Rapport de Projet - Solution IoT ESEO

Table des matières

Chapitre 1

Bilan d'Avancement Réel

Le projet suit une architecture **Clean Architecture** robuste. La majorité des modules fonctionnels sont opérationnels, bien que certaines exigences de sécurité et de règles métier complexes restent à finaliser.

1.1 Tableau de synthèse global

Module	État	Complétion	Commentaires
Step 1 : Unité de Mesure		100%	CRUD complet, filtrage fonctionnel.
Step 2 : Utilisateur		90%	Gestion des rôles OK. **Hachage des mots de passe manquant**.
Step 3 : Système Partenaire		100%	Import/Export fonctionnel.
Step 4 : Sonde		100%	Héritage TPH et relations validées.
Step 5 : Relevé		95%	Graphiques et pagination OK. **Règle de suppression liée aux alertes manquante**.
Step 6 : Seuil d'Alerte		100%	Logique d'activation exclusive opérationnelle.
Step 7 : Alerte		100%	Cycle de vie et résolution auto validés.
Step 8 : Actionneur		100%	Contrôle dynamique (sliders) implémenté.
Step 9 : Import Externe		100%	Basic Auth et gestion des doublons OK.
Step 10 : Dashboard Global		100%	Widgets temps réel et auto-refresh (30s) OK.
Step 11 : Communication Device		80%	HttpPull/Push OK. **MQTT et SignalR non implémentés** (optionnels).

Chapitre 2

Analyse des Fonctionnalités par Module

2.1 Step 1 : Unité de Mesure

— **Implémenté** : CRUD complet via `UniteMesureService` et `UniteMesureController`. Interface Blazor avec validation.
— **Douteux** : La navigation vers `Sonde` est définie dans l'entité mais désactivée dans la configuration Fluent API (`UniteMesureConfiguration.cs`).

2.2 Step 2 : Utilisateur

— **Implémenté** : Distinction rôles Admin/Utilisateur. Recherche dynamique.
— **Manquant** : Le hachage des mots de passe. Les données sont stockées en clair, ce qui représente un risque de sécurité majeur (`UserService.cs`).

2.3 Step 3 : Système Partenaire

— **Implémenté** : Support des modes Appelant/Appelé. Gestion sécurisée des credentials pour les partenaires externes.

2.4 Step 4 : Sonde

— **Implémenté** : Utilisation du pattern Table-Per-Hierarchy (TPH) pour les dispositifs. Validation stricte des plages de mesure (ValeurMin $<$ ValeurMax).

2.5 Step 5 : Relevé

— **Implémenté** : Intégration de Chart.js pour la visualisation. Pagination côté serveur performante.
— **Manquant** : Blocage de la suppression d'un relevé si une alerte y est rattachée (intégrité des données).

2.6 Step 6 : Seuil d'Alerte

— **Implémenté** : Mécanisme de désactivation automatique des anciens seuils lors de l'activation d'un nouveau pour une sonde donnée.

2.7 Step 7 : Alerte

— **Implémenté** : Cycle de vie complet (Active -> Acquittée -> Résolue). Résolution automatique dès le retour à la normale d'une sonde.
— **Douteux** : Absence de `Mapperly` pour ce module spécifique, contrairement aux standards du projet.

2.8 Step 8 : Actionneur & État

— **Implémenté** : Synchronisation 1-à-1 entre l'actionneur et son état physique. Interface de contrôle dynamique dans Blazor.

2.9 Step 9 : Import Externe

— **Implémenté** : Client HTTP robuste utilisant `IHttpClientFactory` pour l'importation de sondes depuis des systèmes tiers.

2.10 Step 10 : Dashboard Global

— **Implémenté** : Vue synthétique agrégeant les statistiques de tous les modules. Rafraîchissement automatique optimisé.

2.11 Step 11 : Communication Device

— **Implémenté** : Service d'arrière-plan `HttpPullBackgroundService` pour l'interrogation cyclique des capteurs. Support des Webhooks pour le mode Push.
— **Manquant** : Protocoles MQTT et SignalR (restés au stade de code commenté dans les Enums).

Chapitre 3

Suivi des Problèmes Rencontrés

3.1 Problèmes de Code et Sécurité

— **Sécurité Critique** : Le stockage des mots de passe en clair est le problème le plus urgent. Une implémentation de `IPasswordHasher` est nécessaire.
— **Validation métier** : Plusieurs règles de validation inter-modules (comme la suppression de relevés liés à des alertes) sont manquantes, risquant de créer des orphelins en base de données.

3.2 Problèmes d'Architecture et Dette Technique

— **Incohérence des Mappers** : L'usage mixte de `Mapperly` et de mapping manuel complexifie la maintenance.
— **Configurations EF Core** : Certaines relations de navigation sont commentées, ce qui limite les capacités de requêtage complexe sans code supplémentaire.

Chapitre 4

Bilan Fonctionnel Honnête

4.1 État Global

Le projet est fonctionnel à environ **93%**. Les flux de données principaux (capteurs -> relevés -> alertes -> dashboard) fonctionnent parfaitement. L'interface Blazor est fluide et respecte les maquettes.

4.2 Points Forts

— **Clean Architecture** : Très bien structurée, facilitant l'ajout de nouveaux modules.
— **Dashboard** : Très réactif et visuellement complet.
— **Héritage TPH** : Implémentation élégante de la hiérarchie des dispositifs.

4.3 Points Faibles

— **Sécurité** : Lacunes sur la gestion des secrets et des mots de passe.
— **Tests** : Absence de tests unitaires pour la logique métier des services.
— **Protocoles** : La communication reste limitée au HTTP, les protocoles temps réel (MQTT) n'ayant pas été finalisés.