



# Agrishield\_\_app\_conception

*cahier de conception de l'application agrishield*

Jean Marc Wogue  
“check it out ”

---

<b>Introduction.....</b>	<b>2</b>
1. Contexte du Projet AgriShield.....	2
2. Problématique et Besoin Logiciel.....	2
3. Approche Architecturale : Le modèle "Offline-First".....	3
<b>1. Application Mobile (Client Riche - Flutter).....</b>	<b>3</b>
1.1 Objectifs.....	3
1.2 Besoins Fonctionnels.....	4
A. Module d'Interface IoT (Acquisition Local ).....	4
B. Module Dashboard & Visualisation (Traitement Local).....	4
C. Module de Persistance (Stockage Local).....	5
D. Module de Synchronisation (Sync Manager).....	5
E. Module de Configuration (Mode Technicien).....	6
1.3 Besoins Non Fonctionnels.....	6
<b>2. Application Backend (Cerveau Central - Java).....</b>	<b>7</b>
2.1 Objectifs.....	7
2.2 Besoins Fonctionnels.....	7
A. Service de Synchronisation de Données (API Gateway).....	8
B. Moteur d'Intelligence Artificielle (Architecture RAG).....	8
C. Moteur d'Analyse Prédictive (Background Workers).....	9
D. Service Météorologique (Proxy).....	9
E. Gestion des Profils de Culture.....	9
2.3 Besoins Non Fonctionnels.....	10
<b>3. Proposition d'Endpoints (API Backend).....</b>	<b>10</b>
3.1 Gestion de l'Authentification.....	10
3.2 Synchronisation des Données (Sync Up).....	11
3.3 Services Intelligents & Météo (Sync Down).....	12
3.4 Gestion des Configurations (IoT).....	13
<b>4. Détails des Pages de l'Application.....</b>	<b>14</b>
Page 1 : Accueil & Synthèse (Home Dashboard).....	14
Page 2 : Assistant de Connexion IoT (Connection Wizard).....	15
Page 3 : Analyse & Historique (Data Analytics).....	15
Page 4 : Assistant Agronome (Chat IA - Mode Online).....	16
Page 5 : Configuration & Profils (Mode Technicien).....	16
<b>Conclusion.....</b>	<b>17</b>
1. Une complémentarité stratégique.....	17
2. Une réponse à la fracture numérique.....	18

# Introduction

## 1. Contexte du Projet AgriShield

Le projet **AgriShield** est une réponse technologique au défi de la sécurité alimentaire et de la résilience climatique dans les zones rurales. Il vise à équiper les petits agriculteurs, qui perdent actuellement jusqu'à 30 à 40 % de leurs récoltes en raison de maladies détectées trop tardivement, d'un système de surveillance autonome et abordable.

Le cœur du système repose sur un boîtier IoT (Internet of Things) alimenté par énergie solaire, capable de mesurer les paramètres climatiques et physiologiques critiques (température, humidité de l'air et du sol). Grâce à un traitement local ("Edge Computing"), ce dispositif alerte immédiatement l'agriculteur via des signaux visuels et sonores en cas de risque avéré pour la culture, sans dépendre d'une connexion Internet ou GSM.

## 2. Problématique et Besoin Logiciel

Bien que le module AgriShield soit conçu pour fonctionner en totale autonomie pour la protection immédiate, il présente des limitations inhérentes à son absence de connectivité permanente :

- **Perte de données historiques** : La mémoire locale du module est limitée (stockage circulaire) et ne permet pas une analyse sur toute une saison.
- **Absence de prédictivité avancée** : Le module réagit à l'instant présent (réactif), mais ne peut pas anticiper les risques futurs basés sur des modèles complexes ou la météo.
- **Isolement numérique** : L'agriculteur en zone rurale ("zone blanche") ne peut pas bénéficier des outils d'Intelligence Artificielle modernes directement sur le champ.

C'est pour combler ces lacunes que le volet logiciel (Application Mobile et Backend)

---

intervient. L'objectif n'est pas de contrôler le module en temps réel, mais d'agir comme une extension intelligente capable de collecter, archiver et analyser les données.

