devenir compliqué et chronophage avec la rédaction de mails et l'actualisation de son suivie à la main.

1.3 Objectifs

L'objectif de JobTracker est de faciliter le suivi de ses candidatures en centralisant tous les outils au sein d'une seule et même application afin de résoudre les problématiques liées au manque d'organisation, au oubli mais aussi pour réduire le temps passé dessus.

1.4. Public cible / utilisateurs visés

JobTracker s'adresse avant tout pour les personnes en recherche d'emploi, qu'il s'agisse de jeunes diplômés, de personnes en reconversion professionnelle ou de demandeurs d'emploi.

2. Expression des besoins :

2.1 Besoins fonctionnels

Ces éléments définissent les principales fonctionnalités qui seront intégrées dans Jobtracker au niveau de l'interaction entre l'utilisateur et l'application.

Authentification des utilisateurs :

- Inscription *(cryptage du mot de passe)*
- Connexion

Gestion des candidatures :

- CRUD des candidatures *(Création, Modification, Suppression)*
- Filtres et recherches des candidatures
- Panneau de rappels des candidatures

Affichages des différentes vues :

- Tableau de bord avec affichage multiples *(Carte, Kanban, Tableau)*
- Details pour les informations et actions sur les candidatures
- Profil pour la gestion des informations personnelles
- Administration *(Backoffice)* pour la gestion des utilisateurs

Outils annexe :

- Graphique des données sur les candidatures
- Exportation au format PDF ou CSV *(tableur)* des candidatures
- Génération automatique de mail via des modèles prédéfinis *(relance, remerciement, candidature spontanée)*

2.2 Cas d’usage (user stories)

En tant qu'utilisateur non connecté, je veux :

- US-01 : Créer un nouveau compte avec mes informations personnelles
- US-02 : Me connecter avec mon email et mon mot de passe pour accéder à mon espace personnel

En tant qu'utilisateur connecté, je veux :

- US-03 : Ajouter une nouvelle candidature pour suivre mes démarches
- US-04 : Consulter la liste de toutes mes candidatures pour avoir une vue d'ensemble
- US-05 : Modifier les informations d’une candidature pour la mettre à jour
- US-06 : Supprimer une candidature que je ne souhaite plus suivre
- US-07 : Rechercher mes candidatures par mots-clés pour les retrouver rapidement
- US-08 : Filtrer mes candidatures par statut, entreprise ou dates pour organiser ma recherche
- US-09 : Visualiser mes candidature selon la vue que je souhaite *(Cartes, Tableau, Kanban)*
- US-10 : Déplacer une candidature entre les colonnes Kanban pour changer rapidement son statut *(en attente, entretien, accepté, refusé)*
- US-11 : Être averti des rappels de candidatures pour effectuer une action
- US-12 : Visualiser des statistiques sur mes candidatures pour analyser mon avancement dans mes démarches
- US-13 : Exporter mes candidatures au format PDF ou CSV pour partager ou archiver mes données
- US-14 : Utiliser des templates d'email pré-remplis pour gagner du temps dans la rédaction des mails *(relance, remerciement, candidature spontanée)*
- US-15 : Consulter et modifier mon profil pour maintenir mes informations personnelles à jour

En tant qu'administrateur, je veux :

- US-16 : Accéder à une vue d’administration *(Backoffice)* pour gérer les utilisateurs *(consulter, modifier les rôles, supprimer)*
- US-17 : Visualiser les statistiques globales de la plateforme pour suivre l'activité

2.3 Contraintes spécifiques

Le projet devra respecter un ensemble d'exigences définies par le client, réparties en 3 points : *(fonctionnelles, techniques et réglementaires)*

Contraintes fonctionnelles :

- Développement d'un espace d'administration complet en + de l'application avec à minima un CRUD opérationnel dans l'interface.
- Communication asynchrone avec le serveur *(API)* pour actualiser les données des candidatures sur l'interface sans recharger la page.
- L'application doit être responsive et s'adapter aux différentes tailles d'écran possible *(mobile, tablette, desktop)*.
- Mise en place du système d'inscription, de connexion et gérer la session de l'utilisateur de manière sécurisé.

