Guide de Déploiement - Générateur de CIU WhatsApp

Health Equity & Resilience Observatory (HERO), UBC

January 2026

Table des matières

1 Générateur de CIU WhatsApp - Guide de Déploiement

Guide complet pour l'équipe en République Démocratique du Congo

Un système de bot WhatsApp de qualité production pour générer des Codes Identificateurs Uniques (CIU) dans le contexte de la santé en RDC. Construit avec FastAPI, Twilio et SQLite.

1.1 Table des matières

2. Prérequis techniques
3. Enregistrement avec Meta (WhatsApp Business API)
4. Configuration du serveur
5. Installation de l'application
6. Configuration des variables d'environnement
7. Déploiement en production
8. Test et validation
9. Maintenance et surveillance
10. Dépannage
11. Support

1.2 Vue d'ensemble

1.2.1 Qu'est-ce que ce système fait ?

Ce bot WhatsApp conduit une conversation interactive pour collecter les informations des utilisateurs et générer un Code Identificateur Unique (CIU) déterministe et préservant la confidentialité. La même personne recevra toujours le même CIU lorsqu'elle fournit les mêmes informations.

1.2.2 Fonctionnalités principales

— ** Préservation de la confidentialité** : Utilise le hachage SHA-256 avec sel pour générer des codes anonymes mais déterministes
— ** Adapté au contexte de la RDC** : Gère les accents français, diverses orthographes de noms et données de zones de santé locales
— ** Flux interactif** : Flux de conversation naturel avec validation et gestion des erreurs
— ** Qualité production** : Journalisation structurée, persistance de base de données, gestion complète des erreurs
— ** Détection des doublons** : Empêche automatiquement les enregistrements en double
— ** Déploiement facile** : Fonctionne avec sandbox Twilio et ngrok pour POC, prêt pour la production

1.2.3 Flux de conversation

Le bot pose 5 questions en séquence. **Les questions actuelles sont des PLACEHOLDERS** – personnalisez-les pour votre cas d'usage !

Questions placeholder actuelles (les utilisateurs fournissent l'information complète) : 1. **En quelle année êtes-vous né(e) ?** → Le système extrait les 3 derniers chiffres 2. **Où est née votre mère ?** → Le système extrait les 4 premières lettres 3. **Quel est votre prénom ?** → Le système extrait les 3 premières lettres 4. **Quel jour du mois êtes-vous né(e) ?** → Le système utilise le numéro 5. **Quel est votre nom de famille ?** → Le système extrait les 4 premières lettres

1.2.4 Format du CIU

Le format CIU (basé sur les questions placeholder) : `AAA-MMMM-JJ-PNNN-LLLL-HHHHH`

— **AAA** : Les 3 derniers chiffres de l'année de naissance (Q1)
— **MMMM** : Les 4 premières lettres du lieu de naissance de la mère (Q2)
— **JJ** : Jour du mois (Q4)
— **PNNN** : Les 3 premières lettres du prénom (Q3)
— **LLLL** : Les 4 premières lettres du nom de famille (Q5)
— **HHHHH** : Hash de 5 caractères pour l'unicité

Exemple : `985-KINS-15-JEA-KABI-A3F9D`

1.3 Prérequis techniques

Avant de commencer le déploiement, assurez-vous d'avoir :

1.3.1 Matériel serveur recommandé

Pour une utilisation en production (100-1000 utilisateurs/jour) :

— **CPU** : 2 cœurs minimum (4 cœurs recommandés)
— **RAM** : 2 GB minimum (4 GB recommandés)
— **Stockage** : 20 GB minimum (SSD préféré)
— **Réseau** : Connexion stable avec IP statique

1.3.2 Logiciels requis

— **Système d'exploitation** : Ubuntu 22.04 LTS (recommandé) ou similaire
— **Python** : Version 3.12 ou supérieure
— **Base de données** : PostgreSQL 14+ (pour production) ou SQLite (pour test)
— **Serveur web** : Nginx (pour reverse proxy)
— **Certificat SSL** : Let's Encrypt (gratuit)

1.3.3 Comptes et services externes

— **Compte Meta Business** : Pour WhatsApp Business API
— **Compte Twilio** : Pour l'intégration WhatsApp (alternative si Meta direct n'est pas disponible)
— **Nom de domaine** : Un domaine pour votre serveur (ex: `whatsapp.votre-organisation.cd`)

1.3.4 Compétences techniques requises

La personne qui installe ce système devrait avoir :

— Connaissance de base de Linux en ligne de commande
— Expérience avec l'installation de logiciels sur serveur
—

1.4 Compréhension des concepts de base de réseau (DNS, ports, SSL)

1.5 Enregistrement avec Meta (WhatsApp Business API)

1.5.1 Option 1 : Utilisation de Twilio (Recommandé pour commencer)

Twilio offre une intégration simplifiée avec l'API WhatsApp Business.

1.5.1.1 Étape 1.1 : Créer un compte Twilio

1. Allez sur https://www.twilio.com/try-twilio
2. Cliquez sur **"Sign Up"** (S'inscrire)
3. Remplissez le formulaire :
   — Prénom et nom
   — Adresse e-mail professionnelle
   — Mot de passe fort
   — Numéro de téléphone pour vérification
4. Vérifiez votre adresse e-mail
5. Vérifiez votre numéro de téléphone (vous recevrez un SMS avec un code)

1.5.1.2 Étape 1.2 : Configuration initiale du compte Twilio

1. Connectez-vous à la Console Twilio
2. Complétez le questionnaire de bienvenue :
   — Sélectionnez **"Messaging"** comme produit principal
   — Sélectionnez **"With code"** pour la méthode de développement
   — Sélectionnez **"Python"** comme langage
3. Passez l'interface d'introduction

1.5.1.3 Étape 1.3 : Accéder au sandbox WhatsApp Pour les tests (Phase POC) :

1. Dans la Console Twilio, allez à **"Messaging"** → **"Try it out"** → **"Send a WhatsApp message"**
2. Vous verrez :
   — Un numéro de sandbox (ex : `+1 415 523 8886`)
   — Un code de jointure (ex : `join side-orbit`)
3. Sur votre téléphone WhatsApp :
   — Enregistrez le numéro du sandbox dans vos contacts
   — Envoyez le code de jointure à ce numéro
   — Attendez le message de confirmation

** Limitation du sandbox :** Le sandbox est excellent pour les tests, mais tous les utilisateurs doivent rejoindre le sandbox avant d'utiliser le bot. Pour la production, vous devez passer à un numéro WhatsApp Business vérifié.

