Documentation Complète du Site Web de l'Ambassade de la République Centrafricaine

Version: 1.0
Date: 6 janvier 2026
Ambassade de la République Centrafricaine aux États-Unis d'Amérique

---

Résumé Exécutif

Ce document fournit une documentation technique complète pour le site web de l'Ambassade de la République Centrafricaine à Washington, D.C. Le système est une application web full-stack moderne conçue pour faciliter les services consulaires, gérer les demandes, et fournir des informations aux citoyens centrafricains et aux visiteurs potentiels.

Caractéristiques Principales

- **Système de Gestion des Utilisateurs** : Inscription, connexion, et authentification sécurisées
- **Traitement des Demandes** : Demandes de visa, certificats de mariage, certificats de naissance, et laissez-passer de voyage
- **Widget de Chat Multilingue** : Support en temps réel avec sélection de langue (anglais/français)
- **Suivi des Demandes** : Système de suivi par code-barres pour toutes les demandes
- **Récupération de Mot de Passe** : Système de réinitialisation par email sécurisé
- **Tableau de Bord Administrateur** : Outils de gestion complets pour le personnel de l'ambassade
- **Génération de PDF** : Génération automatique de documents avec codes-barres
- **Design Responsive** : Entièrement optimisé pour les ordinateurs de bureau, tablettes et appareils mobiles

---

Architecture du Système

Stack Technologique

Frontend

- **React 18.2.0** : Bibliothèque UI avec Hooks pour la gestion de l'état
- **React Router 6.x** : Routage côté client
- **CSS3** : Styles personnalisés avec queries média
- **Axios** : Client HTTP pour les appels API

Backend

- **Node.js** : Environnement d'exécution
- **Express.js** : Framework de serveur web
- **MySQL** : Système de gestion de base de données relationnelle
- **MySQL2** : Driver MySQL avec support des promesses

Sécurité et Authentification

- **bcrypt** : Hachage de mot de passe (10 tours de sel)
- **jsonwebtoken (JWT)** : Authentification basée sur les tokens
- **express-validator** : Validation et assainissement des entrées

Génération de Documents

- **PDFKit** : Création de documents PDF
- **bwip-js** : Génération de codes-barres Code 128

Services Email

- **SendGrid** : Service de livraison d'emails transactionnels

Déploiement

- **Docker** : Conteneurisation
- **Dokploy** : Orchestration de déploiement
- **Traefik** : Proxy inverse avec terminaison SSL/TLS

- **Git LFS** : Gestion des fichiers volumineux (vidéos)

Architecture de Déploiement

```
Internet
    ↓
Traefik (Proxy Inverse + SSL)
    ↓
Docker Network
    ├── Frontend Container (React App)
    ├── Backend Container (Node.js + Express)
    └── MySQL Container (Base de Données)
```

Aperçu des Fonctionnalités

1. Widget de Chat Multilingue

Le widget de chat fournit une assistance en temps réel aux utilisateurs avec support complet en anglais et français.

Fonctionnalités :

- Sélection de langue au démarrage (anglais/français)
- Système de traduction dynamique avec fonction `t()`
- Suivi des demandes intégré avec recherche par code-barres
- Modal responsive pour la saisie du code de suivi
- Stockage de la préférence de langue dans localStorage
- Mode plein écran sur petits téléphones (≤480px)
- Taille de police d'entrée de 16px pour éviter le zoom automatique iOS

Fichiers :

- `src/component/chat/ChatWidget.js` : Composant principal
- `src/component/chat/chatWidget.css` : Styles avec breakpoints responsive

Flux de Conversation :

1. Sélection de la langue (anglais/français)
2. Nom de l'utilisateur
3. Email de l'utilisateur
4. Support par chat avec options de suivi

2. Système de Suivi des Demandes

Permet aux utilisateurs de suivre l'état de leurs demandes en utilisant des codes-barres uniques.

