Générateur CIU WhatsApp - Guide de Déploiement

Guide Complet pour l'Équipe de la République Démocratique du Congo

Un système de bot WhatsApp de qualité production pour générer des Codes d'Identification Unique (CIU) dans le contexte des services de santé en RDC. Construit avec FastAPI et conçu pour le déploiement avec l'API Cloud WhatsApp de Meta.

Important : Ce système utilise **Twilio pour le développement/les tests** et **l'API Cloud de Meta pour la production**. Voir la section Environnements de Déploiement pour plus de détails.

Note Windows : Ce guide est conçu pour le déploiement Linux/Ubuntu. Pour le déploiement sur Windows Server, consultez README_FR_WINDOWS_ADDENDUM.md.

Table des Matières

Aperçu

Que fait ce système ?

Ce bot WhatsApp mène une conversation interactive pour collecter les informations de l'utilisateur et générer un Code d'Identification Unique (CIU) déterministe et préservant la confidentialité. La même personne recevra toujours le même CIU lorsqu'elle fournira les mêmes informations.

Pour Commencer

Consultez le fichier QUICKSTART.md pour les instructions de démarrage rapide.

Prérequis

- git
- Python 3.12+
- Compte Twilio (développement uniquement)
- ngrok installé
- Compte Meta Cloud

REMARQUE : Ce README.md est préparé pour un déploiement sur un serveur Linux. Pour un déploiement sur un serveur Windows Server, consultez le guide complémentaire dans FR/README_FR_WINDOWS_ADDENDUM.md.

Cloner le Dépôt

```
git clone https://github.com/drjforrest/whatsapp-uic-generator.git
cd whatsapp-uic-generator
```

1. Installer les Dépendances

```
python3 -m venv .venv
source .venv/bin/activate  # Sur Windows : .venv\Scripts\activate
pip install -e .
```

2. Générer le Sel de Sécurité

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

Copiez la sortie.

3. Configurer l'Environnement

REMARQUE : Cette configuration est UNIQUEMENT pour l'environnement de développement. Twilio vous permet de configurer le bot WhatsApp et d'avoir une application fonctionnelle sans avoir à vous inscrire auprès de Meta Cloud. Ce n'est pas destiné au déploiement en production. Voir les étapes ci-dessous pour la configuration de l'environnement de production.

```
cp .env.example .env
# Modifiez .env avec :
# - Le sel généré
# - Les identifiants Twilio (depuis console.twilio.com)
```

4. Initialiser la Base de Données

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

5. Démarrer les Services

Terminal 1 - FastAPI :

```
./scripts/run_dev.sh
# OU : uvicorn app.main:app --reload
```

Terminal 2 - ngrok :

```
ngrok http 8000
# Copiez l'URL HTTPS
```

Fonctionnalités Clés

- **Préservation de la confidentialité** : Utilise le hachage SHA-256 avec sel pour générer des codes anonymes mais déterministes
- **Adapté à la RDC** : Gère les accents français, diverses orthographes de noms et données des zones de santé locales
- **Flux interactif** : Flux de conversation naturel en français avec validation et gestion des erreurs
- **Qualité de production** : Journalisation structurée, persistance en base de données, gestion complète des erreurs
- **Détection des doublons** : Empêche automatiquement les inscriptions en double
- **Codes QR optionnels** : Génération de codes QR scannables pour un accès facile au CIU
- **Déploiement facile** : Fonctionne avec le bac à sable Twilio pour les tests, l'API Cloud Meta pour la production

Flux de Conversation

Le bot pose 5 questions en français pour générer le code CIU.