1.5.1.4 Étape 1.4 : Obtenir un numéro WhatsApp Business (Production) Pour la production :

1. Dans la Console Twilio, allez à **"Messaging"** → **"WhatsApp senders"**

2. Cliquez sur **“Get WhatsApp Enabled”**
3. Choisissez votre option :
   — **Option A** : Utiliser un numéro Twilio existant
   — **Option B** : Acheter un nouveau numéro Twilio
4. Suivez le processus de vérification Meta :
   — Nom de votre entreprise
   — Site web de votre organisation
   — Description de votre cas d’usage
   — Documents d’enregistrement de l’entreprise (si requis)
5. Attendez l’approbation (généralement 1-3 jours ouvrables)

1.5.1.5 Étape 1.5 : Récupérer vos identifiants Twilio

1. Dans la Console Twilio, allez à **“Account”** → **“Dashboard”**
2. Notez ces valeurs **importantes** :
   — **Account SID** : Commence par `AC...` (ex : `AC1234567890abcdef1234567890abcd`)
   — **Auth Token** : Cliquez sur “Show” pour révéler (ex : `1234567890abcdef1234567890abcd`)
   — **WhatsApp Number** : Votre numéro sandbox ou business (ex : `+14155238886`)

** IMPORTANT :** Ces identifiants sont confidentiels. Ne les partagez jamais publiquement et ne les committez pas dans Git.

** PLACEHOLDER - À REMPLIR PAR L’ÉQUIPE CANADIENNE :**

```
TWILIO_ACCOUNT_SID="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
TWILIO_AUTH_TOKEN="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
TWILIO_WHATSAPP_NUMBER="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
```

1.5.2 Option 2 : Accès direct à l’API Meta (Avancé)

Si vous voulez vous connecter directement à Meta sans Twilio :

1.5.2.1 Étape 2.1 : Créer un compte Meta Business

1. Allez sur https://business.facebook.com/
2. Cliquez sur **“Create account”**
3. Fournissez les informations de votre organisation
4. Vérifiez votre entreprise avec les documents requis

1.5.2.2 Étape 2.2 : Configurer l’application WhatsApp Business

1. Dans le gestionnaire Meta Business, créez une nouvelle application
2. Ajoutez le produit **“WhatsApp”**
3. Configurez votre profil Business :
   — Nom de l’entreprise
   — Description
   — Catégorie (sélectionnez “Healthcare” ou similaire)
   — Photo de profil
4. Obtenez votre numéro de téléphone WhatsApp Business

1.5.2.3 Étape 2.3 : Obtenir les clés API

1. Dans les paramètres de l'application WhatsApp, récupérez :
   — **WhatsApp Business Account ID**
   — **Phone Number ID**
   — **Access Token** (permanent)
2. Configurez le webhook (nous le ferons plus tard)

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE (Si utilisation de Meta direct) :**

```
META_WHATSAPP_TOKEN="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
META_PHONE_NUMBER_ID="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
META_VERIFY_TOKEN="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
```

** Recommandation :** Pour débuter, nous recommandons l'Option 1 (Twilio) car elle est plus simple à configurer et à gérer.

1.6 Configuration du serveur

1.6.1 Étape 1 : Choisir votre hébergement

Vous avez plusieurs options pour héberger ce système :

1.6.1.1 Option A : Serveur cloud (Recommandé) Fournisseurs recommandés : - DigitalOcean : Simple et économique ($12-24/mois) - **AWS EC2** : Flexible et évolutif - **Google Cloud Platform** : Bons outils d'intégration - **Heroku** : Le plus simple mais plus cher

Pour DigitalOcean (recommandé pour débuter) :

1. Créez un compte sur digitalocean.com
2. Créez un "Droplet" (serveur virtuel) :
   — **Image** : Ubuntu 22.04 LTS
   — **Plan** : Basic (2 GB RAM / 2 vCPUs) - $18/mois
   — **Datacenter region** : Choisissez le plus proche (ex : Amsterdam ou Francfort pour la RDC)
   — **Authentication** : SSH Keys (plus sécurisé) ou Password
3. Attendez que le droplet soit créé (1-2 minutes)
4. Notez l'adresse IP publique de votre serveur

1.6.1.2 Option B : Serveur local/sur site Si vous avez votre propre infrastructure :

1. Installez Ubuntu Server 22.04 LTS sur votre machine
2. Assurez-vous que le serveur a :
   — Une connexion Internet stable
   — Une adresse IP publique statique ou un service DNS dynamique
   — Les ports 80 et 443 ouverts dans votre pare-feu
3. Configurez le routeur pour rediriger les ports 80 et 443 vers votre serveur

1.6.2 Étape 2 : Connexion SSH au serveur

Depuis votre ordinateur local :

```
# Remplacez YOUR_SERVER_IP par l'adresse IP réelle de votre serveur
ssh root@YOUR_SERVER_IP

# Si vous utilisez une clé SSH
ssh -i /chemin/vers/votre/cle.pem root@YOUR_SERVER_IP
```

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
ADRESSE_IP_SERVEUR="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
UTILISATEUR_SSH="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
```

1.6.3 Étape 3 : Mise à jour du système

Une fois connecté au serveur :

```
# Mise à jour de la liste des paquets
sudo apt update

# Mise à niveau des paquets installés
sudo apt upgrade -y

# Redémarrage si nécessaire
sudo reboot
```

Reconnectez-vous après le redémarrage.

1.6.4 Étape 4 : Installation de Python 3.12+

```
# Vérifier la version de Python installée
python3 --version

# Si la version est inférieure à 3.12, installer depuis deadsnakes PPA
sudo apt install software-properties-common -y
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install python3.12 python3.12-venv python3.12-dev -y

# Vérifier l'installation
python3.12 --version
```

1.6.5 Étape 5 : Installation des dépendances système

```
# Installer pip pour Python 3.12
sudo apt install python3-pip -y

# Installer Git
```

```
sudo apt install git -y

# Installer Nginx (serveur web)
sudo apt install nginx -y

# Installer PostgreSQL (base de données)
sudo apt install postgresql postgresql-contrib -y

# Installer certbot pour SSL (Let's Encrypt)
sudo apt install certbot python3-certbot-nginx -y
```

1.6.6 Étape 6 : Configuration de PostgreSQL

```
# Démarrer PostgreSQL
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Créer un utilisateur et une base de données pour l'application
sudo -u postgres psql << EOF
CREATE USER whatsapp_bot WITH PASSWORD 'VOTRE_MOT_DE_PASSE_FORT';
CREATE DATABASE whatsapp_uic_db OWNER whatsapp_bot;
GRANT ALL PRIVILEGES ON DATABASE whatsapp_uic_db TO whatsapp_bot;
\q
EOF
```

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
DB_PASSWORD="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE - Mot de passe fort pour PostgreSQL]"
```

1.6.7 Étape 7 : Configuration du nom de domaine

Vous avez besoin d'un nom de domaine pour SSL (HTTPS).

