

JobMajunga — Cahier des Charges

Projet de Gestion des Offres d'Emploi

1 MOIS (MARS 2026)

DESKTOP & MOBILE

Nom du projet

JobMajunga

Type

MVP Desktop + Mobile

Application Desktop

C# / .NET 8 / WPF

Application Mobile

Flutter

Backend

Node.js / Express.js

Base de données

MySQL

1. Introduction et Contexte

1.1 Présentation du projet

JobMajunga est un écosystème applicatif complet dédié à la gestion des offres d'emploi. Il se compose de trois composantes interconnectées partageant une même base de données MySQL via une API REST centralisée :



Application Desktop C# WPF

Destinée aux recruteurs et administrateurs RH



Application Mobile Flutter

Destinée aux candidats en recherche d'emploi



Backend Node.js/Express/MySQL

Exposant tous les services métier

Cette architecture garantit une source unique de vérité, une cohérence des données en temps réel et une expérience utilisateur adaptée à chaque contexte d'usage.


1.2 Problématique

Les processus de recrutement actuels souffrent d'une fragmentation des outils : les recruteurs jonglent entre plusieurs logiciels de suivi, et les candidats doivent créer des comptes sur de multiples plateformes. JobMajunga répond à ce besoin en proposant un écosystème unifié où chaque acteur dispose d'une interface taillée sur mesure pour son usage.


1.3 Objectifs

1. Offrir aux recruteurs une interface desktop puissante pour gérer offres et candidatures
2. Offrir aux candidats une application mobile intuitive pour chercher un emploi et postuler
3. Centraliser toutes les données dans une API REST unique et sécurisée
4. Garantir la cohérence des données entre les deux plateformes
5. Assurer la sécurité et la conformité RGPD des données utilisateurs

1.4 Périmètre

 **Inclus**

Application WPF recruteur basique, Application Flutter candidat basique, API REST Node.js, BDD MySQL

 **Cible OS**

Windows 10/11 (WPF) · Android 9+ & iOS 14+ (Flutter) · Linux/Windows (serveur Node.js)

 **Facultatif**

- Export PDF/CV
- Authentification avancée
- Notifications push
- Recherche avancée avec filtres complexes
- Paiement en ligne
- IA de matching
- Messagerie temps réel

2. Acteurs et Rôles

2.1 Identification des acteurs

Acteur	Plateforme	Fonctions principales
Candidat	Mobile Flutter	Consulter les offres, gérer son CV et suivre ses candidatures.
Recruteur	Desktop WPF	Publier des offres d'emploi, gérer les candidatures et consulter les profils.

3.1 Module Authentification

Commun aux deux plateformes. L'API REST gère l'authentification par JWT. Chaque application consomme les mêmes endpoints.

Inscription

Email, mot de passe (bcrypt), rôle (candidat / recruteur)

Connexion

Retour d'un access token JWT (1h) + refresh token (7j)

Renouvellement

Renouvellement automatique du token par les deux apps

Déconnexion

Avec invalidation du refresh token en base

Récupération mot de passe

Par email (token temporaire)

Sécurité

Blocage après 5 tentatives échouées (rate limiting)

3.2 Application Desktop C# WPF — Recruteur

L'application WPF est l'outil quotidien des recruteurs, conçue pour une gestion efficace et productive des offres et des candidatures.

Gestion des offres d'emploi

- Création, modification, duplication et suppression d'offres
- Suivi du cycle de vie : Brouillon, Publiée, Expirée, Archivée
- Recherche et tri par statut, date ou catégorie

Gestion des candidatures

- Tableau de bord de suivi des candidatures reçues
- Consultation intégrée des CV des candidats
- Filtrage simple des candidats par compétences

3.3 Application Mobile Flutter — Candidat

L'application Flutter est optimisée pour la mobilité, permettant au candidat de gérer simplement sa recherche d'emploi depuis son smartphone.

1

Consulter les offres d'emploi

Parcourez les offres disponibles et accédez aux détails de chaque poste pour trouver les opportunités correspondant à votre profil.

2

Postuler à une offre

Soumettez votre candidature directement depuis l'application en quelques clics pour les offres qui vous intéressent.

