CoHabitat

Cahier des charges

Thomas Spencer

Table des matières

1. Contexte du projet

La gestion des bâtiments résidentiels repose encore trop souvent sur des échanges physiques, informels ou non structurés : fiches papier, appels téléphoniques, messages éparpillés, etc. Cette approche entraîne des pertes d'informations, un manque de réactivité dans la résolution des incidents, et une difficulté à centraliser les données utiles tant pour les gardiens que pour les locataires.

Dans un monde de plus en plus connecté, il devient essentiel d'outiller numériquement cette relation. Le projet **CoHabitat** répond à ce besoin en proposant une application mobile moderne et intuitive, destinée à fluidifier les échanges et la gestion quotidienne des bâtiments.

Techniquement, le cœur de cette solution repose sur une **API RESTful développée sur mesure**. Cette API, conçue manuellement sans l'usage de services tiers, permet une maîtrise totale de la structure des données, de la logique métier et des flux de communication entre le front-end (React Native/Expo) et la base de données. Elle garantit également une évolutivité du projet et une meilleure adaptation aux besoins spécifiques du terrain.

En optant pour une architecture API-first, **CoHabitat** s'assure d'un découplage clair entre l'interface utilisateur et les traitements serveur, facilitant la maintenance, les mises à jour futures et l'extension à d'autres plateformes (par exemple, une interface web ou une borne en pied d'immeuble).

2. Objectifs du projet

- Créer une application mobile intuitive pour faciliter la gestion des bâtiments par les gardiens et l'accès à l'information par les locataires.
- Centraliser les informations relatives aux bâtiments, incidents, et contacts.
- Accélérer le traitement des signalements grâce à un système de suivi clair.
- Favoriser une communication bidirectionnelle efficace entre gardiens et locataires.
- Mettre en place une infrastructure technique moderne, sûre et évolutive.

3. Périmètre du projet

Le projet concerne exclusivement la création d'une application mobile multiplateforme (iOS/Android) ainsi que le développement de son backend via une API REST personnalisée. Les fonctionnalités incluent :

- Gestion des bâtiments et des utilisateurs (gardiens/locataires)
- Système de signalement et de suivi des incidents
- Profils utilisateurs
- Accès aux informations de contact et de maintenance

Sont exclues du projet initial :

- Une interface web de gestion
- Intégration avec des services tiers (Firebase, etc.)
- Paiement en ligne ou autres services financiers
- Test unitaires / integration
- Notifications internes

4. Aspects Fonctionnels

L'application CoHabitat est conçue pour répondre aux besoins spécifiques de deux profils d'utilisateurs : les **gardiens d'immeuble** et les **locataires**.

Fonctionnalités pour les gardiens :

- Consultation et gestion des bâtiments attribués
- Suivi et traitement des signalements envoyés par les locataires
- Accès aux fiches de contact des résidents
- Modification de son profil personnel
- Visualisation des historiques d'intervention

Fonctionnalités pour les locataires :

- Signalement rapide d'un incident ou problème
- Suivi de l'état d'avancement du signalement
- Accès aux informations du bâtiment (règlement, équipements, etc.)
- Consultation des coordonnées du gardien
- Mise à jour de son profil utilisateur

Fonctionnalités transversales :

- Authentification sécurisée
- Navigation fluide entre les écrans (Expo Router)
- Interface responsive adaptée aux smartphones
- Design clair, intuitif et accessible

Chaque fonctionnalité vise à réduire les frictions dans les échanges, automatiser certaines tâches récurrentes, et centraliser les informations pertinentes dans un seul outil accessible à tout moment.

5. Aspects techniques

Le projet CoHabitat repose sur une architecture technique moderne, modulaire et évolutive. Elle s'appuie sur des technologies libres et couramment utilisées dans l'industrie du développement mobile et backend.

