**Chapitre 5 : Conception et Développement** Chapitre 5 : Conception et Développement

**5.1 Organisation et gestion du projet avec Scrum**

Le projet **“Système Numérique de Dépôt et Suivi des Candidatures aux Concours Gabonais”** a été développé en suivant la méthodologie **Scrum**, adaptée aux projets nécessitant flexibilité et réactivité face à l’évolution des besoins des utilisateurs. La durée totale du projet est de **3 mois**, structurée en plusieurs sprints de 2 à 4 semaines, permettant de livrer progressivement un système fonctionnel et testable.

**5.1.1 Rôles Scrum**

* **Product Owner (PO)** : FIAD Monde, responsable de la vision du produit, de la priorisation des fonctionnalités et de la validation des livrables.
* **Scrum Master** : Mr LAWSON, garant de l’application correcte de Scrum, facilitateur et résolution des obstacles.
* Une image contenant texte, capture d’écran, diagramme, Police

  Le contenu généré par l’IA peut être incorrect.**Équipe de développement** : moi-même, responsable de l’architecture, du développement frontend (React + TypeScript), backend (Node.js + Express) et base de données (MySQL), ainsi que de l’intégration future des API de paiement mobile.

*Figure 1 : Schéma illustrant les rôles Scrum et leurs interactions*

**5.2 Product Backlog et planification des sprints**

**5.2.1 Product Backlog**

Le Product Backlog a été constitué dès le début du projet et mis à jour régulièrement. Chaque élément représente une fonctionnalité à développer ou un correctif. Les fonctionnalités principales développées ou planifiées sont :

|  |  |  |
| --- | --- | --- |
| Fonctionnalité | Description | Priorité |
| Création compte candidat | Permet au candidat de créer un compte sécurisé | Élevé |
| Connexion candidat | Connexion via le NUPCAN pour accéder au suivi de candidature | Élevé |
| Parcours candidature complet | Choix concours → filière → création candidature → NUPCAN → dépôt dossier/dashboard | Élevé |
| Paiement statique | Validation du flux complet avant intégration Mobile Money | Élevé |
| Tableau de bord admin | Suivi des candidatures, documents et paiements | Élevé |
| Intégration future API MyPVIT | Paiement via Mobile Money | Élevé |
| Notifications | Alertes sur statut candidature et paiement via NUPCAN | Moyenne |

*Figure 2 : Product Backlog et priorisation des fonctionnalités*

**5.2.2 Planification des sprints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fonctionnalité | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 |
| Création compte candidat | X |  |  |  |
| Connexion candidat | X | X |  |  |
| Parcours candidature complet |  | X | X |  |
| Paiement statique |  |  | X |  |
| Tableau de bord admin |  |  | X | X |
| Notifications |  |  |  | X |
| Intégration API MyPVIT |  |  |  | X |

**5.3 Analyse des besoins et exigences**

**5.3.1 Besoins fonctionnels**

Le système doit permettre :

* **Parcours candidat clair et uniforme** :
  1. Choix du concours.
  2. Sélection de la filière.
  3. Création de la candidature et génération du **NUPCAN**.
  4. Orientation : dépôt immédiat du dossier ou accès au dashboard.
  5. Téléversement des documents (CNI, diplômes, attestations).
  6. Validation des documents par les administrateurs.
  7. Paiement statique pour valider le flux complet.
  8. Suivi en temps réel via le dashboard.
* **Gestion administrative** : suivi des candidatures, validation/rejet des documents et paiements.
* **Notifications** : alertes et messages envoyés aux candidats via le NUPCAN.

Une image contenant texte, capture d’écran, diagramme, Police

Le contenu généré par l’IA peut être incorrect.

*(Figure 4 : Schéma du parcours candidat étape par étape)*

**5.3.2 Besoins non fonctionnels**

* **Sécurité** : protection des données et authentification.
* **Performance** : traitement rapide des candidatures et paiements simultanés.
* **Accessibilité** : utilisation web et responsive mobile.
* **Évolutivité** : intégration future de Mobile Money (Airtel Money, Moov Money).
* **Traçabilité** : historique complet des actions pour candidats et administrateurs.