Contraintes techniques :

- Conception de l'application en "***mobile firs***t" avec utilisation de media queries pour le responsive.
- Utilisation de préprocesseur CSS *(SCSS)* avec une architecture modulaire au lieu de framework préétablie.
- Nommage des variables par rapport aux normes établies *(camelCase, kebab-case, snake_case)*.
- Vérification et validation des champs de formulaire pour éviter les injections de mauvais caractères *(faille XSS, SQL)*.
- Documentation du code avec schéma de la base de données, des choix techniques et de l'architecture de l'application.

Contraintes réglementaires :

- Validation du code HTML/CSS par rapport aux standards W3C et des standards du web dans le développement.
- Respect des bonnes pratiques d'accessibilité pour les personnes en situation de handicap.
- Respect des règles de sécurité pour le stockage des mots de passe *(cryptage obligatoire)*.

3. Veille technique :

UseMemo – (voir annexe)

3.1 Contexte

useMemo est un hook React qui mémorise le résultat d'un calcul avant que le rendu du composant ne soit exécuté. Ça permet de mettre à jour une valeur seulement si elle change ce qui évite les recalcule inutiles lors des re-render avec React. L'utilisation de ce hook améliore les performances de l'application, notamment sur les supports mobile/tablette disposant de moins de puissance au niveau du CPU.

3.2 Cas d'utilisation

Dans JobTracker, useMemo sera utilisé à plusieurs endroits stratégiques :

- Filtrage des candidatures : Le filtrage ne se recalculera que quand les candidatures, la recherche ou les filtres changent, évitant les recalculs à chaque frappe dans la barre de recherche.
- Calcul des statistiques : Les données du graphique seront recalculées uniquement quand les candidatures sont mises à jour pour éviter le rechargement des animations des graphiques.

3.3 Exemple d'intégration

```
const candidaturesFiltres = useMemo(() => {
   return candidatures.filter(c =>
     c.poste.toLowerCase().includes(recherche.toLowerCase())
  );
}, [candidatures, recherche]);
```

useMemo(() => {}, []) indique à React de mémoriser le résultat du filtrage.
La fonction candidatures.filter(...) ne sera recalculée que si l'une des dépendances suivante change :

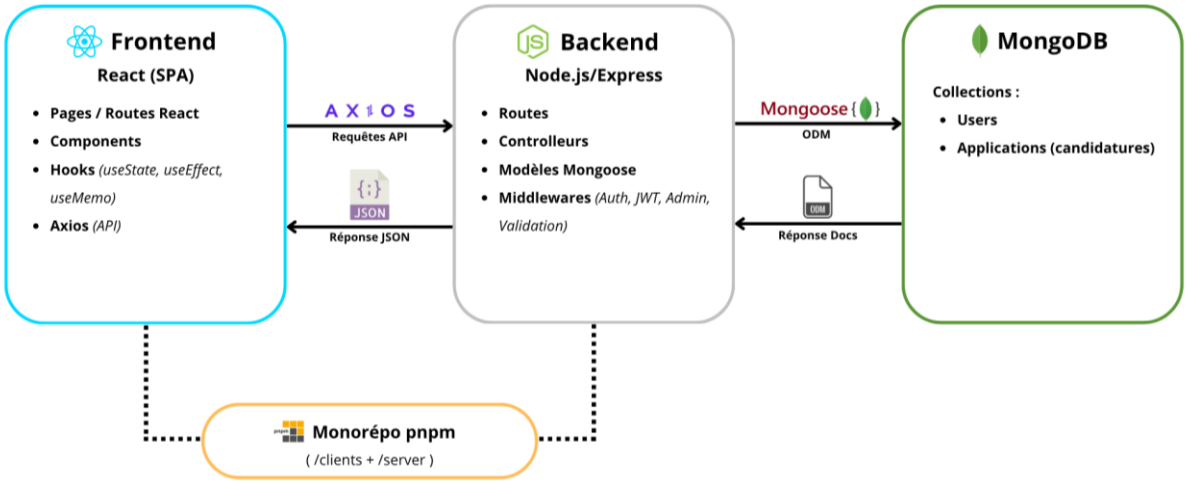
- candidatures
- recherche

4. Spécifications techniques :

4.1. Architecture de la solution

JobTracker sera développé selon l'architecture client/serveur suivante :