Technologies utilisées :

- **React Native + Expo** : pour le développement de l'interface mobile multiplateforme (Android/iOS), avec navigation via **Expo Router**.
- **TypeScript** : pour la robustesse, l'autocomplétion et la clarté du code source front-end.
- **Node.js** avec **Express.js** : pour la création d'une API RESTful sur mesure côté serveur.
- **SQLite** : base de données locale, simple et légère, adaptée au prototypage rapide.
- **Zod** : pour la validation des données côté backend.
- **JWT (jsonwebtoken)** : pour l'authentification sécurisée des utilisateurs (gardiens et locataires).
- **Bcrypt** : pour le hachage sécurisé des mots de passe.
- **Morgan** : pour le logging des requêtes HTTP côté serveur (middleware de journalisation).
- **Railway** : pour l'hébergement de l'API backend.

Architecture technique :

- API REST découplée du front-end, avec endpoints documentés.
- Stockage local des données sans utilisation de services cloud tiers (conformité RGPD).
- Contrôle d'accès par rôles (gardien / locataire).

Bonnes pratiques de développement :

- Utilisation du typage statique avec TypeScript sur toute la stack.
- Organisation du projet par modules fonctionnels.
- Suivi de version avec Git et documentation continue.

Cette approche permet un déploiement rapide, une bonne maintenabilité du code et une flexibilité pour évoluer vers des solutions plus complexes (PostgreSQL, ORM, hébergement Docker, etc.) à l'avenir.

6. Aspects design / UI

L'interface utilisateur de l'application **CoHabitat** est pensée pour offrir une expérience simple, fluide et accessible à tous les utilisateurs, qu'ils soient locataires ou gardiens. Les maquettes interactives du design sont disponibles à l'adresse suivante :

https://www.figma.com/design/9SM0obyROKXe2hgnjnfkkS/Cohabitat?node-id=0-1&t=eFH3UKrZVIc73B0f-1

Principes de design appliqués :

- **Clarté & Hiérarchie visuelle**
  - Utilisation d'une palette de couleurs contrastée pour distinguer les rôles et les statuts (ex. incidents signalés).
  - Titres, boutons et icônes sont visuellement hiérarchisés pour guider l'utilisateur dans son parcours.
- **Design accessible & mobile-first**
  - Taille des éléments adaptée aux usages mobiles (gros boutons, padding suffisant).
  - Icônes explicites accompagnées de libellés pour les utilisateurs non familiers.
- **Navigation intuitive**
  - Navigation structurée autour d'un système de **tabs** (menu inférieur) pour un accès rapide aux sections principales : Accueil, Signalements, Notifications, Profil.
  - Utilisation de **React Navigation avec Expo Router** pour une gestion fluide des transitions.
- **Cohérence graphique**
  - Le style général suit une ligne graphique minimaliste, inspirée des standards iOS/Android (Material + Fluent design).
  - Composants réutilisables et harmonisés (boutons, formulaires, listes).

7. Ressources

Pour mener à bien le développement de l'application **CoHabitat**, plusieurs types de ressources humaines, techniques et logicielles sont nécessaires.

Ressources humaines :

- **Développeur Front-end mobile** : responsable de l'interface utilisateur en React Native (profil : Thomas Spencer).
- **Développeur Back-end** : chargé de concevoir l'API RESTful, la base de données et l'authentification.

Ressources matérielles et logicielles :

- **Ordinateur de développement** (macOS, Windows ou Linux)
- **Émulateur Android Studio** et/ou **simulateur iOS**
- **Outils de développement :**
  - Visual Studio Code
  - Node.js et npm
  - Expo CLI
  - Git et GitHub
- **Hébergement back-end** : plateforme Railway(base gratuite ou payante selon besoin)
- **Base de données** : SQLite (stockage local côté serveur)
- **Bibliothèques principales** :
  - React Native, Expo, Expo Router
  - Express.js, Zod, JWT, Bcrypt, Morgan

Documentation et outils complémentaires :

- Documentation de l'API REST Stoplight
- Tutoriels Expo / Express.js / TypeScript

8. Livrables attendus

Le projet **CoHabitat** donnera lieu à plusieurs livrables essentiels, tant techniques que documentaires. Ces livrables permettront de valider les différentes étapes de conception, de développement, et de mise en production de l'application.

