

Cahier des Charges Détaillé - Projet TaxasGE

Présentation du projet

TaxasGE est une application mobile disponible sur iOS et Android qui permet aux citoyens, entreprises et administrations de Guinée Équatoriale de consulter facilement les taxes fiscales du pays. L'application offre une interface intuitive, un chatbot intégré basé sur l'intelligence artificielle, et fonctionne en mode hors ligne.

Date de création: 26 mars 2025 Auteur: Emac Sah Version: 1.0.0

Objectifs du projet

- 1. Faciliter l'accès aux informations fiscales par une interface intuitive
- 2. Permettre la consultation des taxes en mode hors ligne
- 3. Proposer un assistant virtuel (chatbot) pour répondre aux questions
- 4. Offrir des fonctionnalités de recherche avancée et personnalisation
- 5. Assurer la mise à jour automatique des informations fiscales

Tableau récapitulatif du projet

Aspect	Description
Public cible	Citoyens, entreprises, administrations
Plateformes	iOS, Android
Technologie	Flutter, Node.js, PostgreSQL, TensorFlow Lite
Mode d'utilisation	Principalement hors ligne, synchronisation ponctuelle
Langues	Français, Espagnol

Durée développement 37 semaines (8 phases)

Configuration de l'environnement de développement

Environnement de développement

- 1. GitHub et Outils de gestion
 - Repository GitHub: taxasge-app
 - GitHub Projects pour la gestion des tâches
 - GitHub Actions pour l'intégration continue
- 2. Environnement de développement
 - GitHub Codespaces (environnement cloud)
 - Extension Flutter & Dart pour VS Code

- Émulateurs Web pour les tests (à compléter par des tests locaux sur appareils réels pour les phases finales)

3. Infrastructure backend

- Node.js / Express.js pour l'API REST
- PostgreSQL pour la base de données principale
- Redis pour le système de cache

Architecture technique détaillée

Architecture globale

L'application TaxasGE suit une architecture en couches avec une séparation claire entre:

- 1. Couche présentation (UI, écrans, widgets)**
- 2. Couche domaine (logique métier, modèles)**
- 3. Couche données (sources de données, services)**

Composants principaux

1. Application mobile Flutter

- Interface utilisateur
- Logique de présentation
- Stockage local (SQLite, JSON)
- Modèle NLP embarqué

2. Backend API REST

- Authentication et autorisation
- Endpoints CRUD pour les données fiscales
- Système de synchronisation
- Service de mise à jour delta

3. Base de données

- PostgreSQL (backend)
- SQLite (mobile)
- Cache Redis

4. Services externes

- Firebase Cloud Messaging (notifications)
- Firebase Analytics (suivi d'utilisation)

Diagramme de flux de données

L'application fonctionne selon trois modes principaux:

1. Mode hors ligne:

- Lecture des données depuis SQLite
- Traitement local des requêtes utilisateur
- Mise en cache des interactions avec le chatbot

2. Mode synchronisation:

- Détection des modifications locales
- Envoi des modifications au backend
- Réconciliation des conflits
- Téléchargement des mises à jour

3. Mode en ligne complet:

- Interaction directe avec le backend
- Utilisation du modèle NLP avancé (si disponible)
- Mise à jour en temps réel

Architecture fonctionnelle détaillée

Modules fonctionnels

1. Module de gestion utilisateur

- Création et gestion de compte
- Authentification et autorisation
- Préférences et favoris

2. Module de recherche et navigation

- Recherche simple et avancée
- Navigation par catégories
- Filtres et tris personnalisés

3. Module de consultation des taxes

- Visualisation détaillée des taxes
- Historique de consultation
- Partage et export

4. Module chatbot

- Interface conversationnelle
- Traitement du langage naturel

- Suggestions et recommandations

5. Module synchronisation et mise à jour

- Détection et résolution de conflits
- Téléchargement delta
- Gestion des versions

Cas d'utilisation principaux

1. Consulter une taxe spécifique
2. Rechercher des taxes par critères
3. Poser une question au chatbot
4. Ajouter une taxe aux favoris
5. Partager les détails d'une taxe
6. Synchroniser les données

Modèle de données

Structure de données générale

La structure de données est hiérarchique, reflétant l'organisation des taxes fiscales:

Ministère

└─ Secteur

└─ Service

└─ Taxe

└─ Taux d'expédition

└─ Taux de renouvellement

└─ Documents requis

└─ Procédures

Schéma JSON

```
{
  "ministères": [
    {
      "id": "min_001",
      "nom": "Ministère de la Sécurité Nationale",
      "secteurs": [
        {
```

```
{
  "id": "sec_001",
  "nom": "Secteur de Sécurité Nationale",
  "services": [
    {
      "id": "srv_001",
      "nom": "Document d'Identité Personnel",
      "taxes": [
        {
          "id": "tax_001",
          "nom": "Passeport ordinaire",
          "montant_expedition": 7500,
          "montant_renouvellement": 7500,
          "documents_requis": ["Formulaire de demande", "Pièce d'identité", "Photos d'identité"],
          "procedure": "Se présenter au bureau d'immigration...",
          "mots_cles": ["passeport", "voyage", "document"]
        }
      ]
    }
  ]
}
```

Schéma de base de données SQLite

1. Table Ministère

- **id (PRIMARY KEY)**
- **nom**
- **description**
- **date_maj**

2. Table Secteur

- id (PRIMARY KEY)
- ministere_id (FOREIGN KEY)
- nom
- description

3. Table Service

- id (PRIMARY KEY)
- secteur_id (FOREIGN KEY)
- nom
- description

4. Table Taxe

- id (PRIMARY KEY)
- service_id (FOREIGN KEY)
- nom
- montant_expedition
- montant_renouvellement
- description
- procedure
- date_maj

5. Table DocumentRequis

- id (PRIMARY KEY)
- taxe_id (FOREIGN KEY)
- nom
- description

6. Table MotCle

- id (PRIMARY KEY)
- taxe_id (FOREIGN KEY)
- mot

7. Table Favori

- id (PRIMARY KEY)
- user_id
- taxe_id (FOREIGN KEY)

- **date_ajout**

Description détaillée des phases et tâches

Phase 0: Configuration de l'infrastructure (Déjà réalisée)

Tâche 0.1: Configuration du repository GitHub

- **Création du repository**
- **Mise en place des workflows GitHub Actions**
- **Configuration des branches (main, develop)**

Tâche 0.2: Configuration de l'environnement de développement

- **Configuration de GitHub Codespaces**
- **Installation de Flutter et des extensions**
- **Initialisation du projet Flutter**

Tâche 0.3: Configuration des outils de gestion de projet

- **Création des milestones sur GitHub**
- **Configuration des issues et labels**
- **Mise en place du tableau Kanban (GitHub Projects)**

Phase 1: Préparation et conception (5 semaines)

Tâche 1.1: Analyse du document de loi fiscale (1.5 semaines)

- **Extraction et classification des données du Excel**
- **Identification des structures et hiérarchies**
- **Mapping des entités et relations**
- **Définition des métadonnées et attributs**

Livrables:

- **Script d'extraction des données du Excel**
- **Document de structure des données**
- **Catalogue des taxes classifiées**

Tâche 1.2: Conception de l'architecture technique (1.5 semaines)

- **Élaboration du diagramme d'architecture globale**
- **Définition des composants et leurs interactions**
- **Spécification des interfaces de communication**
- **Documentation des choix technologiques**

Livrables:

- Diagramme d'architecture technique
- Document de spécifications techniques
- Plan d'intégration des composants

Tâche 1.3: Conception du modèle de données (1 semaine)

- Définition des entités et attributs
- Élaboration du schéma JSON
- Conception du schéma SQLite
- Planification des stratégies de mise à jour

Livrables:

- Schéma JSON complet
- Schéma SQLite avec relations
- Documentation du modèle de données

Tâche 1.4: Design UX/UI (1 semaine)

- Création des wireframes des écrans principaux
- Élaboration de la charte graphique
- Design des composants d'interface
- Définition des parcours utilisateurs

Livrables:

- Wireframes de tous les écrans
- Charte graphique (couleurs, typographie, composants)
- Maquettes haute-fidélité des écrans principaux

Phase 2: Développement de la base de données (4 semaines)

Tâche 2.1: Extraction et normalisation des données (1.5 semaines)

- Développement du script d'extraction final
- Nettoyage et validation des données
- Normalisation et structuration
- Enrichissement avec métadonnées

Livrables:

- Script d'extraction optimisé
- Données structurées au format JSON
- Documentation du processus de normalisation

Tâche 2.2: Implémentation du schéma SQLite (1 semaine)

- Création des tables et relations
- Configuration des indices et contraintes
- Implémentation des migrations
- Tests de performance

Livrables:

- Code complet du schéma SQLite
- Scripts de migration
- Documentation de la structure

Tâche 2.3: Développement des services d'accès aux données (1.5 semaines)

- Création des repositories pour chaque entité
- Implémentation des méthodes CRUD
- Développement des requêtes avancées
- Mise en place du système de cache local

Livrables:

- Classes DAO pour chaque entité
- Service de gestion de base de données
- Tests unitaires des méthodes CRUD

Phase 3: Développement du modèle NLP (5 semaines)

Tâche 3.1: Préparation du corpus d'entraînement (1.5 semaines)

- Génération de paires question-réponse
- Annotation des données
- Augmentation du dataset
- Validation du corpus

Livrables:

- Corpus d'entraînement au format JSON
- Script de génération du corpus
- Documentation du processus d'annotation

Tâche 3.2: Développement du modèle NLP (2 semaines)

- Sélection de l'architecture du modèle
- Entraînement du modèle de base

- Fine-tuning avec les données fiscales
- Évaluation des performances

Livrables:

- Code d'entraînement du modèle
- Modèle entraîné (fichiers checkpoints)
- Rapport d'évaluation des performances

Tâche 3.3: Optimisation pour mobile (1.5 semaines)

- Quantification du modèle
- Conversion au format TensorFlow Lite
- Optimisation des inférences
- Tests de performance sur mobile

Livrables:

- Modèle TFLite optimisé
- Benchmarks de performance
- Documentation d'intégration

Phase 4: Développement de l'application Flutter - Core (6 semaines)

Tâche 4.1: Structure de base et navigation (1.5 semaines)

- Mise en place du système de navigation (routes)
- Implémentation du theme global
- Création du squelette d'application
- Configuration de l'état global

Livrables:

- Structure de navigation complète
- Thème global avec composants de base
- Configuration de l'état (Provider/Bloc)

Tâche 4.2: Écrans principaux (2 semaines)

- Développement de l'écran d'accueil
- Implémentation de l'écran de catégories
- Création de l'écran de détail de taxe
- Développement de l'écran de recherche simple

Livrables:

- Composants UI pour tous les écrans principaux
- Intégration avec les services de données
- Navigation entre les écrans

Tâche 4.3: Fonctionnalités de base (1.5 semaines)

- Implémentation de la recherche simple
- Développement du système de favoris basique
- Création du mode hors ligne de base
- Mise en place des fonctionnalités de partage

Livrables:

- Fonctionnalité de recherche avec résultats
- Système de favoris local
- Mode hors ligne fonctionnel

Tâche 4.4: Tests unitaires et d'intégration (1 semaine)

- Création des tests unitaires des modèles
- Développement des tests des services
- Mise en place des tests d'interface
- Vérification de la couverture

Livrables:

- Suite de tests unitaires
- Tests d'intégration des composants principaux
- Rapport de couverture de code

Phase 5: Développement des fonctionnalités avancées (6 semaines)

Tâche 5.1: Système de compte utilisateur (1.5 semaines)

- Mise en place de l'authentification locale
- Développement du profil utilisateur
- Implémentation de la synchronisation des préférences
- Sécurisation des données utilisateur

Livrables:

- Système d'authentification complet
- Écrans de profil et paramètres
- Service de gestion des préférences

Tâche 5.2: Recherche avancée et filtres (1.5 semaines)

- Développement de l'interface de recherche avancée
- Implémentation des filtres combinés
- Création du système de suggestion
- Optimisation des performances de recherche

Livrables:

- Interface de recherche avancée
- Fonctionnalités de filtrage multi-critères
- Système d'historique de recherche

Tâche 5.3: Intégration du chatbot (2 semaines)

- Développement de l'interface conversationnelle
- Intégration du modèle TensorFlow Lite
- Implémentation du système de suggestions
- Optimisation des performances

Livrables:

- Interface conversationnelle complète
- Intégration du modèle NLP
- Système de suggestions intelligentes

Tâche 5.4: Outils pratiques (1 semaine)

- Développement de la calculatrice fiscale
- Création du calendrier des échéances
- Implémentation des fonctionnalités d'export
- Intégration des notifications locales

Livrables:

- Calculatrice fiscale fonctionnelle
- Calendrier avec rappels
- Fonction d'export de données

Phase 6: Système de mise à jour et backend (4 semaines)

Tâche 6.1: Développement de l'API REST (1.5 semaines)

- Création des endpoints d'authentification
- Développement des routes CRUD pour les données

- Mise en place des middlewares de sécurité
- Documentation de l'API

Livrables:

- API REST complète
- Documentation Swagger
- Tests automatisés de l'API

Tâche 6.2: Base de données backend (1 semaine)

- Mise en place de PostgreSQL
- Configuration des schémas et relations
- Implémentation des migrations
- Optimisation des requêtes

Livrables:

- Schéma PostgreSQL complet
- Scripts de migration
- Requêtes optimisées

Tâche 6.3: Système de synchronisation (1.5 semaines)

