

JobMajunga — Cahier des Charges

Projet de Gestion des Offres d'Emploi

VERSION 1.0 — MARS 2026

ÉCOSYSTÈME COMPLET · DESKTOP & MOBILE

Nom du projet

JobMajunga

Type

Écosystème Desktop + Mobile + API
REST

Application Desktop

C# / .NET 8 / WPF — Recruteur /
Admin

Application Mobile

Flutter 3.x / Dart — Candidat

Backend

Node.js 20 LTS + Express.js 4

Base de données

MySQL 8.0+

Authentification

JWT (jsonwebtoken + bcryptjs)

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, Application Flutter candidat, API REST Node.js, BDD MySQL, Auth JWT, Export PDF/CV

🖥 Cible OS

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

✗ Exclut

- Application web navigateur
- Messagerie temps réel (WebSocket)
- Paiement en ligne
- IA de matching

2. Acteurs et Rôles

2.1 Identification des acteurs

Acteur	Plateforme	Rôle	Responsabilités principales
Candidat	Flutter	Demandeur d'emploi	Consulter offres, créer CV, postuler, suivre candidatures
Recruteur	WPF	Représentant entreprise	Publier offres, gérer candidatures, consulter CV candidats
Administrateur	WPF	Gestionnaire plateforme	Accès complet, modération, gestion utilisateurs et statistiques
Visiteur	Flutter	Non authentifié	Consultation des offres publiques uniquement

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 / Admin

L'application WPF est l'outil de travail quotidien des recruteurs. Elle exploite la taille de l'écran pour offrir des interfaces denses et productives impossibles sur mobile.

3.2.1 Gestion des offres d'emploi

- Créer une offre : titre, description rich-text, type de contrat, localisation, salaire, compétences requises, date d'expiration
- Statuts d'une offre : **Brouillon** → **Publiée** → **Expirée** → **Archivée**
- Modifier, dupliquer, supprimer ou archiver une offre existante
- Vue liste avec tri, filtres (statut, date, catégorie) et recherche full-text locale
- Statistiques par offre : nombre de vues, nombre de candidatures reçues

3.2.2 Gestion des candidatures

- Tableau de bord Kanban : colonnes par statut (Reçue, Vue, En examen, Entretien, Acceptée, Refusée)
- Consultation du CV du candidat intégré dans l'interface
- Changement de statut par glisser-déposer ou menu contextuel
- Prise de notes internes par candidature (non visible du candidat)
- Planification d'entretien avec date, heure et lieu
- Export de la liste des candidats en Excel et PDF
- Filtres multi-critères : date, statut, compétences, score

3.2.3 Gestion du profil recruteur / entreprise

- Informations entreprise : nom, logo, description, secteur, site web
- Gestion de plusieurs recruteurs rattachés à la même entreprise
- Paramètres de notification (email, fréquence des récapitulatifs)

3.2.4 Tableau de bord et analytics (Admin)

- KPIs globaux : utilisateurs actifs, offres publiées, candidatures du jour
- Graphiques de tendance (LiveCharts2) : évolution sur 30/90 jours
- Taux de conversion offres → candidatures → entretiens → embauches
- Gestion des utilisateurs : activation, désactivation, changement de rôle
- Modération des offres : validation, signalement, suppression
- Logs d'activité et d'erreurs consultables

3.2.5 Fonctionnalités techniques WPF

- Architecture MVVM stricte avec CommunityToolkit.Mvvm
- HttpClient singleton avec gestion automatique du refresh token
- Cache local (SQLite) pour consultation hors-ligne des données récentes
- Polling automatique toutes les 30 secondes pour les données critiques
- Thème clair / sombre avec persistance du choix
- Navigation par onglets et raccourcis clavier documentés