Si vous avez déjà un domaine :

1. Allez dans les paramètres DNS de votre fournisseur de domaine
2. Créez un enregistrement A :
   - **Nom** : `whatsapp` (ou le sous-domaine de votre choix)
   - **Type** : A
   - **Valeur** : L'adresse IP de votre serveur
   - **TTL** : 3600 (ou valeur par défaut)
3. Attendez que les DNS se propagent (5 minutes à 24 heures)

Si vous n'avez pas de domaine :

Vous pouvez en acheter un chez : - Namecheap ($8-15/an) - Google Domains (~$12/an) - OVH (fournisseur européen) ** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
NOM_DE_DOMAINE="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE - ex: whatsapp.votre-org.cd]"
```

1.6.8 Étape 8 : Obtenir un certificat SSL

```
# Remplacez votre-domaine.cd par votre nom de domaine réel
sudo certbot --nginx -d votre-domaine.cd

# Suivez les instructions :
# 1. Entrez votre adresse e-mail
# 2. Acceptez les conditions d'utilisation
# 3. Choisissez si vous voulez partager votre e-mail avec EFF
# 4. Certbot configurera automatiquement Nginx avec SSL
```

Le certificat SSL sera automatiquement renouvelé par certbot.

1.7 Installation de l'application

1.7.1 Étape 1 : Créer un utilisateur pour l'application

Pour des raisons de sécurité, ne lancez pas l'application en tant que root.

```
# Créer un nouvel utilisateur
sudo adduser whatsappbot

# Ajouter l'utilisateur au groupe sudo (optionnel)
sudo usermod -aG sudo whatsappbot

# Passer à ce nouvel utilisateur
su - whatsappbot
```

1.7.2 Étape 2 : Cloner le dépôt depuis GitHub

```
# Créer un dossier pour les applications
mkdir -p ~/applications
cd ~/applications

# Cloner le dépôt
git clone https://github.com/drjforrest/whatsapp-uic-generator.git

# Entrer dans le dossier
cd whatsapp-uic-generator
```

1.7.3 Étape 3 : Créer l'environnement virtuel Python

```
# Créer l'environnement virtuel avec Python 3.12
python3.12 -m venv .venv

# Activer l'environnement virtuel
source .venv/bin/activate
```

```
# Vous devriez voir (.venv) au début de votre ligne de commande
```

1.7.4 Étape 4 : Installer les dépendances

```
# Mettre à jour pip
pip install --upgrade pip

# Installer l'application en mode éditable
pip install -e .

# Vérifier l'installation
python -c "import fastapi; print('FastAPI installé avec succès')"
```

1.7.5 Étape 5 : Générer le sel de sécurité (Salt)

Le sel est crucial pour la sécurité et l'unicité des CIU.

```
# Générer un sel sécurisé
python scripts/generate_salt.py
```

Vous verrez une sortie comme :

```
 Générateur de sel pour CIU
==================================================

Sel généré (longueur : 43 caractères) :

  xY9kL2mN4pQ6rS8tU0vW1xY3zA5bC7dE9fG1hI3jK5

 Ajoutez ceci à votre fichier .env :

  UIC_SALT="xY9kL2mN4pQ6rS8tU0vW1xY3zA5bC7dE9fG1hI3jK5"

  Gardez ceci secret et ne le committez jamais dans le contrôle de version !
```

Copiez le sel généré – vous en aurez besoin à l'étape suivante.

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
UIC_SALT="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE - Utilisez le script generate_salt.py]"
```

1.8 Configuration des variables d'environnement

1.8.1 Étape 1 : Créer le fichier .env

```
# Copier le fichier d'exemple
cp .env.example .env
```

```
# Éditer le fichier avec nano
nano .env
```

1.8.2 Étape 2 : Remplir les valeurs

Voici le fichier `.env` complet avec les placeholders :

```
# ============================================
# CONFIGURATION DE L'APPLICATION
# ============================================

APP_NAME="Générateur de CIU WhatsApp"
APP_VERSION="0.1.0"

# Environnement : production ou development
ENVIRONMENT=production

# Mode debug (mettre à False en production)
DEBUG=False

# ============================================
# SÉCURITÉ - CRITIQUE !
# ============================================

# Sel pour le hachage des CIU (minimum 16 caractères)
# Générez avec : python scripts/generate_salt.py
UIC_SALT="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"

# ============================================
# BASE DE DONNÉES
# ============================================

# Pour PostgreSQL en production
DATABASE_URL="postgresql://whatsapp_bot:[DB_PASSWORD]@localhost/whatsapp_uic_db"

# Remplacez [DB_PASSWORD] par le mot de passe PostgreSQL créé précédemment
# Exemple : postgresql://whatsapp_bot:MonMotDePasse123!@localhost/whatsapp_uic_db

# ============================================
# CONFIGURATION TWILIO
# ============================================

# Obtenez ces valeurs depuis : https://console.twilio.com/
TWILIO_ACCOUNT_SID="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"
TWILIO_AUTH_TOKEN="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"

# Numéro WhatsApp Twilio
```

```
# Pour sandbox : whatsapp:+14155238886
# Pour production : votre numéro vérifié
TWILIO_WHATSAPP_NUMBER="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]"

# ============================================
# CONFIGURATION WEBHOOK
# ============================================

# Chemin du webhook (ne pas modifier sauf si nécessaire)
WEBHOOK_PATH="/whatsapp/webhook"

# ============================================
# GESTION DES SESSIONS
# ============================================

# Durée d'expiration des sessions en minutes
SESSION_TIMEOUT_MINUTES=15

# ============================================
# JOURNALISATION (LOGGING)
# ============================================

# Niveau de log : DEBUG, INFO, WARNING, ERROR, CRITICAL
LOG_LEVEL=INFO

# Format JSON pour les logs (recommandé en production)
LOG_JSON=True
```

1.8.3 Étape 3 : Sauvegarder et fermer

Dans nano : - Appuyez sur `CTRL + X` - Appuyez sur `Y` pour confirmer - Appuyez sur `ENTRÉE` pour sauvegarder

1.8.4 Étape 4 : Vérifier la configuration

```
# Tester que la configuration se charge correctement
python -c "from app.config import settings; print(f'Configuration OK : {settings.app_name}')"
```