Fonctionnalités :

- Recherche dans tous les types de demandes (visa, mariage, naissance, laissez-passer)
- Codes-barres Code 128 sur tous les PDF
- Modal de suivi dans le widget de chat
- Point de terminaison API : `/api/track/:trackingNumber`
- Affichage des détails : type, statut, nom du demandeur, dates

Implémentation :

```
// Backend (embassy-backend/index.js)
app.get('/api/track/:trackingNumber', async (req, res) => {
  const { trackingNumber } = req.params;

  // Recherche dans toutes les tables de demandes
  const [visaResults] = await pool.query(
    `SELECT * FROM visa_applications WHERE tracking_number = ?`,
    [trackingNumber]
  );

  const [marriageResults] = await pool.query(
    `SELECT * FROM marriage_applications WHERE tracking_number = ?`,
    [trackingNumber]
  );

  // ... certificats de naissance et laissez-passer de voyage

  const allResults = [...visaResults, ...marriageResults, ...birthResults, ...travelResults];

  if (allResults.length === 0) {
    return res.status(404).json({ error: 'Demande non trouvée' });
  }

  res.json(application);
});
```

3. Système de Récupération de Mot de Passe

Système complet de réinitialisation de mot de passe basé sur les emails avec validation de token.

Fonctionnalités :

- Demande de réinitialisation par email
- Génération de token JWT sécurisé
- Expiration du token après 1 heure
- Notification email via SendGrid
- Pages de confirmation de réinitialisation
- Redirection automatique après succès

Fichiers :

- `src/component/services/ForgotPassword.js` : Page de demande de réinitialisation
- `src/component/services/ResetPassword.js` : Page de confirmation de réinitialisation
- `embassy-backend/db.js` : Table `password_resets`

Flux :

1. Utilisateur demande la réinitialisation avec l'email
2. Backend génère un token JWT et l'enregistre dans la base de données
3. Email envoyé avec le lien de réinitialisation
4. Utilisateur clique sur le lien, entre un nouveau mot de passe
5. Token validé, mot de passe mis à jour, token supprimé
6. Redirection vers la page de connexion

4. Traitement des Demandes

Système complet de traitement pour quatre types de demandes :

Types de Demandes :

1. **Demandes de Visa**
2. **Certificats de Mariage**
3. **Certificats de Naissance**
4. **Laissez-Passer de Voyage (Urgence)**

Fonctionnalités :

- Formulaires avec validation
- Téléchargement de documents
- Génération automatique de numéros de suivi
- Génération de PDF avec codes-barres

- Mises à jour de statut (en attente, en cours, approuvé, refusé)
- Notifications par email

5. Tableau de Bord Administrateur

Accès restreint au personnel de l'ambassade (admin@usrcaembassy.org uniquement).

Fonctionnalités :

- Vue d'ensemble des statistiques (total des utilisateurs, demandes, messages)
- Gestion des demandes (révision, approbation, refus)
- Gestion des messages du chat
- Gestion des utilisateurs
- Paramètres du système

Restriction d'Accès :

```
// embassy-backend/index.js
const isAdmin = user.username.toLowerCase() === 'admin@usrcaembassy.org';
```

6. Tableau de Bord Utilisateur

Interface personnalisée pour les utilisateurs enregistrés.

Fonctionnalités :

- Aperçu des demandes (total, en attente, approuvées)
- Demandes récentes avec statut
- Navigation rapide vers les formulaires de demande
- Détails de profil
- Informations de compte

Schéma de Base de Données

Table : `login`

Stocke les comptes utilisateurs et les identifiants de connexion.

```
CREATE TABLE login (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(100) NOT NULL,
  lastname VARCHAR(100) NOT NULL,
  username VARCHAR(150) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_username (username)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `password_resets`

Gère les tokens de réinitialisation de mot de passe.

```
CREATE TABLE password_resets (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_id INT UNSIGNED NOT NULL,
  token TEXT NOT NULL,
  expires_at DATETIME NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  UNIQUE KEY unique_user_id (user_id),
  INDEX idx_token (token(255)),
  INDEX idx_expires (expires_at)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `visa_applications`

Stocke les demandes de visa.

```sql
CREATE TABLE visa_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(150) NOT NULL,
  tracking_number VARCHAR(50) UNIQUE,
  visa_type VARCHAR(100) NOT NULL,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  date_of_birth DATE NOT NULL,
  passport_number VARCHAR(50) NOT NULL,
  nationality VARCHAR(100) NOT NULL,
  email VARCHAR(150) NOT NULL,
  phone VARCHAR(20),
  status ENUM('pending', 'processing', 'approved', 'rejected') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  INDEX idx_user (user_name),
  INDEX idx_tracking (tracking_number),
  INDEX idx_status (status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `marriage_applications`

Stocke les demandes de certificats de mariage.