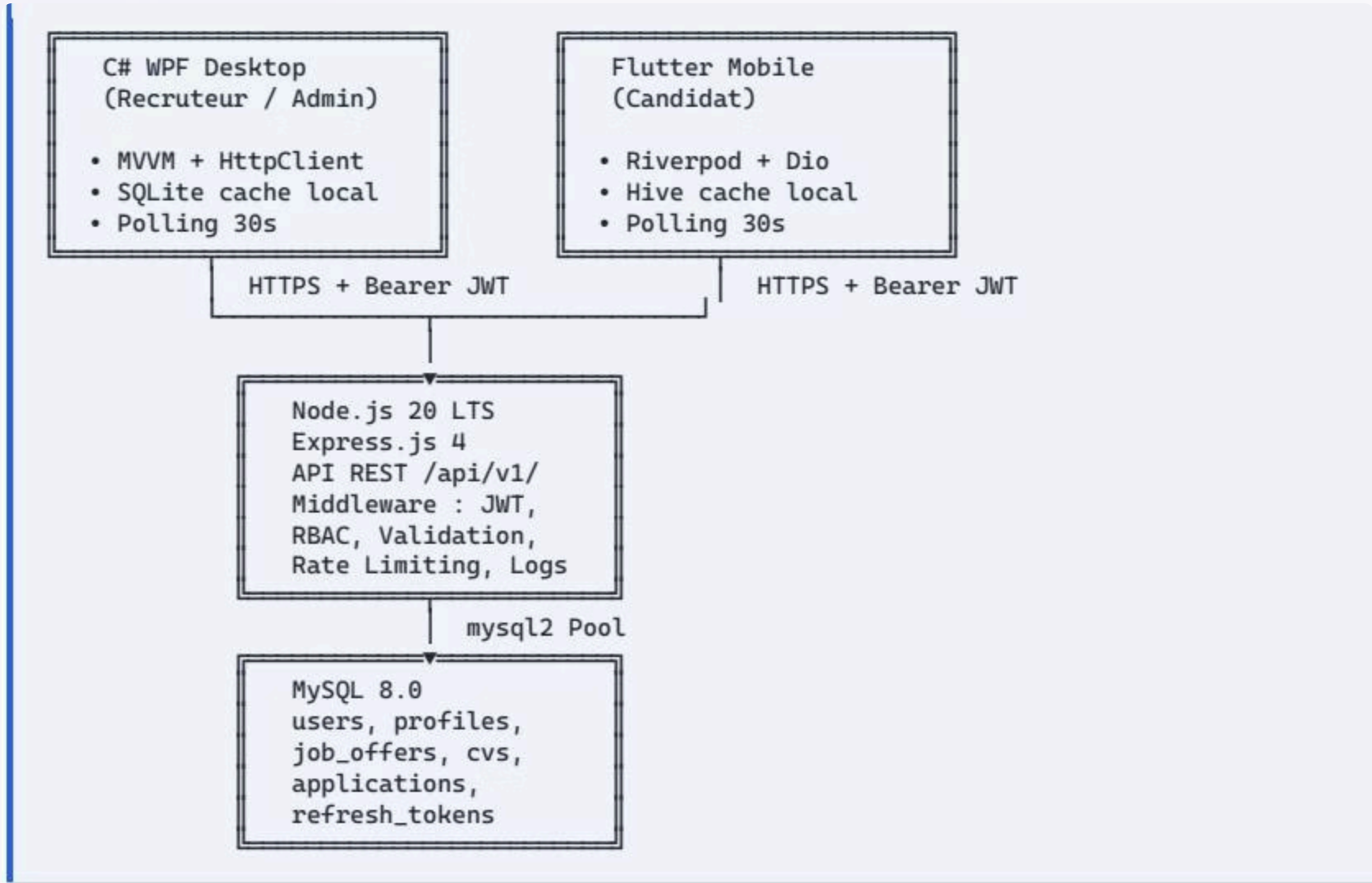
3

Voir l'historique de ses candidatures

Consultez la liste de vos candidatures passées pour suivre l'avancement de vos démarches de recherche d'emploi.