Vous devriez voir : `Configuration OK : Générateur de CIU WhatsApp`

1.8.5 Étape 5 : Sécuriser le fichier .env

```
# Définir les permissions appropriées
chmod 600 .env

# Vérifier
ls -la .env
```

```
# Devrait montrer : -rw------- (lisible/écrivable uniquement par le propriétaire)
```

1.9 Déploiement en production

1.9.1 Étape 1 : Initialiser la base de données

```
# S'assurer que l'environnement virtuel est activé
source .venv/bin/activate

# Exécuter le script d'initialisation
python scripts/init_db.py
```

Vous devriez voir :

```
 Tables de base de données créées avec succès !
Emplacement de la base de données PostgreSQL : configuré
```

1.9.2 Étape 2 : Tester localement

Avant de déployer en production, testons que tout fonctionne :

```
# Démarrer le serveur en mode test
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

Dans un autre terminal SSH :

```
# Tester l'endpoint de santé
curl http://localhost:8000/health
```

Si tout fonctionne, vous verrez :

```
{
  "status": "healthy",
  "service": "Générateur de CIU WhatsApp",
  "version": "0.1.0"
}
```

Arrêtez le serveur avec `CTRL + C`.

1.9.3 Étape 3 : Créer un service systemd

Pour que l'application démarre automatiquement :

```
# Créer le fichier de service (en tant que root ou avec sudo)
sudo nano /etc/systemd/system/whatsapp-uic.service
```

Copiez ce contenu :

```
[Unit]
Description=WhatsApp UIC Generator Service
After=network.target postgresql.service
```

```
[Service]
Type=simple
User=whatsappbot
Group=whatsappbot
WorkingDirectory=/home/whatsappbot/applications/whatsapp-uic-generator
Environment="PATH=/home/whatsappbot/applications/whatsapp-uic-generator/.venv/bin"
ExecStart=/home/whatsappbot/applications/whatsapp-uic-generator/.venv/bin/uvicorn app.main:app

Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Sauvegardez et fermez (`CTRL + X`, `Y`, `ENTRÉE`).

1.9.4 Étape 4 : Activer et démarrer le service

```
# Recharger systemd pour voir le nouveau service
sudo systemctl daemon-reload

# Activer le service au démarrage
sudo systemctl enable whatsapp-uic

# Démarrer le service
sudo systemctl start whatsapp-uic

# Vérifier le statut
sudo systemctl status whatsapp-uic
```

Vous devriez voir `Active: active (running)` en vert.

1.9.5 Étape 5 : Configurer Nginx comme reverse proxy

```
# Créer la configuration Nginx
sudo nano /etc/nginx/sites-available/whatsapp-uic
```

Copiez ce contenu (remplacez `votre-domaine.cd` par votre domaine réel) :

```
server {
    listen 80;
    server_name votre-domaine.cd;

    # Rediriger HTTP vers HTTPS
    return 301 https://$server_name$request_uri;
}
```

```
server {
    listen 443 ssl http2;
    server_name votre-domaine.cd;

    # Certificats SSL (générés par certbot)
    ssl_certificate /etc/letsencrypt/live/votre-domaine.cd/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/votre-domaine.cd/privkey.pem;

    # Paramètres SSL recommandés
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    # Logs
    access_log /var/log/nginx/whatsapp-uic-access.log;
    error_log /var/log/nginx/whatsapp-uic-error.log;

    # Reverse proxy vers FastAPI
    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Protection contre les attaques
    client_max_body_size 1M;
}```

Sauvegardez et fermez.

```bash
# Activer la configuration
sudo ln -s /etc/nginx/sites-available/whatsapp-uic /etc/nginx/sites-enabled/

# Tester la configuration Nginx
sudo nginx -t

# Si OK, recharger Nginx
sudo systemctl reload nginx
```
```

1.9.6 Étape 6 : Configurer le pare-feu

```
# Autoriser SSH (important !)
sudo ufw allow OpenSSH

# Autoriser HTTP et HTTPS
sudo ufw allow 'Nginx Full'

# Activer le pare-feu
sudo ufw enable

# Vérifier le statut
sudo ufw status
```

1.9.7 Étape 7 : Configurer le webhook Twilio

1. Connectez-vous à la Console Twilio
2. Allez à **"Messaging"** → **"WhatsApp senders"** ou **"Sandbox"**
3. Trouvez **"Webhook configuration"**
4. Pour **"When a message comes in"** :
   — URL : `https://votre-domaine.cd/whatsapp/webhook`
   — Méthode : **POST**
5. Cliquez sur **"Save"**

Twilio testera automatiquement le webhook. Vous devriez voir une coche verte .

1.9.8 Étape 8 : Vérifier les logs

```
# Voir les logs du service
sudo journalctl -u whatsapp-uic -f

# Voir les logs Nginx
sudo tail -f /var/log/nginx/whatsapp-uic-access.log
sudo tail -f /var/log/nginx/whatsapp-uic-error.log
```

Appuyez sur `CTRL + C` pour arrêter de suivre les logs.

1.10 Test et validation

1.10.1 Test 1 : Vérifier que le service fonctionne

```
# Vérifier l'endpoint de santé
curl https://votre-domaine.cd/health
```

Devrait retourner :

```
{
  "status": "healthy",
```

```
  "service": "Générateur de CIU WhatsApp",
  "version": "0.1.0"
}
```

1.10.2 Test 2 : Tester le webhook

```
# Envoyer une requête test au webhook
curl -X POST https://votre-domaine.cd/whatsapp/webhook \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "From=whatsapp:+243999999999&Body=Test"
```

Devrait retourner un message de bienvenue en XML.