```sql
CREATE TABLE marriage_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(150) NOT NULL,
  tracking_number VARCHAR(50) UNIQUE,
  spouse1_first_name VARCHAR(100) NOT NULL,
  spouse1_last_name VARCHAR(100) NOT NULL,
  spouse1_dob DATE NOT NULL,
  spouse2_first_name VARCHAR(100) NOT NULL,
  spouse2_last_name VARCHAR(100) NOT NULL,
  spouse2_dob DATE NOT NULL,
  marriage_date DATE NOT NULL,
  marriage_location VARCHAR(255) NOT NULL,
  status ENUM('pending', 'processing', 'approved', 'rejected') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  INDEX idx_user (user_name),
  INDEX idx_tracking (tracking_number)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `birth_certificate_applications`

Stocke les demandes de certificats de naissance.

```sql
CREATE TABLE birth_certificate_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(150) NOT NULL,
  tracking_number VARCHAR(50) UNIQUE,
  child_first_name VARCHAR(100) NOT NULL,
  child_last_name VARCHAR(100) NOT NULL,
  child_dob DATE NOT NULL,
  birth_location VARCHAR(255) NOT NULL,
  parent1_name VARCHAR(200) NOT NULL,
  parent2_name VARCHAR(200),
  status ENUM('pending', 'processing', 'approved', 'rejected') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  INDEX idx_user (user_name),
  INDEX idx_tracking (tracking_number)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `travel_pass_applications`

Stocke les demandes de laissez-passer de voyage d'urgence.

```sql
CREATE TABLE travel_pass_applications (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(150) NOT NULL,
  tracking_number VARCHAR(50) UNIQUE,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  date_of_birth DATE NOT NULL,
  reason TEXT NOT NULL,
  destination VARCHAR(255) NOT NULL,
  travel_date DATE NOT NULL,
  status ENUM('pending', 'processing', 'approved', 'rejected') DEFAULT 'pending',
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  INDEX idx_user (user_name),
  INDEX idx_tracking (tracking_number)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Table : `messages`

Stocke les messages du widget de chat.

```sql
CREATE TABLE messages (
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  session_id VARCHAR(100),
  sender_type ENUM('user', 'bot') NOT NULL,
  sender_name VARCHAR(100),
  message TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_session (session_id),
  INDEX idx_created (created_at)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Documentation de l'API

Points de Terminaison d'Authentification

POST `/api/signup`

Créer un nouveau compte utilisateur.

Corps de la Requête :

```json
{
  "firstname": "Jean",
  "lastname": "Dupont",
  "username": "jean.dupont@email.com",
  "password": "motdepasse123"
}
```

Réponse :

```json
{
  "success": true,
  "message": "Compte créé avec succès"
}
```

POST `/api/login`

Authentifier un utilisateur et recevoir un token JWT.

Corps de la Requête :

```json
{
  "username": "jean.dupont@email.com",
  "password": "motdepasse123"
}
```

Réponse :

```
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "user": {
    "id": 1,
    "fullName": "Jean Dupont",
    "username": "jean.dupont@email.com",
    "isAdmin": false
  }
}
```

POST /api/password-reset/request

Demander une réinitialisation de mot de passe.

Corps de la Requête :

```
{
  "email": "jean.dupont@email.com"
}
```

Réponse :

```
{
  "success": true,
  "message": "Si un compte existe avec cet email, un lien de réinitialisation a été envoyé."
}
```

POST /api/password-reset/reset

Réinitialiser le mot de passe avec un token.

Corps de la Requête :

```
{
  "token": "eyJhbGciOiJIUzI1NiIs...",
  "newPassword": "nouveaumotdepasse123"
}
```

Réponse :

```
{
  "success": true,
  "message": "Le mot de passe a été réinitialisé avec succès"
}
```

Points de Terminaison des Demandes

POST /api/applications/visa

Soumettre une nouvelle demande de visa.

En-têtes :

```
Authorization: Bearer <token_jwt>
```

Corps de la Requête :

```
{
  "visa_type": "Tourisme",
  "first_name": "Jean",
  "last_name": "Dupont",
  "date_of_birth": "1990-01-15",
  "passport_number": "P123456",
  "nationality": "Française",
  "email": "jean.dupont@email.com",
  "phone": "+33123456789"
}
```

Réponse :

