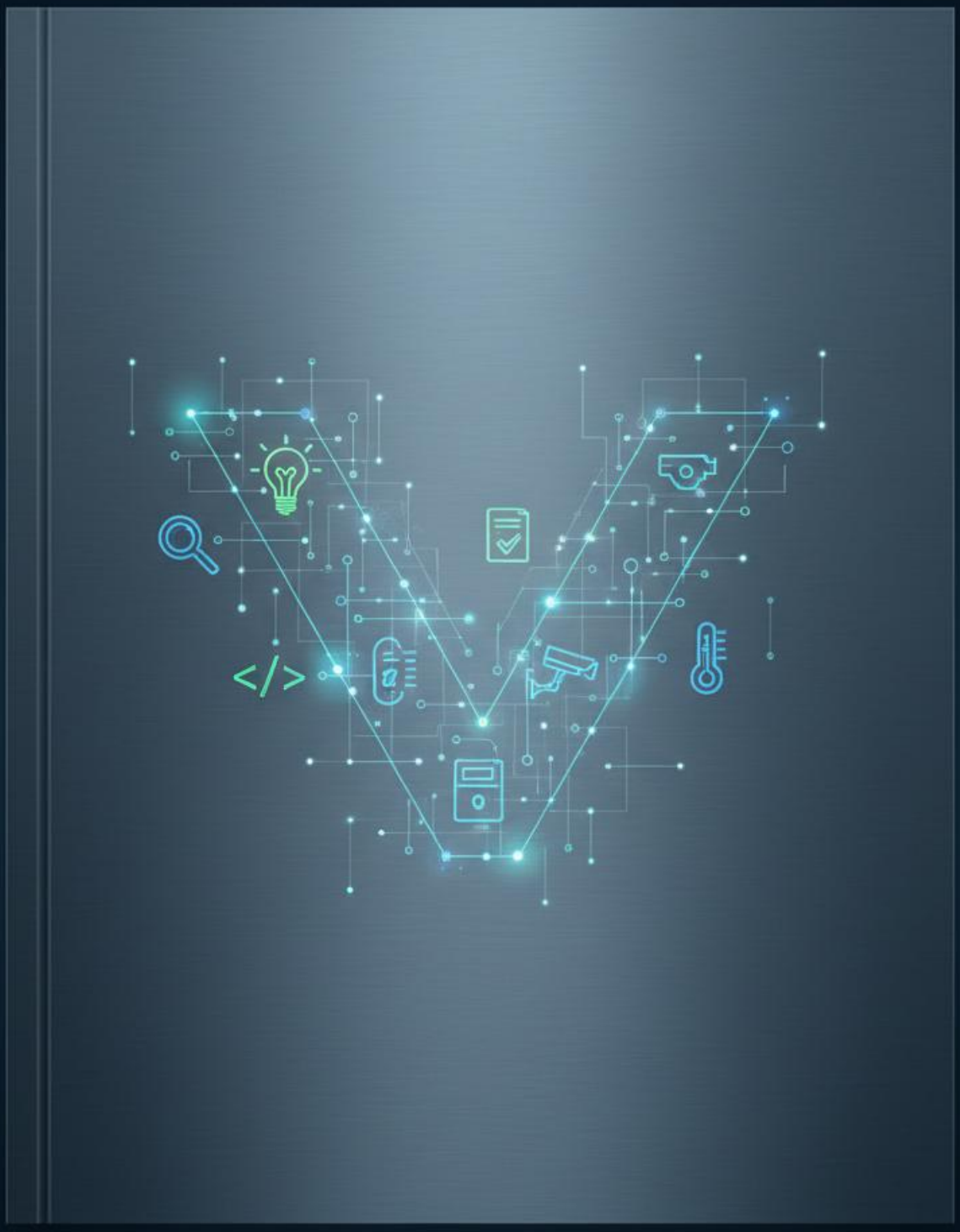