- **Frontend:** Application React servie par Vite, communication avec l'API via Axios
- **Backend:** API REST Node.js/Express structurée en MVC *(routes, contrôleurs, modèles)*
- **Base de données:** MongoDB avec Mongoose pour la modélisation et les requêtes
- **Organisation:** Monorepo avec pnpm workspaces *(dossiers client/, server/)*



4.2. Justification des choix techniques

React + Vite : L'utilisation de React pour l'interface de JobTracker permettra de découper l'application en composants réutilisables et propose aussi des packages comme React Router pour la navigation. Vite sera utilisé comme outil de build pour faciliter le développement *(compilation rapide, hot reload, configuration plus simple que Webpack)*.

Axios : Axios a été choisi plutôt que fetch natif car il offre des intercepteurs pour gérer automatiquement les tokens JWT dans les headers, une gestion simplifiée des erreurs HTTP et une syntaxe plus simple que fetch.

Node.js/Express : Pour le backend, nous utiliserons Express car c'est un framework léger et performant, parfait pour une API REST et qui offre aussi de nombreux middlewares pour la sécurité *(Helmet, rate limiting)* et la validation.

MongoDB + Mongoose : Nous opterons pour MongoDB plutôt qu'à une base SQL car les données de candidatures ont une structure variable. Le schéma flexible de MongoDB s'adapte mieux à ces variations et Mongoose apporte la validation des données et simplifie les requêtes.

pnpm : pnpm permettra de partager les dépendances entre les packages du monorepo et de faciliter l'exécution des scripts entre le client et le serveur.

5. Organisation de l’équipe projet :

5.2. Rôles et périmètre d'action

Pour la phase de développement de JobTracker, une équipe réduite sera mise en place :

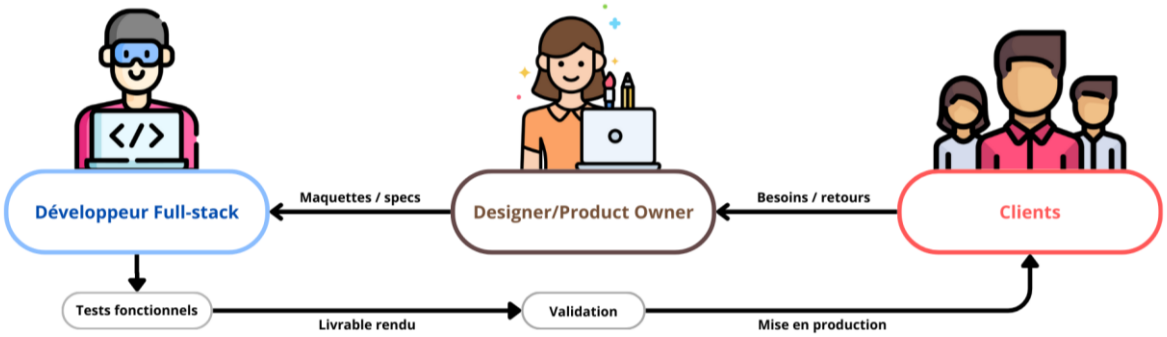
Développeur full-stack :

- Périmètre : Développement complet de l'application
- Responsabilités :
    - Architecture technique et structure du projet
    - Développement des fonctionnalités (authentification, gestion des candidatures, statistiques)
    - Intégration frontend/backend
    - Configuration de la base de données MongoDB
    - Mise en place de la sécurité *(JWT, validation formulaire)*

Designer UX/UI *(product owner par défault)* **:**

- Périmètre : Conception de l'interface et de l'expérience utilisateur et recueil des besoins
- Responsabilités :
    - Création des wireframes et maquettes
    - Optimisation de l'expérience utilisateur *(parcours, interactions)*
    - Validation de l'accessibilité visuelle
    - Écoute des besoins du client