```
{
  "success": true,
  "applicationId": 42,
  "trackingNumber": "VIS-2026-00042",
  "message": "Demande de visa soumise avec succès"
}
```

GET `/api/track/:trackingNumber`

Suivre une demande par son numéro de suivi.

Exemple :

```
GET /api/track/VIS-2026-00042
```

Réponse :

```
{
  "id": 42,
  "type": "Visa",
  "typeDetail": "Tourisme",
  "applicantName": "Jean Dupont",
  "status": "processing",
  "createdAt": "2026-01-06T10:30:00.000Z",
  "updatedAt": "2026-01-06T14:22:00.000Z"
}
```

GET `/api/user/applications`

Obtenir toutes les demandes pour l'utilisateur connecté.

En-têtes :

```
Authorization: Bearer <token_jwt>
```

Réponse :

```
{
  "visas": [...],
  "marriages": [...],
  "births": [...],
  "travelPasses": [...]
}
```

Points de Terminaison Administrateur

GET `/api/admin/applications`

Obtenir toutes les demandes (admin uniquement).

En-têtes :

```
Authorization: Bearer <token_jwt_admin>
```

Réponse :

```
{
  "visas": [...],
  "marriages": [...],
  "births": [...],
  "travelPasses": [...]
}
```

PUT `/api/admin/applications/visa/:id`

Mettre à jour le statut d'une demande de visa.

En-têtes :

```
Authorization: Bearer <token_jwt_admin>
```

Corps de la Requête :

```
{
  "status": "approved"
}
```

Réponse :

```
{
  "success": true,
  "message": "Statut de la demande mis à jour"
}
```

Points de Terminaison du Chat

POST `/api/messages`

Envoyer un message de chat.

Corps de la Requête :

```
{
  "session_id": "session_123",
  "sender_type": "user",
  "sender_name": "Jean",
  "message": "Bonjour, j'ai besoin d'aide"
}
```

Réponse :

```
{
  "success": true,
  "messageId": 456
}
```

GET `/api/messages/:sessionId`

Récupérer les messages du chat pour une session.

Réponse :

```
[
  {
    "id": 455,
    "session_id": "session_123",
    "sender_type": "bot",
    "sender_name": "Bot de l'Ambassade",
    "message": "Bonjour ! Bienvenue...",
    "created_at": "2026-01-06T10:00:00.000Z"
  },
  {
    "id": 456,
    "sender_type": "user",
    "sender_name": "Jean",
    "message": "Bonjour, j'ai besoin d'aide",
    "created_at": "2026-01-06T10:01:00.000Z"
  }
]
```

Guide de Déploiement

Variables d'Environnement

Backend (embassy-backend/.env)

```
# Base de Données
DB_HOST=mysql
DB_USER=root
DB_PASSWORD=votre_mot_de_passe_sécurisé
DB_NAME=car_embassy
DB_PORT=3306

# JWT
JWT_SECRET=votre_clé_secrète_jwt_très_longue_et_aléatoire

# SendGrid
SENDGRID_API_KEY=votre_clé_api_sendgrid
CONTACT_FROM=noreply@usrcaembassy.org

# Frontend
FRONTEND_URL=https://usrcaembassy.org

# Serveur
PORT=5000
NODE_ENV=production
```

Frontend (.env)

```
REACT_APP_API_URL=https://backend.usrcaembassy.org
```

Configuration Docker

Dockerfile Frontend

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
RUN npm install -g serve
EXPOSE 3000
CMD ["serve", "-s", "build", "-l", "3000"]
```

Dockerfile Backend

```
FROM node:18-alpine
WORKDIR /app
COPY embassy-backend/package*.json ./
RUN npm install --production
COPY embassy-backend/ ./
EXPOSE 5000
CMD ["node", "index.js"]
```

Configuration Dokploy

1. **Créer un Nouveau Projet**
    - Nom : Embassy Website
    - Branche Git : main
    - Référentiel : https://github.com/groupeouambo/embassy-website
2. **Configurer le Service Frontend**
    - Type : Application Web
    - Port : 3000
    - Domaine : usrcaembassy.org
    - Dockerfile : Dockerfile (racine)
3. **Configurer le Service Backend**
    - Type : API
    - Port : 5000
    - Domaine : backend.usrcaembassy.org
    - Dockerfile : embassy-backend/Dockerfile