Livrables techniques :

- **Application mobile fonctionnelle** (APK pour Android, testable sur iOS via Expo Go)
- **Code source complet du front-end** (React Native / Expo) versionné sur GitHub
- **Code source complet du back-end** (API Express.js) versionné et documenté
- **Base de données SQLite** structurée et pré-remplie pour la démo

Livrables de test :

- **Jeux de données fictifs** (exemples de bâtiments, incidents, utilisateurs)
- **Suite de tests automatisés** (Jest + Supertest) avec taux de couverture
- **Rapport de tests unitaires et d'intégration**
- **Recette fonctionnelle** (liste de vérification des fonctionnalités livrées)

Livrables documentaires :

- **Cahier des charges complet** (ce document)
- **Guide de déploiement de l'API** (sur Railway ou serveur local)
- **Guide d'installation et de démarrage de l'application mobile**
- **Documentation technique du code source** (README + commentaires)
- **Tutoriel d'utilisation simplifié** pour les gardiens et les locataires

Livrables annex :

- Vidéo de démonstration du fonctionnement de l'application
- Maquettes d'écrans / prototypes (Figma)

9. Délais (Planification)

Étape	Description	Échéance estimée
Spécification fonctionnelle	Rédaction du cahier des charges	15h
Conception UI/UX	Maquettes / wireframes	17h
Développement	Implémentation des fonctionnalités	20h
Tests & corrections	Phase de test utilisateur	10h
Livraison finale	Livraison de l’application	10h

10. Critères de validation

- Interface cohérente
- Fonctionnalité de signalement d'incident opérationnelle
- Navigation stable
- Application fluide
- Respect des bonnes pratiques de code

11. Architecture Front-end

L'interface mobile de **CoHabitat** repose sur **React Native avec Expo** et suit une architecture modulaire par domaine fonctionnel, facilitant la lisibilité, la maintenance et l'évolutivité du code.

```
/app
├── (auth)/                → Gestion de l'authentification
|   ├── login.tsx
|   ├── register.tsx
|   ├── gardian-login.tsx
|   ├── gardian-register.tsx
|   ├── forgot-password.tsx
|   └── splash.tsx

├── (signalements)/        → Création, suivi et gestion des incidents
|   ├── signalement.tsx
|   ├── incidents.tsx
|   ├── gerer-incidents.tsx
|   └── suivresignal.tsx

├── (batiments)/           → Informations sur les bâtiments
|   ├── batiments.tsx
|   ├── mon-batiment.tsx
|   ├── mon-gardien.tsx

├── (profil)/              → Paramètres et fiche utilisateur
|   ├── profil.tsx
|   ├── parametres.tsx

├── (notifications)/       → Affichage des notifications
|   └── notifications.tsx

├── (accueil)/             → Pages d'accueil et tableau de bord
|   ├── home.tsx
|   └── accueil.tsx

├── _layout.tsx            → Layout global pour la navigation
└── +not-found.tsx         → Page d'erreur personnalisée
```

12. Architecture Back-end

```
Backend/
|
├── node_modules/               # Modules npm installés
├── cohabiat.db                 # Fichier de base de données SQLite
├── package.json                # Dépendances et scripts du projet
├── package-lock.json           # Verrouillage des versions de dépendances
├── nodemon.json                # Config de Nodemon pour le hot-reload
├── server.js                   # Point d'entrée de l'application (serveur HTTP)
├── app.js                      # Configuration globale de l'app (middlewares, routes)
|
├── src/
|   ├── controllers/            # Logique métier (ex: createUser, getProfile)
|   |   └── user.controller.js
|   |
|   ├── db/                     # Connexion à la base de données
|   |   └── database.js
|   |
|   ├── middleware/             # Middleware custom (auth, logger, error handler)
|   |   └── auth.middleware.js
|   |
|   ├── models/                 # Définition des schémas ou classes métiers
|   |   └── user.model.js
|   |
|   └── routes/                 # Définition des routes (API REST)
|       └── user.routes.js
|
```

13. Sécurité et RGPD

La sécurité est un enjeu central du projet **CoHabitat**, en particulier en raison de la gestion d'informations personnelles et de transactions financières. Le backend intègre plusieurs mesures de sécurité :