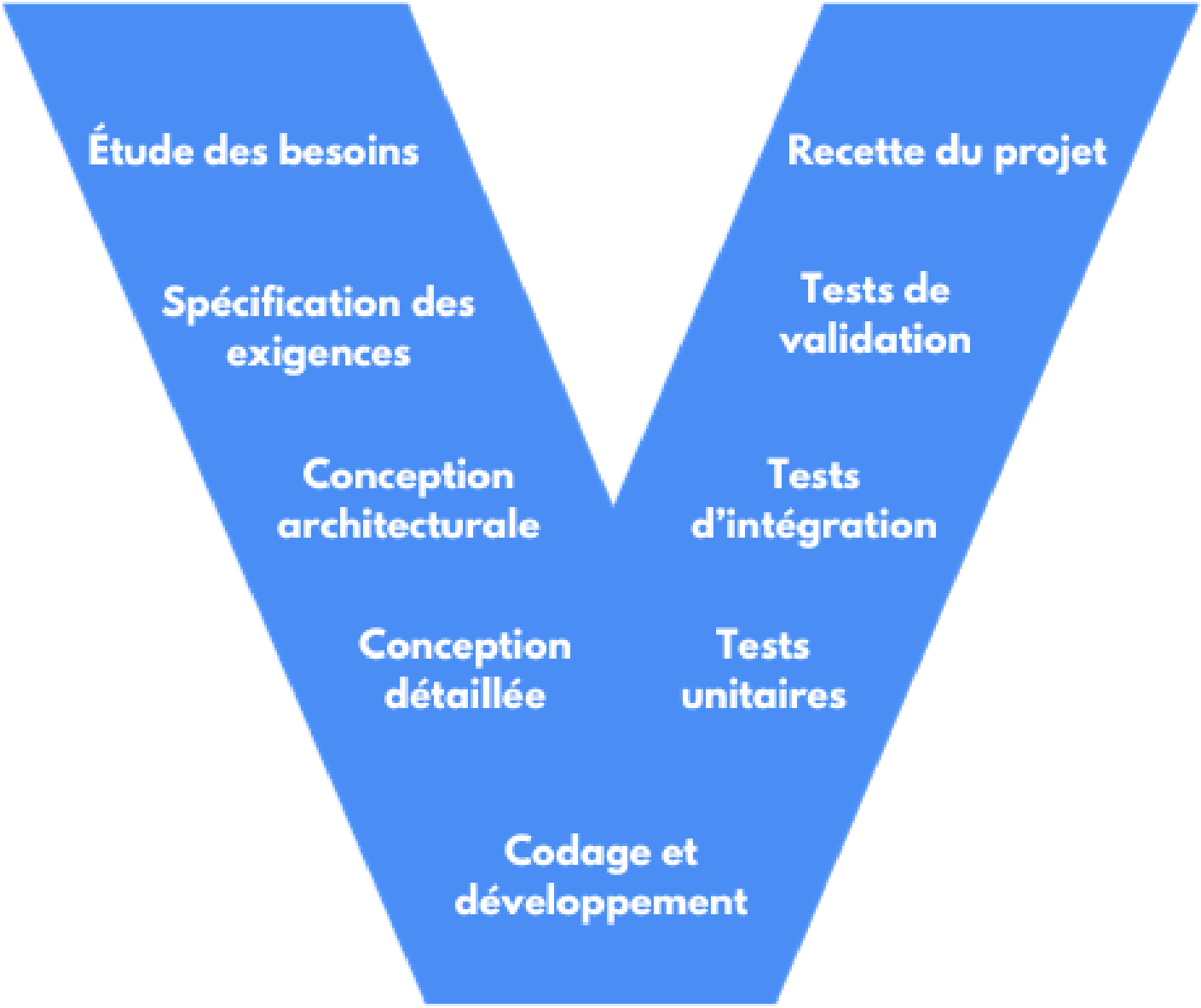


