Document de Spécifications Techniques: Application INTIA

# Nom du projet :

Application de gestion des clients et assurances – INTIA Assurance

# Client :

INTIA Assurance

# Technologie principale :

Django

# Version :

1.0

# Date :

11 avril 2025

# 1. Objectif du projet

Mettre en place une application web sécurisée qui permet à la société INTIA Assurance de :  
- Créer, modifier, consulter et supprimer des clients  
- Gérer les contrats d’assurance associés à chaque client  
- Gérer les utilisateurs par agence (Direction générale, Douala, Yaoundé)  
- Contrôler l’accès aux données selon le rôle et l’agence de l’utilisateur

# 2. Architecture de la solution

Backend (API REST):

- Framework : Django 4+  
- API : Django REST Framework (DRF)  
- Authentification : Django author AbstractUser  
- Base de données : PostgreSQL(SQL pour la présentation)  
- Tests : pytest, drf-spectacular pour documentation OpenAPI

# 3. Modélisation des données

🔹 Client  
class Client(models.Model):  
 nom = models.CharField(max\_length=255)  
 email = models.EmailField(unique=True)  
 téléphone = models.CharField(max\_length=20)  
 adresse = models.TextField()  
 date\_inscription = models.DateField(auto\_now\_add=True)  
 agence = models.ForeignKey('Agence', on\_delete=models.CASCADE)

🔹 Assurance  
class Assurance(models.Model):  
 TYPE\_CHOICES = [('auto', 'Auto'), ('vie', 'Vie'), ('santé', 'Santé')]  
  
 client = models.ForeignKey(Client, on\_delete=models.CASCADE, related\_name='assurances')  
 type\_assurance = models.CharField(choices=TYPE\_CHOICES, max\_length=20)  
 date\_debut = models.DateField()  
 date\_fin = models.DateField()  
 montant = models.DecimalField(max\_digits=10, decimal\_places=2)  
 statut = models.CharField(max\_length=20, default='actif')

🔹 Agence  
class Agence(models.Model):  
 nom = models.CharField(max\_length=100, unique=True)  
 ville = models.CharField(max\_length=100)

🔹 Utilisateur personnalisé  
class CustomUser(AbstractUser):  
 ROLE\_CHOICES = [('admin', 'Admin'), ('agent', 'Agent'), ('directeur', 'Directeur')]  
 role = models.CharField(choices=ROLE\_CHOICES, max\_length=20)  
 agence = models.ForeignKey(Agence, on\_delete=models.SET\_NULL, null=True, blank=True)

# 4. Fonctionnalités de l’API

/api/auth/login/ : Connexion avec JWT  
/api/clients/ : Lister ou ajouter un client  
/api/clients/<id>/ : Modifier / supprimer / consulter un client  
/api/assurances/ : Ajouter ou voir les assurances  
/api/assurances/<id>/ : Modifier / supprimer / consulter une assurance  
/api/agences/ : Voir la liste des agences  
/api/users/ : Gérer les utilisateurs

# 5. Sécurité et permissions

- JWT Authentication avec djangorestframework-simplejwt  
- Permissions personnalisées :  
 \* Un agent ne peut accéder qu’aux clients et assurances de son agence  
 \* Un admin peut accéder à toutes les données  
 \* Un directeur peut voir les données globales mais ne peut pas modifier

# 6. Outils supplémentaires

- drf-spectacular : documentation Swagger automatique  
- Django Admin : gestion rapide des données en back-office  
- Pagination DRF : pour les listes de clients et assurances  
- Filtres : recherche par nom, agence, statut d’assurance, etc.

# 7. Déploiement

- Hébergement : Render, Heroku, Railway, AWS  
- Base de données : PostgreSQL hébergé (Supabase, ElephantSQL, etc.)  
- Sécurité : HTTPS, Auth JWT, Permissions DRF  
- Backup : Automatique via l’hébergeur

# 8. Roadmap (MVP)

- Conception des modèles : 2 jours  
- Création de l’API REST : 4 jours  
- Authentification + rôles : 2 jours  
- Tests + documentation : 2 jours  
- Déploiement : 1 jour  
Total estimé : 11 jours

# 9. Évolutions possibles

- Intégration frontend (React, Vue, etc.)  
- Génération de rapports PDF mensuels  
- Notifications par email ou SMS pour renouvellements  
- Application mobile (via React Native ou Flutter)  
- Tableau de bord analytique