- Chiffrement des mots de passe avec la bibliothèque Bcrypt avant stockage en base de données.
- Authentification par tokens JWT pour sécuriser les connexions et les échanges entre client et serveur.
- Middleware d'autorisation pour restreindre l'accès aux routes en fonction du rôle de l'utilisateur (Gardiens / Locataires).
- Mise à jour régulière des dépendances pour éviter les failles connues dans les packages tiers

Le projet Cohabitat respectera aussi les principes du RGPD pour garantir la confidentialité et la protection des données des utilisateurs européens :

- Consentement explicite lors de l'inscription (CGU, politique de confidentialité).
- Droit d'accès aux données personnelles via l'espace utilisateur.
- Droit de modification et de suppression des informations personnelles (profil, avis).
- Suppression complète du compte utilisateur à la demande.
- Minimisation des données collectées : seules les données nécessaires sont enregistrées (email, nom, mot de passe, numero de telephone).
- Stockage sécurisé dans une base de données relationnelle protégée.

14. Critère de réussite MVP

Pour que le Minimum Viable Product (MVP) du projet Cohabitat soit considéré comme abouti et validé, les critères suivants doivent être remplis à l'issue du dernier sprint :

14.1 Fonctionnalités

- Toutes les routes de l'API prévues dans les spécifications fonctionnelles sont implémentées et opérationnelles (gestion des incidents, bâtiments, notifications, utilisateurs, etc.).
- Un locataire peut signaler un incident, suivre son évolution, consulter l'historique de ses signalements et recevoir des notifications.
- Un gardien peut consulter la liste des incidents à traiter, mettre à jour le statut des incidents, ajouter des commentaires de suivi, et recevoir des notifications.
- L'administrateur peut gérer l'ensemble des bâtiments, des utilisateurs (locataires et gardiens), et superviser les incidents signalés.
- Le système de connexion/inscription avec sécurisation (hashage des mots de passe, vérification des identifiants) est fonctionnel pour tous les types d'utilisateurs.
- Les rôles (gardien / locataire) sont bien gérés côté authentification et autorisation, avec des accès différenciés selon le profil.
- Le suivi des statuts d'incidents est tracé et historisé, permettant de consulter les changements de statut et les commentaires associés.

14.2 Techniques

- Le code respecte les bonnes pratiques de structuration et de lisibilité.
- Aucune faille de sécurité évidente (injection SQL, accès non autorisé, mot de passe non hashé…).
- Le projet est versionné proprement sur GitHub avec un historique de commits clair.

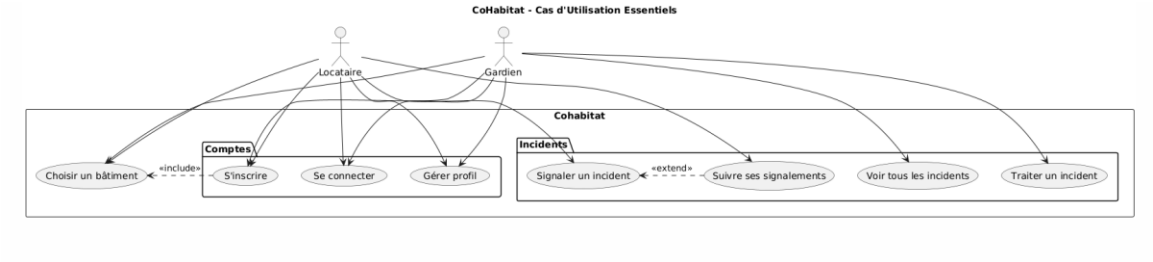
14.3 Déploiements

- Le backend est déployé sur une plateforme comme Railway.
- Une démonstration de l'API (via Postman ou un front de test) est réalisable.

14.4 Documentations

- Une documentation technique complète est disponible (README, instructions d'installation).
- Un guide d'utilisation de l'API est rédigé pour les développeurs mobiles.

15. Les Diagrammes


CoHabitat - Cas d'Utilisation Essentiels
Locataire
Gardien
Cohabitat
Comptes
Incidents
Choisir un bâtiment
«include»
S'inscrire
Se connecter
Gérer profil
Signaler un incident
«extend»
Suivre ses signalements
Voir tous les incidents
Traiter un incident