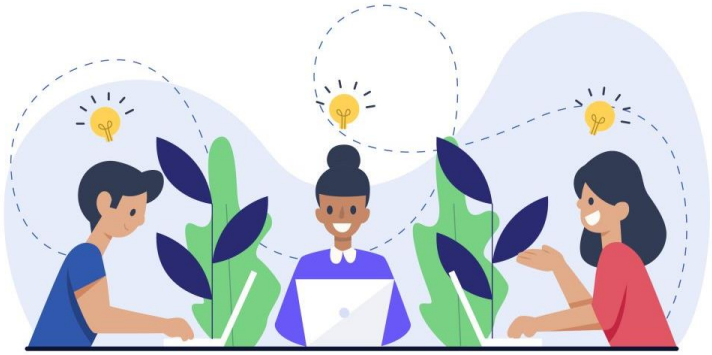
5.2. Interactions et méthodes de travail



Méthode de développement :

- Le designer UX/UI fournit les maquettes selon les demandes utilisateurs
- Le développeur full-stack implémente les fonctionnalités selon les maquettes
- Tests fonctionnels par le développeur et validation avec le designer
- Itération et améliorations basées sur les retours utilisateurs

6. Outils collaboratifs et méthodes :



6.1. Gestion de projet

- Outil de gestion de projet *(GitHub Projects) :*
    - Backlog : Fonctionnalités priorisées par rapport au MVP
    - En cours : Tâches en cours de développement *(tests fonctionnels compris*)
    - Terminé : Fonctionnalités validées

- Méthodologie de travail (*approche itérative ~ 1 sprint par semaine*)
- Communication *(Discord, Teams)* pour les réunions avec le client

6.2. Gestion de version et collaboration

- Utilisation de Git pour le versionning de l'application
- GitHub/SmartGit pour l'hébergement du projet
- Branches principales : master (*production*), dev (*intégration*)
- Branches features : feature/nom-fonctionnalite (*rattaché à la branche dev*)

6.3. Environnement de développement

- Outil de développement Visual Studio Code, Postman *(requête API)*
- Documentation du projet *(README.md, docs/, cahier des charges)*
- Documentation de l'API via Redoc CLI
- Commentaires dans le code

7. Rétroplanning :

Durée totale : 14 semaines (Octobre 2024 - Janvier 2025)

Phase 1 : Conception du projet (Semaines 1-3)

Semaine	Rôle	Tâches principales	Livrables
S1	Développeur + Designer	• Faire une recherche concurrentielle • Brainstormer sur les idées du futur projet	• Listes de fonctionnalités
S1-S2	Développeur	• Rédiger user stories détaillées • Définir un MVP avec features priorisées	• Product Backlog & MVP établis
S2	Designer	• Créer des wireframes/maquettes	• Wireframes/Maquettes
S3	Développeur	• Définir l'architecture de l'application	• Liste des Stacks définis
S3	Développeur	• Configurer l'environnement technique (Git, Github Project)	• Github initialisé
S1-S3	Développeur	• Finaliser le cahier des charges	• CDG complet

Phase 2 : Développement MVP (Semaines 4-8)

Semaine	Type	Tâches principales	Livrables	User Stories
S4	Backend + DB	• Modéliser la BDD (MongoDB) • Initialiser le serveur Express (API) • Créer les endpoints d'authentification (register, login, logout)	• BDD opérationnelle • Backend initialisé • Endpoints authentification	US01, US02
S5	Frontend	• Créer les pages Login, Register, Dashboard • Mettre en place le routing avec React	• React app initialisé • Pages Login, Register et Dashboard	US01, US02
S6	Backend + Frontend	• Créer les endpoints CRUD pour les candidatures • Créer la page détails d'une candidature • Intégrer le CRUD sur le front	• Endpoints candidatures • CRUD candidatures complet • Page Details	US03, US04, US05, US06
S7	Frontend	• Ajouter les vues Cartes, Kanban, Tableau • Faire le système drag & drop pour la vue Kanban • Intégrer le switch entre les vues sur le dashboard	• Nouvelles vues (Kanban,Cartes,Tableau)	US09, US10
S8	Backend + Frontend	• Ajouter le système de recherche et de filtrage (statut, entreprise, dates) • Optimiser les perfs avec useMemo	• Recherche et filtrages	US07, US08