### 3. Approche Architecturale : Le modèle "Offline-First"

La contrainte majeure de ce projet est l'absence de réseau Internet fiable sur les parcelles agricoles. En conséquence, l'architecture logicielle repose sur un modèle hybride strict défini par les spécifications techniques :

1. **Au Champ (Mode Déconnecté)** : L'application mobile agit comme un terminal de collecte ("Data Harvester"). Elle communique avec le module via un réseau WiFi local (SoftAP) pour extraire les données brutes, servant de passerelle temporaire.
2. **À Distance (Mode Connecté)** : Une fois la connectivité Internet retrouvée, l'application synchronise les données avec un Backend centralisé. C'est à ce niveau que l'intelligence déportée (IA générative, analyse prédictive, météo) est activée pour fournir des conseils agronomiques à haute valeur ajoutée.

Ce document détaille la conception technique de cette double architecture logicielle : une application mobile **Flutter** robuste pour le terrain et un backend **Java** puissant pour l'intelligence, reliés par un contrat de données strict.

# 1. Application Mobile (Client Riche - Flutter)

## 1.1 Objectifs

L'application mobile AgriShield est conçue comme un terminal d'extension pour le module IoT. Contrairement à une application IoT classique qui pilote un appareil en temps réel via Internet, celle-ci répond à une contrainte d'isolation numérique ("Offline-First").

Ses objectifs principaux sont :

1. **Agir comme une "passerelle de collecte" (Data Harvester)** : Récupérer physiquement les données du module via un réseau local temporaire, palliant

- l'absence de GSM/Internet sur le module.
2. **Offrir une visualisation immédiate** : Permettre à l'agriculteur de consulter l'état de ses cultures et les graphiques historiques "au pied de la plante", sans attendre une synchronisation cloud.
  3. **Assurer la continuité du service** : Garantir l'accès aux données historiques et aux prévisions météorologiques (préalablement téléchargées)

## 1.2 Besoins Fonctionnels

L'application est structurée en modules fonctionnels distincts pour gérer le cycle de vie de la donnée, de son acquisition locale à sa synchronisation.

### A. Module d'Interface IoT (Acquisition Local)

Ce module gère la communication directe et éphémère avec le firmware ESP32. Il respecte strictement le contrat d'interface défini dans la spécification V1.

- **Gestion de la Connexion WiFi (SoftAP) :**
  - L'application doit scanner les réseaux WiFi environnants pour identifier le SSID normalisé **AgriShield\_XXXX**.
  - Elle doit guider l'utilisateur pour basculer son téléphone sur ce réseau (IP Gateway : **192.168.4.1**).
- **Vérification de Santé (Handshake) :**
  - Avant tout échange de données, l'application doit interroger l'endpoint **GET /health** pour confirmer le statut opérationnel du module (**status : OK**).
- **Lecture de l'État Instantané :**
  - L'application appelle **GET /status** pour afficher les valeurs temps réel (T°, Humidité, Batterie) et le niveau d'alerte actuel sans nécessiter de téléchargement complet.
- **Téléchargement de l'Historique (Source de Vérité) :**
  - L'application télécharge le fichier JSON complet via **GET /data/log**. Ce fichier est considéré comme la source de vérité unique et contient l'intégralité des enregistrements stockés par le module.

### B. Module Dashboard & Visualisation (Traitement Local)

Une fois les données récupérées, l'application agit comme un "Client Lourd" pour traiter

et afficher les informations sans aide du serveur.

- **Parsing et Intégrité des Données :**
  - L'application doit parser le JSON reçu en respectant le mappage strict des champs (`ts`, `t`, `h`, `soil`, `bat`, `alert`) défini par le firmware.
  - Elle ne doit jamais modifier les données brutes issues du capteur.
- **Affichage des Alertes (Logique Passive) :**
  - L'application affiche le niveau de risque (`GREEN`, `ORANGE`, `RED`) tel qu'évalué par le firmware.
  - **Contrainte critique :** L'application ne doit pas recalculer les alertes elle-même pour éviter toute divergence avec les LED du boîtier. Elle se contente de traduire le code d'alerte en interface visuelle (ex: Bannière Rouge pour "RED").
- **Visualisation Graphique :**
  - Génération de courbes interactives pour la température de l'air, l'humidité (air/sol) et la tension batterie.
  - Indication visuelle de l'état de charge solaire (`solar_charging: true`).