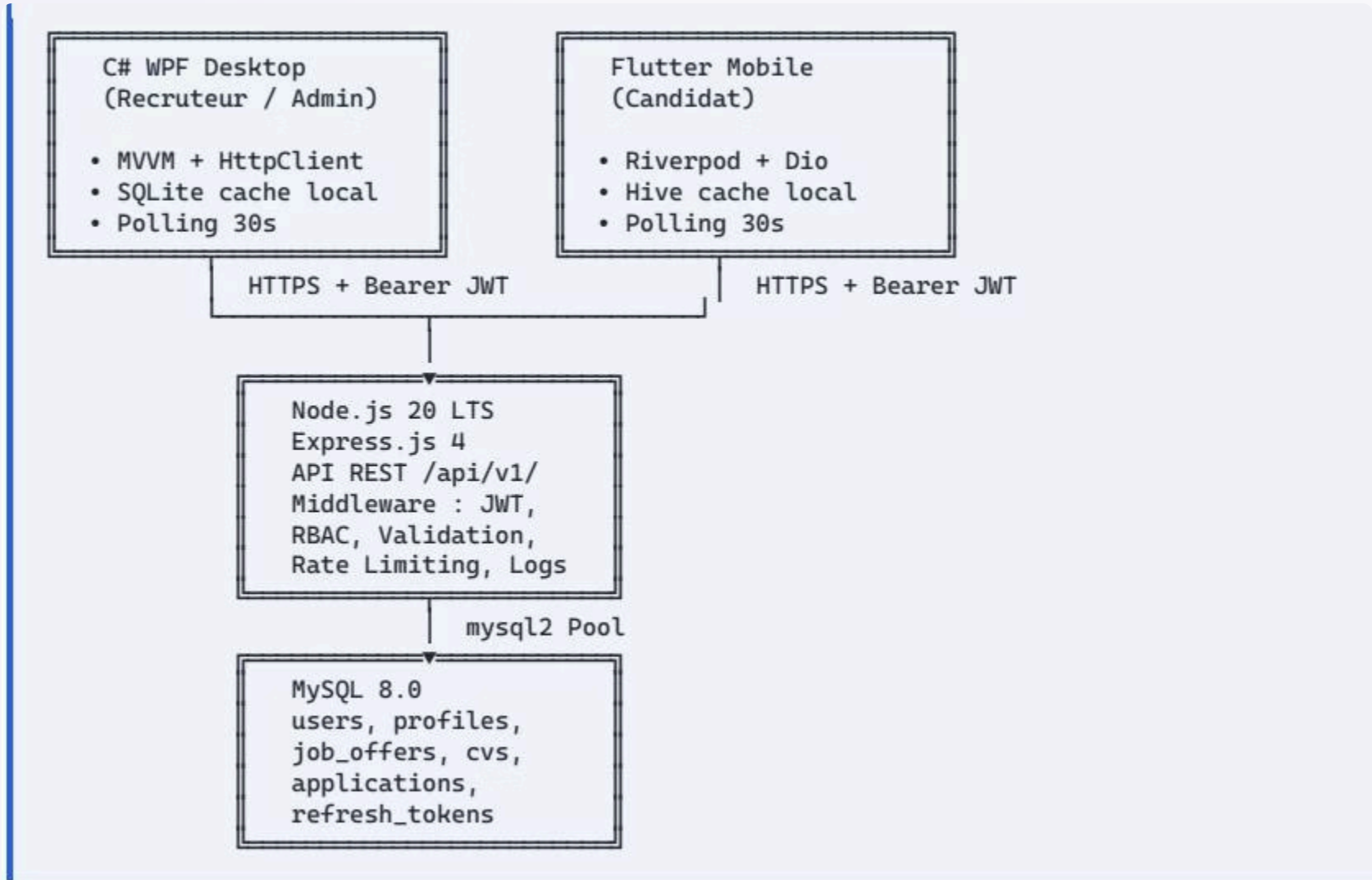
3.3 Application Mobile Flutter — Candidat

L'application Flutter est optimisée pour la mobilité. Elle permet au candidat de gérer toute sa recherche d'emploi depuis son smartphone.

