# **📘 Cahier des charges – Application de gestion de projets (type Jira)**

## **1. Présentation du projet**

Créer une application web et mobile destinée à la gestion de projets, inspirée de Jira, permettant la planification, le suivi, la gestion des tâches et des équipes, avec une flexibilité sur les workflows agiles (Scrum, Kanban).

## **2. Objectifs du projet**

* Offrir une solution intuitive pour la gestion des projets en équipe.
* Permettre la personnalisation des workflows.
* Intégrer des fonctionnalités de collaboration en temps réel.
* Fournir un système de suivi des bugs, sprints et tâches.

## **3. Fonctionnalités principales**

### **A. Gestion des utilisateurs et des permissions**

* Authentification (email/mot de passe, SSO, OAuth2).
* Rôles (admin, chef de projet, membre, invité).
* Gestion des équipes et affectation des membres aux projets.

### **B. Gestion de projets**

* Création de projets.
* Configuration de workflows (Scrum, Kanban, etc.).
* Paramétrage des statuts, transitions, types de tâches.

### **C. Gestion des tâches (issues)**

* Création de tâches avec :  
  + Titre, description, statut, priorité, type (bug, tâche, story, epic).
  + Assignation à un utilisateur.
  + Deadlines, étiquettes, pièces jointes.
* Historique des modifications.
* Commentaires et mentions (@).

### **D. Boards visuels**

* Tableaux Kanban / Scrum personnalisables.
* Drag & drop des tâches entre colonnes.
* Filtres (par statut, personne, label, etc.).

### **E. Sprints et backlog**

* Planification de sprint.
* Estimations (points, heures).
* Visualisation du backlog.
* Burn-down chart, velocity chart.

### **F. Notifications**

* Notifications push/email/in-app.
* Système de rappel et de suivi de tâches.

### **G. Rapports et statistiques**

* Graphiques d’avancement (burn-down, burn-up, etc.).
* Reporting personnalisé par projet/utilisateur.

### **H. Intégrations**

* Intégration Git (GitHub, GitLab, Bitbucket).
* Intégration Slack, Teams, Google Calendar.
* API REST pour interagir avec l’application.

## **4. Technologies proposées**

### **A. Frontend**

* Framework : React / Vue.js
* Mobile : React Native / Flutter
* UI : TailwindCSS, Material UI

### **B. Backend**

* Langage : Node.js / Python (Django, FastAPI) / Java (Spring)
* API REST ou GraphQL

### **C. Base de données**

* PostgreSQL (relationnelle)
* Redis (cache, files d’attente)

### **D. Infrastructure**

* Hébergement : AWS / GCP / Azure / VPS
* CI/CD : GitHub Actions, GitLab CI
* Docker + Kubernetes pour le déploiement

## **5. Architecture logicielle**

* Architecture microservices ou modulaire MVC.
* Authentification centralisée.
* Séparation claire entre frontend, backend, API.

## **6. Sécurité**

* Authentification sécurisée (hashing bcrypt, JWT).
* Autorisation par rôle (RBAC).
* Gestion des erreurs, validation des entrées.

## **7. UX/UI**

* Interface intuitive, responsive, accessible.
* Mode sombre/clair.
* Navigation fluide entre les sections.

## **8. Phasage du développement**

| **Phase** | **Objectif** | **Durée estimée** |
| --- | --- | --- |
| Phase 1 | Authentification, UI basique | 2 semaines |
| Phase 2 | Gestion de projets & tâches | 3 semaines |
| Phase 3 | Boards Kanban/Scrum | 3 semaines |
| Phase 4 | Sprints, backlog, rapports | 3 semaines |
| Phase 5 | Notifications & intégrations | 2 semaines |
| Phase 6 | Optimisation, QA, déploiement | 2 semaines |

## **9. Livrables**

* Application web responsive.
* Application mobile (si prévue).
* Documentation technique et utilisateur.
* API publique (documentation Swagger/OpenAPI).

## **10. Maintenance & évolutions futures**

* Roadmap évolutive.
* Ajout de fonctionnalités (AI assistant, automation).
* Améliorations UX.