4. Architecture Technique

4.1 Vue d'ensemble

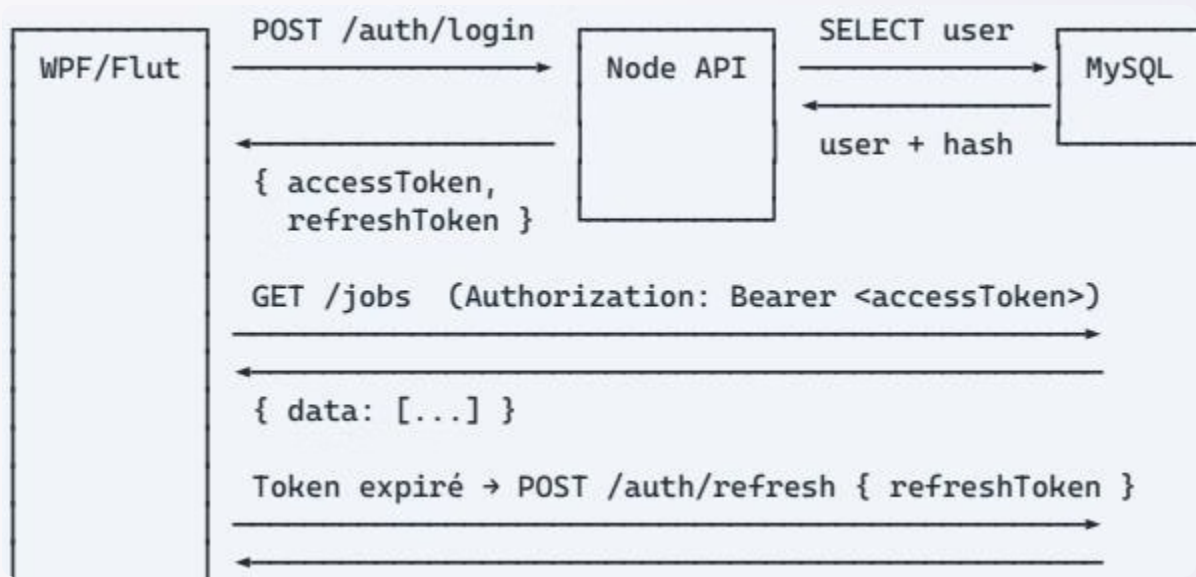


L'architecture repose sur un modèle client-serveur simple : deux applications clientes (WPF et Flutter) communiquent avec un backend central Node.js. Ce backend expose une API REST qui assure la gestion des données via une base de données unique MySQL 8.0.

4.2 Stack technologique

Couche	Clients	Backend
Langage	C# (.NET 8) / Dart 3	Node.js 20 LTS
Communication	HTTP / REST	Express.js 4
Données	—	MySQL 8.0
Auth	JWT	jsonwebtoken

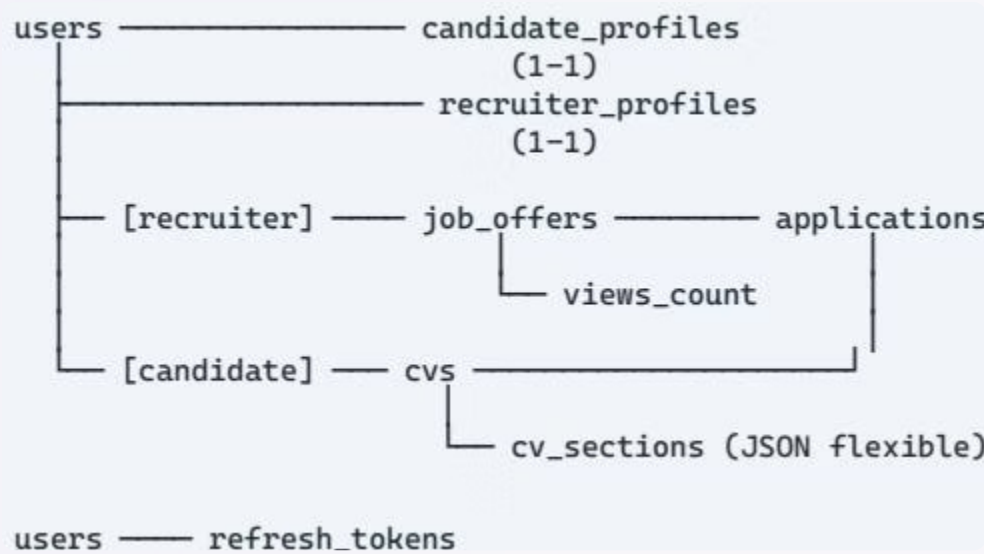
4.4 Flux d'authentification



Le flux d'authentification suit un cycle classique : authentification par identifiants, émission de jetons JWT, et accès aux ressources protégées par vérification du jeton sur l'API.