Objectif : MVP fonctionnel avec les fonctionnalités essentielles (authentification, gestion candidatures, recherche, filtres)

Phase 3 : Fonctionnalités avancées (Semaines 9-12)

Semaine	Type	Tâches principales	Livrables	User Stories
S9	Backend + Frontend	• Ajouter la vue des rappels • Intégrer la logique des rappels • Ajouter la vue des statistiques (Recharts)	• Nouvelles vues (Rappel, Graphique)	US11, US12
S9	Frontend	• Créer l'export PDF, CSV des candidatures • Intégrer le graphique dans le PDF	• Export CSV et PDF	US13
S10-S11	Frontend + Backend	• Créer la page de génération d'email • Ajouter la vue Profil utilisateur • Ajouter les endpoints pour la modification du profil de l'utilisateur	• Page Composer Email • Nouvelle vue (Profil) • Endpoints profil	US14, US15
S12	Backend + Frontend	• Créer la page Admin • Créer les endpoints CRUD pour le backoffice • Intégrer le CRUD sur le backoffice	• Backoffice	US16, US17

Phase 4 : Finalisation du projet (Semaines 13-14)

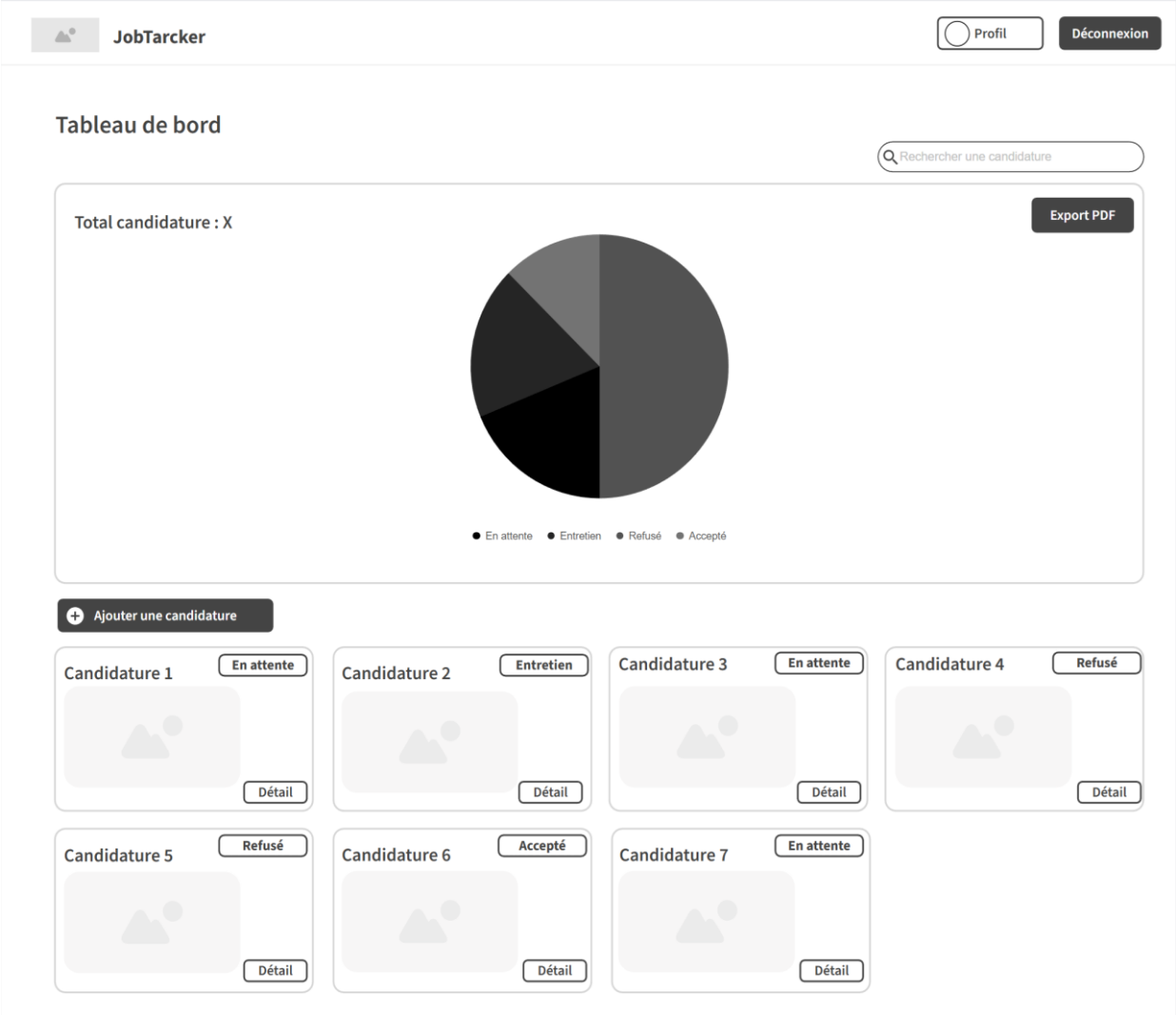
Semaine	Rôle	Tâches principales	Livrables
S13	Frontend	• Intégrer le changement de thèmes (clair/sombre) • Finaliser le responsive design • Accessibilité (ARIA, navigation clavier) • Réaliser les tests WCAG 2.1 AA	• Interface accessible • Responsive OK • Darkmode
S14	Frontend + Backend	• Corriger les derniers bugs • Rédiger la documentation technique complète • Finaliser le déploiement	• Documentation complète • Version déployable