1	<div>3.3.1 Recherche et consultation des offres</div> <ul style="list-style-type: none">Liste des offres avec filtres : localisation, type de contrat, salaire, secteur, mots-clésMoteur de recherche full-text avec suggestions en temps différéCarte interactive (Google Maps) affichant les offres géolocalisées autour de soiFiche détaillée d'une offre avec toutes les informations et bouton PostulerSystème de favoris pour sauvegarder des offresAlertes configurables : nouvelles offres correspondant au profil (polling FCM optionnel)
2	<div>3.3.2 Profil candidat</div> <ul style="list-style-type: none">Informations personnelles : nom, prénom, photo, titre professionnel, localisationCompétences, langues, expériences et formations résuméesParamètres de confidentialité et de notifications
3	<div>3.3.3 Création et gestion de CV</div> <ul style="list-style-type: none">Éditeur de CV simplifié adapté au mobile (sections scrollables)Sections : résumé, expériences, formations, compétences, langues, projetsPlusieurs CV par utilisateur (multi-CV) avec nommage libreChoix d'un template parmi 4 modèles (Classique, Moderne, Minimaliste, Créatif)Export PDF généré par l'API et téléchargeable sur le téléphonePartage du CV par lien public unique
4	<div>3.3.4 Candidatures</div> <ul style="list-style-type: none">Postuler en sélectionnant un CV existant + lettre de motivation optionnelleVérification anti-doublon (impossible de postuler deux fois à la même offre)Tableau de bord de suivi des candidatures avec statuts colorésHistorique complet par candidatureNotifications push locales lors du polling de statut
5	<div>3.3.5 Fonctionnalités techniques Flutter</div> <ul style="list-style-type: none">State management : Riverpod 2.xAppels HTTP : Dio avec intercepteur JWT refresh automatiqueNavigation : GoRouter avec deep linksStockage local : Hive (profil, favoris, CV en cache)Support Android 9+ et iOS 14+, adaptatif tabletteMode hors-ligne partiel : consultation des offres et CV en cache Hive

4. Architecture Technique

4.1 Vue d'ensemble de l'écosystème

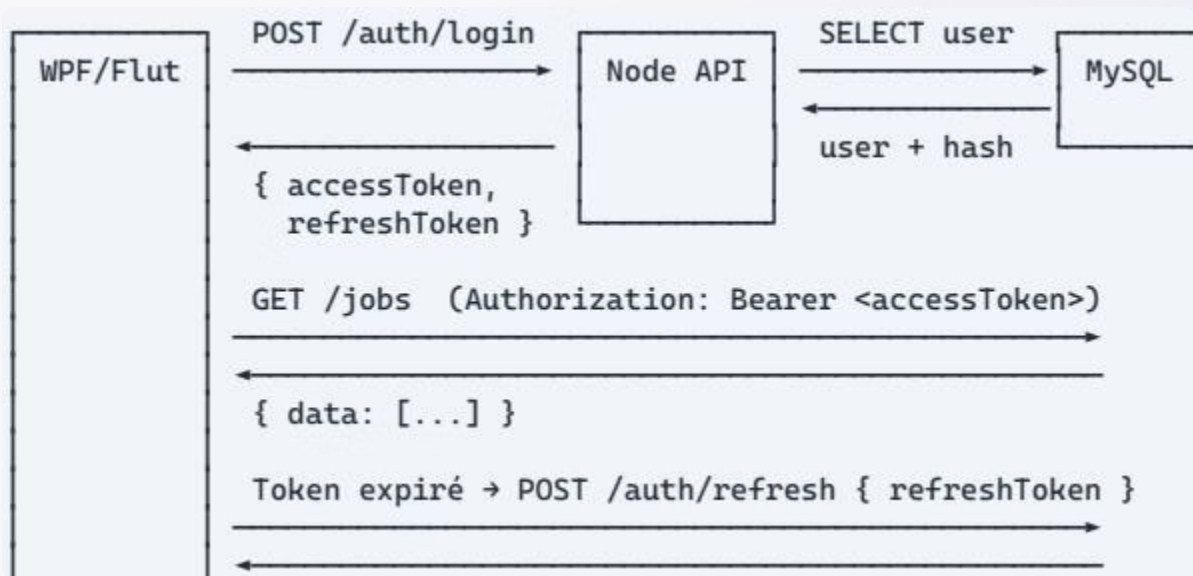


Le diagramme illustre l'architecture du système. Deux applications clientes communiquent avec un backend central Node.js 20 LTS / Express.js 4 via HTTPS et des tokens Bearer JWT. Le backend expose une API REST à /api/v1/ avec des middlewares JWT, RBAC, validation, rate limiting et logging. Il se connecte ensuite à une base de données MySQL 8.0 via un pool mysql2, stockant users, profiles, job_offers, CVs, applications et refresh_tokens.