5. Schéma Base de Données MySQL

5.1 Vue simplifiée du schéma



Le schéma a été simplifié pour se concentrer sur les entités métier essentielles. Il repose désormais sur quatre tables principales : `users` pour la gestion des comptes, `job_offers` pour les annonces, `applications` pour le suivi des candidatures et `skills` pour la gestion des compétences.

5.2 Script SQL simplifié

```
-- TABLE : users
CREATE TABLE users (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  role ENUM('candidate', 'recruiter') NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- TABLE : job_offers
CREATE TABLE job_offers (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  recruiter_id INT UNSIGNED NOT NULL,
  title VARCHAR(255) NOT NULL,
  description TEXT,
  contract_type VARCHAR(50),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (recruiter_id) REFERENCES users(id) ON DELETE CASCADE
);

-- TABLE : applications
CREATE TABLE applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  candidate_id INT UNSIGNED NOT NULL,
  job_offer_id INT UNSIGNED NOT NULL,
  status ENUM('pending', 'accepted', 'rejected') DEFAULT 'pending',
  applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (candidate_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (job_offer_id) REFERENCES job_offers(id) ON DELETE CASCADE
);

-- TABLE : skills
CREATE TABLE skills (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL UNIQUE
);
```

6. Endpoints API REST

 **Base URL :** `https://api.jobmajunga.local/api/v1` — Format JSON — Auth : `Authorization: Bearer` — Pagination : `?page=1&limit=20`

6.2 Auth & Profil

Méthode	Route	Accès	Description
POST	/auth/register	Public	Inscription
POST	/auth/login	Public	Connexion
GET	/profile/me	Auth	Mon profil
PUT	/profile/me	Auth	Mettre à jour profil

6.3 Offres d'emploi

Méthode	Route	Accès	Description
GET	/jobs	Public	Liste des offres
GET	/jobs/:id	Public	Détail offre
POST	/jobs	Recruteur	Créer offre
PUT	/jobs/:id	Recruteur	Modifier offre
DELETE	/jobs/:id	Recruteur	Supprimer offre

6.4 Candidatures

Méthode	Route	Accès	Description
POST	/applications	Candidat	Postuler
GET	/applications/mine	Candidat	Mes candidatures
GET	/applications/received	Recruteur	Candidatures reçues
PATCH	/applications/:id/status	Recruteur	Changer statut

7. Exigences Non Fonctionnelles

7.1 Performance

Temps de réponse API < 500ms

7.2 Sécurité

HTTPS obligatoire et validation stricte des entrées (input)

7.3 Compatibilité

Windows 10+, Android 9+, iOS 14+

8. Planning Réaliste — 1 Mois (4 Semaines)

Ce planning détaillé sur quatre semaines présente les étapes clés pour le développement et le déploiement du projet JobMajunga, en adoptant une approche structurée pour maximiser l'efficacité.

Semaine 1 : Setup & Architecture

1

- Jour 1-2 : Configuration environnements (Node.js, .NET, Flutter, MySQL)
- Jour 3-4 : Design API REST et schéma BDD
- Jour 5 : Mise en place Git, CI/CD basique

Semaine 3 : Applications Clientes

3

- Jour 12-14 : Application WPF (UI + connexion API)
- Jour 15-17 : Application Flutter (UI + connexion API)
- Jour 18 : Tests d'intégration WPF-Flutter-API

2

Semaine 2 : Backend & BDD

- Jour 6-7 : Création tables MySQL (Users, JobOffers, Applications)
- Jour 8-10 : Endpoints API (Auth, Jobs CRUD, Applications CRUD)
- Jour 11 : Tests API et déploiement staging

4

Semaine 4 : Tests & Déploiement

- Jour 19-20 : Tests fonctionnels et corrections bugs
- Jour 21 : Déploiement production
- Jour 22 : Documentation et handover