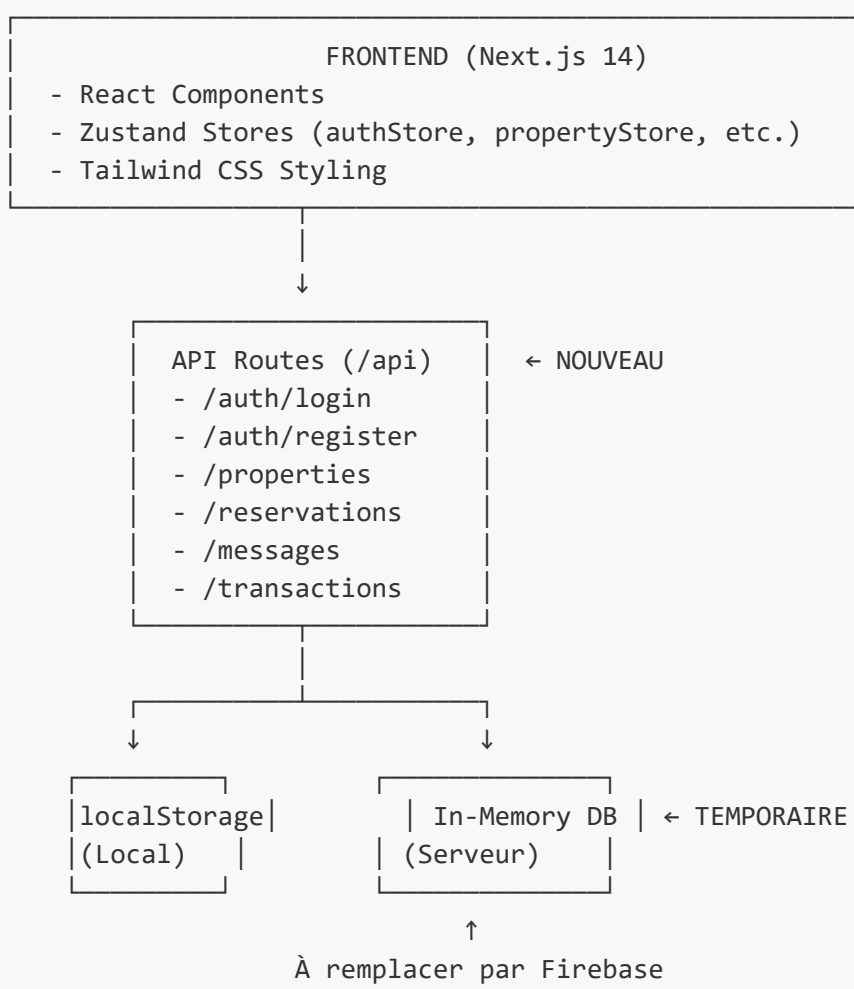
🏗️ Architecture - HabitatsConnect

📊 Flux de données actuellement



🔄 Flux d'authentification

```
1. Utilisateur se connecte
   ↓
2. authStore.login() appelé
   ↓
3. D'abord: Essayer l'API (/api/auth/login)
   ↓
4. Si l'API échoue: Fallback à localStorage
   ↓
5. Utilisateur connecté → Token stocké
   ↓
6. Composants peuvent accéder à user via useAuthStore()
```

💾 Où sont les données?

Actuellement:

- **localStorage**: Données du navigateur local (spécifique à chaque navigateur)
- **API In-Memory**: Données du serveur (perdues au redémarrage)

À implémenter:

- **Firebase Firestore**: Base de données NoSQL cloud
- **Firebase Authentication**: Gestion des utilisateurs et tokens

📝 Stores Zustand

authStore

- user: Utilisateur connecté
- token: JWT token
- isAuthenticated: État de connexion
- login(email, password, role): Connexion
- register(userData): Inscription
- logout(): Déconnexion

propertyStore

- properties: Liste des propriétés
- addProperty(property): Ajouter une propriété
- updateProperty(id, data): Modifier
- deleteProperty(id): Supprimer
- getPropertyById(id): Récupérer une

reservationStore

- reservations: Liste des réservations
- addReservation(reservation): Ajouter
- updateReservation(id, data): Modifier
- getReservationsByPropertyId(id): Filtre

messageStore

- conversations: Conversations
- messages: Messages par conversation
- addConversation(conversation): Ajouter
- addMessage(conversationId, message): Message

transactionStore

- transactions: Transactions
- addTransaction(transaction): Ajouter
- getTransactionsByOwnerId(id): Filtrer

🔗 API Routes (Nouvelles)

Authentification

- POST /api/auth/login : Connexion utilisateur
- POST /api/auth/register : Inscription utilisateur

Propriétés

- GET/POST /api/properties : Gestion des propriétés
- GET/PUT/DELETE /api/properties/[id] : Propriété spécifique

Réservations

- GET/POST /api/reservations : Gestion des réservations

Messages

- GET/POST /api/messages : Gestion des conversations

Transactions

- GET/POST /api/transactions : Gestion des transactions

🔐 Authentification Flow

```
1. User clique "Connexion"
2. Form envoyé à POST /api/auth/login
3. API vérifie dans Firebase Auth
4. Si valide: Retourne user + token
5. Token stocké dans authStore
6. Utilisateur redirigé vers dashboard
7. Toutes les requêtes incluent le token
8. Token stocké en localStorage aussi (pour persistance)
```

🚨 Problèmes actuels

- Données en mémoire (API) perdues au redémarrage
- Pas de vraie base de données persistante
- Pas de hachage de mots de passe sécurisé
- Pas de tokens JWT appropriés
- Pas de validation côté serveur
- Pas de configuration CORS

☑️ À faire pour production

Étape 1: Base de données

Configuration de Firebase Firestore comme base de données principale.

Étape 2: Authentification sécurisée

Implémentation de Firebase Authentication avec gestion des mots de passe.

Étape 3: Tokens sécurisés

Utilisation des tokens Firebase pour l'authentification.

Étape 4: Middleware d'authentification

Mise en place de la validation des tokens côté serveur.

📦 Variables d'environnement nécessaires

Configuration des clés Firebase et paramètres d'environnement pour le développement et la production.

🎯 Prochaines étapes

1. Configurer Firebase comme base de données
2. Migrer les stores pour utiliser Firebase
3. Implémenter l'authentification sécurisée
4. Déployer sur une plateforme cloud
5. Tester en production