Formule du CIU : LLLFFFYCG (10 caractères)

Questions posées (en français) : 1. **Quelles sont les 3 premières lettres de votre nom de famille ?** → 3 premières lettres du nom de famille (LLL) 2. **Quelles sont les 3 premières lettres de votre prénom ?** → 3 premières lettres du prénom (FFF) 3. **Quel est le dernier chiffre de votre année de naissance ?** → Dernier chiffre de l'année de naissance (Y) 4. **Quel est le code de votre ville de naissance ?** → Code de ville à 2 lettres (CC) 5. **Quel est votre code de genre ?** → Code de genre : 1, 2, 3, ou 4 (G)

Format du CIU

Le format du CIU suit : `LLLFFFYCCG` (10 caractères au total)

- **LLL** : 3 premières lettres du code du nom de famille (par ex., MBE de Mbengue)

- **FFF** : 3 premières lettres du code du prénom (par ex., IBR de Ibrahima)
- **Y** : Dernier chiffre de l'année de naissance (par ex., 7 de 1997)
- **CC** : Code de ville à 2 lettres (par ex., DA pour Dakar)
- **G** : Code de genre - 1 chiffre
  - 1 = Homme
  - 2 = Femme
  - 3 = Trans
  - 4 = Autre

Exemple : `MBEIBR7DA1` - MBE : Code du nom de famille « Mbengue » - IBR : Code du prénom « Ibrahima » - 7 : Né dans une année se terminant par 7 (par ex., 1997) - DA : Code de ville « Dakar » - 1 : Code de genre « Homme »

Environnements de Déploiement

Ce système est conçu pour fonctionner dans deux environnements distincts :

Environnement de Développement/Test (Twilio)

Objectif : Configuration rapide pour les tests, la preuve de concept et le développement

Comment ça fonctionne : - Utilise le bac à sable WhatsApp de Twilio ou l'API Business - Twilio agit comme intermédiaire entre votre serveur et WhatsApp - Les webhooks pointent vers votre serveur - Twilio gère la livraison des messages

Avantages : - Configuration rapide (15 minutes) - Aucune vérification Meta Business requise initialement - Fonctionne avec ngrok pour les tests locaux - Mode bac à sable disponible immédiatement - Toutes les fonctionnalités fonctionnent, y compris les codes QR

Limitations : - Le bac à sable nécessite que les utilisateurs « rejoignent » d'abord (envoyer un code) - La production Twilio WhatsApp nécessite de toute façon la vérification Meta - Couche de coût supplémentaire (frais Twilio en plus de Meta)

Quand l'utiliser : - Test du bot localement - Démonstrations de preuve de concept - Développement et débogage - Déploiement initial avant la fin de la vérification Meta

Environnement de Production (API Cloud Meta)

Objectif : Intégration directe avec WhatsApp pour le déploiement en production

Comment ça fonctionne : - Intégration directe avec l'API Cloud WhatsApp de Meta - Aucun service intermédiaire - Votre serveur communique directement avec l'API de Meta - Les webhooks proviennent directement de Meta

Avantages : - Aucun frais par message (uniquement les frais de Meta) - Accès direct à l'API - Meilleures performances - Solution officielle WhatsApp Business - Aucune limitation de bac à sable

Exigences : - Vérification du compte Meta Business - Compte WhatsApp Business - Domaine avec certificat SSL - Point de terminaison webhook accessible publiquement

Quand l'utiliser : - Déploiement en production - Après la fin de la vérification Meta - Lors de la mise à l'échelle au-delà de la phase de test - Pour le déploiement officiel du système de santé en RDC

Choisir Votre Voie

Approche Recommandée :

1. **Commencer avec Twilio** (Semaine 1-2)
    - Configurer le bac à sable Twilio
    - Tester toutes les fonctionnalités
    - Former les utilisateurs
    - Vérifier l'intégration DHIS-2
2. **Transition vers Meta Avant le Déploiement Officiel** (Semaine 3+)
    - Compléter la vérification Meta Business
    - Mettre à jour la configuration du webhook
    - Migrer vers l'API Meta directe
    - Supprimer la dépendance Twilio

Notes Importantes : - Le code source utilise actuellement le SDK de Twilio pour l'envoi de messages - Pour la production avec l'API Cloud Meta, vous devrez adapter le fichier `webhook.py` pour utiliser le format API de Meta - Tout le reste du code (génération CIU, base de données, validation) reste identique - Voir la section Guide de Migration pour les modifications de code spécifiques nécessaires

