

DIGITAL TWIN AUGMENTÉ AVEC IA GENERATIVE

PRESENTÉ PAR:

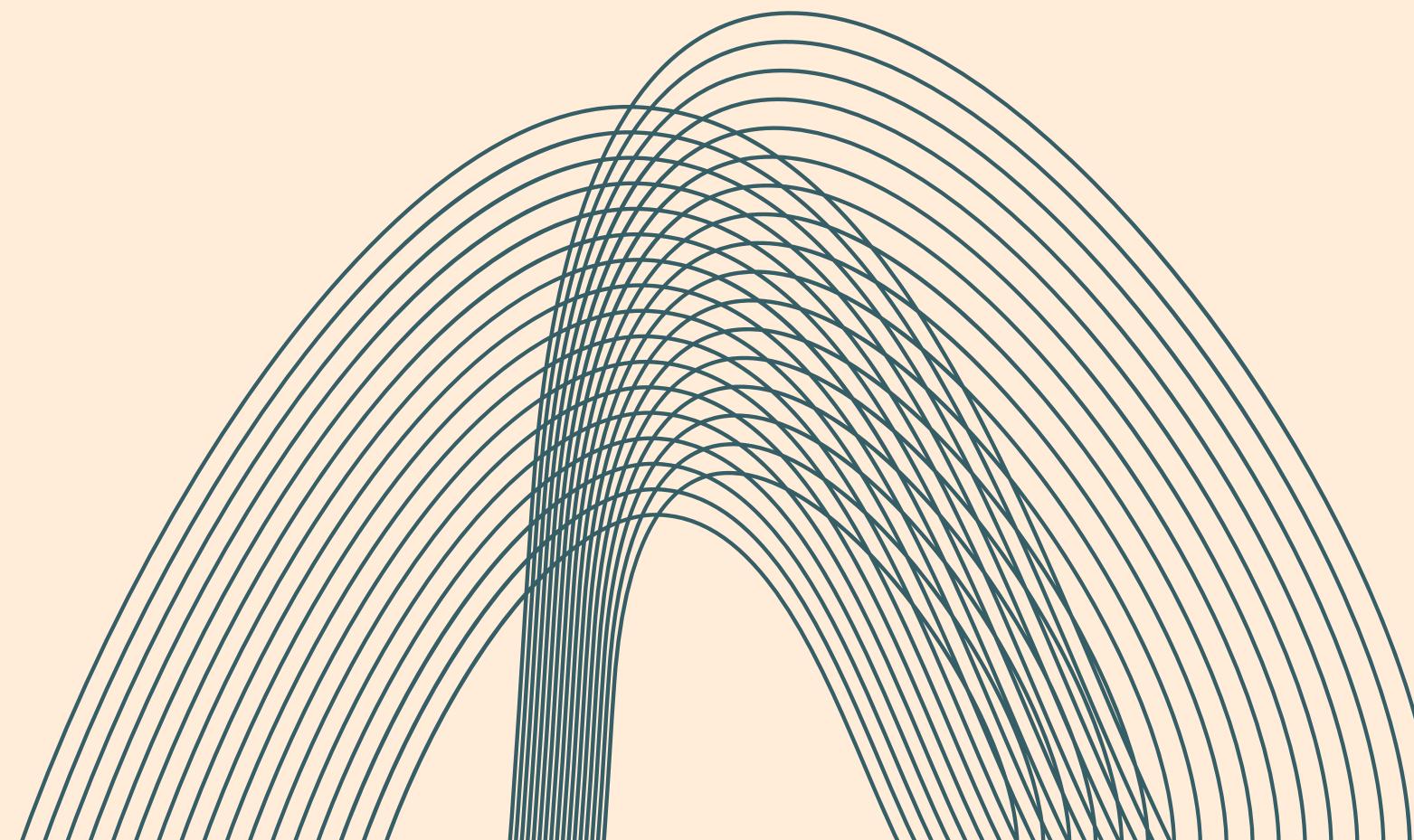
- KARDI ABDOSSATTAR
- NAAIMI AYMAN
- LOUZALI ABDELHAMID