**5.4 Conception du système**

**5.4.1 Diagramme de cas d’utilisation**

**Acteurs principaux** :

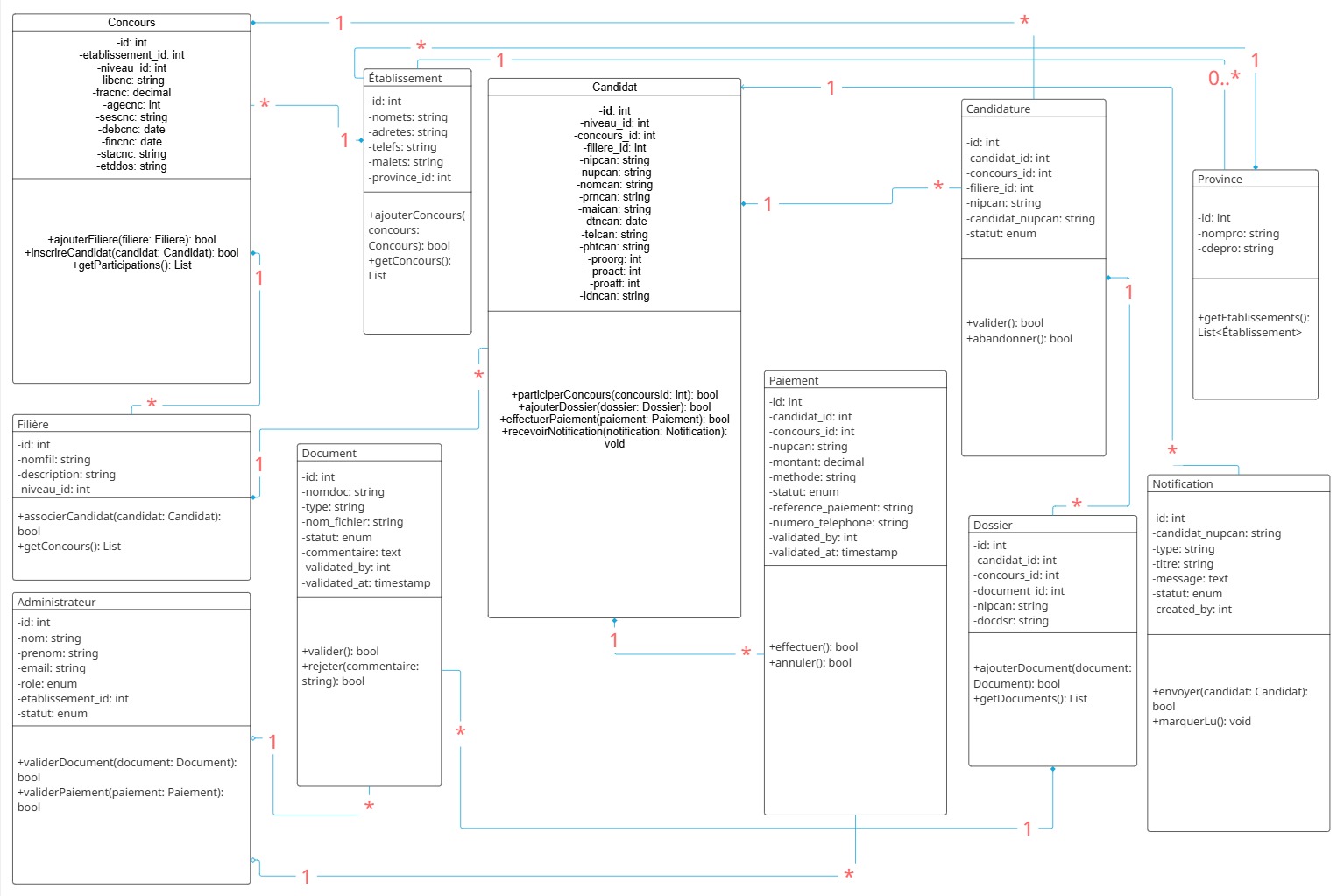
* **Candidat** : choix concours/filière, création candidature, dépôt documents, suivi, paiement statique.
* **Administrateur** : validation des candidatures
* **Super administrateur** : supervision de l’ensemble des concours et utilisateurs.

*(Figure 5 : Diagramme de cas d’utilisation)*

**5.4.2 Diagramme de classes**

Classes principales :

* **Candidat** (id, nom, email, téléphone, …)
* **Concours** (id, nom, établissement, niveau, frais)
* **Filière** (id, nom, concours\_id, places)
* **Candidature** (id, candidat\_id, concours\_id, filiere\_id, nupcan, statut)
* **Dossier** (id, candidature\_id, statut)
* **Document** (id, dossier\_id, type, chemin, statut)
* **Paiement** (id, candidature\_id, montant, statut)
* **Notification** (id, candidat\_nupcan, message, statut)



*(Figure 6 : Diagramme de classes UML)*

**5.4.3 MCD**

Relations principales :

* **Candidature** lie **Candidat → Concours + Filière** et possède **NUPCAN unique**.
* **Dossier** est associé à **une Candidature**.
* **Document** appartient à **un Dossier**.
* **Paiement** est lié à **une Candidature**.
* **Notification** est envoyée via **NUPCAN**.

*Figure 7 : MCD*

**5.4.4 Schéma relationnel MySQL**

Tables principales :

candidats, concours, filieres, candidatures, dossiers, documents, paiements, notifications.