Prérequis Techniques

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

Matériel Serveur Recommandé

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

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

Logiciels Requis

- **Système d'exploitation** : Ubuntu 22.04 LTS (recommandé) ou similaire
- **Python** : Version 3.12 ou supérieure
- **Base de données** : PostgreSQL 14+ (pour la production) ou SQLite (pour les tests)
- **Serveur Web** : Nginx (pour le proxy inverse)
- **Certificat SSL** : Let's Encrypt (gratuit)

Comptes et Services Externes

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

Compétences Techniques Requises

La personne installant ce système devrait avoir :

- Connaissance de base de la ligne de commande Linux
- Expérience de l'installation de logiciels serveur
- Compréhension des concepts réseau de base (DNS, ports, SSL)

Inscription Meta (API WhatsApp Business)

Option 1 : Utilisation de Twilio (Recommandé pour débuter)

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

Étape 1.1 : Créer un Compte Twilio

1. Allez sur https://www.twilio.com/try-twilio *(https://www.twilio.com/try-twilio)*
2. Cliquez sur **« Inscription »**
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)

Étape 1.2 : Configuration Initiale du Compte Twilio

1. Connectez-vous à la Console Twilio *(https://console.twilio.com/)*
2. Complétez le questionnaire de bienvenue :
    - Sélectionnez **« Messagerie »** comme produit principal
    - Sélectionnez **« Avec code »** pour la méthode de développement
    - Sélectionnez **« Python »** comme langage
3. Passez l'interface d'introduction

Étape 1.3 : Accéder au Bac à Sable WhatsApp

Pour les tests (Phase POC) :

1. Dans la Console Twilio, allez dans **« Messagerie »** → **« Essayer »** → **« Envoyer un message WhatsApp »**
2. Vous verrez :
    - Un numéro de bac à sable (par ex., `+1 415 523 8886`)

- Un code d'adhésion (par ex., `join side-orbit`)

3. Sur votre téléphone WhatsApp :
   - Enregistrez le numéro de bac à sable dans vos contacts
   - Envoyez le code d'adhésion à ce numéro
   - Attendez le message de confirmation

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

Étape 1.4 : Obtenir un Numéro WhatsApp Business (Production)

Pour la production :

1. Dans la Console Twilio, allez dans **« Messagerie »** → **« Expéditeurs WhatsApp »**
2. Cliquez sur **« Activer WhatsApp »**
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 du cas d'utilisation
   - Documents d'enregistrement d'entreprise (si requis)
5. Attendez l'approbation (généralement 1-3 jours ouvrables)

Étape 1.5 : Récupérer Vos Identifiants Twilio

1. Dans la Console Twilio, allez dans **« Compte »** → **« Tableau de bord »**
2. Notez ces valeurs **importantes** :
   - **Account SID** : Commence par `AC...` (par ex., `AC1234567890abcdef1234567890abcd`)
   - **Auth Token** : Cliquez sur « Afficher » pour révéler (par ex., `1234567890abcdef1234567890abcd`)
   - **Numéro WhatsApp** : Votre numéro de bac à sable ou d'entreprise (par ex., `+14155238886`)

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

Enregistrez Ces Valeurs : Copiez ces trois valeurs de la Console Twilio. Vous en aurez besoin lors de la configuration du fichier `.env` dans la section Configuration des Variables d'Environnement.

```
# Depuis votre Tableau de Bord Twilio
TWILIO_ACCOUNT_SID="AC..."

# Depuis votre Tableau de Bord Twilio
TWILIO_AUTH_TOKEN="..."

# Votre numéro de bac à sable ou de production
TWILIO_WHATSAPP_NUMBER="+..."
```

Option 2 : Accès Direct à l'API Meta (Avancé)

Si vous souhaitez vous connecter directement à Meta sans Twilio :