4. **Configurer MySQL**
    - Type : Base de Données
    - Version : 8.0
    - Volume : /var/lib/mysql
    - Variables d'environnement : DB_PASSWORD, MYSQL_DATABASE
5. **Configuration Traefik**
    - Certificats SSL automatiques (Let's Encrypt)
    - Redirection HTTP vers HTTPS
    - Routage basé sur le domaine

Considérations de Sécurité

1. Authentification et Autorisation

- Mots de passe hachés avec bcrypt (10 tours de sel)
- Authentification JWT avec tokens expirables
- Middleware de vérification de token sur toutes les routes protégées
- Restriction admin basée sur l'email exact

2. Validation des Entrées

- Validation côté serveur avec express-validator
- Assainissement de toutes les entrées utilisateur
- Requêtes préparées pour toutes les requêtes de base de données
- Protection contre l'injection SQL

3. Sécurité des Données

- Variables d'environnement pour les données sensibles
- Mots de passe non stockés en clair
- Tokens de réinitialisation de mot de passe expirables
- Communication sécurisée (HTTPS uniquement)

4. Protection contre les Attaques

- Limitation de débit sur les points de terminaison d'authentification
- Protection CORS appropriée
- En-têtes de sécurité HTTP
- Validation des téléchargements de fichiers

5. Sécurité de la Base de Données

- Indexation appropriée pour les performances
- Sauvegardes régulières
- Accès en lecture seule pour l'utilisateur de l'application
- Connexions chiffrées

Responsive Mobile

Points de Rupture

- **1024px+** : Disposition de bureau complète
- **769px-1023px** : Disposition tablette
- **481px-768px** : Grande disposition mobile
- **≤480px** : Petite disposition téléphone

Optimisations Clés

Barre de Navigation

- Logo : 70px (tablette) → 50px (mobile) → 45px (petit)
- Titre : 0.9rem → 0.75rem → 0.68rem
- Menu hamburger apparaît à 900px
- Titres se replient naturellement sans débordement
- `flex-shrink: 0` sur le logo

- `min-width: 0` sur le conteneur de titres

Page d'Accueil

- Disposition à une seule colonne sur mobile
- Le carrousel affiche un élément à la fois
- Hauteur de la galerie : 520px → 360px → 280px → 200px
- Toutes les sections empilées verticalement
- Police compacte (14px titre, 13px paragraphe)

Widget de Chat

- Mode plein écran sur téléphones (≤480px)
- Police d'entrée de 16px (évite le zoom iOS)
- Boutons de langue pleine largeur
- Fenêtre : 100vw x 100vh sur mobile

Formulaires d'Authentification

- Police d'entrée de 16px (évite le zoom iOS)
- Padding réduit sur mobile
- Boutons pleine largeur
- Labels accessibles

Cibles Tactiles

- Taille minimale : 44x44px (Apple HIG)
- Recommandé : 48x48px (Material Design)
- Espacement adéquat entre les éléments

Maintenance et Support

Surveillance

- Logs des erreurs du serveur
- Surveillance de la base de données
- Métriques de performance
- Suivi de l'uptime

Tâches Régulières

- Sauvegardes de la base de données (quotidiennes)
- Mises à jour de sécurité (hebdomadaires)
- Rotation des logs (mensuelle)
- Rotation des secrets JWT (trimestrielle)

Résolution des Problèmes

Problème : Les utilisateurs ne peuvent pas se connecter

- Vérifier les identifiants de la base de données
- Vérifier le secret JWT
- Vérifier les logs du serveur
- Vérifier les connexions réseau

Problème : Les emails ne sont pas envoyés

- Vérifier la clé API SendGrid
- Vérifier l'adresse email de l'expéditeur
- Vérifier les limites SendGrid
- Vérifier les logs du serveur

Problème : La génération de PDF échoue

- Vérifier les permissions des fichiers
- Vérifier l'espace disque
- Vérifier les dépendances PDFKit
- Vérifier les logs d'erreur

Contact et Support

Ambassade de la République Centrafricaine
1618 22nd Street NW
Washington, D.C. 20008
États-Unis d'Amérique

Email : info@usrcaembassy.org
Téléphone : +1 (202) 483-7800
Site Web : https://usrcaembassy.org

Support Technique :
Email : support@usrcaembassy.org

Fin de la Documentation