Spécifications Techniques de l’Application INTIA

1. Introduction

L'application web proposée pour la société INTIA assurance permet de gérer les agences, les clients et les assurances. Elle offrira aux administrateurs la possibilité d'ajouter, de modifier, de supprimer et de consulter les informations des agences, clients, et contrats d'assurance. L’architecture repose sur Next.js pour le frontend et backend, Prisma pour la gestion de la base de données MongoDB, NextAuth pour l'authentification, et un modèle de gestion des accès basé sur les rôles (RBAC).

Objectifs du projet:

- Centraliser la gestion des agences, des clients et des contrats d’assurance.
- Offrir une interface intuitive pour faciliter l’ajout, la modification, la suppression et la consultation des données.
- Garantir un contrôle d'accès basé sur les rôles (administrateurs, managers, agents).

2. Fonctionnalités Principales

2.1 Gestion des Agences

- Ajouter une nouvelle agence.
- Modifier et supprimer une agence existante.
- Consulter la liste des agences.

2.2 Gestion des Clients

- Ajouter un client à une agence.
- Modifier les informations d'un client.
- Supprimer un client.
- Consulter la liste des clients d’une agence.

2.3 Gestion des Assurances

- Ajouter une nouvelle assurance pour un client.
- Modifier et supprimer une assurance existante.
- Consulter les assurances d’un client.

2.4 Authentification et Accès

- Authentification des utilisateurs par la Session.
- Gestion des rôles: Admin, Manager, Agent.
- Accès restreint en fonction du rôle (RBAC):
  - Admin: Gère toutes les agences, clients et assurances.
  - Manager: Gère les clients et assurances dans l'agence attribuée.
  - Agent: Accès en lecture uniquement pour consulter les clients et assurances de leur agence.

3. Architecture Système

3.1 Frontend - Next.js

- Technologie: Next.js avec rendu côté serveur (Server-Side Rendering).
- Pages statiques pour les pages publiques.
- Pas de gestion d'état global (utilisation d'actions serveur pour toutes les interactions).

3.2 Backend - Next.js & Prisma

- ORM: Prisma pour interagir avec la base de données MongoDB.
- Authentification: Utilisation de la session pour gérer l’authentification et les rôles d’accès.

3.3 Base de Données - MongoDB

- Base de données NoSQL hébergée sur MongoDB Atlas(Pour le cas de ce projet) ou une instance locale, assurant une gestion efficace des documents pour les clients, assurances et agences.

3.4 Hébergement

- Déploiement sur Vercel (recommandé pour les applications Next.js) ou Heroku pour l'hébergement du backend et du frontend.
- Git pour la gestion du code source et le contrôle de version.

4. Modèle de Données (Prisma)

4.1 Agence

```
model Agence {
 id       String   @id @default(auto()) @map("_id") @db.ObjectId
 nom      String
 clients Client[]
 userId String @unique @db.ObjectId
 user   User   @relation(fields: [userId], references: [id])
 assurences Assurance[]
 createdAt DateTime @default(now())
 updatedAt DateTime @default(now()) @updatedAt
}
```

4.2 Client

```
model Client {
 id     String @id @default(auto()) @map("_id") @db.ObjectId
 nom    String
 prenom String
 email  String @unique
 phone  String
 agencyId String @db.ObjectId
 agence   Agence @relation(fields: [agencyId], references: [id])
 assurances Assurance[]
 userId    String   @unique @db.ObjectId
 createdAt DateTime @default(now())
 updatedAt DateTime @default(now()) @updatedAt
}
```

4.3 Assurance

```
model Assurance {
 id       String @id @default(auto()) @map("_id") @db.ObjectId
 name     String
 amount Float
 clientId String @db.ObjectId
 client   Client @relation(fields: [clientId], references: [id])
 useId    String @db.ObjectId
 createBy User   @relation(fields: [useId], references: [id])
```

```
 agencyId String @db.ObjectId
 agency   Agence @relation(fields: [agencyId], references: [id])
 createdAt DateTime @default(now())
 updatedAt DateTime @default(now()) @updatedAt
}
```

4.4 User (Gestion des rôles)

```
model User {
 id        String   @id @default(auto()) @map("_id") @db.ObjectId
 email     String   @unique
 password  String
 firstName String
 lastName  String
 role      Role     @default(USER)
 createdAt DateTime @default(now())
 updatedAt DateTime @default(now()) @updatedAt
 agency     Agence?
 assurances Assurance[]
}
enum Role {
 ADMIN
 MANAGER
 AGENT
 CLIENT
 USER
}
```

5. Sécurité et Gestion des Accès

- Authentification: Utilisation de la session pour contrôler l'accès sécurisé aux données.
- RBAC (Role-Based Access Control):
  - L'accès à chaque ressource est contrôlé en fonction du rôle de l'utilisateur.
  - Un middleware sera implémenté pour vérifier les rôles avant d'accéder aux routes sensibles.

6. Interface Utilisateur (UI/UX)

L'interface sera simple et fonctionnelle pour une meilleure expérience utilisateur :

- Pages de gestion des agences : Créer, modifier et supprimer des agences.
- Pages de gestion des clients : Créer, modifier, consulter et supprimer des clients dans chaque agence.
- Pages de gestion des assurances: Gestion des contrats d'assurance pour chaque client.
- Tableaux interactifs: pour afficher les listes d'agences, de clients et d'assurances, avec des filtres de recherche.

7. Conclusion

La solution proposée avec Next.js, Prisma, et MongoDB répond aux exigences de la société INTIA pour la gestion centralisée de ses agences, clients et assurances, tout en assurant une sécurité robuste grâce à l'authentification basée sur les rôles.