## C. Module de Persistance (Stockage Local)

Pour garantir l'approche "Offline-First", aucune donnée ne doit être perdue si l'application est fermée ou si le téléphone redémarre.

- **Base de Données Embarquée :**
  - Intégration d'une base de données locale performante (ex: SQLite ou Hive).
- **Gestion des Doubloons :**
  - Lors de l'importation du JSON `DataLog`, l'application doit utiliser le champ `timestamp` comme clé unique pour ne pas enregistrer deux fois la même mesure lors de synchronisations successives.

## D. Module de Synchronisation (Sync Manager)

Ce module assure le pont entre le mode "Terrain" et le mode "Connecté".

- **Détection de Connectivité :**
  - Un service d'arrière-plan surveille le retour d'une connexion Internet (4G ou

WiFi domestique).

- **Sync Up (Montant) :**
  - Envoi automatique des nouvelles mesures non synchronisées vers le Backend Java via `POST /api/measurements`.
- **Sync Down (Descendant - Météo & IA) :**
  - Téléchargement des prévisions météo locales (J+3) et des rapports d'analyse IA générés par le serveur.
  - Ces données sont stockées en cache local pour être consultables lors de la prochaine visite au champ sans internet.

## E. Module de Configuration (Mode Technicien)

Fonctionnalité restreinte permettant de modifier le comportement du module IoT.

- **Envoi de Profil de Culture :**
  - Interface permettant de sélectionner un profil (ex: "Tomato\_Nursery") et de l'envoyer au module via `POST /config`.
  - Ce profil met à jour les seuils (`humidity_critical`, `temperature_max`) et la fréquence d'échantillonnage.

## 1.3 Besoins Non Fonctionnels

- **Expérience Utilisateur (UX) - Gestion du SoftAP :**
  - La connexion au point d'accès du module (qui coupe internet sur le téléphone) est une friction majeure. L'application doit guider l'utilisateur pas à pas et gérer la reconnexion automatique au réseau mobile une fois le transfert terminé.
- **Robustesse aux interruptions :**
  - Le téléchargement du log (`/data/log`) doit gérer les interruptions (timeout, perte de signal WiFi) sans corrompre la base de données locale.
- **Compatibilité Ascendante :**
  - Le format JSON est versionné (V1). L'application doit vérifier le champ `firmware_version` dans le JSON pour s'assurer qu'elle peut lire les données.
- **Efficacité Énergétique :**
  - L'application ne doit pas scanner le WiFi en permanence en arrière-plan

---

pour préserver la batterie du smartphone de l'agriculteur.

---

Voici la rédaction détaillée de la **Partie 2 : Application Backend**, structurée selon notre plan de conception validé et intégrant l'architecture **IA/RAG** (Retrieval-Augmented Generation) demandée.

---

## 2. Application Backend (Cerveau Central - Java)

### 2.1 Objectifs

Alors que le module IoT gère l'urgence au champ et que l'application mobile assure la collecte locale, le Backend (développé en Java/Spring Boot) agit comme le **cerveau analytique** du système. Il s'active uniquement lorsque l'utilisateur dispose d'une connexion Internet.

Ses objectifs stratégiques sont :

1. **Centralisation et Sécurisation** : Constituer un entrepôt de données ("Data Warehouse") pérenne regroupant l'historique de toutes les parcelles et utilisateurs, au-delà de la capacité de stockage limitée du téléphone ou du boîtier.
2. **Intelligence Déportée (IA)** : Fournir des capacités d'analyse avancée inaccessibles au processeur du module IoT, notamment via un moteur d'Intelligence Artificielle générative pour le conseil agronomique.
3. **Intégration de Services Tiers** : Agir comme une passerelle unique vers les API externes, principalement pour l'acquisition des prévisions météorologiques locales.

### 2.2 Besoins Fonctionnels

Le backend est architecturé autour d'une API RESTful qui expose des services aux

---

applications mobiles authentifiées.

