Brief de la soutenance croisée 2 2025/2026

Projet : JobFinder

Contexte du Projet

JobFinder est une application de recherche d'emplois qui permet aux chercheurs d'emploi de consulter des offres d'emploi provenant de plusieurs sources internationales via une ou plusieurs APIs publiques. Le projet doit être développé comme une Single Page Application (SPA) en utilisant Angular en frontend uniquement ce qui veut dire qu il n y aura pas de backend custom développé, mais utilisation de JSON Server pour simuler une API REST et persister les données (utilisateurs, favoris, candidatures).

L'application permet aux utilisateurs de :

- Rechercher des offres d'emploi via des APIs publiques
- Sauvegarder leurs offres favorites
- Suivre l'état de leurs candidatures
- Gérer leur profil personnel

Fonctionnement de l'application

1. Inscription et Connexion

Les chercheurs d’emploi peuvent créer un compte en renseignant les informations suivantes :

- Nom
- Prénom
- Email
- Mot de passe

L’interface de connexion est unique et repose sur une authentification basique par email et mot de passe, sans nécessité de choisir le type d’utilisateur à l’avance.

L’interface d’inscription est accessible à tous les nouveaux utilisateurs.

Chaque utilisateur peut :

- Modifier ses informations personnelles (nom, prénom, email, mot de passe)
- Supprimer son compte si nécessaire

Mécanisme de fake authentification

Les comptes utilisateurs sont stockés dans JSON Server (table "users").

À la connexion :

1. Vérification email/password contre la table "users" de db.json
2. Si valide : l'objet utilisateur (sans le mot de passe) est stocké dans le `sessionStorage ou dans le localStorage` afin de protéger les données sensibles côté frontend.
3. Ce token local est utilisé pour vérifier l'accès aux routes protégées via authGuard

Cette approche permet de simuler un backend sécurisé avec JSON Server, tout en conservant les fonctionnalités de CRUD pour les utilisateurs et l'authentification côté frontend.

2. Recherche d'Emplois :

Le chercheur d'emploi (même s'il n'est pas authentifié) peut effectuer une recherche d'offres d'emploi avec les critères suivants :

Filtres de recherche obligatoires :

- Mots clés (titre du poste) : champ texte (input) dans une barre de recherche
- Localisation (ville, pays, région) : liste déroulante (select) ou champ texte (input) pour saisir la localisation souhaitée

Exigences métier :

- La recherche par mot clé doit retourner uniquement les offres dont le titre contient le(s) mot(s)clé(s) saisi(s). Les offres qui contiennent le mot clé uniquement dans la description ne doivent pas être affichées
- Les résultats doivent être triés par date de publication (du plus récent au plus ancien)
- Un indicateur de chargement doit être affiché pendant la recherche.

Note : D'autres filtres optionnels peuvent être ajoutés selon les possibilités offertes par l'API choisie (type de contrat, niveau d'expérience, salaire minimum, etc.)

Affichage des résultats :

Liste des offres d'emploi paginée (10 résultats par page par défaut)

Pour chaque offre, afficher :

- Titre du poste
- Nom de l'entreprise
- Localisation
- Date de publication
- Description courte
- Lien vers l'offre complète
- Salaire (si disponible)

- Un bouton "Voir l'offre" qui permet de rediriger vers le site source de l'offre
- Un bouton "Ajouter aux favoris" (visible uniquement pour les utilisateurs authentifiés)
- Un bouton "Suivre cette candidature" (visible uniquement pour les utilisateurs authentifiés)

3. Gestion des Favoris

L'accès à cette fonctionnalité est réservé aux utilisateurs authentifiés.

Fonctionnalités

- Ajouter une offre à ses favoris depuis la liste des offres via le bouton "favoris" (visible uniquement pour les utilisateurs authentifiés)
- Consulter ses offres favorites dans une page dédiée.
- Supprimer une offre de ses favoris.

Exigences métier

- Un utilisateur ne peut pas ajouter deux fois la même offre à ses favoris.
- Un indicateur visuel doit signaler si une offre est déjà présente dans ses favoris.

Mécanisme technique : Gérée par JSON Server

- Chaque favori est associé à un **`userId`** pour identifier l'utilisateur propriétaire de l'offre.
- Les offres favorites sont stockées dans JSON Server dans une table `favoritesOffers` sous la forme :

```
{
  "id": 1,
  "userId": 2,
  "offerId": 101,
  "title": "Développeur Angular",
  "company": "Entreprise A",
  "location": "Casablanca"
}
```

Cette partie de gestion des favoris doit être gérer surtout avec NgRX

4. Suivi des Candidatures

L'accès à cette fonctionnalité nécessite que l'utilisateur soit **authentifié**.

Fonctionnalités pour le chercheur d'emploi

- Ajouter une offre au suivi des candidatures à partir de la liste des offres via le bouton "Suivre cette candidature" (visible uniquement pour les utilisateurs authentifiés)
- Consulter la liste de toutes les candidatures suivies.
- Ajouter des notes personnelles pour chaque candidature (optionnel).
- Supprimer une candidature de sa liste de suivi.