4.2 Stack technologique complète

Couche	C# WPF	Flutter	Backend / Infra
Runtime / SDK	.NET 8	Dart 3 / Flutter 3.x	Node.js 20 LTS
Framework UI	WPF (XAML)	Material 3 / Cupertino	—
Architecture	MVVM (CommunityToolkit)	Riverpod + Clean Arch	MVC (Express)
HTTP Client	HttpClient (.NET)	Dio 5.x	—
Cache local	SQLite (EF Core Lite)	Hive 2	—
Base de données	—	—	MySQL 8.0
ORM / Pilote DB	—	—	mysql2/promise
Auth	JWT (stocké SecureStorage)	JWT (flutter_secure_storage)	jsonwebtoken + bcryptjs
Graphiques	LiveCharts2	fl_chart	—
Export PDF	iTextSharp	pdf (dart)	pdfkit (Node)
Maps	—	google_maps_flutter	—
Validation	FluentValidation	flutter_form_builder	express-validator
Tests	xUnit + Moq	flutter_test + Mockito	Jest + Supertest

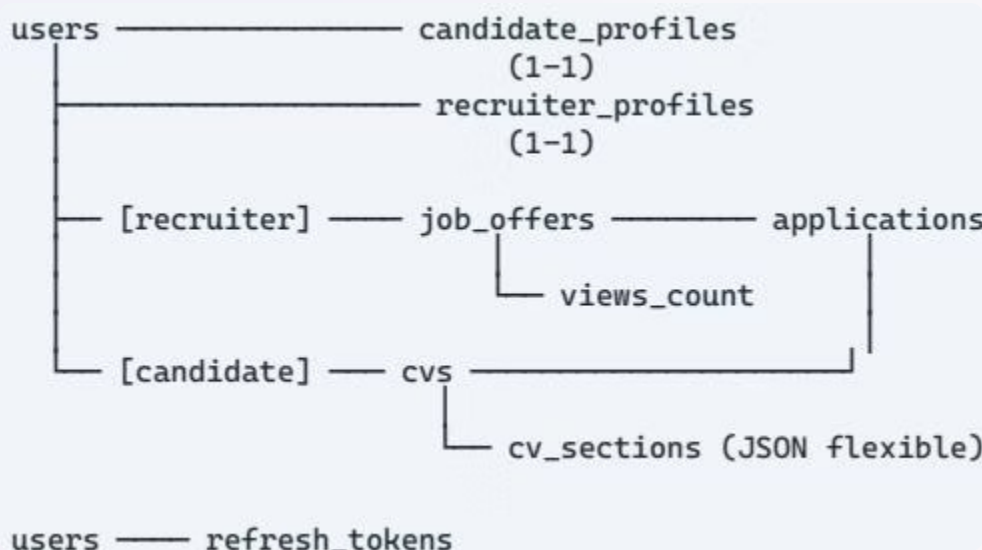
4.4 Flux d'authentification JWT



Le diagramme de séquence illustre le flux d'authentification JWT entre WPF/Flutter, le Node API et MySQL : login → récupération user + hash → retour { accessToken, refreshToken } → appel API protégé → renouvellement automatique via POST /auth/refresh → retour du nouvel accessToken.

5. Schéma Base de Données MySQL

5.1 Diagramme des entités



Le diagramme illustre les relations entre les entités de la base de données. Les tables users, candidate_profiles, recruiter_profiles, job_offers, applications, cvs et cv_sections sont connectées par des lignes. Les relations sont marquées par des multiplicités (1-1) entre users et les autres entités. La table refresh_tokens est également liée à users. La table cv_sections est décrite comme étant flexible au format JSON.