8. Annexes :

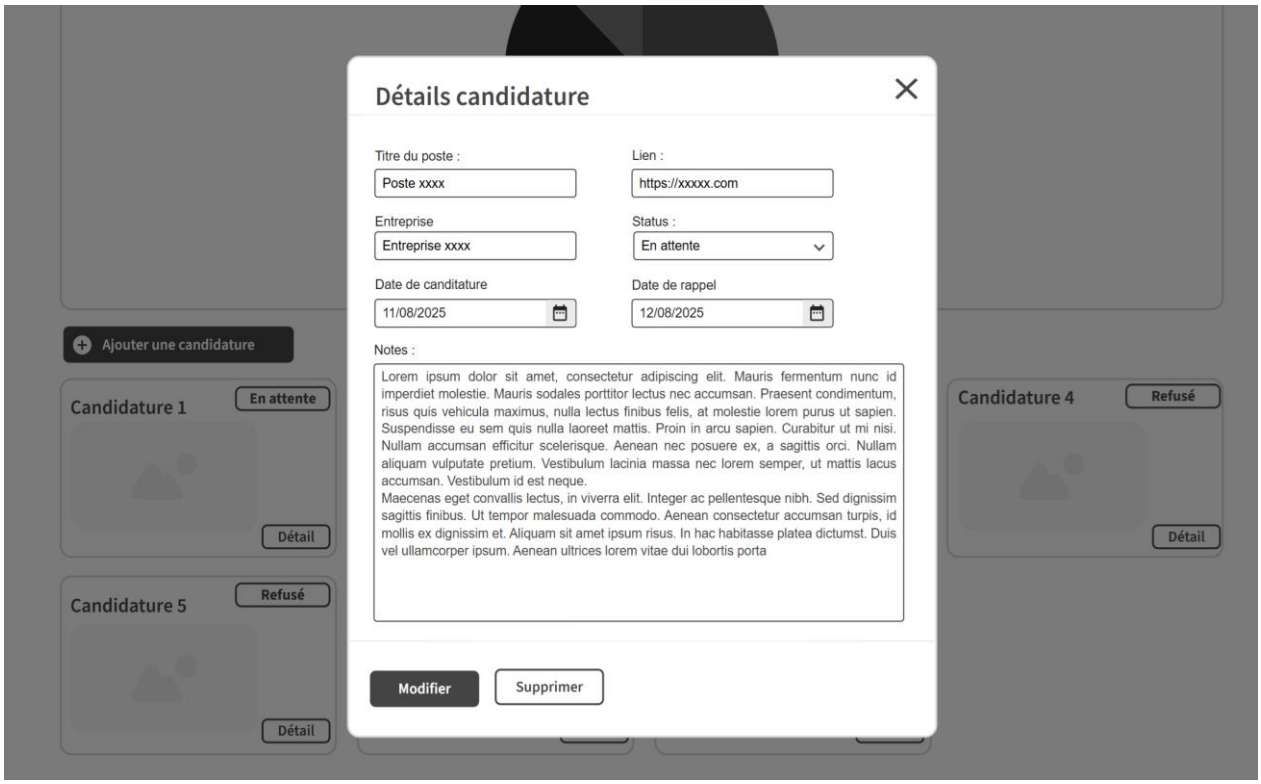
8.1. Wireframes

Les wireframes des principales pages ont été créée avec *Moqups* et sont