PLAN DE MISE EN ŒUVRE PROJET MAISON CONNECTÉE

PLANIFICATION DES PHASES, LIVRABLES ET TESTS POUR LE PROJET

KONSTANTINE GAROZASHVILI

LAPLATEFORME.IO



INTRODUCTION

Suite à l'approbation de l'utilisation du Cycle en V, ce document détaille le plan d'action pour chaque phase du projet de développement du système de maison connectée.

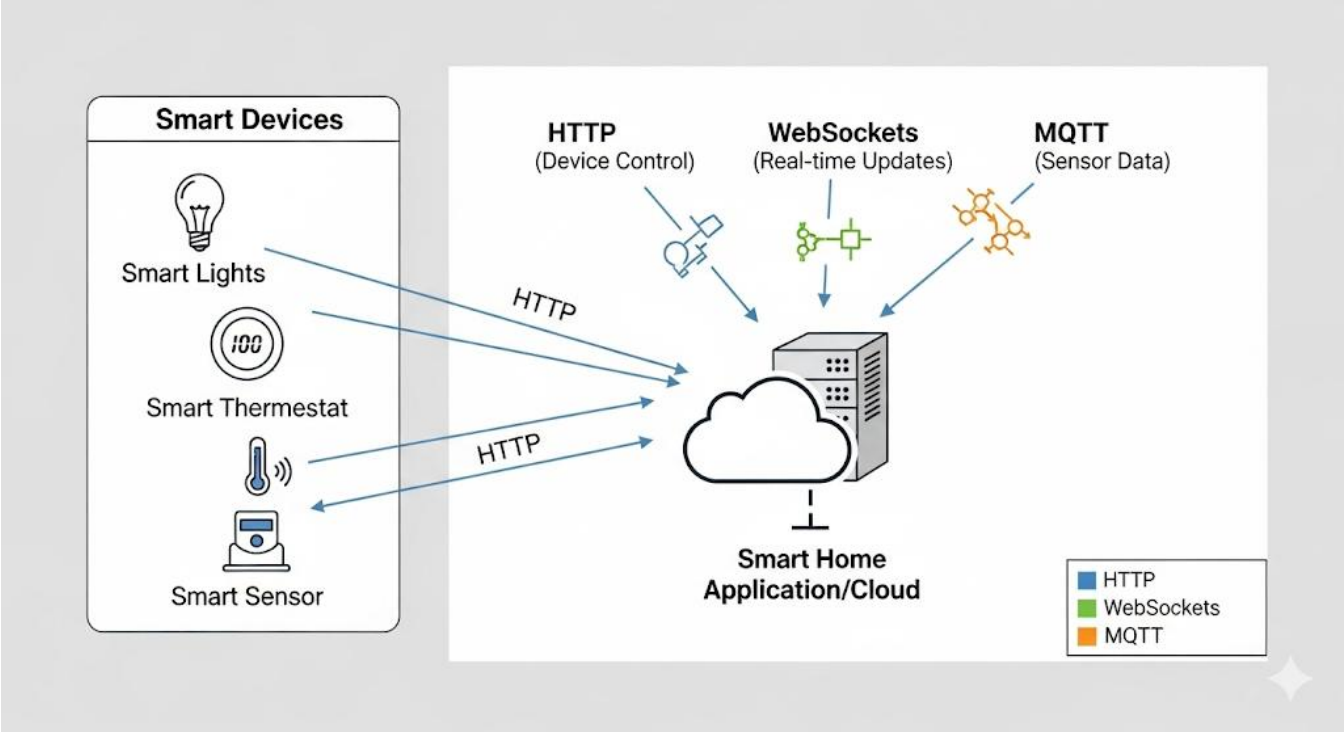
PROTOCOLES DE COMMUNICATION

Pour un système de maison connectée, le choix des protocoles est primordial pour assurer la fiabilité et la performance. Nous avons opté pour une approche hybride pour couvrir l'ensemble des besoins :

- **HTTPS pour les requêtes standards** : Ce protocole est utilisé pour les communications classiques telles que l'authentification **(/auth/login)** et la récupération initiale des données de configuration. Il est fiable et bien établi pour les échanges de type requête-réponse. Nous l'utilisons également pour d'autres points d'accès essentiels, comme :

- **GET /api/devices** : Pour récupérer la liste des appareils connectés.
- **POST /api/devices/:id/toggle** : Pour allumer ou éteindre un appareil.
- **POST /api/devices/:id/climate** : Pour contrôler le thermostat ou le climatiseur.