1.10.3 Test 3 : Test WhatsApp complet

Sur votre téléphone WhatsApp :

1. **Rejoindre le sandbox** (si vous utilisez le sandbox) :
   — Envoyez le code de jointure au numéro Twilio
   — Attendez la confirmation
2. **Démarrer une conversation** :
   — Envoyez n'importe quel message
   — Vous devriez recevoir le message de bienvenue
3. **Compléter le flux** :
   — Répondez aux 5 questions
   — Vous devriez recevoir votre CIU
4. **Tester les commandes** :
   — Envoyez `RESTART` → Nouvelle conversation
   — Envoyez `HELP` → Message d'aide
5. **Tester la détection de doublons** :
   — Envoyez `RESTART`
   — Répondez avec exactement les mêmes informations
   — Vous devriez recevoir votre CIU existant avec un message indiquant qu'il a déjà été généré

1.10.4 Test 4 : Vérifier la base de données

```
# Se connecter à PostgreSQL
sudo -u postgres psql whatsapp_uic_db

# Voir tous les CIU générés
SELECT uic_code, phone_number, created_at FROM uic_records;

# Compter le total de CIU
SELECT COUNT(*) FROM uic_records;

# Voir les sessions actives
SELECT phone_number, current_step FROM conversation_sessions;
```

```
# Quitter
\q
```

1.11 Maintenance et surveillance

1.11.1 Surveillance quotidienne

1.11.1.1 Vérifier les logs

```
# Logs du service (dernières 100 lignes)
sudo journalctl -u whatsapp-uic -n 100

# Logs avec erreurs uniquement
sudo journalctl -u whatsapp-uic -p err

# Logs depuis aujourd'hui
sudo journalctl -u whatsapp-uic --since today
```

1.11.1.2 Vérifier l'utilisation des ressources

```
# Utilisation CPU et mémoire
htop

# Espace disque
df -h

# Vérifier l'état du service
sudo systemctl status whatsapp-uic
```

1.11.2 Maintenance hebdomadaire

1.11.2.1 Nettoyer les sessions expirées

```
# Appeler l'endpoint de nettoyage
curl -X POST https://votre-domaine.cd/whatsapp/cleanup
```

1.11.2.2 Sauvegarder la base de données

```
# Créer un dossier pour les sauvegardes
mkdir -p ~/backups

# Sauvegarder PostgreSQL
pg_dump -U whatsapp_bot -d whatsapp_uic_db > ~/backups/backup-$(date +%Y%m%d).sql

# Avec mot de passe
PGPASSWORD='[DB_PASSWORD]' pg_dump -U whatsapp_bot -h localhost whatsapp_uic_db > ~/backups/bac
```

```
# Compresser la sauvegarde
gzip ~/backups/backup-$(date +%Y%m%d).sql
```

1.11.2.3 Nettoyer les anciennes sauvegardes

```
# Garder seulement les 30 derniers jours
find ~/backups -name "backup-*.sql.gz" -mtime +30 -delete
```

1.11.3 Maintenance mensuelle

1.11.3.1 Mettre à jour le système

```
# Mettre à jour les paquets
sudo apt update
sudo apt upgrade -y

# Redémarrer si nécessaire
sudo reboot
```

1.11.3.2 Analyser les métriques d'utilisation

```
# Statistiques PostgreSQL
sudo -u postgres psql whatsapp_uic_db << EOF
-- Nombre total de CIU générés
SELECT COUNT(*) as total_uics FROM uic_records;

-- CIU par jour (derniers 30 jours)
SELECT DATE(created_at) as date, COUNT(*) as count
FROM uic_records
WHERE created_at >= NOW() - INTERVAL '30 days'
GROUP BY DATE(created_at)
ORDER BY date DESC;

-- Utilisateurs uniques (par numéro de téléphone)
SELECT COUNT(DISTINCT phone_number) as unique_users FROM uic_records;
EOF
```

1.11.4 Rotation des logs

Les logs peuvent devenir très volumineux. Configurez la rotation :

```
# Créer la configuration de rotation
sudo nano /etc/logrotate.d/whatsapp-uic
```

Ajoutez :

```
/var/log/nginx/whatsapp-uic-*.log {
    daily
```

```
    rotate 30
    compress
    delaycompress
    notifempty
    create 0640 www-data adm
    sharedscripts
    postrotate
        [ -f /var/run/nginx.pid ] && kill -USR1 `cat /var/run/nginx.pid`
    endscript
}
```

1.11.5 Automatiser les sauvegardes avec cron

```
# Éditer le crontab
crontab -e

# Ajouter cette ligne pour sauvegarder tous les jours à 2h du matin
0 2 * * * PGPASSWORD='[DB_PASSWORD]' pg_dump -U whatsapp_bot -h localhost whatsapp_uic_db | gzi
```

1.12 Dépannage

1.12.1 Problème : Le service ne démarre pas

Symptômes : - `sudo systemctl status whatsapp-uic` montre “failed” ou “inactive”

Solutions :

1. Vérifier les logs détaillés :

```
sudo journalctl -u whatsapp-uic -n 50
```

2. Vérifier la configuration .env :

```
python -c "from app.config import settings; print(settings.uic_salt)"
```

3. Vérifier que PostgreSQL fonctionne :

```
sudo systemctl status postgresql
```

4. Tester manuellement :

```
cd ~/applications/whatsapp-uic-generator
source .venv/bin/activate
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

1.12.2 Problème : Erreur de configuration du sel

Symptômes : - Message d'erreur : "Configuration error: UIC salt must be at least 16 characters"

Solutions :

1. Générer un nouveau sel :

```
python scripts/generate_salt.py
```

2. Copier le sel dans `.env` :

```
nano .env
# Mettre à jour UIC_SALT="le-nouveau-sel-généré"
```

3. Redémarrer le service :

```
sudo systemctl restart whatsapp-uic
```

1.12.3 Problème : Le webhook retourne 404

Symptômes : - Les messages WhatsApp ne reçoivent pas de réponse - Logs Twilio montrent erreur 404

Solutions :

1. Vérifier que Nginx fonctionne :

```
sudo systemctl status nginx
```

2. Vérifier la configuration Nginx :

```
sudo nginx -t
```

3. Vérifier l'URL du webhook dans Twilio :
   — Doit être : `https://votre-domaine.cd/whatsapp/webhook`
   — Doit utiliser POST
4. Tester directement :

```
curl -X POST https://votre-domaine.cd/whatsapp/webhook \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "From=whatsapp:+243999999999&Body=Test"
```

1.12.4 Problème : Sessions expirées trop rapidement

Symptômes : - Utilisateurs reçoivent fréquemment "Session expired"

Solutions :

1. Augmenter le délai d'expiration dans `.env` :

```
SESSION_TIMEOUT_MINUTES=30  # Au lieu de 15
```

2. Redémarrer le service :

```
sudo systemctl restart whatsapp-uic
```

1.12.5 Problème : Erreurs de base de données

Symptômes : - Erreurs mentionnant PostgreSQL dans les logs - Messages comme “connection refused” ou “authentication failed”

Solutions :

1. Vérifier que PostgreSQL fonctionne :

```
sudo systemctl status postgresql
```

2. Tester la connexion :

```
psql -U whatsapp_bot -d whatsapp_uic_db -h localhost
# Entrez le mot de passe quand demandé
```

3. Vérifier DATABASE_URL dans `.env` :

```
grep DATABASE_URL .env
```

4. Réinitialiser la base de données si nécessaire :

```
python scripts/init_db.py
```

1.12.6 Problème : Certificat SSL expiré“‘

Sauvegardez et fermez.

