📋 CAHIER DES CHARGES COMPLET : “InfoSMS Burundi”

1. PROJET
   1.1 Nom du Projet
   InfoSMS Burundi - Plateforme d’Information Locale par SMS

1.2 Contexte
Au Burundi, l’accès à l’information fiable (emplois, prix marchés, opportunités) reste difficile malgré la pénétration mobile élevée. Les solutions existantes (réseaux sociaux) souffrent de désinformation, manque de ciblage et exigent une connexion internet coûteuse.

1.3 Objectifs
Principal : Fournir une information locale vérifiée et ciblée via SMS.
Secondaires

Faciliter l’accès à l’emploi

Réduire la fracture numérique

Générer des données locales fiables

1.4 Public Cible
Phase 1 (Lancement) :
Chercheurs d’emploi urbains (Bujumbura, Gitega)

Petites et moyennes entreprises locales

18-45 ans, toutes éducations

Phase 2 (Expansion) :
Population rurale

Commerçants

Institutions publiques

2. FONCTIONNALITÉS DÉTAILLÉES
   2.1 Pour les UTILISATEURS (Abonnés)
   2.1.1 Inscription/Abonnement
   text
   Code : INSCRIPTION
   → Envoyer “ABO” au 5555
   → Recevoir instructions paiement
   → Confirmer paiement par “OK”
   → Recevoir confirmation + aide

2.1.2 Profil Utilisateur (Basique)
Métier principal (max 3 mots)

Zone géographique (quartier/ville)

Niveau d'éducation (optionnel)

Langue préférée (Kirundi/Français)

2.1.3 Réception d'Informations
2-3 alertes emploi/jour (ciblées)

Format SMS optimisé (160 caractères max)

Heures fixes de diffusion (8h, 12h, 16h)

Option "stop" temporaire

2.1.4 Interactions
text
Commandes disponibles :

- STOP : suspend abonnement
- INFO : voir solde jours
- METIER [nom] : changer métier
- ZONE [nom] : changer zone
- AIDE : voir toutes commandes
  2.2 Pour les ANNONCEURS (Entreprises)
  2.2.1 Publication d'Offres
  Interface web simplifiée

Formulaire structuré :

Titre poste (30 caractères)

Description courte (100 caractères)

Salaire (fourchette)

Lieu

Contact (numéro)

Catégorie (prédéfinie)

2.2.2 Gestion des Offres
Tableau de bord des offres actives

Statistiques basiques (vues estimées)

Historique des publications

Renouvellement d'offre

2.2.3 Profil Entreprise
Nom entreprise

Secteur d'activité

Contacts

Historique de publications

2.3 Pour l'ADMINISTRATEUR
2.3.1 Gestion Contenu
Validation manuelle des offres (Phase 1)

Catégorisation automatique

Base de données des métiers/zones

Modération des contenus

2.3.2 Gestion Utilisateurs
Liste abonnés actifs/expirés

Historique des interactions

Désactivation manuelle

Export données basique

2.3.3 Monitoring Système
Logs des SMS envoyés/reçus

État des paiements

Alertes techniques

Backup automatique

3. EXIGENCES TECHNIQUES
   3.1 Architecture Globale
   text
   Utilisateurs → SMS → Opérateur Télécom → API SMS → Backend Java → Base Données
   ↑ ↓
   ←←←←←←←←←←←←←←← Réponse SMS ←←←←←←←←←←←←←←←←←←←←←←←←←←←
   3.2 Backend (Java Spring Boot)
   3.2.1 Contrôleurs Principaux
   java
   // 1. SMS Controller
   @Controller
   public class SmsController {
   @PostMapping("/api/sms/receive")
   public void receiveSms(@RequestBody SmsRequest request) {
   // Traitement SMS entrant
   }
   @PostMapping("/api/sms/send")
   public void sendSms(@RequestBody SmsToSend sms) {
   // Envoi SMS sortant
   }
   }

// 2. User Management
@Controller
public class UserController {
// Gestion abonnés
// Profils
// Abonnements
}

// 3. Job Management
@Controller
public class JobController {
// CRUD offres
// Matching
// Publication
}
3.2.2 Services
java
@Service
public class SubscriptionService {
// Gestion cycles abonnement
// Renouvellements
// Expirations
}

```
@Service
public class MatchingService {
// Matching offre/utilisateur
// Ciblage géographique
// Filtrage par métier
}

@Service
public class PaymentService {
// Intégration Mobile Money
// Vérification paiements
// Historique transactions
}
```

