Projet de remise à niveau – B3 Tech

Titre du projet : Mini-site de gestion de candidatures
Dates : Du 15 au 19 septembre 2025
Durée : 5 jours
Travail en binômes

Contexte

L'entreprise fictive **TalentJump** est en pleine croissance et recrute activement des stagiaires et alternants. Claire, la responsable du recrutement, est débordée. Elle gère actuellement toutes les candidatures via sa boîte mail, ce qui entraîne des oublis et des pertes de temps.

Votre mission est de développer le prototype d'une **application web interne** qui lui permettra de centraliser et de gérer les candidatures de manière simple et efficace.

Objectif

Ce projet vise à réactiver et consolider vos compétences fondamentales en développement web et en programmation orientée objet. L'objectif est de construire une application simple mais complète, en séparant clairement la partie **présentation (HTML/CSS)** de la partie **logique métier (Java)**.

Le projet couvrira le cycle de vie complet d'une candidature :

- **Ajouter** un nouveau candidat.
- **Visualiser** la liste de tous les candidats.
- **Modifier** le statut d'un candidat (ex: "En attente", "Entretien", "Refusé").
- **Trier** et **filtrer** les candidats pour retrouver une information rapidement.

Livrables attendus

1. Interface Web (Front-End Statique)

Une interface claire, responsive et professionnelle.

- `index.html` : Une page d'accueil présentant l'outil TalentJump.
- `candidats.html` : Une page affichant la liste des candidats. La présentation sous forme de **cartes (cards)** est encouragée pour un design moderne.
- `ajouter.html` : Un formulaire soigné pour saisir les informations d'un candidat :
  - Prénom / Nom (`text`)
  - Email (`email`)

    - Poste visé (ex: "Développeur Java", "UI/UX Designer") (`text`)
    - Compétences principales (saisies séparées par des virgules) (`text`)
    - Lien vers le CV/Portfolio (`url`, facultatif)

2. Traitement des données (Back-End Console)

Une application en ligne de commande (console) codée en Java.

- **Classe `Candidat.java`** : Un modèle clair contenant les attributs, un constructeur, et les getters/setters. Pensez à utiliser les types de données appropriés (`String`, `List<String>` pour les compétences, etc.).
- **Classe `GestionCandidats.java`** (ou service équivalent) :
    - Contient la structure de données (`ArrayList<Candidat>`).
    - Implémente les méthodes de gestion (CRUD) :
        - `ajouterCandidat()`
        - `listerCandidats()`
        - `modifierStatutCandidat()`
        - `trierParNom()` ou `trierParPoste()`
        - `filtrerParCompetence()`
- **Lien entre Java et Web :** Le programme Java doit être capable de **générer un fichier HTML** (ex: `rapport_candidats.html`) qui affiche la liste à jour des candidats. C'est la "liaison" entre votre logique et le rendu web.

3. Documentation et Code

- **Dépôt Git** (sur GitHub, GitLab...) partagé par le binôme. Les commits devront être réguliers et explicites.
- **`README.md`** complet contenant :
    - Une brève description du projet.
    - La répartition des tâches au sein du binôme.
    - **Instructions claires** pour compiler et exécuter le programme Java.
    - Les choix de conception que vous avez faits.

4. Soutenance Orale

- **10 minutes de présentation** + 5 minutes de questions/réponses.
- **Démo live** :
    1. Montrer le code Java.
    2. Lancer le programme en console pour ajouter/modifier un candidat.
    3. Lancer la génération du fichier HTML.
    4. Ouvrir le fichier HTML généré dans un navigateur pour montrer le résultat.

- Expliquer vos choix techniques, les difficultés rencontrées et comment vous les avez surmontées.

Critères d'évaluation

Votre projet sera évalué sur les points suivants :

- **Fonctionnalités Java (40%) :** Toutes les fonctionnalités (ajout, tri, filtre, modification de statut) sont implémentées et fonctionnelles.
- **Qualité du code (30%) :**
  - Le code est clair, commenté et respecte les conventions de nommage.
  - Bonne utilisation de la POO (classes, méthodes).
  - L'historique Git est propre et pertinent.
- **Rendu HTML/CSS (20%) :**
  - L'interface est propre et responsive.
  - Le formulaire est ergonomique.
- **Soutenance et Documentation (10%) :** La présentation est claire, la démo fonctionne et le `README.md` est complet.

Contraintes et Outils

- **Front-End :** HTML5, CSS3. **Aucun framework CSS (Bootstrap, Tailwind) ou JS n'est autorisé.**
- **Back-End :** Java (version 11 ou supérieure). **Aucune interface graphique Java (Swing, JavaFX) ni serveur d'application (Tomcat) n'est requis.**
- **Outils :**
  - Un IDE (IntelliJ, Eclipse, VS Code).
  - **Git** pour la gestion de version.
  - Le stockage des données se fait en mémoire (`ArrayList`). Pas de base de données.

Bonus (pour aller plus loin)

Si vous terminez en avance, voici quelques idées pour améliorer votre projet :

- **Suppression d'un candidat**.
- **Sauvegarde et chargement** de la liste de candidats dans un fichier simple (ex: `.json` ou `.csv`).
- Utilisation d'une `enum` en Java pour gérer le **statut de la candidature** (`EN_ATTENTE`, `ENTRETIEN_PLANIFIE`, `ACCEPTE`, `REFUSE`).
- Ajout d'une fonctionnalité de **recherche par mot-clé** dans la liste des candidats.
- Création d'une page de **statistiques simples** (ex: nombre de candidats par poste).

Bon projet à tous !