```
# Activer la configuration
sudo ln -s /etc/nginx/sites-available/whatsapp-uic /etc/nginx/sites-enabled/

# Tester la configuration Nginx
sudo nginx -t

# Si OK, recharger Nginx
sudo systemctl reload nginx
```

1.12.7 Étape 6 : Configurer le pare-feu

```
# Autoriser SSH (important !)
sudo ufw allow OpenSSH

# Autoriser HTTP et HTTPS
sudo ufw allow 'Nginx Full'

# Activer le pare-feu
sudo ufw enable
```

```
# Vérifier le statut
sudo ufw status
```

1.12.8 Étape 7 : Configurer le webhook Twilio

1. Connectez-vous à la Console Twilio
2. Allez à **"Messaging"** → **"WhatsApp senders"** ou **"Sandbox"**
3. Trouvez **"Webhook configuration"**
4. Pour **"When a message comes in"** :
   — URL : `https://votre-domaine.cd/whatsapp/webhook`
   — Méthode : **POST**
5. Cliquez sur **"Save"**

Twilio testera automatiquement le webhook. Vous devriez voir une coche verte .

1.12.9 Étape 8 : Vérifier les logs

```
# Voir les logs du service
sudo journalctl -u whatsapp-uic -f

# Voir les logs Nginx
sudo tail -f /var/log/nginx/whatsapp-uic-access.log
sudo tail -f /var/log/nginx/whatsapp-uic-error.log
```

Appuyez sur `CTRL + C` pour arrêter de suivre les logs.

1.13 Test et validation

1.13.1 Test 1 : Vérifier que le service fonctionne

```
# Vérifier l'endpoint de santé
curl https://votre-domaine.cd/health
```

Devrait retourner :

```
{
  "status": "healthy",
  "service": "Générateur de CIU WhatsApp",
  "version": "0.1.0"
}
```

1.13.2 Test 2 : Tester le webhook

```
# Envoyer une requête test au webhook
curl -X POST https://votre-domaine.cd/whatsapp/webhook \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "From=whatsapp:+243999999999&Body=Test"
```

Devrait retourner un message de bienvenue en XML.

1.13.3 Test 3 : Test WhatsApp complet

Sur votre téléphone WhatsApp :

1. **Rejoindre le sandbox** (si vous utilisez le sandbox) :
   — Envoyez le code de jointure au numéro Twilio
   — Attendez la confirmation
2. **Démarrer une conversation** :
   — Envoyez n'importe quel message
   — Vous devriez recevoir le message de bienvenue
3. **Compléter le flux** :
   — Répondez aux 5 questions
   — Vous devriez recevoir votre CIU
4. **Tester les commandes** :
   — Envoyez `RESTART` → Nouvelle conversation
   — Envoyez `HELP` → Message d'aide
5. **Tester la détection de doublons** :
   — Envoyez `RESTART`
   — Répondez avec exactement les mêmes informations
   — Vous devriez recevoir votre CIU existant avec un message indiquant qu'il a déjà été généré

1.13.4 Test 4 : Vérifier la base de données

```
# Se connecter à PostgreSQL
sudo -u postgres psql whatsapp_uic_db

# Voir tous les CIU générés
SELECT uic_code, phone_number, created_at FROM uic_records;

# Compter le total de CIU
SELECT COUNT(*) FROM uic_records;

# Voir les sessions actives
SELECT phone_number, current_step FROM conversation_sessions;

# Quitter
\q
```

1.14 Maintenance et surveillance

1.14.1 Surveillance quotidienne

1.14.1.1 Vérifier les logs

```
# Logs du service (dernières 100 lignes)
sudo journalctl -u whatsapp-uic -n 100

# Logs avec erreurs uniquement
sudo journalctl -u whatsapp-uic -p err

# Logs depuis aujourd'hui
sudo journalctl -u whatsapp-uic --since today
```

1.14.1.2 Vérifier l'utilisation des ressources

```
# Utilisation CPU et mémoire
htop

# Espace disque
df -h

# Vérifier l'état du service
sudo systemctl status whatsapp-uic
```

1.14.2 Maintenance hebdomadaire

1.14.2.1 Nettoyer les sessions expirées

```
# Appeler l'endpoint de nettoyage
curl -X POST https://votre-domaine.cd/whatsapp/cleanup
```

1.14.2.2 Sauvegarder la base de données

```
# Créer un dossier pour les sauvegardes
mkdir -p ~/backups

# Sauvegarder PostgreSQL
pg_dump -U whatsapp_bot -d whatsapp_uic_db > ~/backups/backup-$(date +%Y%m%d).sql

# Avec mot de passe
PGPASSWORD='[DB_PASSWORD]' pg_dump -U whatsapp_bot -h localhost whatsapp_uic_db > ~/backups/bac

# Compresser la sauvegarde
gzip ~/backups/backup-$(date +%Y%m%d).sql
```

1.14.2.3 Nettoyer les anciennes sauvegardes

```
# Garder seulement les 30 derniers jours
find ~/backups -name "backup-*.sql.gz" -mtime +30 -delete
```

1.14.3 Maintenance mensuelle

1.14.3.1 Mettre à jour le système

```
# Mettre à jour les paquets
sudo apt update
sudo apt upgrade -y

# Redémarrer si nécessaire
sudo reboot
```

1.14.3.2 Analyser les métriques d'utilisation

```
# Statistiques PostgreSQL
sudo -u postgres psql whatsapp_uic_db << EOF
-- Nombre total de CIU générés
SELECT COUNT(*) as total_uics FROM uic_records;

-- CIU par jour (derniers 30 jours)
SELECT DATE(created_at) as date, COUNT(*) as count
FROM uic_records
WHERE created_at >= NOW() - INTERVAL '30 days'
GROUP BY DATE(created_at)
ORDER BY date DESC;

-- Utilisateurs uniques (par numéro de téléphone)
SELECT COUNT(DISTINCT phone_number) as unique_users FROM uic_records;
EOF
```

1.14.4 Rotation des logs

Les logs peuvent devenir très volumineux. Configurez la rotation :

```
# Créer la configuration de rotation
sudo nano /etc/logrotate.d/whatsapp-uic
```

Ajoutez :

```
/var/log/nginx/whatsapp-uic-*.log {
    daily
    rotate 30
    compress
    delaycompress
    notifempty
    create 0640 www-data adm
    sharedscripts
    postrotate
        [ -f /var/run/nginx.pid ] && kill -USR1 `cat /var/run/nginx.pid`
    endscript
```