- **WebSockets pour la communication en temps réel** : Pour que l'application puisse afficher l'état des appareils de manière instantanée, nous utiliserons les WebSockets. Cela permet une connexion bidirectionnelle constante entre le client et le serveur, évitant les requêtes répétitives.
- **MQTT (Message Queuing Telemetry Transport)** : Ce protocole de messagerie léger est conçu pour l'IoT. Les appareils l'utiliseront pour publier leur état (ex: "température: 21°C") à un "broker" MQTT, qui se chargera de distribuer l'information à l'application.



Dans une approche pédagogique, HTTP sert de base pour comprendre la communication client-serveur. Cependant, il transmet des données non chiffrées, le rendant vulnérable.

HTTPS est la seule option sécurisée et obligatoire pour tout système réel de maison connectée. Il garantit la confidentialité et l'intégrité des données grâce au chiffrement TLS/SSL, protégeant ainsi les informations sensibles de l'utilisateur. En résumé, **HTTP est un concept**, **HTTPS est le protocole de production**.

DÉTAIL DU PLAN D'ACTION

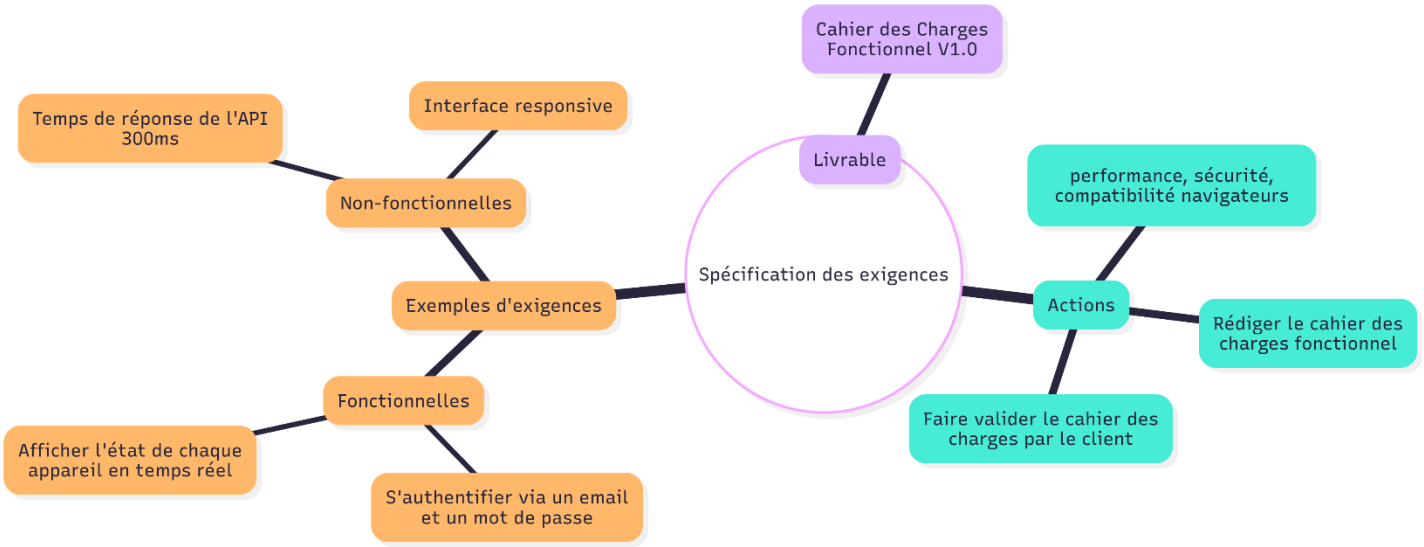
L'OBJECTIF EST DE TRADUIRE LES BESOINS EN EXIGENCES CLAIRES ET VÉRIFIABLES

ÉTUDE DES BESOINS

• **Actions**: Organiser des ateliers avec le client pour lister toutes les fonctionnalités souhaitées.
• **Exemples**: L'utilisateur doit pouvoir contrôler toutes les lumières et appareils de sa maison depuis une application web.