inclus dans le dossier **docs/wireframes/** du projet :

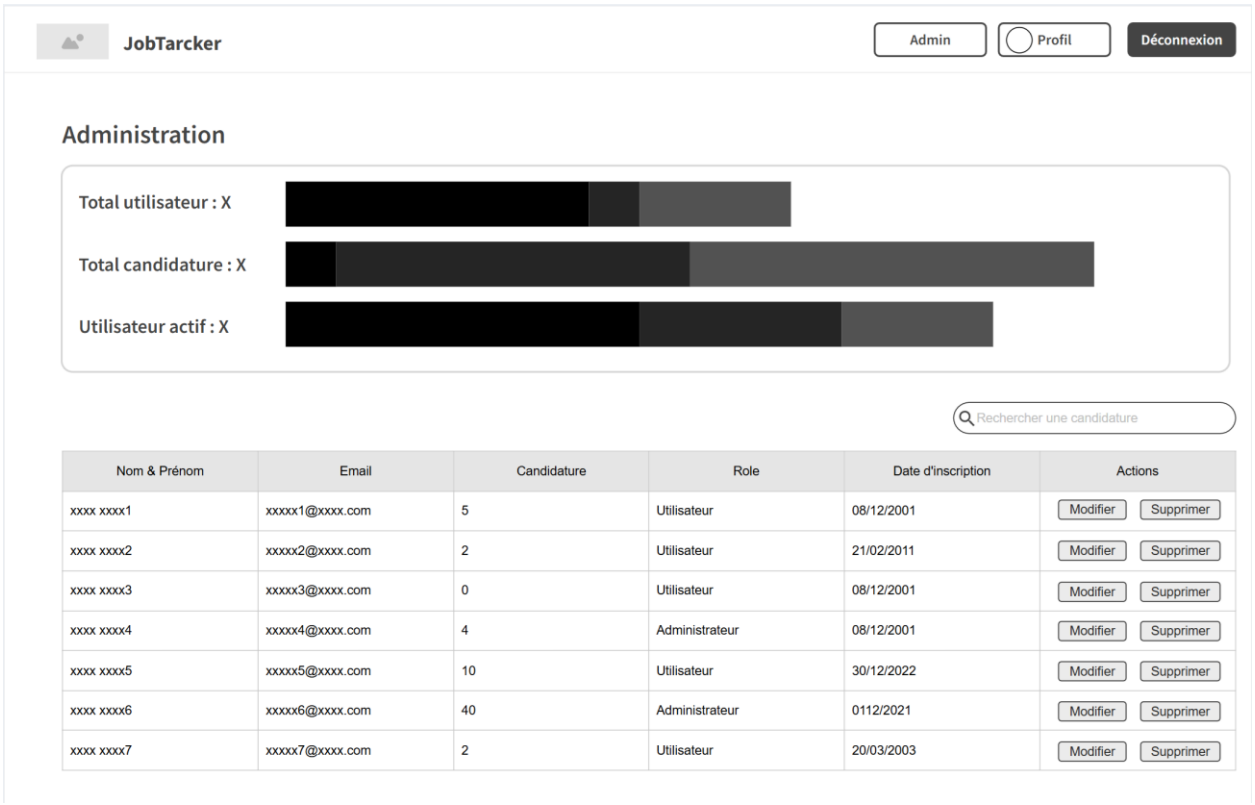
Tableau de bord *(vue principale)* **:**



