📁 Structure Backend TITAN V31.4

```
server/
├── _core/
│   ├── context.ts          # Contexte tRPC avec RLS
│   ├── cookies.ts          # Configuration cookies
│   ├── env.ts              # Variables d'environnement
│   ├── errors.ts           # Gestion erreurs
│   ├── sdk.ts              # OAuth Manus (existant)
│   ├── systemRouter.ts     # Router système
│   └── trpc.ts             # Configuration tRPC + middleware RLS
│
├── db/
│   ├── database.ts         # Connexion Drizzle
│   ├── rls.ts              # 🆕 Contexte RLS PostgreSQL
│   └── migrations/         # Migrations Alembic/Drizzle
│
├── services/
│   ├── pgp.ts              # 🆕 Service chiffrement PGP
│   ├── auth.ts             # 🆕 Service authentification (crypt)
│   ├── search.ts           # 🆕 Service recherche full-text
│   └── stats.ts            # 🆕 Service vues statistiques
│
├── routers/
│   ├── auth.ts             # 🆕 Auth complet (login/register/me)
│   ├── users.ts            # 🆕 CRUD users + RLS
│   ├── centers.ts          # 🆕 CRUD centres
│   ├── patients.ts         # 🆕 CRUD patients + search
│   ├── appointments.ts     # 🆕 CRUD appointments + PGP
│   ├── encounters.ts       # 🆕 CRUD encounters + PGP
│   ├── invoices.ts         # 🆕 CRUD invoices + triggers
│   ├── doctors.ts          # 🆕 CRUD doctors
│   ├── pharmacy.ts         # 🆕 CRUD pharmacie + stock
│   ├── prescriptions.ts    # 🆕 CRUD prescriptions
│   ├── documents.ts        # 🆕 CRUD documents + upload
│   ├── roles.ts            # 🆕 CRUD roles/permissions
│   ├── stats.ts            # 🆕 Vues statistiques
│   └── index.ts            # 🆕 Agrégation routers
│
├── db.ts                   # Fonctions DB helper (existant, à modifier)
├── oauth.ts                # OAuth callback (existant)
└── index.ts                # Point d'entrée Express
```

```
drizzle/
└── schema.ts           # Schéma Drizzle (existant, complet)

.env.example            # Variables d'environnement
package.json
tsconfig.json
```

🔑 Fichiers à créer (🆕) ou modifier

Priorité CRITIQUE

1. `server/db/rls.ts` - Contexte RLS
2. `server/services/pgp.ts` - Chiffrement PGP
3. `server/services/auth.ts` - Auth PostgreSQL
4. `server/_core/trpc.ts` - Middleware RLS (modifier)
5. `server/routers/auth.ts` - Router auth complet

Priorité HAUTE

6. `server/routers/patients.ts` - CRUD + search
7. `server/routers/appointments.ts` - CRUD + PGP
8. `server/routers/encounters.ts` - CRUD + PGP
9. `server/routers/invoices.ts` - CRUD + triggers
10. `server/services/search.ts` - Recherche full-text

Priorité MOYENNE

11. `server/routers/users.ts` - CRUD users
12. `server/routers/centers.ts` - CRUD centres
13. `server/routers/doctors.ts` - CRUD doctors
14. `server/routers/pharmacy.ts` - CRUD pharmacie
15. `server/routers/stats.ts` - Vues statistiques

📝 Variables d'environnement requises

bash

```
# Database
DATABASE_URL=postgresql://titan:password@localhost:5432/titan_emr

# Auth
JWT_SECRET=your-super-secret-jwt-key-change-in-production
COOKIE_SECRET=your-cookie-secret-key

# OAuth Manus (existant)
OAUTH_SERVER_URL=https://oauth.example.com
APP_ID=your-app-id

# Centre par défaut (pour les nouveaux users)
DEFAULT_CENTER_ID=00000000-0000-0000-0000-000000000000
```

🚀 Ordre d'implémentation recommandé

1. **Phase 1 : Infrastructure Sécurité** (2-3 jours)
   - RLS Context (`db/rls.ts`)
   - PGP Service (`services/pgp.ts`)
   - Auth Service (`services/auth.ts`)
   - Middleware tRPC (modifier `_core/trpc.ts`)
2. **Phase 2 : Auth & Users** (1-2 jours)
   - Router Auth (`routers/auth.ts`)
   - Router Users (`routers/users.ts`)
   - Router Centers (`routers/centers.ts`)
3. **Phase 3 : Données Cliniques** (2-3 jours)
   - Router Patients + Search (`routers/patients.ts`)
   - Router Appointments + PGP (`routers/appointments.ts`)
   - Router Encounters + PGP (`routers/encounters.ts`)
4. **Phase 4 : Finance & Pharmacie** (2-3 jours)
   - Router Invoices + Triggers (`routers/invoices.ts`)
   - Router Pharmacy + Stock (`routers/pharmacy.ts`)
   - Router Doctors + Commissions (`routers/doctors.ts`)
5. **Phase 5 : Stats & Documents** (1-2 jours)

- Router Stats + Vues (`routers/stats.ts`)
- Router Documents + Upload (`routers/documents.ts`)

✅ Tests à effectuer

```bash
# 1. Migration DB
pnpm db:push

# 2. Créer un centre
psql -d titan_emr -c "INSERT INTO centers (name, code) VALUES ('Test Center', 'TC001') RETURNING id;"

# 3. Créer un user
pnpm tsx server/scripts/create-user.ts

# 4. Tester RLS
# - User A voit seulement son centre
# - Superadmin voit tous les centres

# 5. Tester PGP
# - Créer appointment avec notes
# - Vérifier chiffrement en DB
# - Vérifier déchiffrement en API

# 6. Tester Triggers
# - Ajouter invoice_items → vérifier total_amount
# - Créer medical_acts → vérifier commissions
# - Ajouter pharmacy_transaction → vérifier stock
```

🎯 Résultat final

Un backend **100% fonctionnel** avec :

- ✅ Isolation multi-centres automatique (RLS)
- ✅ Chiffrement transparent (PGP)
- ✅ Authentification PostgreSQL native
- ✅ Recherche full-text performante
- ✅ Calculs automatiques (triggers)
- ✅ Statistiques temps réel (vues)

- ✅ Documentation auto (tRPC + OpenAPI)