Statuts des candidatures

- En attente (par défaut) : candidature envoyée, en attente de retour
- Accepté : l'utilisateur met le statut manuellement lorsque la candidature est acceptée (convoqué pour entretien ou offre acceptée)
- Refusé : l'utilisateur met le statut manuellement si la candidature est rejetée

Mécanisme technique : Gérée par JSON Server

Chaque candidature est associée à un userId et contient les informations de l'offre suivie ainsi que son statut et les notes personnelles.

Les candidatures sont stockées dans JSON Server dans une table applications sous la forme

```
{
  "id": 1,
  "userId": 2,
  "offerId": "101",
  "apiSource": "adzuna",
  "title": "Développeur Angular",
  "company": "Entreprise A",
  "location": "Casablanca",
  "url": "https://...",
  "status": "en_attente",
  "notes": "Candidature envoyée le 10/02/2025. Relancer dans 2 semaines.",
  "dateAdded": "2026-02-10T10:30:00Z"
}
```

Note : Le champ "notes" peut être vide initialement (optionnel).

5. Choix des APIs

Dans cette application on doit utiliser AU MINIMUM une API parmi les 4 APIs gratuites proposées ci-dessous. on est libre d'utiliser plusieurs APIs simultanément si on le souhaite (pour agréger les résultats de plusieurs sources), mais au moins UNE SEULE API est obligatoire.

Ressource complète des APIs à utiliser : https://job-finder-api-nine.vercel.app/

Concepts technologiques à utiliser

- Angular version 17 ou plus (module ou standalone au choix)
- Gestion d'état NgRx : pour gérer au minimum la partie favoris
- RxJS/Observables
- Injection de dépendance
- Formulaire via Reactive Forms ou Template Driven Forms
- Bootstrap ou Tailwind
- Guards, resolvers
- Databinding
- Service, Pipes, Parent/Child components, Routing
- Lazy Loading : au moins une route doit être chargée en lazy loading
- Composition de composants : chaque page doit contenir au minimum 2 composants (parent/child ou composants frères)

Authentification :

- L'objet utilisateur (sans mot de passe) est stocké dans sessionStorage ou localStorage afin de gérer l'état de connexion et de sécuriser l'accès aux routes protégées via authGuard

  Note :

  - ➢ sessionStorage : session active uniquement pendant la navigation
  - ➢ localStorage : session persistante même après fermeture du navigateur

Le choix entre ces deux options doit être justifié lors de la soutenance

Données métier :

- Les informations liées aux **favoris** et aux **candidatures** (statuts, notes, association avec `userId`) sont **persistées** via **JSON Server** (`db.json`)

Résumé des données stockées :

- **SessionStorage ou localStorage** : profil utilisateur pour l'authentification et contrôle d'accès.
- **Côté JSON Server (`db.json`)** : favoris des utilisateurs, candidatures avec statuts et notes
- Validations métier avec des messages d'erreurs à afficher
- Responsive design
- HTTP Client pour consommer les APIs RESTful
- Gestion des erreurs HTTP
- Intercepteurs HTTP pour gérer les API keys (optionnel) ou pour gérer les erreurs HTTP de manière centralisée

Compétences techniques visées

- C1N2 : Installer et configurer son environnement de travail en fonction du projet
- C2N2 : Développer des interfaces utilisateur
- C3N2 : Développer des composants métier
- C5N2 : Analyser les besoins et maquetter une application
- C6N2 : Définir l'architecture logicielle d'une application

Compétences transversales visées

- C6N2 : Présenter un travail réalisé en synthétisant ses résultats, sa démarche et en répondant aux questions afin de le restituer au commanditaire
- C8N2 : Interagir dans un contexte professionnel de façon respectueuse et constructive pour favoriser la collaboration

Modalités pédagogiques

- Un travail individuel
- Date de lancement : 09/02/2026
- Deadline : 13/02/2026
- Durée : 5j

Livrables :

- Lien GitHub contenant le code source complet
- un bon ReadMe

Modalités d'évaluation

45 min réparties comme suit :

- 05 minutes : Présentation + démonstration des fonctionnalités de l'application
- 05 minutes : Explication du code et de son organisation
- 20 minutes : évaluation des savoirs(Q/A)
- 15 minutes : Mise en situation

Consignes

Avant le début de la soutenance :

- Préparer votre IDE (Ouvrir le projet)
- Ouvrir votre dépôt GitHub
- Préparer et ouvrir des slides de présentation simples et claires
- Pas d'auto-complétion IA lors de la mise en situation
- Utiliser Redux DevTools (extension) pour NgRx

Déroulement de la soutenance

- Introduction : Commencez par une brève présentation du projet, son objectif, son utilité ainsi que les technologies
- Architecture : Expliquez l'architecture globale du projet, notamment la structure des composants, les relations entre eux
- Démonstration
- Mise en situation : on vous demandera de coder une méthode ou d'apporter une modification à votre code (une logique métier, introduire une nouvel implémentation d'un service, etc.)
- Évaluation des savoirs : Le formateur évaluateur vous posera des questions pour évaluer votre degré de maîtrise des concepts et technologies abordés

Critères de performance

- Implémentation complète des fonctionnalités frontend
- Code propre et bien structuré
- Interface responsive et ergonomique
- Gestion d'état NgRx correctement implémentée
- Validation rigoureuse des données et gestions des erreurs
- Une bonne qualité du code et structure du projet