Étape 2.1 : Créer un Compte Meta Business

1. Allez sur https://business.facebook.com/ *(https://business.facebook.com/)*
2. Cliquez sur **« Créer un compte »**
3. Fournissez les informations de votre organisation
4. Vérifiez votre entreprise avec les documents requis

Étape 2.2 : Configurer l'Application WhatsApp Business

1. Dans Meta Business Manager, créez une nouvelle application
2. Ajoutez le produit **« WhatsApp »**
3. Configurez votre profil Business :
    - Nom de l'entreprise
    - Description
    - Catégorie (sélectionnez « Santé » ou similaire)
    - Photo de profil
4. Obtenez votre numéro de téléphone WhatsApp Business

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)

Enregistrez Ces Valeurs : Si vous utilisez l'API Meta directe, copiez ces valeurs. Vous en aurez besoin lors de la configuration du fichier `.env`.

```
# Votre Jeton d'Accès
META_WHATSAPP_TOKEN="..."

# Votre ID de Numéro de Téléphone
META_PHONE_NUMBER_ID="..."

# Créez une chaîne aléatoire sécurisée pour la vérification du webhook
META_VERIFY_TOKEN="..."
```

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

Intégration DHIS-2

Important : Points de Terminaison DHIS-2 Tracker

⚠️ **CRITIQUE POUR LE DÉPLOIEMENT :** Le code de l'application contient des URL de point de terminaison fictives qui **DOIVENT être remplacées** par vos points de terminaison réels de l'instance DHIS-2 Tracker avant l'utilisation en production.

Points de Terminaison DHIS-2 Requis

Votre implémentation DHIS-2 Tracker doit exposer les points de terminaison suivants. Remplacez les valeurs fictives dans le code source par votre hôte et points de terminaison DHIS-2 réels :

1. Point de Terminaison de Génération de CIU

- **Objectif** : Générer un nouveau CIU dans DHIS-2 Tracker
- **Fictif** : `/uic/generate`
- **Votre point de terminaison** : `https://votre-hote-dhis2.org/api/uic/generate`
- **Méthode** : POST

- **Requis** : Oui

2. Point de Terminaison de Validation du CIU

- **Objectif** : Vérifier si un CIU existe déjà avant de créer
- **Fictif** : `/uic/validate`
- **Votre point de terminaison** : `https://votre-hote-dhis2.org/api/uic/validate`
- **Méthode** : GET ou POST
- **Requis** : Oui (pour la prévention des doublons)

3. Point de Terminaison de Recherche de Client

- **Objectif** : Rechercher des clients existants dans DHIS-2 par attributs
- **Fictif** : `/clients/search`
- **Votre point de terminaison** : `https://votre-hote-dhis2.org/api/clients/search`
- **Méthode** : GET ou POST
- **Requis** : Oui (pour la vérification des doublons)

4. Point de Terminaison de Création de Client

- **Objectif** : Créer un nouvel enregistrement de client dans DHIS-2 Tracker
- **Fictif** : `/clients/create`
- **Votre point de terminaison** : `https://votre-hote-dhis2.org/api/clients/create`
- **Méthode** : POST
- **Requis** : Oui

5. Point de Terminaison de Fusion des Doublons

- **Objectif** : Fusionner les enregistrements de clients en double lorsqu'ils sont identifiés
- **Fictif** : `/duplicates/merge`
- **Votre point de terminaison** : `https://votre-hote-dhis2.org/api/duplicates/merge`
- **Méthode** : POST
- **Requis** : Optionnel (mais recommandé pour la qualité des données)

