■ Guide Complet d'Organisation

Application de Gestion Locative

Next.js 14 • TypeScript • PostgreSQL • Prisma

■ Vue d'ensemble

Ce guide complet vous accompagne dans l'organisation et le développement de votre application SaaS de gestion locative. Vous y trouverez la structure complète du projet, les conventions à suivre, et toutes les commandes essentielles.

■ Installation Rapide

Méthode Automatique (Recommandée)
1. Téléchargez le script init-gestion-locative.sh
2. Rendez-le exécutable : chmod +x init-gestion-locative.sh
3. Exécutez-le : ./init-gestion-locative.sh

Le script crée automatiquement toute la structure, installe les dépendances, configure Prisma et initialise Git.

■ Structure Principale

app/ - Cœur de l'application Next.js 14
• (auth)/ - Routes d'authentification (login, register)
• (dashboard)/ - Espaces owner, tenant, admin
• api/ - API Routes pour le backend

components/ - Composants React réutilisables
• ui/ - Composants de base (button, card, input, modal)
• dashboard/ - Composants spécifiques dashboards
• forms/ - Tous les formulaires
• layout/ - Structure (header, sidebar, footer)

lib/ - Utilitaires et configurations
• prisma.ts - Client Prisma singleton
• auth.ts - Configuration NextAuth
• validations.ts - Schémas Zod

• utils.ts - Fonctions utilitaires

prisma/ - Base de données
• schema.prisma - Modèles de données
• migrations/ - Historique des migrations

types/ - Types TypeScript
• index.ts, user.ts, property.ts, lease.ts

■ Conventions de Nommage

Fichiers et Dossiers
• Composants : kebab-case (property-card.tsx)
• Pages Next.js : kebab-case (page.tsx, layout.tsx)
• API Routes : kebab-case (route.ts)
• Dossiers : kebab-case (dashboard/, api/)

Code TypeScript/React
• Composants : PascalCase (PropertyCard, PaymentForm)
• Fonctions : camelCase (fetchProperties, validateUser)
• Variables : camelCase (userName, propertyList)
• Constantes : UPPER_SNAKE_CASE (MAX_UPLOAD_SIZE)
• Types : PascalCase (User, Property, LeaseStatus)

■ Commandes Essentielles

Next.js
npm run dev - Serveur développement
npm run build - Build production
npm run start - Lancer en production
npm run lint - Vérifier le code

Prisma
npx prisma migrate dev - Créer migration
npx prisma generate - Générer client
npx prisma studio - Interface graphique DB
npx prisma db push - Sync rapide (dev)

Git
git checkout -b feature/nom - Nouvelle branche
git add . - Tout ajouter
git commit -m "message" - Commit
git push origin branch - Push

■ Configuration VS Code

Extensions Essentielles
1. ESLint - Détection d'erreurs
2. Prettier - Formatage automatique
3. Prisma - Support Prisma
4. Tailwind CSS IntelliSense - Autocomplétion
5. Error Lens - Erreurs inline

Un popup apparaîtra automatiquement dans VS Code pour installer ces extensions lors de l'ouverture du projet.

■ Variables d'Environnement

Fichier .env (NE PAS COMMIT)
DATABASE_URL - PostgreSQL connection string
NEXTAUTH_URL - URL de l'application
NEXTAUTH_SECRET - Secret (générer avec openssl)
STRIPE_SECRET_KEY - Clé Stripe
CLOUDINARY_* - Config Cloudinary
RESEND_API_KEY - API key Resend

Générer un secret
openssl rand -base64 32

■ Workflow de Développement

Démarrage Quotidien
Terminal 1 : npm run dev
Terminal 2 : npx prisma studio (optionnel)
Terminal 3 : Commandes Git

Créer une Feature
1. Créer branche : git checkout -b feature/nom
2. Développer la fonctionnalité
3. Tester : npm run build && npm run type-check
4. Commit : git commit -m "feat: description"
5. Push : git push origin feature/nom

Workflow Prisma
1. Modifier prisma/schema.prisma
2. npx prisma migrate dev --name nom_migration
3. npx prisma generate
4. Tester avec npx prisma studio

■ Checklist de Démarrage

■ Projet créé avec create-next-app
■ Dépendances installées
■ Prisma initialisé
■ Structure des dossiers créée
■ .env configuré
■ .gitignore configuré
■ VS Code configuré avec extensions
■ PostgreSQL installé et configuré
■ Première migration Prisma créée
■ Git initialisé
■ Premier commit effectué
■ npm run dev fonctionne

■ Roadmap (Rappel)

Phase 1 (Semaines 1-3) - Fondations
Setup, authentification, landing page

Phase 2 (Semaines 4-7) - MVP Propriétaire
CRUD propriétés, upload images, dashboard

Phase 3 (Semaines 8-10) - MVP Locataire
Modèle Lease, dashboard locataire, maintenance

Phase 4 (Semaines 11-12) - Admin
Panel admin, gestion utilisateurs

Phase 5 (Semaines 13-18) - Avancé
Stripe, messagerie, notifications

Phase 6 (Semaines 19-20) - Production
Tests, optimisation, déploiement Vercel

■ Bon développement ! ■

Ce guide est évolutif. Mettez-le à jour au fur et à mesure.