*(Figure 8 : Schéma relationnel MySQL)*

**5.5 Architecture de l’application**

* **Frontend** : React + TypeScript, composants interactifs, formulaires, dashboard candidats et admins.
* **Backend** : Node.js + Express, API REST sécurisée, logique métier.
* **Base de données** : MySQL, stockage des candidats, concours, filières, documents, paiements et notifications.

*Figure 9 : Architecture 3 couches*

**5.6 Développement du module de candidature**

Parcours interactif et progressif :

|  |  |  |
| --- | --- | --- |
| Étape | Action | Résultat attendu |
| 1 | Choix du concours | Affichage et sélection enregistrée |
| 2 | Choix de la filière | Liste disponible et choix validé |
| 3 | Création candidature | **NUPCAN généré** |
| 4 | Orientation | Dépôt dossier immédiat ou accès dashboard |
| 5 | Dépôt dossier | Téléversement documents requis |
| 6 | Validation documents | Administrateurs valident ou rejettent |
| 7 | Paiement statique | Paiement enregistré et validé |
| 8 | Suivi dashboard | Visualisation temps réel |

*Figure 10 : Cycle détaillé d’une candidature)*

**5.7 Modules**

**5.7.1 Suivi candidature**

* Candidat : statut documents, paiements, état global.
* Administrateur : supervision et validation.

*(Figure 11 : Interfaces candidat vs administrateur)*

**5.7.2 Paiement statique**

* Paiement simulé pour valider le parcours.
* Préparation à l’intégration Mobile Money.

*(Figure 12 : Flux paiement statique)*

**PARTIE 3 :**

**Résultats et Bilan du Projet**

*Cette dernière partie évalue les résultats obtenus en comparant la solution aux objectifs définis. Elle examine les points forts, les limites, et les perspectives d’amélioration pour une adoption à plus grande échelle. Enfin, elle conclut avec un bilan global de l’expérience et des enseignements tirés du projet.*

**5.1 Organisation et gestion du projet avec Scrum**

Le projet *Système Numérique de Dépôt et Suivi des Candidatures aux Concours Gabonais* a été développé en suivant la méthodologie **Scrum**, adaptée aux projets nécessitant flexibilité et réactivité face à l’évolution des besoins des utilisateurs.  
La durée totale du projet est de **trois mois**, structurée en **plusieurs sprints de deux à quatre semaines**, permettant de livrer progressivement un système fonctionnel et testable.

**5.1.1 Rôles Scrum**

* **Product Owner (PO)** : Oxylium, représentant du besoin utilisateur, responsable de la vision du produit, de la priorisation des fonctionnalités et de la validation des livrables.
* **Scrum Master** : M. Yacouba, garant de l’application correcte de Scrum, facilitateur et résolution des obstacles.
* **Équipe de développement** : Moi-même (M. Makosso), responsable de l’architecture globale, du développement **frontend (React + TypeScript)**, **backend (Node.js + Express)** et de la **base de données MySQL**, ainsi que de l’intégration future de l’API de paiement **MyPVIT**.

*(Figure 1 : Schéma illustrant les rôles Scrum et leurs interactions)*

**5.2 Product Backlog et planification des sprints**

**5.2.1 Product Backlog**

Le **Product Backlog** a été défini dès le début du projet et mis à jour à chaque sprint. Chaque élément correspond à une fonctionnalité ou un besoin prioritaire.

|  |  |  |
| --- | --- | --- |
| Fonctionnalité | Description | Priorité |
| Création compte candidat | Permet au candidat de créer un compte sécurisé | Élevé |
| Connexion candidat | Connexion via le NUPCAN pour accéder au suivi de candidature | Élevé |
| Parcours candidature complet | Choix concours → filière → création candidature → NUPCAN → dépôt dossier/dashboard | Élevé |
| Paiement statique | Validation du flux complet avant intégration Mobile Money | Élevé |
| Tableau de bord admin | Suivi des candidatures, documents et paiements | Élevé |
| Intégration API MyPVIT | Paiement via Mobile Money | Élevé |
| Notifications | Alertes sur statut candidature et paiement via NUPCAN | Moyenne |

*(Figure 2 : Product Backlog et priorisation des fonctionnalités)*

**5.2.2 Planification des sprints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fonctionnalité | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 |
| Création compte candidat | X |  |  |  |
| Connexion candidat | X | X |  |  |
| Parcours candidature complet |  | X | X |  |
| Paiement statique |  |  | X |  |
| Tableau de bord admin |  |  | X | X |
| Notifications |  |  |  | X |
| Intégration API MyPVIT |  |  |  | X |

**5.3 Analyse des besoins et exigences**

* **5.3.1 Besoins fonctionnels**

Le système doit permettre :

1. Choix du concours.
2. Sélection de la filière.
3. Création de la candidature et génération du NUPCAN.
4. Téléversement des documents requis.
5. Validation des documents par les administrateurs.
6. Paiement statique (simulation du flux).
7. Suivi temps réel du statut via dashboard.
8. Gestion et affectation d’administrateurs aux établissements.