SPÉCIFICATION DES EXIGENCES

• **Actions**: Définir les exigences non-fonctionnelles (performance, sécurité, compatibilité navigateurs). Rédiger le cahier des charges fonctionnel et le faire valider par le client.
• **Livrable**: Cahier des Charges Fonctionnel V1.0



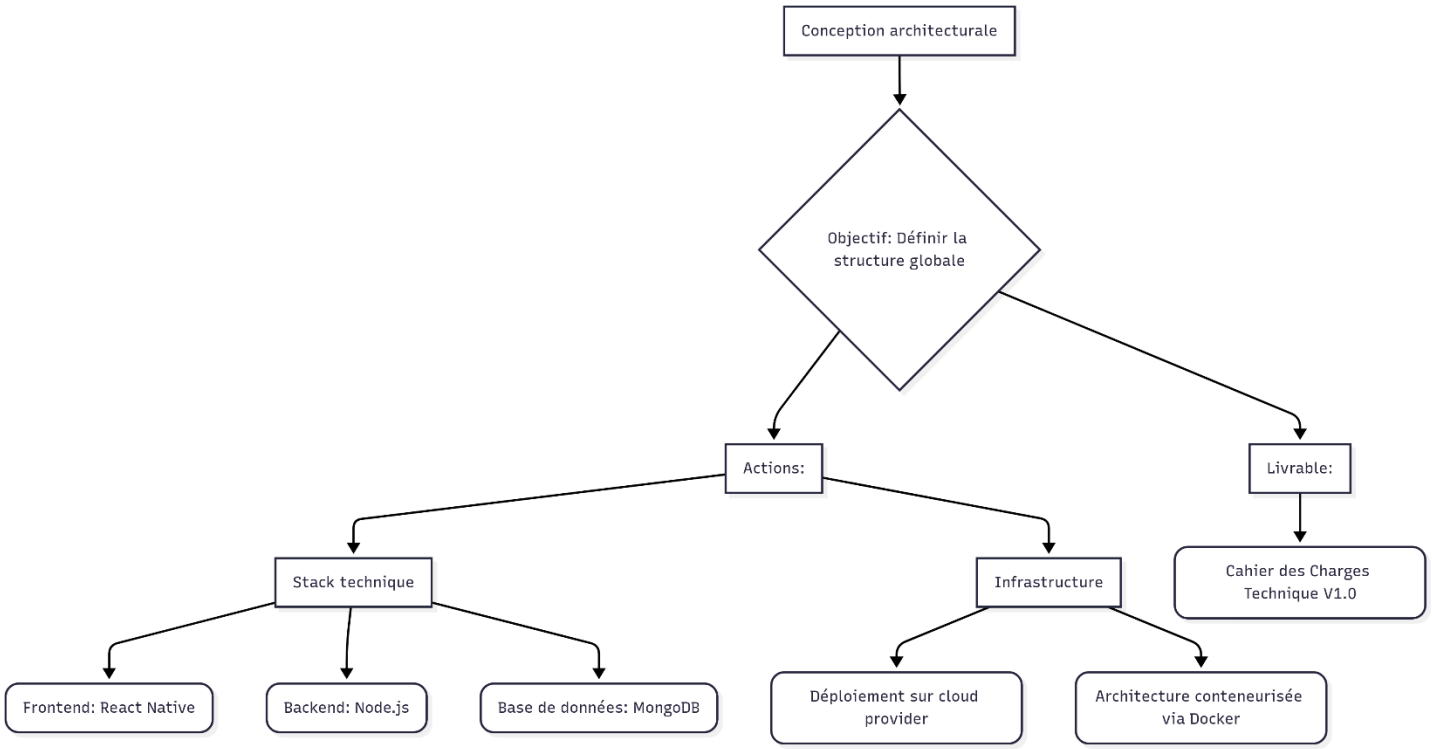
- **Fonctionnelles**: L'utilisateur peut s'authentifier via un email et un mot de passe ; l'application doit afficher l'état de chaque appareil en temps reel et modifier les fonctionnalité des appareil par example : lights turn off/on et regle la temperature de choffage .

- **Non-fonctionnelles**: Le temps de réponse de l'API doit être inférieur à 300ms ; l'interface doit être responsive.

CONCEPTION ARCHITECTURALE

L'OBJECTIF EST DE DÉFINIR LA STRUCTURE TECHNIQUE GLOBALE DU SYSTÈME.