## A. Service de Synchronisation de Données (API Gateway)

Ce module est le point d'entrée unique pour les données remontées du terrain.

- **Réception des Mesures (Sync Up) :**
  - Exposition d'un endpoint `POST /api/measurements` acceptant des lots (batchs) d'enregistrements JSON.
  - **Validation stricte** : Le backend doit valider la conformité des données reçues avec le schéma JSON "Source de Vérité" défini dans la spécification IoT (champs `ts, t, h, soil, bat, alert`).
  - **Dédoubleillage** : Vérification de l'unicité des enregistrements basée sur le couple `device_id + timestamp` pour éviter de corrompre l'historique lors de synchronisations multiples.
- **Envoi des Contextes (Sync Down) :**
  - Envoi des données descendantes vers l'application mobile : prévisions météo mises à jour et nouveaux profils de culture disponibles.

## B. Moteur d'Intelligence Artificielle (Architecture RAG)

C'est le cœur de la valeur ajoutée "Online". Il utilise une approche **RAG (Retrieval-Augmented Generation)** pour fournir des réponses fiables basées sur les données agronomiques du projet, et non sur des connaissances génériques.

- **Base de Connaissances Vectorielle :**
  - Le système doit indexer et vectoriser les documents techniques agronomiques (Source 1 à 7) contenant les symptômes des maladies (Mildiou, Oïdium, Flétrissement bactérien) et les méthodes de détection.
- **Contextualisation Dynamique :**
  - Lorsqu'un utilisateur pose une question via l'endpoint `POST /api/chat`, le système doit injecter automatiquement dans le "prompt" les **dernières mesures** de cet utilisateur (ex: "Humidité sol: 85%", "Alerte récente: WARNING").
- **Génération de Réponses :**
  - L'IA doit générer une réponse qui croise la question de l'utilisateur, l'état réel de sa plante (capteurs) et la connaissance scientifique (base

vectorielle).

- *Exemple :* Si l'utilisateur demande "Dois-je arroser ?", l'IA vérifie l'humidité du sol en base et répond "Non, votre sol est à 60% d'humidité, ce qui est suffisant. Un excès d'eau risquerait de favoriser le flétrissement bactérien."

## C. Moteur d'Analyse Prédictive (Background Workers)

Ce module travaille en tâche de fond, même si l'utilisateur n'est pas actif sur l'application.

- **Détection de Tendances à Risque :**

- Analyse des séries temporelles pour identifier des motifs prolongés favorables aux maladies, définis dans les sources agronomiques.
- *Règle métier (Exemple) :* Si **Température < 20°C ET Humidité > 80%** pendant 48h → Générer une notification de risque de "Mildiou (Downy Mildew)".

- **Génération de Notifications Push :**

- Envoi d'alertes proactives sur le mobile pour inciter l'utilisateur à aller vérifier son champ : "Risque Mildiou élevé détecté. Inspectez le dessous des feuilles".

## D. Service Météorologique (Proxy)

- **Agrégation Météo :**

- Connexion à une API tierce (type OpenWeatherMap) pour récupérer les prévisions à J+3 correspondant aux coordonnées GPS des modules installés.
- Stockage de ces prévisions en base pour les servir à l'application mobile lors de la prochaine synchronisation (permettant la consultation offline ultérieure).

## E. Gestion des Profils de Culture

- **Référentiel Centralisé :**

- Stockage et versioning des profils de configuration JSON (ex: **Tomato\_Nursery, Pepper\_OpenField**).
- Permettre aux administrateurs de créer de nouveaux profils (nouveaux

---

seuils d'alerte) qui seront redescendus vers les applications mobiles puis vers les modules IoT.

## 2.3 Besoins Non Fonctionnels

- **Scalabilité Multi-Tenant :**
  - Le système doit supporter une architecture multi-utilisateurs (agriculteurs indépendants) et multi-organisations (coopératives), assurant que les données de chaque entité sont cloisonnées.
- **Sécurité et Authentification :**
  - Mise en œuvre d'un protocole sécurisé (OAuth2 / JWT) pour l'API. Seules les applications mobiles authentifiées peuvent envoyer des données vers le Cloud.
- **Haute Disponibilité :**
  - Le service doit être disponible 24/7 pour permettre la synchronisation à tout moment, dès que l'agriculteur trouve une zone de couverture réseau.