5.2 Script SQL complet

```
-- 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', 'admin') NOT NULL DEFAULT 'candidate',
  is_active BOOLEAN NOT NULL DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- TABLE : candidate_profiles
CREATE TABLE candidate_profiles (
  user_id INT UNSIGNED PRIMARY KEY,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  phone VARCHAR(30),
  title VARCHAR(200),
  location VARCHAR(150),
  bio TEXT,
  photo_url VARCHAR(500),
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- TABLE : recruiter_profiles
CREATE TABLE recruiter_profiles (
  user_id INT UNSIGNED PRIMARY KEY,
  company_name VARCHAR(200) NOT NULL,
  logo_url VARCHAR(500),
  description TEXT,
  website VARCHAR(300),
  sector VARCHAR(100),
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- 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 NOT NULL,
  contract_type ENUM('CDI','CDD','Freelance','Stage','Alternance') NOT NULL,
  location VARCHAR(150),
  latitude DECIMAL(10,8),
  longitude DECIMAL(11,8),
  salary_min DECIMAL(10,2),
  salary_max DECIMAL(10,2),
  category VARCHAR(100),
  skills JSON,
  status ENUM('draft','published','expired','archived') DEFAULT 'draft',
  views_count INT UNSIGNED DEFAULT 0,
  expires_at DATE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (recruiter_id) REFERENCES users(id) ON DELETE CASCADE,
  FULLTEXT INDEX ft_jobs (title, description)
);

-- TABLE : cvs
CREATE TABLE cvs (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  candidate_id INT UNSIGNED NOT NULL,
  title VARCHAR(255) NOT NULL,
  template_id VARCHAR(50) DEFAULT 'classic',
  color_theme VARCHAR(20) DEFAULT '#2563EB',
  is_public BOOLEAN DEFAULT FALSE,
  public_token VARCHAR(36) UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (candidate_id) REFERENCES users(id) ON DELETE CASCADE
);

-- TABLE : cv_sections (contenu JSON flexible)
CREATE TABLE cv_sections (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  cv_id INT UNSIGNED NOT NULL,
  section_type ENUM('summary','experience','education','skills','languages','certifications','projects','interests') NOT NULL,
  display_order TINYINT UNSIGNED DEFAULT 0,
  is_visible BOOLEAN DEFAULT TRUE,
  content JSON NOT NULL,
  FOREIGN KEY (cv_id) REFERENCES cvs(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,
  cv_id INT UNSIGNED NOT NULL,
  cover_letter TEXT,
  status ENUM('sent','viewed','reviewing','interview','accepted','rejected') DEFAULT 'sent',
  recruiter_notes TEXT,
  interview_date DATETIME,
  applied_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  UNIQUE KEY uq_application (candidate_id, job_offer_id),
  FOREIGN KEY (candidate_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (job_offer_id) REFERENCES job_offers(id) ON DELETE CASCADE,
  FOREIGN KEY (cv_id) REFERENCES cvs(id)
);

-- TABLE : refresh_tokens
CREATE TABLE refresh_tokens (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id INT UNSIGNED NOT NULL,
  token VARCHAR(512) NOT NULL UNIQUE,
  expires_at TIMESTAMP NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- INDEX de performance
CREATE INDEX idx_jobs_status ON job_offers(status);
CREATE INDEX idx_jobs_recruiter ON job_offers(recruiter_id);
CREATE INDEX idx_jobs_location ON job_offers(location);
CREATE INDEX idx_apps_candidate ON applications(candidate_id);
CREATE INDEX idx_apps_job ON applications(job_offer_id);
CREATE INDEX idx_apps_status ON applications(status);
CREATE INDEX idx_cvs_candidate ON cvs(candidate_id);
```


6. Endpoints API REST

🔖 **Base URL** : `https://api.jobmajunga.local/api/v1` — Format JSON exclusivement (Content-Type: application/json) — Auth : `Authorization: Bearer` sur toutes les routes protégées — Versioning : chemin `/api/v1/`, `/api/v2/...` — Pagination : `?page=1&limit=20` — Erreur type : `{ "error": "message", "code": "ERR_CODE" }`

6.2 Auth

Méthod e	Route	Accès	Description
POST	/auth/register	Public	Inscription — crée user + profil
POST	/auth/login	Public	Connexion — retourne access + refresh token
POST	/auth/refresh	Public	Renouveler le JWT avec le refresh token
POST	/auth/logout	Authentifié	Invalider le refresh token
POST	/auth/forgot-password	Public	Envoyer email de réinitialisation
POST	/auth/reset-password	Public	Réinitialiser le mot de passe

6.3 Profils

Méthod e	Route	Accès	Description
GET	/profile/me	Authentifié	Obtenir son propre profil
PUT	/profile/me	Authentifié	Mettre à jour son profil
POST	/profile/photo	Authentifié	Upload photo de profil (multipart)