• **Actions**: Définition de la stack technique (Frontend : React Native, Backend : Node.js, Base de données : MongoDB), et de l'infrastructure (déploiement sur un cloud provider avec une architecture conteneurisée via Docker).
• **Livrable**: Cahier des Charges Technique V1.0

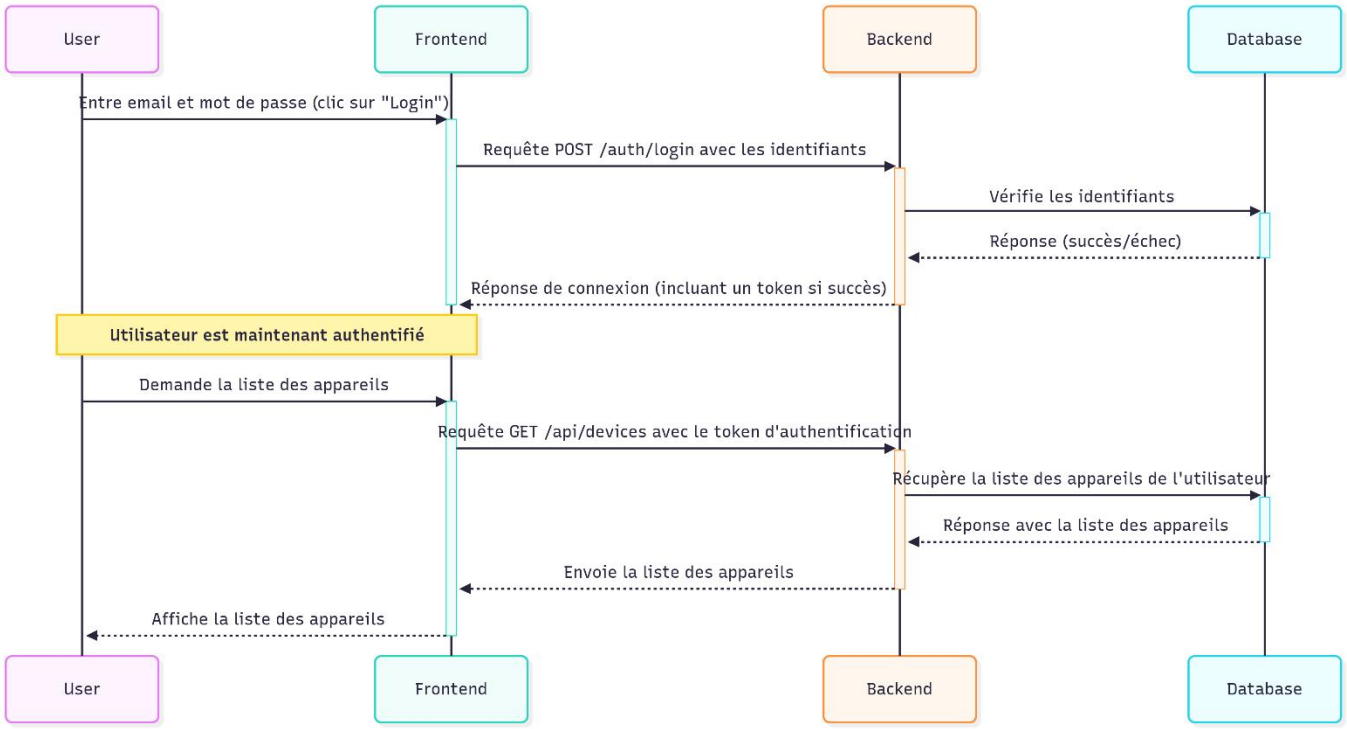


CONCEPTION DÉTAILLÉE

L'OBJECTIF EST DE SPÉCIFIER LE FONCTIONNEMENT DE CHAQUE COMPOSANT DU SYSTÈME.

• **Actions:** Définition des routes de l'API Backend (ex : **POST /auth/login, GET /api/devices . POST /api/devices/:id/toggle .etc**). Découpage de l'interface Frontend en composants réutilisables (<login>. <dashboard>).