## 3. Proposition d'Endpoints (API Backend)

Cette section détaille l'interface de programmation (API REST) exposée par le serveur central. Ces endpoints sont consommés par l'application mobile uniquement lorsqu'une connexion Internet est disponible (Mode Online).

L'architecture respecte les standards RESTful et utilise le format JSON pour les échanges de données, en cohérence avec la structure de données définie par le module IoT.

### 3.1 Gestion de l'Authentification

*Sécurisation de l'accès aux données des agriculteurs et des coopératives.*

Méthode	Endpoint	Description	Payload (Corps) / Réponse
POST	/api/auth/login	<b>Connexion.</b> Authentifie l'utilisateur (Agriculteur ou Technicien) et délivre un jeton de session sécurisé.	<b>Req:</b> {"username": "...", "password": "..."} <b>Res:</b> {"token": "JWT_XYZ...", "role": "FARMER"}
POST	/api/auth/refresh	<b>Rafraîchissement.</b> Renouvelle le jeton d'accès sans nécessiter une resaisie du mot de passe.	<b>Req:</b> {"refresh_token": "..."}  

## 3.2 Synchronisation des Données (Sync Up)

Réception et archivage des données collectées par l'application mobile sur le terrain.

Méthode	Endpoint	Description	Payload (Corps) / Réponse
POST	/api/measurements	<b>Envoi de Mesures.</b> L'application mobile pousse les lots de données (batch) récupérés depuis le module IoT. Le backend valide l'intégrité et dédouble les enregistrements avant archivage.	<b>Req:</b> Tableau d'objets <b>SensorRecord</b> incluant <b>device_id</b> , <b>timestamp</b> , <b>soil_moisture</b> , etc. <b>Res:</b> {"status": "SYNCED", "count": 45}

<b>POST</b>	<b>/api/alerts</b>	<b>Remontée d'Alertes.</b> Permet de signaler immédiatement une alerte critique (ex: "RED") détectée au champ, pour notification aux superviseurs de la coopérative.	<b>Req:</b> { "device_id": "AS-01", "level": "RED", "reason": "Drought" }
-------------	--------------------	--	---

### 3.3 Services Intelligents & Météo (Sync Down)

*Distribution de la valeur ajoutée (IA, Prévisions) vers le mobile pour mise en cache.*

Méthode	Endpoint	Description	Payload (Corps) / Réponse
<b>GET</b>	<b>/api/weather</b>	<b>Prévisions Météo.</b> Le backend agit comme proxy vers une API externe (ex: OpenWeather). Il fournit les prévisions J+3 localisées pour les parcelles de l'utilisateur.	<b>Res:</b> { "forecast": [ { "day": "Mon", "rain_prob": 80, "temp_max": 28}, ... ] }
<b>POST</b>	<b>/api/chat</b>	<b>Assistant IA (RAG).</b> L'utilisateur pose une question. Le backend injecte le contexte (dernières mesures du sol/air) et interroge la base de connaissances (maladies, symptômes)	<b>Req:</b> { "question": "Mes feuilles jaunissent, pourquoi ?"} <b>Res:</b> { "answer": "Vu l'humidité élevée (85%) relevée hier, cela ressemble à un }

		pour générer une réponse.	<code>début de Mildiou... ", "sources": [ "Doc Mildiou" ]}</code>
GET	<code>/api/analysis/ report</code>	<b>Rapport Hebdomadaire.</b> Récupère un résumé généré par l'IA sur l'état de santé global de la parcelle (Tendances, Risques détectés).	<code>Res: {"status": "RISKY", "advice": "Réduire irrigation", "disease_risk": "High"}</code>

