Test Technique Recrutement Développeur Backend

1. Objectif du Test

Évaluer la capacité du candidat à concevoir et développer une API RESTful **modulaire**, **sécurisée** et **documentée** en utilisant **NestJS** avec **MongoDB** et **Mongoose**.

2. Cas d'usage – API de gestion des offres d'emploi et des candidatures

Développer une **API RESTful** permettant de gérer des offres d'emploi, des utilisateurs (admin, recruteur, candidat) et les candidatures soumises.

3. Fonctionnalités à développer

A. Authentification (JWT)

- `POST /auth/register` → Créer un utilisateur (admin uniquement)
- `POST /auth/login` → Connexion, retour accessToken (JWT)
- Middleware de protection JWT
- Gestion des rôles : `admin`, `recruteur`, `candidat`

B. Offres d'emploi

- `POST /jobs` → Créer une offre (recruteur uniquement)
- `GET /jobs` → Lister toutes les offres
- `GET /jobs/:id` → Voir le détail d'une offre
- `PUT /jobs/:id` → Modifier une offre (recruteur propriétaire uniquement)
- `DELETE /jobs/:id` → Supprimer une offre (recruteur propriétaire uniquement)

C. Candidatures

- `POST /jobs/:id/apply` → Postuler à une offre (candidat uniquement)
- `GET /jobs/:id/applications` → Voir les candidatures reçues (recruteur propriétaire uniquement)
- `GET /me/applications` → Voir ses propres candidatures

---

4. Schéma de base de données (MongoDB / Mongoose)

4.1. `User`

```
Unset
{
  _id: ObjectId,
  email: string,
  password: string (haché),
  role: 'admin' | 'recruteur' | 'candidat',
  createdAt: Date,
}
```

4.2. `Job`

```
Unset
{
  _id: ObjectId,
  title: string,
  description: string,
  recruiter: ObjectId (référence vers User),
  createdAt: Date,
}
```

4.3. `Application`

```
Unset
{
  _id: ObjectId,
```

```
  job: ObjectId (référence vers Job),
  candidate: ObjectId (référence vers User),
  cvUrl: string,
  appliedAt: Date,
}
```

5. Contraintes techniques

- NestJS avec architecture modulaire
- MongoDB + Mongoose
- Validation avec `class-validator` + `DTO`
- Authentification par JWT (access token)
- Swagger à `/api-docs`
- Logger pour suivre les requêtes
- Middleware/Guard pour contrôle d'accès basé sur les rôles
- Gestion centralisée des erreurs

6. Tests (facultatif)

- Au moins 3 tests unitaires ou end-to-end avec **Jest**

7. Livrables attendus

- Code source dans un dossier zippé
- README.md :
    - Setup du projet

    - Exemple `.env`
    - Commandes de lancement (`npm run start:dev`)
    - Accès Swagger
- Export `.bson` ou `.json` pour peupler les données (facultatif)

8. Durée du test

Délai maximum de soumission : **4 jours après réception**