- Développement du mécanisme de détection de modifications
- Implémentation de la synchronisation bidirectionnelle
- Gestion des conflits
- Optimisation du transfert de données

Livrables:

- Service de synchronisation complet
- Algorithme de résolution de conflits
- Tests de robustesse

Phase 7: Tests et assurance qualité (4 semaines)

Tâche 7.1: Tests unitaires et d'intégration (1 semaine)

- Complétion des tests unitaires
- Développement des tests d'intégration
- Mise en place des tests de bout en bout
- Validation de la couverture

Livrables:

- Suite de tests complète
- Rapport de couverture >80%
- Documentation des tests

Tâche 7.2: Tests de performance et optimisation (1 semaine)

- Benchmarking des fonctionnalités critiques
- Profilage de l'utilisation mémoire
- Optimisation des goulots d'étranglement
- Tests sur différents appareils

Livrables:

- Rapport de performance
- Optimisations documentées
- Benchmarks comparatifs

Tâche 7.3: Beta testing (2 semaines)

- Déploiement de la version beta
- Collecte des retours utilisateurs
- Correction des bugs identifiés
- Ajustements finaux

Livrables:

- Version beta stabilisée
- Rapport de retours utilisateurs
- Liste des corrections appliquées

Phase 8: Déploiement et suivi (3 semaines)

Tâche 8.1: Préparation au déploiement (1 semaine)

- Finalisation des assets pour les stores
- Création des fiches descriptives
- Configuration de l'analytique
- Préparation du matériel marketing

Livrables:

- Assets complets pour les stores
- Fiches descriptives optimisées
- Configuration de l'analytique

Tâche 8.2: Déploiement sur les stores (1 semaine)

- Création des builds de production
- Soumission à Google Play
- Soumission à l'App Store
- Suivi du processus de validation

Livrables:

- Applications déployées sur les stores
- Documentation du processus de déploiement
- Plan de communication de lancement

Tâche 8.3: Suivi et maintenance (1 semaine)

- Mise en place du monitoring
- Configuration des alertes
- Planification des mises à jour
- Élaboration de la stratégie de support

Livrables:

- Dashboard de monitoring
- Plan de maintenance
- Stratégie de mises à jour

Design UX/UI

Charte graphique

1. Couleurs

- Couleur principale: Bleu (#4285F4)
- Couleur secondaire: Vert (#34A853)
- Couleur d'accentuation: Rouge (#EA4335)
- Couleurs neutres: Blanc (FFFFFF), Gris clair (#F5F5F5), Gris foncé (#333333)

2. Typographie

- Titres: Roboto Bold
- Corps de texte: Roboto Regular
- Données numériques: Roboto Mono
- Taille de base: 16px avec échelle modular

3. Composants

- Boutons arrondis avec ombre légère
- Cartes avec élévation variable selon importance
- Barres de navigation avec icônes et labels
- Champs de recherche proéminents

Navigation

1. Structure principale

- Barre de navigation inférieure avec 4 onglets:
 - Accueil
 - Recherche
 - Favoris
 - Profil

2. Navigation secondaire

- Drawer latéral pour options avancées
- Navigation par catégories hiérarchiques
- Bouton retour contextuel

Écrans principaux

1. Écran d'accueil

- Barre de recherche proéminente
- Accès au chatbot
- Catégories de ministères avec icônes
- Taxes récemment consultées

2. Écran de catégories

- Navigation hiérarchique (Ministère > Secteur > Service)
- Indicateurs visuels de niveau
- Compteurs d'éléments par catégorie

3. Écran de détail de taxe

- Information claire sur les montants
- Onglets pour informations supplémentaires (procédures, documents)
- Boutons d'action (favoris, partage, calcul)

4. Écran de recherche

- Champ de recherche avec suggestions

- Filtres avancés accessibles
- Résultats groupés par pertinence

5. Écran de chatbot

- Interface conversationnelle
- Suggestions de questions
- Liens vers les taxes mentionnées

Dépendances techniques

Frontend (Flutter)

1. Packages essentiels

- http: Requêtes réseau
- shared_preferences: Stockage simple
- sqflite: Base de données SQLite
- path: Gestion des chemins
- provider: Gestion d'état
- flutter_local_notifications: Notifications locales

2. Packages d'interface

- flutter_svg: Support des SVG
- intl: Internationalisation
- url_launcher: Ouverture de liens
- share_plus: Partage de contenu

3. Packages d'intégration

- tfmlite_flutter: Intégration TensorFlow Lite
- connectivity_plus: Détection de connectivité
- path_provider: Accès au système de fichiers