```
}
```

1.14.5 Automatiser les sauvegardes avec cron

```
# Éditer le crontab
crontab -e

# Ajouter cette ligne pour sauvegarder tous les jours à 2h du matin
0 2 * * * PGPASSWORD='[DB_PASSWORD]' pg_dump -U whatsapp_bot -h localhost whatsapp_uic_db | gzi
```

1.15 Dépannage

1.15.1 Problème : Le service ne démarre pas

Symptômes : - `sudo systemctl status whatsapp-uic` montre "failed" ou "inactive"

Solutions :

1. Vérifier les logs détaillés :

```
sudo journalctl -u whatsapp-uic -n 50
```

2. Vérifier la configuration .env :

```
python -c "from app.config import settings; print(settings.uic_salt)"
```

3. Vérifier que PostgreSQL fonctionne :

```
sudo systemctl status postgresql
```

4. Tester manuellement :

```
cd ~/applications/whatsapp-uic-generator
source .venv/bin/activate
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

1.15.2 Problème : Erreur de configuration du sel

Symptômes : - Message d'erreur : "Configuration error: UIC salt must be at least 16 characters"

Solutions :

1. Générer un nouveau sel :

```
python scripts/generate_salt.py
```

2. Copier le sel dans `.env` :

```
nano .env
# Mettre à jour UIC_SALT="le-nouveau-sel-généré"
```

3. Redémarrer le service :

```
sudo systemctl restart whatsapp-uic
```

1.15.3 Problème : Le webhook retourne 404

Symptômes : - Les messages WhatsApp ne reçoivent pas de réponse - Logs Twilio montrent erreur 404

Solutions :

1. Vérifier que Nginx fonctionne :

```
sudo systemctl status nginx
```

2. Vérifier la configuration Nginx :

```
sudo nginx -t
```

3. Vérifier l'URL du webhook dans Twilio :
   — Doit être : `https://votre-domaine.cd/whatsapp/webhook`
   — Doit utiliser POST
4. Tester directement :

```
curl -X POST https://votre-domaine.cd/whatsapp/webhook \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "From=whatsapp:+243999999999&Body=Test"
```

1.15.4 Problème : Sessions expirées trop rapidement

Symptômes : - Utilisateurs reçoivent fréquemment "Session expired"

Solutions :

1. Augmenter le délai d'expiration dans `.env` :

```
SESSION_TIMEOUT_MINUTES=30  # Au lieu de 15
```

2. Redémarrer le service :

```
sudo systemctl restart whatsapp-uic
```

1.15.5 Problème : Erreurs de base de données

Symptômes : - Erreurs mentionnant PostgreSQL dans les logs - Messages comme "connection refused" ou "authentication failed"

Solutions :

1. Vérifier que PostgreSQL fonctionne :

```
sudo systemctl status postgresql
```

2. Tester la connexion :

```
psql -U whatsapp_bot -d whatsapp_uic_db -h localhost
# Entrez le mot de passe quand demandé
```

3. Vérifier DATABASE_URL dans `.env` :

```
grep DATABASE_URL .env
```

4. Réinitialiser la base de données si nécessaire :

```
python scripts/init_db.py
```

1.15.6 Problème : Certificat SSL expiré

Symptômes : - Erreur SSL dans le navigateur - Twilio ne peut pas atteindre le webhook

Solutions :

1. Vérifier l'expiration :

```
sudo certbot certificates
```

2. Renouveler manuellement :

```
sudo certbot renew
```

3. Le renouvellement automatique devrait fonctionner, mais vérifier :

```
sudo systemctl status certbot.timer
```

1.15.7 Problème : Performances lentes

Symptômes : - Réponses WhatsApp lentes - Timeouts

Solutions :

1. Vérifier l'utilisation des ressources :

```
htop
```

2. Augmenter le nombre de workers dans systemd :

```
sudo nano /etc/systemd/system/whatsapp-uic.service
# Changer --workers 2 à --workers 4
sudo systemctl daemon-reload
sudo systemctl restart whatsapp-uic
```

3. Optimiser PostgreSQL (si nécessaire) :

```
sudo nano /etc/postgresql/14/main/postgresql.conf
# Ajuster shared_buffers, effective_cache_size selon votre RAM
sudo systemctl restart postgresql
```

1.15.8 Problème : Accents français mal affichés

Symptômes : - Les noms avec accents (é, è, ç, etc.) ne s'affichent pas correctement

Solutions :

1. C'est normalement géré automatiquement par l'application
2. Vérifier l'encodage de la base de données :

```
sudo -u postgres psql whatsapp_uic_db
\l whatsapp_uic_db
# Devrait montrer UTF8
```

3. Si problème persiste, recréer la base avec bon encodage :

```
sudo -u postgres psql << EOF
DROP DATABASE whatsapp_uic_db;
CREATE DATABASE whatsapp_uic_db OWNER whatsapp_bot ENCODING 'UTF8';
EOF
python scripts/init_db.py
```

1.16 Support

1.16.1 Contact pour questions techniques

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
Personne de contact technique (Canada) :
Nom : [À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]
Email : [À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]
Téléphone : [À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]
Fuseau horaire : [À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]

Heures de disponibilité :
[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE]