PLAN

01

INTRODUCTION



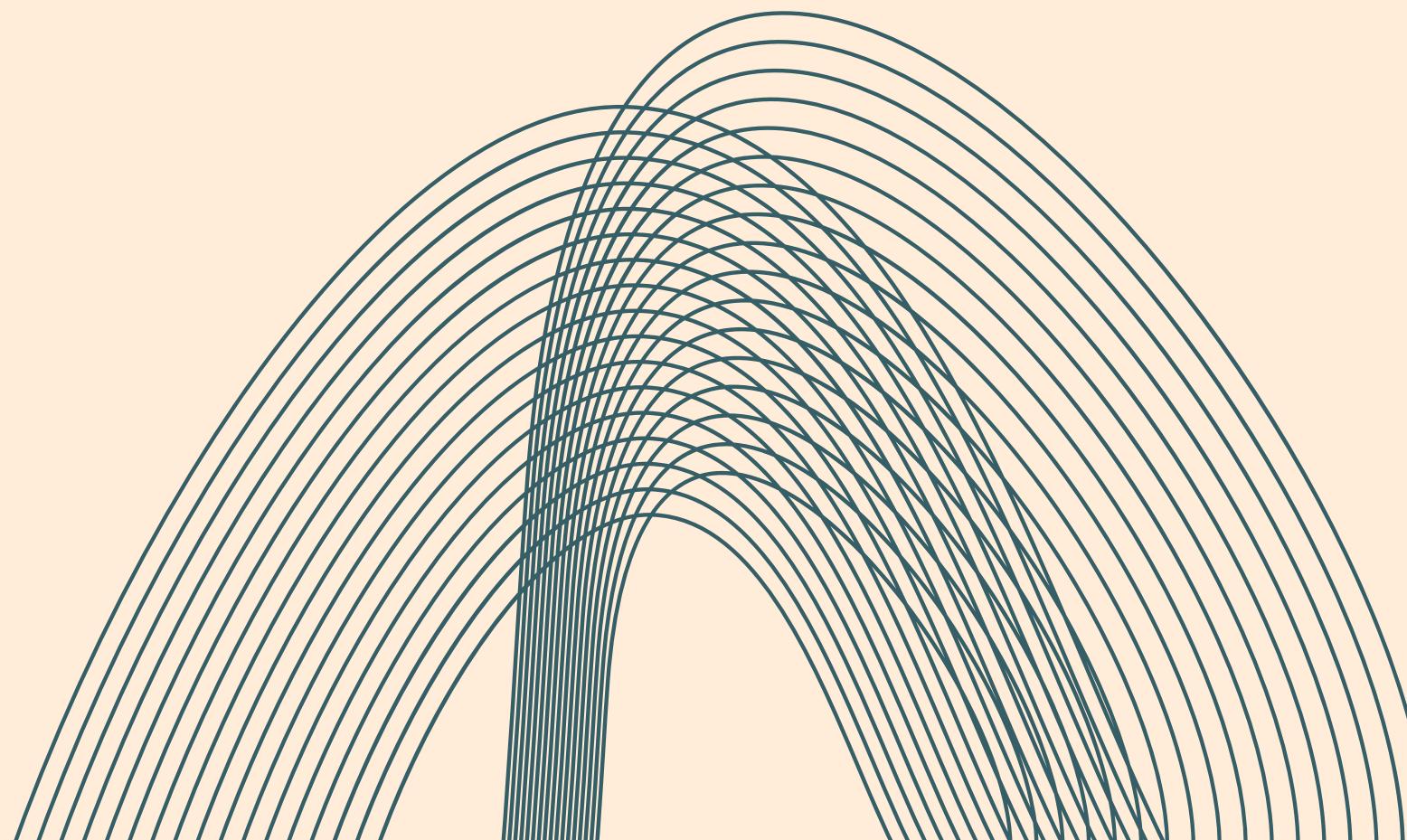
02

CONCEPTS

PLAN

03

PIPELINE



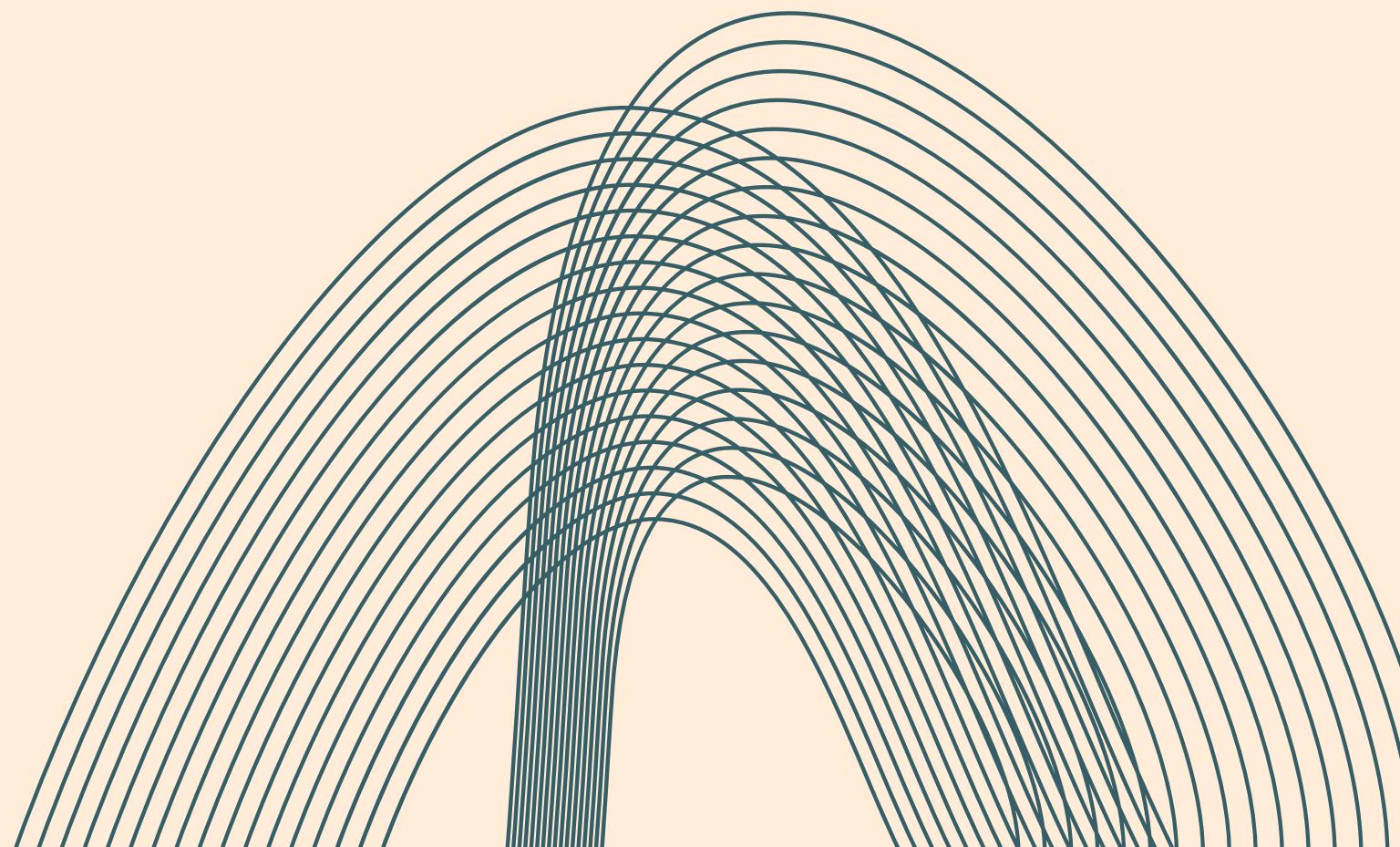
04

APROCHES

PLAN

05

DEMONSTRATION



INTRODUCTION

- L'industrie a connu une évolution progressive, passant de systèmes mécaniques simples à des systèmes de plus en plus automatisés et connectés
- Cette évolution a conduit à l'intégration massive de capteurs capables de mesurer en temps réel l'état des machines

INTRODUCTION

- Avec la généralisation de ces capteurs, est apparue l'idée de collecter et centraliser les données pour mieux comprendre le fonctionnement des systèmes
- L'Internet des Objets (IoT) a alors permis d'interconnecter ces capteurs et de rendre les données accessibles en continu

INTRODUCTION

- Cependant, la simple disponibilité des données ne suffit pas à en extraire du sens ou à prendre des décisions efficaces
- L'intelligence artificielle est devenue nécessaire pour analyser ces données complexes et détecter des anomalies ou tendances

INTRODUCTION

- Afin de représenter et exploiter ces informations de manière structurée, le concept de jumeau numérique s'est imposé
- Enfin, l'intelligence artificielle générative apporte une aide supplémentaire en transformant les données analysées en explications claires et recommandations compréhensibles par l'humain

CONCEPTS

Qu'est-ce qu'un Digital Twin (Jumeau Numérique) ?

- Un digital twin, ou jumeau numérique, est une représentation virtuelle d'un système physique réel (machine, équipement, processus industriel).