Vue de détails des candidatures :

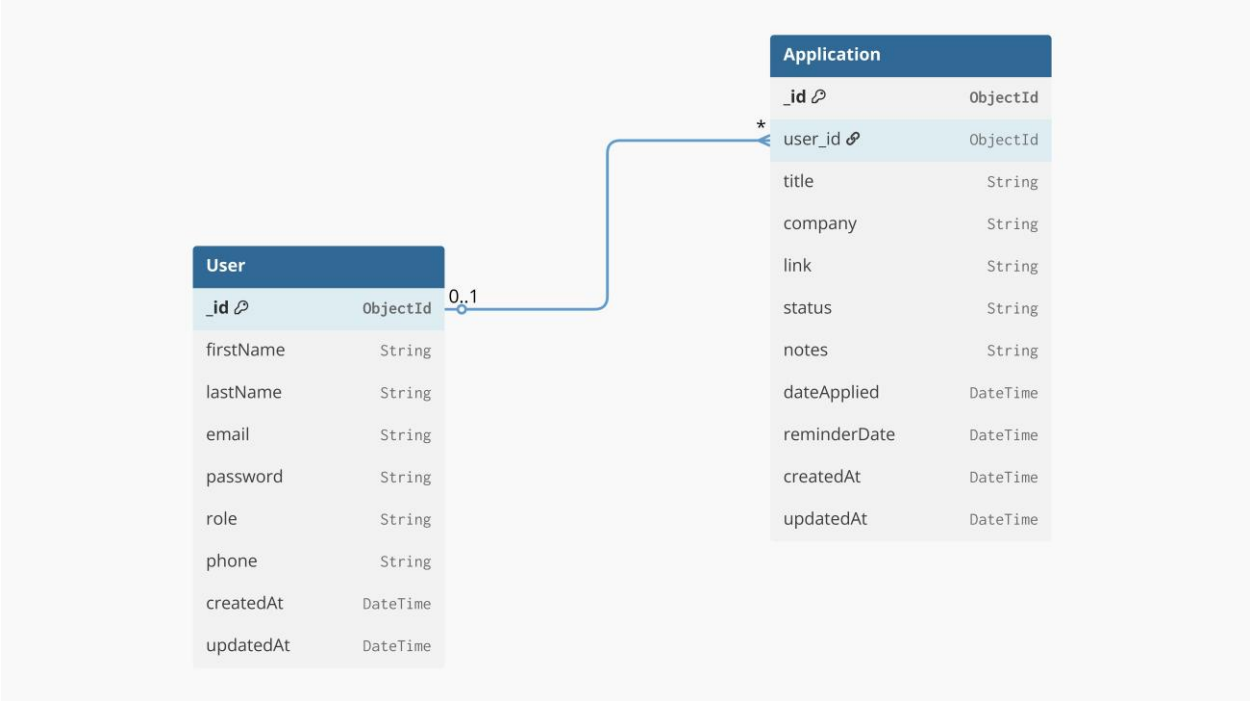


Backoffice *(page d'administration)* **:**



8.2. Diagramme UML

Les diagrammes UML de l'application a été fait avec *dbdiagram.io* et sont inclus dans le dossier **docs/diagrammes/** du projet :



8.3. Référence de veille

- UseMemo : https://fr.react.dev/reference/react/useMemo
- Mongoose : https://mongoosejs.com/docs/index.html
- React Render : https://react.dev/learn/render-and-commit