*(Figure 4 : Schéma du parcours candidat étape par étape)*

**5.3.2 Besoins non fonctionnels**

* **Sécurité** : authentification JWT et cryptage des données sensibles.
* **Performance** : requêtes SQL optimisées et chargement asynchrone des pages.
* **Accessibilité** : responsive design (desktop et mobile).
* **Évolutivité** : intégration future des API Mobile Money.
* **Traçabilité** : historique complet des actions utilisateur.

**5.4 Conception du système**

**5.4.1 Diagramme de cas d’utilisation**

Acteurs principaux :

* **Candidat** : crée la candidature, téléverse les documents, suit son dossier.
* **Administrateur d’établissement** : valide les documents, gère les concours.
* **Super Administrateur** : gère les établissements, administrateurs et supervise le système.

*(Figure 5 : Diagramme de cas d’utilisation global du système)*

* **5.4.2 Diagramme de classes UML**

Classes principales :

* Candidat
* Concours
* Filière
* Candidature
* Dossier
* Document
* Paiement
* Notification
* Administrateur
* Établissement

*(Figure 6 : Diagramme de classes UML du système)*

* **5.4.3 MCD**
* Une **Candidature** relie un **Candidat** à un **Concours** et une **Filière**.
* Un **Dossier** contient plusieurs **Documents**.
* Un **Paiement** est associé à une **Candidature**.
* Une **Notification** est envoyée via le NUPCAN.

*(Figure 7 : MCD du système)*

* **5.4.4 Schéma relationnel MySQL**

Tables : candidats, concours, filieres, candidatures, dossiers, documents, paiements, notifications, admins, etablissements.

*(Figure 8 : Schéma relationnel MySQL)*

**5.5 Architecture de l’application**

* **Frontend** : React + TypeScript, formulaires dynamiques, tableaux de bord candidats et administrateurs.
* **Backend** : Node.js + Express.js, architecture REST, sécurisation JWT.
* **Base de données** : MySQL, gestion relationnelle des entités.

*(Figure 9 : Architecture 3 couches du système)*

**5.6 Développement du module de Candidature**

Ce module permet au candidat de suivre un parcours interactif :

| **Étape** | **Action** | **Résultat attendu** |
| --- | --- | --- |
| 1 | Choix du concours | Affichage et enregistrement du concours choisi |
| 2 | Choix de la filière | Sélection validée |
| 3 | Création de la candidature | Génération automatique du **NUPCAN** |
| 4 | Téléversement des documents | Stockage et validation automatique |
| 5 | Paiement (statique) | Confirmation du paiement simulé |
| 6 | Suivi via dashboard | Visualisation en temps réel |

*(Figure 10 : Diagramme de séquence du module Candidature – choix concours, création, dépôt, paiement)*  
*(Figures suivantes : Captures des interfaces par étape)*

**5.7 Développement des autres modules**

* **5.7.1 Module Gestion des Administrateurs**

Ce module permet au **Super Administrateur** de créer et d’affecter des administrateurs à des établissements.  
Lors de la création, le mot de passe est **généré automatiquement** et les identifiants sont envoyés à l’administrateur par email.  
L’administrateur peut ensuite se connecter à son tableau de bord pour gérer les concours de son établissement.

*(Figure 11 : Diagramme de séquence du module Gestion des Administrateurs)*  
*(Figures suivantes : Interfaces de création et d’affectation des administrateurs)*

* **5.7.2 Module Validation des Documents**

Ce module permet à l’**administrateur d’établissement** de consulter, valider ou rejeter les documents téléversés par les candidats.  
Chaque validation met à jour le statut dans la base de données, et le candidat est notifié du résultat.

*(Figure 12 : Diagramme de séquence du module Validation des Documents)*  
*(Figures suivantes : Interface de vérification et statut des documents)*

* **5.7.3 Module Suivi de Candidature**

Permet au candidat de consulter le statut de ses documents, paiements et candidatures.  
L’administrateur peut suivre l’avancement global des dossiers et générer des rapports.

*(Figure 13 : Interfaces de suivi – candidat vs administrateur)*

* **5.7.4 Module Paiement Statique**

Ce module simule le paiement afin de valider le flux complet de candidature avant l’intégration réelle de **MyPVIT**.  
L’objectif est de tester le processus d’enregistrement, de validation et de confirmation sans appel API externe.

*(Figure 14 : Flux de paiement statique et validation du parcours)*

Souhaites-tu que je te génère maintenant la **version Word complète** de ce chapitre 5 (avec formatage propre, titres, numérotation automatique des figures et zones pour les captures) ?  
Je peux te la produire en .docx prête à insérer dans ton rapport.