Étapes de Configuration

1. **Identifier Votre Hôte DHIS-2**
   - Déterminez votre URL de base DHIS-2 Tracker
   - Exemple : `https://dhis2.health.gov.cd` ou `https://tracker.ministry-health.cd`
2. **Vérifier la Disponibilité des Points de Terminaison**
   - Confirmez que tous les points de terminaison requis sont disponibles sur votre instance DHIS-2
   - Testez chaque point de terminaison avec votre administrateur DHIS-2
   - Documentez les exigences d'authentification (clés API, jetons OAuth, etc.)
3. **Mettre à Jour les Fichiers de Configuration**
   - Localisez le fichier de configuration contenant les points de terminaison fictifs
   - Remplacez chaque fictif par votre URL de point de terminaison réelle
   - Format : `https://votre-hote-dhis2.org/api/[chemin-du-point-de-terminaison]`
4. **Configurer l'Authentification**
   - Ajoutez les identifiants d'authentification DHIS-2 à vos variables d'environnement :

```
DHIS2_BASE_URL="https://votre-hote-dhis2.org"
DHIS2_API_USERNAME="[VOTRE_NOM_UTILISATEUR]"
DHIS2_API_PASSWORD="[VOTRE_MOT_DE_PASSE]"
# OU pour l'authentification par jeton :
DHIS2_API_TOKEN="[VOTRE_JETON_API]"
```

Flux de Travail d'Intégration

Le bot WhatsApp s'intègre avec DHIS-2 comme suit :

1. **L'Utilisateur Complète les Questions** → Le bot collecte les composants du CIU
2. **Vérification de Validation** → Appelle `/uic/validate` pour vérifier un CIU existant
3. **Recherche de Client** → Si nécessaire, appelle `/clients/search` pour trouver des doublons
4. **Génération de CIU** → Appelle `/uic/generate` pour créer le CIU dans DHIS-2
5. **Création de Client** → Appelle `/clients/create` pour enregistrer le client
6. **Réponse à l'Utilisateur** → Le bot livre le CIU à l'utilisateur WhatsApp

Test de l'Intégration DHIS-2

Avant de passer en production, testez l'intégration :

Tester le point de terminaison de validation du CIU :

```
curl -X POST https://votre-hote-dhis2.org/api/uic/validate \
  -H "Authorization: Bearer VOTRE_JETON" \
  -H "Content-Type: application/json" \
  -d '{"uic": "KABJEA512M"}'
```

Tester la recherche de client :

```
curl -X GET https://votre-hote-dhis2.org/api/clients/search?uic=KABJEA512M \
  -H "Authorization: Bearer VOTRE_JETON"
```

Tester la génération de CIU :

```
curl -X POST https://votre-hote-dhis2.org/api/uic/generate \
  -H "Authorization: Bearer VOTRE_JETON" \
  -H "Content-Type: application/json" \
  -d '{
    "last_name_code": "KAB",
    "first_name_code": "JEA",
    "birth_year_digit": "5",
    "city_code": "12",
    "gender_code": "M"
  }'
```

⚠️ NOTES IMPORTANTES

NE PAS déployer en production tant que tous les points de terminaison DHIS-2 ne sont pas : 1. ✅ Correctement configurés avec vos URL réelles 2. ✅ Testés et vérifiés comme fonctionnels 3. ✅ Sécurisés avec une authentification appropriée 4. ✅ Surveillés pour la performance et la disponibilité

Documentation Requise : Votre administrateur DHIS-2 devrait fournir : - Documentation complète des points de terminaison - Détails du mécanisme d'authentification - Formats de demande/réponse attendus - Informations sur la limitation de débit - Procédures de gestion des erreurs

Fonctionnalité Code QR (Optionnel)

Aperçu

Le bot peut optionnellement générer et envoyer des codes QR contenant le CIU de l'utilisateur via WhatsApp. Cela facilite l'enregistrement et le partage des codes par les utilisateurs.