• **Livrable:** Documentation de l'API

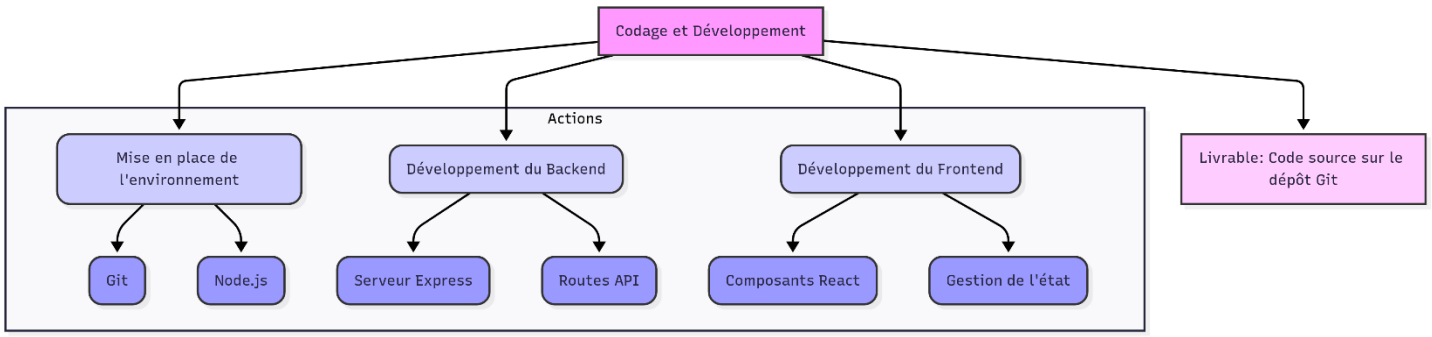


CODAGE ET DÉVELOPPEMENT

L'OBJECTIF EST DE DÉVELOPPER LE CODE SOURCE DU PROJET.

• **Actions:** Mise en place de l'environnement de développement (Git, Node.js). Développement du backend (serveur Express, routes API) et du frontend (création des composants React, gestion de l'état).

• **Livrable:** Code source sur le dépôt Git



TESTS UNITAIRES

L'OBJECTIF EST DE VÉRIFIER LE BON FONCTIONNEMENT DE CHAQUE COMPOSANT D E MANIÈRE ISOLÉE.

• **Actions:** Tester chaque fonction du backend avec Jest. Tester chaque composant React avec React Testing Library pour vérifier son rendu et ses interactions de base.
• **Livrable:** Rapport de couverture des tests unitaires.

```
// Test du composant Login
it('devrait afficher un message de succès après connexion', () => {
  render(<Login />);

  // Simule l'entrée utilisateur
  fireEvent.change(screen.getByLabelText('Email'), { target: { value: 'test@example.com' } });
  fireEvent.change(screen.getByLabelText('Mot de passe'), { target: { value: 'password123' } });

  // Simule le clic
  fireEvent.click(screen.getByRole('button', { name: 'Se connecter' }));

  // Vérifie le résultat
  expect(screen.getByText('Connexion réussie !')).toBeInTheDocument();
});
```

```
// Test de la fonction de récupération des appareils
it('devrait retourner la liste des appareils pour un utilisateur authentifié', () => {
  const user = { id: 123 }; // Utilisateur authentifié
  const response = getDevices(user);
  expect(response.status).toBe(200);
  expect(response.devices).toBeInstanceOf(Array);
});
```

TESTS D'INTÉGRATION

L'OBJECTIF EST DE VÉRIFIER QUE LES DIFFÉRENTS COMPOSANTS DU SYSTÈME COMMUNIQUENT CORRECTEMENT ENTRE EUX.

• **Actions:** Vérifier que le frontend et le backend communiquent correctement. Simuler un clic sur un bouton dans l'interface pour vérifier que l'appel API est bien reçu par le backend.
• **Livrable:** Plan de tests V1.0

TESTS DE VALIDATION

L'OBJECTIF EST DE VÉRIFIER QUE L'APPLICATION COMPLÈTE RÉPOND BIEN AUX EXIGENCES FONCTIONNELLES DÉFINIES AU DÉBUT DU PROJET.

• Actions : Reprendre le cahier des charges fonctionnel et créer un plan de test pour chaque exigence (F-01, F-02, etc.). Le testeur suit un script pour vérifier le comportement de l'application.
• Livrable : Rapports d'exécution des tests.