6.4 Offres d'emploi

Méthod e	Route	Accès	Description
GET	/jobs	Public	Liste paginée + filtres (search, type, location, salary)
GET	/jobs/:id	Public	Détail d'une offre + incrément views_count
GET	/jobs/nearby	Public	Offres proches (?lat=&lng=&radius=km)
POST	/jobs	Recruteur	Créer une offre
PUT	/jobs/:id	Recruteur	Modifier une offre
PATCH	/jobs/:id/status	Recruteur	Changer le statut de l'offre
POST	/jobs/:id/duplicate	Recruteur	Dupliquer une offre
DELET E	/jobs/:id	Recruteur	Supprimer une offre
GET	/jobs/mine	Recruteur	Mes offres publiées (avec stats)

6.5 CV

Méthod e	Route	Accès	Description
GET	/cv	Candidat	Lister mes CV
POST	/cv	Candidat	Créer un CV (avec sections JSON)
GET	/cv/:id	Candidat	Obtenir un CV avec ses sections
PUT	/cv/:id	Candidat	Mettre à jour un CV
DELET E	/cv/:id	Candidat	Supprimer un CV
POST	/cv/:id/duplicate	Candidat	Dupliquer un CV
GET	/cv/:id/export/pdf	Candidat	Exporter en PDF (binaire)
PATCH	/cv/:id/share	Candidat	Activer / désactiver partage public
GET	/cv/public/:token	Public	Voir un CV partagé publiquement

6.6 Candidatures

Méthod e	Route	Accès	Description
POST	/applications	Candidat	Postuler (cvId + jobOfferId + coverLetter)
GET	/applications/mine	Candidat	Mes candidatures + statuts
DELET E	/applications/:id	Candidat	Retirer une candidature
GET	/applications/received	Recruteur	Candidatures reçues (par offre)
GET	/applications/:id/cv	Recruteur	Consulter le CV du candidat
PATCH	/applications/:id/status	Recruteur	Changer le statut
PATCH	/applications/:id/notes	Recruteur	Ajouter notes internes
PATCH	/applications/:id/interview	Recruteur	Planifier un entretien
GET	/applications/export	Recruteur	Export CSV/Excel candidats

6.7 Administration

Méthod e	Route	Accès	Description
GET	/admin/stats	Admin	KPIs globaux (users, offres, candidatures)
GET	/admin/users	Admin	Liste tous les utilisateurs
PATCH	/admin/users/:id	Admin	Activer / désactiver / changer rôle
DELET E	/admin/users/:id	Admin	Supprimer un compte
GET	/admin/jobs/moderation	Admin	Offres signalées à modérer
PATCH	/admin/jobs/:id/approve	Admin	Approuver une offre
DELET E	/admin/jobs/:id	Admin	Supprimer une offre (modération)
GET	/admin/logs	Admin	Logs d'activité

7. Exigences Non Fonctionnelles

7.1 Performance

- Temps de réponse API < 200 ms pour les requêtes simples
- Génération PDF < 3 secondes
- Démarrage de l'app WPF < 3 secondes (cold start)
- Démarrage de l'app Flutter < 2 secondes
- Pagination obligatoire : maximum 20 éléments par page
- Pool MySQL : 10 connexions simultanées minimum

7.4 Disponibilité & Fiabilité

- Disponibilité API cible : 99,5% hors maintenance planifiée
- Backup MySQL automatique quotidien
- Mode hors-ligne partiel pour WPF et Flutter (cache local)
- Polling WPF et Flutter toutes les 30 secondes pour les données critiques

7.2 Sécurité

- Hachage bcrypt avec facteur 12 sur tous les mots de passe
- JWT access token : durée 1h — refresh token : durée 7j stocké en base
- HTTPS obligatoire en production (TLS 1.2+)
- Rate limiting : 5 tentatives / 10 min sur les routes auth
- Rate limiting global : 200 req / 15 min par IP
- Validation et sanitisation de toutes les entrées (express-validator)
- Protection SQL injection via requêtes paramétrées (mysql2)
- Helmet.js : headers sécurisés (CSP, HSTS, X-Frame-Options)
- CORS restreint aux origines autorisées


7.3 Conformité RGPD

- Consentement explicite lors de l'inscription
- Droit à l'effacement : suppression cascade de toutes les données utilisateur
- Portabilité : export JSON de toutes ses données sur demande
- Durée de conservation : 3 ans après inactivité du compte
- Journalisation des accès aux données sensibles

