**Cahier des charges**

# Page de garde

**Nom projet :** Plateforme de gestion des candidatures aux concours

**Nom de code du projet** : CandidaturePlus

# Périmètre du projet

## But

Le but de ce projet est de développer une plateforme en ligne pour dématérialiser le processus de candidature aux concours organisés dans différents centres.

La plateforme doit simplifier et moderniser le processus pour les candidats et l'administration en intégrant les fonctionnalités suivantes :

* **Gestion des candidatures.**
* **Gestion des utilisateurs et des rôles.**
* **Notifications automatiques.**
* **Suivi et validation des candidatures.**
* **Reporting et génération de statistiques.**

## Missions

* **Développer une interface utilisateur ergonomique permettant une navigation intuitive pour les candidats et les gestionnaires.**
* **Mettre en place une API REST avec Spring Boot pour gérer les flux de données entre le front-end et la base de données.**
* **Intégrer un système sécurisé d'authentification et de gestion des rôles.**
* **Automatiser l'envoi de notifications par email pour informer les utilisateurs de l'état de leurs candidatures.**
* **Générer des statistiques et rapports pour les gestionnaires globaux et administrateurs.**

## Contraintes

- Ressources humaines : Une équipe constituée de deux développeur stagiaire sous supervision.

- Délais serrés : Le projet doit être finalisé en 6 semaines.

## Parties prenantes

- Candidats : Soumettent, suivent et consultent leurs candidatures.

- Gestionnaires locaux : Accès restreint à la gestion des candidatures dans leur centre.

- Gestionnaires globaux : Supervision globale des candidatures et des centres.

- Administrateurs : Gèrent les utilisateurs, les centres et les paramètres de la plateforme.

## Livrables d’entrée

- Spécifications fonctionnelles et techniques.

- Architecture technique.

- Maquettes initiales des interfaces utilisateur.

## Livrable de sortie

- Application web fonctionnelle.

- Documentation technique et utilisateur.

## Equipe de travail et rôles

- Stagiaires : Développement de l’application (front-end, back-end, base de données).

- Encadrant : Supervision, validation des sprints, et support technique.

# Etude de l’existant

# Etude fonctionnelle

## Objectifs fonctionnels

1. Permettre aux candidats de soumettre leurs candidatures via un formulaire en ligne.

2. Proposer un tableau de bord personnalisé selon le rôle de l’utilisateur.

3. Permettre aux gestionnaires de valider ou rejeter les candidatures.

4. Générer des rapports et statistiques pour les gestionnaires globaux et administrateurs.

## Besoins fonctionnels : Fonctionnalités (user stories)

### 2.1 Bloc fonctionnel : Gestion des utilisateurs

|  |  |  |
| --- | --- | --- |
| Règle de gestion | Description | Type |
| RG01 | L’authentification se fait via email/mot de passe pour les gestionnaires et administrateurs. | Obligatoire |
| RG02 | Les candidats accèdent à leur candidature via un numéro unique. | Obligatoire |
| RG03 | Les administrateurs gèrent les utilisateurs et leurs droits. | Obligatoire |

### 2.2 Bloc fonctionnel : Gestion des candidatures

|  |  |  |
| --- | --- | --- |
| Règle de gestion | Description | Type |
| RG01 | Les candidats remplissent un formulaire pour soumettre leur candidature. | Obligatoire |
| RG02 | Les gestionnaires locaux valident ou rejettent les candidatures. | Obligatoire |
| RG03 | Une notification est envoyée par email lors de chaque changement d’état. | Important |
| RG04 | Les candidatures sont filtrables par concours, centre, et état. | Important |

### 2.3 Bloc fonctionnel : Reporting et statistiques

|  |  |  |
| --- | --- | --- |
| Règle de gestion | Description | Type |
| RG01 | Les gestionnaires globaux accèdent à des statistiques agrégées. | Obligatoire |
| RG02 | Les rapports sont exportables au format CSV. | Important |

## Acteurs

|  |  |
| --- | --- |
| Acteur | Rôles |
| Candidat | * Soumettre une candidature. * Suivre une candidature. |
| Gestionnaire local | * Valider les candidatures. * Rejeter les candidatures. |
| Gestionnaire global | * Supervision et reporting global |
| Administrateur | * Gestion des utilisateurs et des paramètres. |

# Besoins non fonctionnels

**Performance**

* Temps de chargement des pages inférieur à 2 secondes
* Capacité à gérer 500 utilisateurs simultanés
* Disponibilité de 99,9% (hors maintenance planifiée)

**Sécurité**

* Authentification sécurisée avec double facteur optionnel
* Chiffrement des données sensibles en base de données
* Protection contre les injections SQL et XSS
* Journalisation des événements de sécurité

**Utilisabilité**

* Interface responsive pour tous appareils (desktop, tablette, mobile)
* Navigation intuitive et claire
* Temps d'apprentissage de l'interface inférieur à 10 minutes

**Fiabilité**

* Système de reprise après incident
* Sauvegarde quotidienne des données
* Plan de continuité en cas de panne serveur

# Stratégie de développement et plannification

## Méthodologie de développement

- Utilisation de la méthode \*\*Agile Scrum\*\*.

- Sprints hebdomadaires avec des livrables clairs.

1. Découpage en Lots de Travail\*\*

* Semaine 1 : Initialisation

- Conception et Maquetage

- Configuration des environnements de développement.

- Création de la base de données et des entités JPA.

* Semaine 2 : Back-end

- Développement des API REST pour la gestion des candidatures.

- Implémentation de l’authentification et des rôles.

* Semaine 3 : Front-end

- Création des interfaces principales (formulaire, tableau de bord).

- Intégration avec les API REST.

* Semaine 4 : Notifications et statistiques

- Mise en place des emails.

- Génération de statistiques et rapports.

* Semaine 5 : Tests et validation

- Tests unitaires et d’intégration.

- Validation de bout en bout avec Postman et utilisateurs tests.

* Semaine 6 : Livraison

- Documentation et déploiement sur un serveur local.

# Architecture du système

## Diagrammes UML

## Technologies utilisées

Front-end : React

React : Librairie front-end pour construire une interface utilisateur dynamique et réactive.

React Router : Gestion de la navigation et des routes.

Axios : Pour consommer les API REST.

Formik + Yup : Gestion des formulaires et validation.

Material-UI / Ant Design : Librairie de composants pour un design moderne.

Context API : Gestion de l'état global si nécessaire.

Back-end : Spring Boot

Spring Boot : Framework pour développer rapidement des applications robustes.

Spring Data JPA : Gestion de la persistance et mapping objet-relationnel (ORM).

Spring Security : Gestion de l'authentification et de l'autorisation.

Spring Web (Spring MVC) : Exposition des services REST.

Lombok : Réduction du code boilerplate (getters, setters, etc.).

Java Mail API : Envoi d'emails pour les notifications.

OpenAPI : Documentation des API.

Base de données : MySQL

MySQL : Système de gestion de bases de données relationnelles (RDBMS).

Flyway : Gestion des migrations de base de données.

Autres

Postman : Test des API.

GitHub/GitLab : Suivi du code source et intégration continue.

## Architecture logicielle

- Front-end (React) ↔ API REST (Spring Boot) ↔ Base de données (MySQL).

# Annexe

- Documentation technique et utilisateur.

- Guide d’installation.