4. Packages optionnels

- firebase_core: Base Firebase
- firebase_messaging: Notifications push
- firebase_analytics: Analytique

Backend (Node.js)

1. Packages principaux

- express: Framework web

- **pg:** Client PostgreSQL
- **sequelize:** ORM pour la base de données
- **jsonwebtoken:** Authentification JWT
- **bcrypt:** Hachage de mots de passe

2. Packages utilitaires

- **cors:** Gestion CORS
- **helmet:** Sécurité HTTP
- **morgan:** Logging
- **dotenv:** Variables d'environnement

3. Packages de développement

- **nodemon:** Rechargement automatique
- **jest:** Tests unitaires
- **supertest:** Tests d'API
- **eslint:** Linting

Ressources nécessaires

Équipe de développement

1. Rôles clés

- **Chef de projet**
- **Développeur Flutter**
- **Développeur backend Node.js**
- **Spécialiste NLP/ML**
- **Designer UX/UI**
- **Testeur QA**

Infrastructure technique

1. Environnement de développement

- **GitHub Codespaces**
- **Émulateurs Android/iOS**
- **Dispositifs de test réels**

2. Production

- **Serveur VPS (pour le backend)**
- **Base de données PostgreSQL**

- Redis pour le cache
- Système de CI/CD

Calendrier prévisionnel

Phase	Durée	Date début	Date fin
Phase 0: Configuration	2 semaines	Déjà réalisée	Déjà réalisée
Phase 1: Préparation et conception	5 semaines	À définir	À définir
Phase 2: Développement de la base de données	4 semaines	À définir	À définir
Phase 3: Développement du modèle NLP	5 semaines	À définir	À définir
Phase 4: Développement de l'application Flutter	6 semaines	À définir	À définir
Phase 5: Fonctionnalités avancées	6 semaines	À définir	À définir
Phase 6: Système de mise à jour et backend	4 semaines	À définir	À définir
Phase 7: Tests et assurance qualité	4 semaines	À définir	À définir
Phase 8: Déploiement et suivi	3 semaines	À définir	À définir
Total	37 semaines		

Plan de test

Types de tests

1. Tests unitaires

- Tests des modèles de données
- Tests des services
- Tests des utilitaires

2. Tests d'intégration

- Tests des flux de données
- Tests de synchronisation
- Tests de l'API

3. Tests d'interface

- Tests des composants UI
- Tests de navigation
- Tests d'accessibilité

4. Tests de performance

- Tests de charge

- Tests de consommation batterie
- Tests de mémoire

5. Tests utilisateurs

- Beta testing
- Tests d'acceptation
- Tests d'expérience utilisateur

Critères de qualité

1. Fonctionnels

- Exactitude des données fiscales
- Précision des réponses du chatbot
- Fiabilité de la synchronisation

2. Non-fonctionnels

- Temps de démarrage < 3 secondes
- Réponse du chatbot < 2 secondes
- Consommation mémoire < 100MB
- Fonctionnement hors ligne à 100%

Risques et mitigations

- 1. Risque: Complexité d'extraction des données du PDF Mitigation: Approche hybride d'extraction automatique et vérification manuelle**
- 2. Risque: Taille excessive du modèle NLP Mitigation: Quantification, pruning et optimisation pour mobile**
- 3. Risque: Variations de performance entre appareils Mitigation: Tests sur divers appareils et optimisations spécifiques**
- 4. Risque: Délais d'approbation des stores Mitigation: Planification anticipée et respect strict des guidelines**
- 5. Risque: Problèmes de connectivité en Guinée Équatoriale Mitigation: Priorité au mode hors ligne robuste**

Processus de validation

Chaque phase du projet nécessitera une validation formelle avant de passer à l'étape suivante:

1. Critères de validation

- Tous les livrables ont été fournis
- Les tests associés sont passés
- La documentation est complète

- Les critères de qualité sont respectés

2. Processus formel

- Réunion de présentation des livrables
- Tests de validation
- Approbation formelle
- Documentation de la validation

Conclusion

Ce cahier des charges détaillé constitue la base de référence pour le développement de l'application TexasGE. Il définit les objectifs, l'architecture, les fonctionnalités et le calendrier du projet.

Chaque phase de développement devra s'y référer pour assurer la cohérence de l'application et le respect des exigences. Les équipes travaillant sur les différentes phases pourront s'appuyer sur ce document pour comprendre le contexte global et les interdépendances entre les composants.

Le succès du projet repose sur le respect de ce cahier des charges et la collaboration efficace entre les différentes équipes de développement.