# Conception Générale - HabitatsConnect

## Vue d'ensemble du projet

**HabitatsConnect** est une plateforme moderne de gestion immobilière similaire à Airbnb, permettant aux propriétaires de louer leurs propriétés et aux voyageurs de réserver des hébergements.

## Objectifs principaux

- Fournir une plateforme intuitive pour la location de propriétés
- Assurer une expérience utilisateur fluide et sécurisée
- Intégrer des fonctionnalités avancées de gestion immobilière
- Supporter une architecture scalable et maintenable

## Fonctionnalités clés

### Pour les propriétaires

- Inscription et gestion de profil
- Ajout et gestion de propriétés
- Gestion des réservations et disponibilités
- Suivi des transactions financières
- Communication avec les locataires

### Pour les locataires

- Recherche et filtrage de propriétés
- Réservation en ligne sécurisée
- Système d'avis et de notation
- Messagerie intégrée
- Gestion des favoris

## Architecture technique

- **Frontend**: Next.js 14 avec React 18 et TypeScript
- **Styling**: Tailwind CSS avec thème fluide personnalisé
- **State Management**: Zustand pour la gestion d'état
- **Backend**: API Routes Next.js
- **Base de données**: MongoDB ou PostgreSQL (à configurer)
- **Authentification**: JWT avec bcrypt pour le hachage
- **Déploiement**: Vercel ou Railway

## Structure du projet

```
src/
├── app/              # Pages Next.js
├── components/       # Composants réutilisables
```

```
├── hooks/           # Hooks personnalisés
├── lib/             # Utilitaires et configurations
├── models/          # Modèles de données
├── store/           # Stores Zustand
└── types/           # Types TypeScript
```

## Technologies utilisées

- Next.js 14
- React 18
- TypeScript
- Tailwind CSS
- Zustand
- React Icons
- Lucide Icons

## Phases de développement

1. **Phase initiale**: Configuration et scaffolding
2. **Conception**: Définition des spécifications
3. **Développement**: Implémentation des fonctionnalités
4. **Test**: Validation et débogage
5. **Déploiement**: Mise en production
6. **Maintenance**: Support et évolution

## Critères de succès

- Interface utilisateur intuitive et responsive
- Performance optimale
- Sécurité des données
- Évolutivité de l'architecture
- Satisfaction des utilisateurs finaux