8. Planning et Phasage — Méthode Agile

8.1 Découpage en sprints

Sprint	Durée	Backend (Node/MySQL)	WPF (C#)	Flutter
S0	1 sem	Setup projet, BDD schema.sql, CI/CD, env	Setup solution, architecture MVVM	Setup projet Flutter, Riverpod, Dio
S1	2 sem	Auth : register, login, refresh, logout	Écrans login/inscription, HttpClient JWT	Écrans login/inscription, Dio + SecureStorage
S2	2 sem	CRUD offres, filtres full-text, géoloc	Interface offres recruteur, liste + formulaire	Liste offres, filtres, carte Google Maps
S3	2 sem	CRUD CV + sections JSON, export PDF	Visionneuse CV dans l'appli recruteur	Éditeur CV mobile, multi-CV, sélection template
S4	2 sem	Endpoints candidatures, changement statut	Kanban candidatures, notes, planification	Postuler, tableau de bord candidatures
S5	2 sem	Admin : stats, modération, gestion users	Dashboard analytics, LiveCharts2, admin	Profil candidat, favoris, alertes polling
S6	2 sem	Tests Jest, optimisation requêtes, docs Swagger	Tests xUnit, export Excel/PDF, peaufinage UI	Tests Flutter, mode hors-ligne Hive, peaufinage
S7	1 sem	Déploiement serveur, variables prod	Package MSI, documentation utilisateur	Build APK/IPA, publication stores

 **Durée totale estimée : 15 semaines (~4 mois) · 3 équipes en parallèle (Backend / WPF / Flutter)**

8.2 Jalons clés



9. Risques, Contraintes & Livrables

9.1 Analyse des risques

Risque	Proba.	Impact	Plan de mitigation
Désynchronisation données WPF Flutter	Moyenn e	Haut	Polling 30s + invalidation cache sur mutation API
Performance Flutter sur vieux appareils	Moyenn e	Moyen	Lazy loading, pagination, profiling Flutter DevTools
Sécurité : vol de JWT	Basse	Haut	SecureStorage, refresh token rotation, durée courte (1h)
Dépassement du délai sprint CV/ éditeur	Haute	Moyen	Spike technique S0, découpage MVP puis améliorations
Compatibilité MySQL versions	Basse	Moyen	Docker MySQL 8 en dev, CI test multi-versions
Rejet app Flutter par les stores	Basse	Haut	Respect guidelines dès S0, tests sur devices réels
Non-conformité RGPD	Basse	Haut	Audit légal en S0, Privacy by Design, DPO impliqué

9.2 Contraintes techniques

- Langage desktop obligatoire : C# / .NET 8 avec WPF — pas d'alternative Electron ou web
- Application mobile obligatoire : Flutter / Dart — pas de React Native
- Backend obligatoire : Node.js + Express.js + MySQL — pas d'ORM (mysql2 natif)
- Pas de WebSocket / SignalR — communication exclusivement REST + polling
- Windows 10/11 64-bit requis pour le client WPF
- Android 9+ et iOS 14+ requis pour Flutter

9.3 Contraintes métier

- Un candidat ne peut pas postuler deux fois à la même offre
- Une offre expirée ou archivée ne peut plus recevoir de candidatures
- Le statut d'une candidature ne peut que progresser (pas de retour en arrière)
- Un CV supprimé ne peut pas être utilisé pour une nouvelle candidature
- Un recruteur ne peut consulter le CV d'un candidat que s'il a postulé à l'une de ses offres
- Les offres doivent être approuvées par un Admin avant publication (paramétrable)

10.1 Livrables techniques

- Code source C# WPF sur dépôt Git (solution Visual Studio)
- Code source Flutter sur dépôt Git (avec pubspec.yaml)
- Code source Node.js/Express/MySQL sur dépôt Git
- Script schema.sql + seed.sql de la base de données
- Application WPF compilée : setup MSI ou MSIX
- APK Android signé + IPA iOS (ou lien TestFlight)
- API REST déployée + documentation Swagger/OpenAPI auto-générée
- Pipeline CI/CD GitHub Actions pour les 3 projets

10.2 Livrables documentaires

- Présent cahier des charges (ce document)
- Documentation technique : diagrammes UML (classes, séquence, déploiement)
- Guide d'installation et de déploiement (Backend + WPF + Flutter)
- Manuel utilisateur Recruteur (WPF) et Candidat (Flutter)
- Documentation API Swagger
- Rapport de tests et couverture de code