Statut : Fonctionnalité optionnelle, désactivée par défaut

Comment Ça Fonctionne

Lorsque activé : 1. L'utilisateur complète les 5 questions 2. Le bot génère le code CIU (par ex., `MBEIBR7DA1`) 3. Le bot crée une image de code QR contenant le CIU 4. Le bot envoie à la fois le CIU en texte et l'image du code QR 5. L'utilisateur peut scanner le code QR pour accéder rapidement à son CIU

Activation de la Fonctionnalité

Étape 1 : Installer la Bibliothèque de Code QR

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

# Installer qrcode avec support PIL
pip install qrcode[pil]
```

Étape 2 : Activer dans la Configuration d'Environnement

Ajoutez à votre fichier `.env` :

```
# Activer la génération de code QR
ENABLE_QR_CODE=true
```

Étape 3 : Redémarrer l'Application

```
# Si vous utilisez le service systemd
sudo systemctl restart whatsapp-uic

# Ou si vous exécutez manuellement
# Arrêtez l'application et redémarrez-la
```

Comment les Codes QR Sont Livrés

Avec Twilio (Développement/Test)

Fonctionne immédiatement - Aucune configuration supplémentaire nécessaire

Twilio automatiquement : 1. Récupère l'image du code QR depuis votre serveur 2. La livre à l'utilisateur via WhatsApp

Exigences : - Votre serveur doit être accessible publiquement via HTTPS - Les codes QR sont servis à : `https://votre-serveur.com/static/qr_codes/{CIU}.png`

Avec l'API Cloud Meta (Production)

Nécessite une adaptation - Modifications de code supplémentaires nécessaires

Deux options pour l'API Cloud Meta :

Option 1 : Télécharger vers Meta d'abord (Recommandé)

```
# 1. Télécharger le code QR vers Meta
POST https://graph.facebook.com/v18.0/{phone_number_id}/media

# 2. Obtenir media_id de la réponse
# 3. Envoyer le message avec media_id
POST https://graph.facebook.com/v18.0/{phone_number_id}/messages
```

Option 2 : Utiliser l'URL Publique

```
{
  "messaging_product": "whatsapp",
  "to": "{destinataire}",
  "type": "image",
  "image": {
    "link": "https://votre-serveur.com/static/qr_codes/MBEIBR7DA1.png"
  }
}
```

Voir `docs/QR_CODE_FEATURE.md` pour un guide d'implémentation détaillé pour l'API Cloud Meta.

Stockage et Nettoyage

Où les Codes QR Sont Stockés

```
# Les codes QR sont enregistrés dans :
static/qr_codes/MBEIBR7DA1.png

# Structure de répertoire :
whatsapp-uic-generator/
  static/
    qr_codes/
      MBEIBR7DA1.png
      MOBMAR3KI2.png
      ...
```

Nettoyage Automatique (Optionnel)

Les codes QR sont stockés de façon permanente par défaut. Pour activer le nettoyage automatique, ajoutez une tâche cron :

```
# Éditer crontab
crontab -e

# Ajouter cette ligne pour supprimer les codes QR de plus de 7 jours (s'exécute
        quotidiennement à 2h du matin)
0 2 * * * cd /home/whatsapp-uic-generator && .venv/bin/python -c "from
        app.services.qr_service import QRCodeService;
        QRCodeService().cleanup_old_qr_codes(7)"
```

Test de la Fonctionnalité Code QR

```
# Tester la génération de code QR
python tests/test_qr_service.py

# Sortie attendue :
# Code QR généré avec succès !
# Fichier de code QR vérifié
# Tous les tests de code QR réussis !
```

Considérations de Sécurité

URLs Publiques : Les images de code QR sont accessibles publiquement à des URL prévisibles - Les URLs suivent le modèle : `/static/qr_codes/{CIU}.png` - Toute personne avec l'URL peut voir le code QR - **Atténuation** : Les CIU préservent déjà la confidentialité (pas d'informations personnelles identifiables)

Stockage : Envisagez d'implémenter : - Nettoyage automatique après X jours - Stockage cloud (S3, Cloudflare R2) pour la production - URLs signées temporaires pour une sécurité accrue

Dépannage

Les codes QR n'apparaissent pas ?

1. Vérifiez que la fonctionnalité est activée :

```
grep ENABLE_QR_CODE .env
# Devrait afficher : ENABLE_QR_CODE=true
```

1. Vérifiez la dépendance installée :