3.2.3 Entités Base de Données

java

```
@Entity
public class User {
private Long id;
private String phoneNumber;
private String profession;
private String zone;
private LocalDate subscriptionEnd;
private String language;
private String status; // ACTIVE, EXPIRED, PAUSED
}

@Entity
public class JobOffer {
private Long id;
private String title;
private String description;
private String salary;
private String location;
private String contact;
private String category;
private LocalDateTime publishedAt;
private boolean verified;
}

@Entity
public class SmsLog {
private Long id;
private String phoneNumber;
private String message;
private String direction; // IN/OUT
private LocalDateTime timestamp;
```

private String status;
}
3.3 Base de Données (MySQL)
3.3.1 Schéma Principal
sql
– Table Utilisateurs
CREATE TABLE users (
id BIGINT PRIMARY KEY AUTO_INCREMENT,
phone_number VARCHAR(20) UNIQUE NOT NULL,
profession VARCHAR(50),
zone VARCHAR(50),
subscription_start DATE,
subscription_end DATE,
language VARCHAR(10) DEFAULT 'kirundi',
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

– Table Offres
CREATE TABLE job_offers (
id BIGINT PRIMARY KEY AUTO_INCREMENT,
title VARCHAR(100) NOT NULL,
description TEXT,
salary VARCHAR(50),
location VARCHAR(100),
contact_phone VARCHAR(20),
category VARCHAR(50),
company_name VARCHAR(100),
verified BOOLEAN DEFAULT FALSE,
published_at TIMESTAMP,
expires_at TIMESTAMP
);

– Table Abonnements
CREATE TABLE subscriptions (
id BIGINT PRIMARY KEY AUTO_INCREMENT,
user_id BIGINT,
start_date DATE,
end_date DATE,
payment_amount DECIMAL(10,2),
payment_method VARCHAR(20),
status VARCHAR(20),
FOREIGN KEY (user_id) REFERENCES users(id)
);
3.4 API Externes
3.4.1 API SMS
Fournisseur : Orange Burundi API ou service local

Endpoints requis : Send SMS, Receive SMS, Check Balance

Format : JSON/REST

Budget mensuel estimé : 50,000 FBu pour 5,000 SMS

3.4.2 Mobile Money (Phase 2)
Option 1 : Lumicash API

Option 2 : Orange Money API

Phase 1 alternative : Validation manuelle

3.5 Interface d'Administration
3.5.1 Technologie
Framework : Spring Boot + Thymeleaf

Style : Bootstrap 5

Authentification : Spring Security basique

3.5.2 Pages Principales
Dashboard : Statistiques clés

Gestion Offres : Validation/Modération

Gestion Utilisateurs : Abonnés actifs/expirés

Envoi SMS : Interface manuelle

Rapports : Revenus, croissance

4. SPÉCIFICATIONS FONCTIONNELLES
   4.1 Flux d'Abonnement
   text
   Étape 1 : Utilisateur envoie "ABO" au 5555
   Étape 2 : Système répond "Envoyez 500 FBu au 77-XXX-XXX, puis OK"
   Étape 3 : Utilisateur paie et envoie "OK"
   Étape 4 : Admin valide paiement (manuel Phase 1)
   Étape 5 : Système confirme "Abonné 7 jours. Envoyez METIER [votremetier]"
   Étape 6 : Utilisateur définit son métier
   Étape 7 : Réception alertes commence
   4.2 Flux de Publication Offre
   text
   Étape 1 : Entreprise accède site web

Étape 2 : Remplit formulaire (sans compte Phase 1)
Étape 3 : Système notifie admin par email
Étape 4 : Admin vérifie et valide manuellement
Étape 5 : Offre ajoutée à la file d'envoi
Étape 6 : Matching automatique avec abonnés pertinents
Étape 7 : Envoi SMS aux abonnés ciblés
4.3 Matching Algorithm (Simple)
java
public List<User> findRelevantUsers(JobOffer offer) {
return userRepository.findAll()
.stream()
.filter(user ->
user.isActive() &&
(user.getProfession().contains(offer.getCategory()) ||
offer.getTitle().contains(user.getProfession())) &&
user.getZone().equals(offer.getLocation())
)
.limit(50) // Max 50 envois par offre
.collect(Collectors.toList());
}

5. EXIGENCES NON FONCTIONNELLES
5.1 Performance
Temps réponse API : < 2 secondes

SMS delivery : < 30 secondes

Concurrent users : 1000 utilisateurs simultanés

Database : Optimisé pour lectures fréquentes

5.2 Sécurité
Authentification admin : Mot de passe fort

Données utilisateurs : Chiffrement numéros

API keys : Stockage sécurisé

Backup : Quotidien automatique

5.3 Fiabilité
Uptime : 99% (sauf maintenance)

SMS delivery rate : > 95%

Recovery : Restauration < 1 heure

Monitoring : Logs détaillés

5.4 Utilisabilité
Interface admin : Simple, pas de formation nécessaire

Commandes SMS : Intuitives, aide intégrée

Documentation : Guide opérationnel complet

Support : Numéro WhatsApp pour assistance

6. PLAN DE DÉPLOIEMENT
   6.1 Phase 1 : MVP (Mois 1-2)
   Semaine 1-2 : Développement Core
   Backend Java Spring Boot

Base de données MySQL

Intégration API SMS basique

Interface admin minimale

Semaine 3-4 : Test & Validation
Test avec 10 amis

Correction bugs

Optimisation performance

Documentation technique

Semaine 5-8 : Lancement Beta
50 abonnés gratuits (test)