### 3.4 Gestion des Configurations (IoT)

*Distribution des profils de culture pour la mise à jour des modules.*

Méthode	Endpoint	Description	Payload (Corps) / Réponse
GET	<code>/api/profiles</code>	<b>Catalogue de Profils.</b> Récupère la liste des profils de culture disponibles (ex: "Tomato_Nursery", "Pepper_OpenField") avec leurs seuils d'alerte. L'app les télécharge pour pouvoir configurer un module hors-ligne.	<code>Res: [ {"id": "Tomato_Nursery" , "thresholds": {"air_temp_crit": 34, ...} }]</code>
GET	<code>/api/firmware/ latest</code>	<b>Mise à jour.</b> Vérifie si une nouvelle version du firmware ESP32 est	<code>Res: {"version": "1.2.0", "url":</code>

		disponible (pour notification technicien). " . . ." }
--	--	---

## 4. Détails des Pages de l'Application

L'architecture de l'application repose sur **5 écrans principaux**, accessibles via une barre de navigation simplifiée ou un flux guidé. L'interface s'adapte dynamiquement selon que l'utilisateur est connecté au module (Mode IoT) ou à Internet (Mode Cloud).

### Page 1 : Accueil & Synthèse (Home Dashboard)

Cette page est le point d'entrée. Elle fonctionne totalement hors-ligne et affiche le "dernier état connu" de la parcelle.

- **Utilité** : Offrir une vue d'ensemble immédiate de la santé des cultures et de la météo, sans nécessiter d'action technique.
- **Fonctionnalités Clés :**
  1. **Carte "Dernière Mesure"** : Affiche les dernières données synchronisées (ex: "Hier 16h00").
    - Indicateurs : Température Air, Humidité Sol, Niveau Batterie.
    - Code Couleur : Reprend la dernière alerte enregistrée (Vert/Orange/Rouge) issue du champ `alert_level`.
  2. **Widget Météo (Cache J+3)** : Affiche les prévisions météorologiques (Pluie/Soleil) stockées localement lors de la dernière connexion Internet. C'est crucial pour l'anticipation des maladies fongiques.
  3. **Statut de Synchronisation** : Une icône indique si des données locales sont en attente d'envoi vers le Cloud (ex: "45 mesures à synchroniser").
  4. **Bouton d>Action Principal (FAB)** : Un bouton proéminent "**Scanner ma Plante**" qui lance l'Assistant de Connexion (Page 2).

## Page 2 : Assistant de Connexion IoT (Connection Wizard)

C'est la page technique critique qui gère la transition vers le réseau WiFi du module (SoftAP).

- **Utilité** : Guider l'utilisateur pas-à-pas pour connecter son téléphone au boîtier AgriShield, récupérer les données, et se déconnecter.
- **Fonctionnalités Clés :**
  1. **Étape 1 - Scan Radar** : Animation de recherche du SSID **AgriShield\_XXXX**.
  2. **Étape 2 - Connexion Guidée** : Invite l'utilisateur à rejoindre le réseau WiFi (Android/iOS prompt). Vérifie la connexion via l'endpoint **/health**.
  3. **Étape 3 - Live Monitor (Temps Réel)** : Une fois connecté, affiche les jauge en direct via **/status**.
    - Permet à l'agriculteur de voir les valeurs changer sous ses yeux (utile pour vérifier un arrosage immédiat).
  4. **Étape 4 - Synchronisation Auto** : Barre de progression indiquant le téléchargement du fichier historique (**/data/log**).
  5. **Confirmation** : Message "Données sécurisées !". L'application invite l'utilisateur à se déconnecter du WiFi du module.

## Page 3 : Analyse & Historique (Data Analytics)

L'espace de travail pour comprendre les tendances passées.

- **Utilité** : Permettre à l'agriculteur d'identifier les causes d'une maladie ou d'un stress hydrique en observant l'historique.
- **Fonctionnalités Clés :**
  1. **Filtres Temporels** : Boutons pour changer l'échelle : 24 Heures, 7 Jours, 1 Mois.
  2. **Graphiques Multi-Courbes** :
    - **Climat** : Superposition Température Air vs Humidité Air (pour détecter les conditions "Chaud + Humide" favorables au *Bacterial Wilt*).
    - **Sol** : Courbe d'humidité du sol avec ligne de seuil critique (ex: ligne

rouge à 35%).