Méthode de contact préférée :
[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE - Email, WhatsApp, Slack, etc.]
```

1.16.2 Documentation supplémentaire

Dans ce dépôt, vous trouverez également :

— `README.md` : Documentation technique complète (en anglais)
— `SETUP.md` : Guide de configuration détaillé
— `CUSTOMIZING_QUESTIONS.md` : Comment personnaliser les questions
— `TWILIO_SANDBOX_SETUP.md` : Configuration détaillée du sandbox Twilio

1.16.3 Ressources externes

— **Documentation Twilio** : https://www.twilio.com/docs/whatsapp
— **Documentation FastAPI** : https://fastapi.tiangolo.com/
— **Documentation PostgreSQL** : https://www.postgresql.org/docs/
— **Guide Let's Encrypt** : https://letsencrypt.org/getting-started/

1.16.4 Signaler un problème

Pour signaler un bug ou demander une fonctionnalité :

1. Allez sur : https://github.com/drjforrest/whatsapp-uic-generator/issues
2. Cliquez sur "New Issue"
3. Décrivez le problème en détail :
   — Étapes pour reproduire
   — Comportement attendu
   — Comportement observé
   — Messages d'erreur (si applicable)
   — Captures d'écran (si utile)

1.17 Personnalisation (Optionnel)

1.17.1 Changer les questions

Les questions actuelles sont des **placeholders**. Pour les personnaliser :

1. Éditez le fichier `app/services/flow_manager.py`
2. Modifiez la liste `STEPS` avec vos questions
3. Consultez `CUSTOMIZING_QUESTIONS.md` pour les instructions détaillées

1.17.2 Ajouter le support multilingue

Pour ajouter le support du lingala ou d'autres langues :

1. Dans `app/services/flow_manager.py`, dupliquez les messages
2. Ajoutez une logique de détection de langue
3. Créez des variables pour chaque langue (ex: `WELCOME_MESSAGE_LN` pour lingala)

1.17.3 Modifier le format du CIU

Pour changer le format du CIU généré :

1. Éditez `app/services/uic_service.py`
2. Modifiez la fonction `generate_uic()`
3. Testez soigneusement avant le déploiement

1.18 Considérations de sécurité

1.18.1 Bonnes pratiques

À FAIRE :

- Garder le fichier `.env` secret et sécurisé
- Utiliser des mots de passe forts pour PostgreSQL
- Activer le pare-feu (UFW)
- Maintenir le système à jour
- Sauvegarder régulièrement la base de données
- Utiliser HTTPS/SSL pour toutes les communications
- Limiter l'accès SSH (désactiver root login)
- Monitorer les logs régulièrement

À NE PAS FAIRE :

- Ne jamais committer le fichier `.env` dans Git
- Ne jamais partager les identifiants Twilio publiquement
- Ne pas exécuter l'application en tant que root
- Ne pas désactiver le pare-feu
- Ne pas ignorer les mises à jour de sécurité
- Ne pas utiliser SQLite en production
- Ne pas exposer les ports inutiles

1.18.2 Sécuriser l'accès SSH

```
# Désactiver le login root
sudo nano /etc/ssh/sshd_config
# Changer : PermitRootLogin no

# Utiliser uniquement les clés SSH (plus sécurisé que mots de passe)
# Dans le même fichier :
# PasswordAuthentication no

# Redémarrer SSH
sudo systemctl restart sshd
```

1.18.3 Configurer fail2ban (Protection contre les attaques)

```
# Installer fail2ban
sudo apt install fail2ban -y

# Créer la configuration
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

# Activer pour SSH et Nginx
sudo nano /etc/fail2ban/jail.local
# Sous [sshd] et [nginx-http-auth], mettre : enabled = true
```

```
# Démarrer fail2ban
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

1.19 Métriques et surveillance avancée (Optionnel)

1.19.1 Installation de Prometheus et Grafana

Pour une surveillance avancée (optionnel mais recommandé pour la production) :

```
# Cette section est avancée et peut être mise en place plus tard
# Contactez l'équipe canadienne pour assistance si nécessaire
```

1.19.2 Alertes par email

Configurez des alertes pour être notifié en cas de problème :

```
# Installer mailutils
sudo apt install mailutils -y

# Créer un script d'alerte
cat > ~/alert.sh << 'EOF'
#!/bin/bash
if ! systemctl is-active --quiet whatsapp-uic; then
    echo "Le service WhatsApp UIC est arrêté !" | mail -s "Alerte Service" votre-email@example.
fi
EOF

chmod +x ~/alert.sh

# Ajouter au crontab (vérifier toutes les 5 minutes)
crontab -e
# Ajouter : */5 * * * * ~/alert.sh
```

** PLACEHOLDER - À REMPLIR PAR L'ÉQUIPE CANADIENNE :**

```
EMAIL_ALERTES="[À COMPLÉTER PAR L'ÉQUIPE CANADIENNE - Email pour recevoir les alertes]"
```

1.20 Liste de vérification finale

Avant de mettre en production, vérifiez que :

- ☐ Le serveur est configuré et accessible
- ☐ Python 3.12+ est installé
- ☐ PostgreSQL fonctionne correctement
- ☐ Le dépôt GitHub est cloné

☐ L'environnement virtuel Python est créé et activé
☐ Toutes les dépendances sont installées
☐ Le sel de sécurité (UIC_SALT) est généré
☐ Le fichier .env est configuré avec toutes les valeurs
☐ La base de données est initialisée
☐ Le service systemd est créé et activé
☐ Nginx est configuré comme reverse proxy
☐ Le certificat SSL est installé et valide
☐ Le pare-feu est activé avec les bons ports
☐ Le webhook Twilio est configuré
☐ Les tests WhatsApp fonctionnent
☐ Les sauvegardes automatiques sont configurées
☐ Les logs sont surveillés
☐ L'équipe a les contacts pour le support

1.21 Félicitations !

Vous avez maintenant un système WhatsApp UIC Generator entièrement fonctionnel et déployé en production !

1.21.1 Prochaines étapes suggérées

1. **Tester avec des utilisateurs réels** dans votre zone de santé
2. **Surveiller les logs** quotidiennement pendant la première semaine
3. **Collecter les retours** des utilisateurs
4. **Personnaliser les questions** selon vos besoins spécifiques
5. **Former votre équipe** sur l'utilisation et la maintenance
6. **Documenter vos procédures** locales spécifiques

1.21.2 Formation continue

Organisez des sessions de formation pour : - Les agents de santé qui utiliseront le système - Le personnel technique qui maintiendra le serveur - Les administrateurs qui géreront la base de données

---

1.22 Notes importantes

1.22.1 À propos de la confidentialité

Ce système est conçu pour **préserver la confidentialité** : - Aucune donnée d'identification personnelle n'est stockée en clair - Les CIU sont générés avec un hachage cryptographique sécurisé - Seul le CIU et le numéro de téléphone sont conservés - Le sel de sécurité doit être gardé absolument secret

1.22.2 Conformité légale

Assurez-vous de respecter : - Les lois sur la protection des données en RDC - Les régulations du Ministère de la Santé - Les politiques de votre organisation

1.22.3 Limitations connues

— Le sandbox Twilio nécessite que chaque utilisateur “rejoigne” le sandbox
— Pour la production complète, un numéro WhatsApp Business vérifié est nécessaire
— Les sessions expirent après 15 minutes d’inactivité (configurable)

1.23 Licence et crédits

Auteur : Jamie Forrest, PhD, MPH
Organisation : Health Equity & Resilience Observatory (HERO), UBC

Construit pour les agents de santé et chercheurs en République Démocratique du Congo travaillant sur des systèmes d’identification de patients préservant la confidentialité.

Document créé le : [DATE]
Version : 1.0
Dernière mise à jour : [DATE]

Pour toute question ou assistance, contactez l’équipe canadienne aux coordonnées fournies dans la section Support.

Bonne chance avec votre déploiement !