Validation modèle économique

Collecte feedback

Améliorations mineures

6.2 Phase 2 : Croissance (Mois 3-6)
Automatisation paiements Mobile Money

Amélioration matching algorithm

Analytics avancés

Support multi-opérateurs

6.3 Phase 3 : Maturation (Mois 7-12)
Expansion autres villes

Services additionnels (prix marchés)

API publique pour partenaires

Application mobile (optionnel)

7. STRUCTURE DES COÛTS
   7.1 Développement Initial (Mois 1)
   Poste Coût
   Hébergement serveur 20,000 FBu/mois
   API SMS (démarrage) 10,000 FBu
   Nom de domaine (.bi) 15,000 FBu/an
   Total initial 45,000 FBu
   7.2 Coûts Récurrents (à partir Mois 2)
   Poste Coût Mensuel
   Hébergement 20,000 FBu
   API SMS (5000 SMS) 50,000 FBu
   Maintenance 10,000 FBu
   Total mensuel 80,000 FBu
   7.3 Seuil de Rentabilité
   text
   Avec prix abonnement : 500 FBu/semaine
   → 200 abonnés = 400,000 FBu/mois
   → Coûts : 80,000 FBu
   → Profit : 320,000 FBu
   Seuil : 40 abonnés payants
8. DOCUMENTATION TECHNIQUE
   8.1 Installation Développement
   bash

1. Cloner le projet

git clone https://github.com/tonrepo/infosms-burundi.git

2. Configuration base de données

mysql -u root -p
CREATE DATABASE infosms;
CREATE USER ‘infosms’@‘localhost’ IDENTIFIED BY ‘password’;

3. Configuration application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/infosms
spring.datasource.username=infosms
spring.datasource.password=password

4. Lancer l’application

mvn spring-boot:run
8.2 Variables d’Environnement
properties

API SMS

SMS_API_KEY=ton_api_key
SMS_API_SECRET=ton_secret
SMS_SENDER_ID=INFOSMS

Mobile Money

MM_API_KEY=key_lumicash
MM_API_SECRET=secret_lumicash

Admin

ADMIN_USERNAME=admin
ADMIN_PASSWORD=Password123!
8.3 Structure des Packages
text
src/main/java/com/infosms/
├── config/ # Configurations
├── controller/ # Contrôleurs REST
├── service/ # Logique métier
├── repository/ # Accès données
├── model/ # Entités JPA
├── dto/ # Data Transfer Objects
└── scheduler/ # Tâches planifiées

9. SUPPORT ET MAINTENANCE
   9.1 Procédures Opérationnelles
   Backup : Quotidien automatique à 2h

Monitoring : Vérification SMS non délivrés 2x/jour

Support : Réponse sous 24h via WhatsApp

Mise à jour : Maintenance hebdomadaire dimanche 4h-6h

9.2 Checklist Lancement
API SMS fonctionnelle

Base de données peuplée (métiers, zones)

Interface admin testée

Processus paiement validé

10 offres d'emploi réelles

5 abonnés tests

Documentation opérationnelle complète

10. RISQUES ET MITIGATION
    10.1 Risques Techniques
    Risque Probabilité Impact Mitigation
    API SMS indisponible Moyenne Élevé Contrat avec 2 fournisseurs
    Base de données corrompue Faible Critique Backup quotidien + test restauration
    Attaque sécurité Moyenne Élevé Audit sécurité mensuel, mots de passe forts
    10.2 Risques Business
    Risque Probabilité Impact Mitigation
    Faible adoption Élevée Moyen Offre gratuite 1 mois, parrainage
    Concurrence Moyenne Faible Focus niche, service personnalisé
    Problèmes paiement Élevée Moyen Support téléphonique, processus simple
11. LIVRABLES
    11.1 Phase 1 (J+60)
    Application Spring Boot fonctionnelle

Interface admin complète

Intégration API SMS

Base de données peuplée

Documentation technique

Guide utilisateur SMS

11.2 Phase 2 (J+180)
Intégration Mobile Money automatique

Analytics dashboard

API publique

Application mobile basique

Document version : 1.0
Date : [Date de création]
Auteur : [Ton nom]
Statut : En révision

📝 POUR LE DÉVELOPPEUR :
Points d'Attention Techniques :
Optimisation SMS : 160 caractères max, formatage clair

Gestion erreurs API SMS : Retry logic, fallback manuel

Performance matching : Index sur colonnes fréquemment interrogées

Logging détaillé : Tous les SMS entrants/sortants loggés

Bonnes Pratiques à Implémenter :
Code commenté en français

Tests unitaires pour services critiques

Configuration externalisée (properties files)

Versioning Git avec commits descriptifs

Extensions Possibles (Phase 2+) :
Cache Redis pour performances

Microservices architecture

Notification push (si app mobile)

Machine learning pour meilleur matching