POUREUX

Cahier des charges

Développement d'une application web de gestion associative

Rédigé par : Valentin Sala
Date : Mai 2025

Table des matières

1. Présentation du projet
    2. Objectifs
    3. Périmètre fonctionnel
    4. Contraintes techniques
    5. Planning prévisionnel
    6. Démarche d'apprentissage
    7. Livrables
    8. Technologies
    9. Justification choix des technologies

1. Présentation du projet

Le projet **POUREUX** est une application web de gestion comptable et administrative destinée à une association. Elle a pour but de centraliser la gestion des **adhérents**, des **projets**, des **saisies comptables**, ainsi que des **reçus fiscaux**. Cette application est développée dans une optique de simplification des tâches quotidiennes pour les bénévoles de l'association.

Le projet est réalisé dans la logique d'**auto-formation**, dans le cadre d'un apprentissage progressif des technologies web modernes. Certains choix technologiques sont encore en cours d'exploration.

2. Objectifs

- Interface intuitive pour bénévoles.
- Gestion des adhérents, projets et opérations comptables.
- Génération et envoi de reçus fiscaux.
- Tableau de bord synthétique.
- Sauvegarde cloud automatisée.
- Accès distant sécurisé.

3. Périmètre fonctionnel

Modules principaux :

- M1 : Saisies comptables
- M2 : Adhérents
- M3 : Projets
- M4 : Référentiels
- M5 : Reçus fiscaux
- M6 : Tableau de bord

4. Choix techniques

- Frontend : Vue.js (JavaScript)
- Backend : Java avec Spring Boot + Data (JPA)
- Serveur : Apache Tomcat
- Sécurité : JWT
- BDD : PostgreSQL
- Sauvegarde : Amazon S3 (cloud)

5. Planning prévisionnel

Projet en sprints de 2 semaines

Sprint	**Objectif fonctionnel**	**Structuration des données**	**Actions principales**
Sprint 0	Préparation et conception	Modèle conceptuel global (diagramme entité-relation), choix technologie base de données	Analyse besoins, modélisation des données, setup environnement technique
Sprint 1	Authentification	Table Utilisateurs (id, email, mot de passe hashé, rôles)	Création table utilisateurs, gestion de la sécurité (Spring security + JWT).
Sprint 2	Connexion + API	Optimisation tables utilisateurs, sécurisation API	Implémentation API REST sécurisée, gestion sessions
Sprint 3	CRUD Adhérents	Table Adhérents (id, nom, prénom, statut, contact, etc)	Création table adhérents, API CRUD adhérents
Sprint 4	Saisies	Tables pour saisies diverses (ex : événements, actions associatives)	Conception tables saisies, API gestion saisies
Sprint 5	Projets	Tables Projets (id, nom, description, état, lien adhérents)	Modélisation projets, relations avec adhérents, API projets
Sprint 6	Reçus fiscaux	Tables Reçus fiscaux (id, montant, date, adhérent lié)	Gestion reçus fiscaux, génération et API associée
Sprint 7	Tableau de bord	Tables statistiques, rapports (agrégats, vues synthétiques)	Création vues, calculs statistiques, API tableau de bord
Sprint 8	Déploiement	Stabilisation base données, scripts migration, sauvegardes	Mise en production, sauvegarde, optimisation BD

6. Démarche d'apprentissage

Projet mené dans une logique d'autoformation : apprentissage progressif des technologies choisies, documentation continue, approche incrémentale inspirée de l'agilité.

7. Livrables

- Cahier des charges
- Spécifications fonctionnelles
- Diagrammes d'architecture
- Code source (GitHub)
- Documentation API
- Captures écran
- Exemples de reçus fiscaux

8. Annexe – Technologies

Technologie	Statut	Objectif
Vue.js	En cours	Interface utilisateur moderne et réactive
Java (Spring)	En cours	Backend structuré avec API REST
JPA / Hibernate	À découvrir	Persistance objet-relationnel avec Spring data
JWT	En cours	Sécurisation des échanges via token
Apache Tomcat	En cours	Déploiement de l'application Spring. Serveur intégré à Spring boot
Amazon S3	À explorer	Sauvegarde cloud des données critiques
Docker	À explorer	Conteneurisation de l'application qui contiendra le jar auto-exécutable (fat jar) de spring boot.

9. Justification choix des technologies

Besoin	Technologie choisie	Justification du choix
Interface web moderne	Vue.js	Léger, réactif, adapté aux applications SPA. Facile à intégrer avec une API REST.
Développement du backend	Spring Framework	Puissant, structuré, modulaire. Bien adapté aux API REST et très utilisé en entreprise.
Persistance des données	JPA / Hibernate	Abstraction efficace de la base de données relationnelle, intégré dans Spring data.
Sécurisation de l'API REST	JWT (JSON Web Token)	Solution stateless adaptée aux API REST. Facilement intégrable avec Spring Security.
Déploiement du backend	Apache Tomcat	Léger, rapide et intégré directement à Spring boot.
Sauvegarde de données	Amazon S3	Stockage cloud fiable, sécurisé, et facilement accessible depuis le backend Java.
Conteneurisation	Docker	Standard industriel pour l'isolation et la portabilité des applications. Simplifie le déploiement, la scalabilité et l'intégration continue.