```
pip list | grep qrcode
# Devrait afficher : qrcode 7.4.0 ou supérieur
```

1. Vérifiez que le répertoire static existe :

```
ls -la static/qr_codes/
# Devrait exister avec le fichier .gitkeep
```

1. Vérifiez les journaux de l'application :

```
tail -f logs/app.log | grep "QR code"
# Devrait afficher les messages de génération de QR
```

Pour la documentation complète des codes QR, voir `docs/QR_CODE_FEATURE.md` dans le dépôt.

Migration de Twilio vers l'API Cloud Meta

Lorsque vous êtes prêt à passer de Twilio (développement) à l'API Cloud Meta (production), suivez ce guide.

Prérequis

Avant de migrer : - ✅ Compte Meta Business vérifié - ✅ Compte WhatsApp Business créé - ✅ Numéro de téléphone enregistré avec Meta - ✅ Webhook vérifié avec Meta - ✅ Jeton d'accès obtenu

Étape 1 : Mettre à Jour les Variables d'Environnement

Ajoutez les identifiants de l'API Cloud Meta à `.env` :

```
# Configuration de l'API Cloud Meta
META_ACCESS_TOKEN="votre_jeton_acces_meta_ici"
META_PHONE_NUMBER_ID="votre_id_numero_telephone_ici"
META_WEBHOOK_VERIFY_TOKEN="votre_jeton_verification_choisi_ici"

# Conservez Twilio pour le secours (optionnel)
TWILIO_ACCOUNT_SID="votre_account_sid"
TWILIO_AUTH_TOKEN="votre_auth_token"
```

Étape 2 : Mettre à Jour webhook.py

Le fichier actuel `app/api/webhook.py` utilise le SDK de Twilio. Pour l'API Cloud Meta, vous devrez :

1. **Remplacer la logique d'envoi de message** de Twilio par l'API de Meta
2. **Mettre à jour la vérification du webhook** pour la vérification challenge de Meta

Code Twilio actuel :

```
from twilio.twiml.messaging_response import MessagingResponse

twiml_response = MessagingResponse()
twiml_response.message(response_text)
return Response(content=str(twiml_response), media_type="application/xml")
```

Nouveau code API Cloud Meta :

```
import httpx
from fastapi import Response

# Envoyer un message via l'API Cloud Meta
async with httpx.AsyncClient() as client:
    response = await client.post(
        f"https://graph.facebook.com/v18.0/{META_PHONE_NUMBER_ID}/messages",
        headers={
            "Authorization": f"Bearer {META_ACCESS_TOKEN}",
            "Content-Type": "application/json"
        },
        json={
            "messaging_product": "whatsapp",
            "to": phone_number,  # Sans le préfixe 'whatsapp:'
            "type": "text",
            "text": {"body": response_text}
        }
    )

return Response(content='{"status": "ok"}', media_type="application/json")
```

Étape 3 : Ajouter le Point de Terminaison de Vérification du Webhook

Meta nécessite un point de terminaison de vérification. Ajoutez à `webhook.py` :

```
@router.get("/webhook")
async def verify_webhook(
    hub_mode: str = Query(None, alias="hub.mode"),
    hub_verify_token: str = Query(None, alias="hub.verify_token"),
    hub_challenge: str = Query(None, alias="hub.challenge")
):
    """Vérifier le webhook pour l'API Cloud Meta."""
    if hub_mode == "subscribe" and hub_verify_token == META_WEBHOOK_VERIFY_TOKEN:
        return Response(content=hub_challenge, media_type="text/plain")
    raise HTTPException(status_code=403, detail="Échec de la vérification")
```

Étape 4 : Mettre à Jour l'Analyse des Messages

La structure de charge utile du webhook de Meta est différente de celle de Twilio :

Structure du webhook Meta :