3. **Timeline des Alertes** : Liste chronologique des événements critiques.
  - Ex: 14 Fév - 14:00 : Alerte Rouge (Température > 35°C).
4. **Diagnostic Énergétique** : Graphique de la tension batterie et indicateur binaire de charge solaire (`solar_charging` : `true`) pour vérifier que le panneau fonctionne bien.

## Page 4 : Assistant Agronome (Chat IA - Mode Online)

*Cette page est inactive hors-ligne. Elle devient le centre de conseil une fois Internet retrouvé.*

- **Utilité** : Fournir des diagnostics et des conseils via le moteur RAG du backend.
- **Fonctionnalités Clés** :
  1. **Interface de Chat** : Zone de dialogue textuel simple (Questions utilisateur / Réponses IA).
  2. **Contextualisation Implicite** : Un bandeau indique "L'IA analyse vos données du [Date]". (L'IA sait déjà que le sol est sec, pas besoin de lui dire).
  3. **Boutons de Suggestion (Quick Prompts)** :
    - "Quel est le risque de Mildiou ?"
    - "Dois-je arroser demain ?"
  4. **Fiches Maladies Interactives** : Si l'IA détecte un risque, elle affiche une "Carte Maladie" issue de la documentation (Source 2-6) avec :
    - Photo du symptôme (ex: taches poudreuses pour l'Oïdium).
    - Action recommandée immédiate.

## Page 5 : Configuration & Profils (Mode Technicien)

*Page d'administration pour gérer le matériel et le compte.*

- **Utilité** : Adapter le comportement du module au type de culture plantée.
- **Fonctionnalités Clés** :
  1. **Bibliothèque de Profils (Cloud)** : Liste des profils disponibles téléchargés du serveur (ex: "Tomate Pépinière", "Piment Plein Champ").
  2. **Appliquer au Module (IoT)** : Si connecté au module, bouton pour envoyer

- le profil choisi via `POST /config`.
3. **Paramètres Seuils Manuels** : (Avancé) Possibilité de surcharger manuellement un seuil (`humidity_critical`) si l'agronome le recommande.
  4. **Infos Système** : Affichage de la version du firmware (`1.0.0`) et de l'ID du dispositif (`AS-001...`).

---

Voici la **Conclusion** pour clore votre document de conception. Elle synthétise l'architecture technique tout en rappelant la vision d'impact du projet, conformément au contexte du Hult Prize.

---

## Conclusion

La conception technique présentée dans ce document définit une architecture logicielle robuste et adaptée aux réalités du terrain africain. En adoptant une stratégie "**Offline-First**", le système AgriShield réussit à concilier deux impératifs souvent contradictoires : la **résilience locale** nécessaire en zone rurale et l'**intelligence avancée** offerte par le Cloud.

### 1. Une complémentarité stratégique

Le système repose sur une division claire des responsabilités qui garantit la fiabilité du service :

- **Le Module IoT (Le "Réflexe")** : Il assure la survie immédiate de la plante grâce à des alertes instantanées et une autonomie énergétique totale. Il est l'autorité de terrain, immuable et fiable.
- **L'Application Mobile (La "Mémoire")** : Elle agit comme une passerelle indispensable, permettant de collecter et visualiser les données sans dépendre d'une infrastructure réseau inexistante au champ.
- **Le Backend Java (Le "Cerveau")** : Il apporte la valeur ajoutée sur le long terme (archivage, météo) et démocratise l'accès à l'expertise agronomique grâce à l'IA

générationne (RAG), transformant chaque smartphone en assistant personnel.

## 2. Une réponse à la fracture numérique

Contrairement aux solutions "tout connecté" inadaptées aux zones blanches, cette architecture accepte la déconnexion comme un état normal et non comme une erreur. Le flux de données asynchrone (Collecte locale → Synchronisation différée) permet d'apporter des technologies de pointe (Prédictions maladies, Analyse de tendances) aux petits agriculteurs, contribuant directement à la réduction des pertes de récoltes (ODD 2) et à l'adaptation climatique (ODD 13).

AgriShield ne se contente pas de surveiller les cultures ; grâce à cette architecture logicielle, il dote l'agriculteur d'un outil décisionnel puissant pour sécuriser ses revenus et pérenniser son exploitation face aux défis climatiques.