```
{
  "object": "whatsapp_business_account",
  "entry": [{
    "changes": [{
      "value": {
        "messages": [{
          "from": "1234567890",
          "text": {"body": "Bonjour"}
        }]
      }
    }]
  }]
}
```

Mettez à jour le webhook pour analyser le format de Meta au lieu des données de formulaire de Twilio.

Étape 5 : Mettre à Jour la Livraison de Code QR (si activé)

Pour l'API Cloud Meta, les codes QR doivent d'abord être téléchargés :

```
# 1. Télécharger le code QR vers Meta
async with httpx.AsyncClient() as client:
    with open(qr_path, 'rb') as f:
        files = {'file': ('qr.png', f, 'image/png')}
        upload_response = await client.post(
            f"https://graph.facebook.com/v18.0/{META_PHONE_NUMBER_ID}/media",
            headers={"Authorization": f"Bearer {META_ACCESS_TOKEN}"},
            files=files
        )

    media_id = upload_response.json()['id']

# 2. Envoyer le message avec media_id
await client.post(
    f"https://graph.facebook.com/v18.0/{META_PHONE_NUMBER_ID}/messages",
    headers={
        "Authorization": f"Bearer {META_ACCESS_TOKEN}",
        "Content-Type": "application/json"
    },
    json={
        "messaging_product": "whatsapp",
        "to": phone_number,
        "type": "image",
        "image": {"id": media_id, "caption": response_text}
    }
)
```

Étape 6 : Configurer le Webhook Meta

1. Allez dans la Console Développeur Meta
2. Accédez à votre application WhatsApp Business
3. Configurez le Webhook :
    - **URL de Rappel** : `https://whatsapp.votre-org.cd/whatsapp/webhook`
    - **Jeton de Vérification** : (depuis votre fichier `.env`)
    - **Champs du Webhook** : Sélectionnez `messages`
4. Cliquez sur **Vérifier et Enregistrer**

Étape 7 : Tester la Migration

```
# Tester la vérification du webhook Meta
curl "https://whatsapp.votre-org.cd/whatsapp/webhook?
        hub.mode=subscribe&hub.verify_token=VOTRE_JETON&hub.challenge=test123"
# Devrait retourner : test123

# Envoyer un message test depuis WhatsApp
# Vérifiez les journaux pour confirmer que le webhook Meta fonctionne
tail -f logs/app.log
```

Étape 8 : Supprimer Twilio (Optionnel)

Une fois que l'API Cloud Meta fonctionne :

```
# Supprimer le code spécifique à Twilio
# Mettre à jour .env pour supprimer les variables Twilio
# Désinstaller le SDK Twilio (si désiré)
pip uninstall twilio
```

Liste de Vérification de Migration

- [ ] Compte Meta Business vérifié
- [ ] Jeton d'accès Meta obtenu
- [ ] Variables d'environnement mises à jour
- [ ] webhook.py mis à jour pour l'API Meta
- [ ] Point de terminaison de vérification du webhook ajouté
- [ ] Analyse des messages mise à jour
- [ ] Livraison de code QR mise à jour (si activé)
- [ ] Webhook Meta configuré et vérifié
- [ ]

Messages de test envoyés avec succès

☐ Toutes les fonctionnalités fonctionnent (génération CIU, codes QR)

☐ Identifiants Twilio supprimés (optionnel)

Plan de Retour en Arrière

Si la migration échoue, revenez à Twilio :

```
# Annuler les modifications de webhook.py
git checkout HEAD -- app/api/webhook.py

# Mettre à jour l'URL du webhook Twilio
# Redémarrer l'application
sudo systemctl restart whatsapp-uic
```

Pour les sections complètes sur Configuration du Serveur, Installation de l'Application, Configuration des Variables d'Environnement, Déploiement en Production, Tests et Validation, Maintenance et Surveillance, Dépannage et Support, veuillez consulter le README.md anglais complet ou contactez l'équipe de support technique.

Licence

Remerciements

- Construit avec Python FastAPI, Twilio, et amour depuis le Canada
- Pour le déploiement en République Démocratique du Congo

Dernière Mise à Jour : Janvier 2026 **Version :** 1.0.0