- Il est alimenté en continu par des données provenant de capteurs installés sur le système réel.
- Grâce à ces données, le jumeau numérique reflète en temps réel l'état, le comportement et les performances de l'équipement physique.

CONCEPTS

Rôle du Digital Twin dans la Maintenance Prédictive

- Il permet de surveiller en continu l'état des machines
- Il facilite la détection précoce d'anomalies ou de comportements anormaux
- Il aide à anticiper les pannes avant qu'elles ne surviennent
- Il permet de tester des scénarios (usure, surcharge, conditions extrêmes) sans impacter le système réel
- Il améliore la planification des opérations de maintenance, en réduisant les arrêts non planifiés

CONCEPTS

Limite Sans Intelligence Artificielle

- Sans intelligence artificielle, le digital twin reste principalement un outil de visualisation.
- Il montre des données et des graphiques, mais n'explique pas automatiquement les causes des anomalies ni les actions à entreprendre.
- C'est pourquoi l'intégration de l'intelligence artificielle, et en particulier de l'IA générative, permet de transformer le digital twin en un véritable outil d'aide à la décision.

CONCEPTS

Study case : Machine étudiée

- Notre étude de cas porte sur une pompe centrifuge verticale multistage de type GRUNDFOS CR, largement utilisée dans les systèmes industriels pour le pompage de fluides (eau, circuits industriels, surpression, refroidissement).
- Ce type de machine est critique car son dysfonctionnement peut entraîner des arrêts de production, des pertes énergétiques ou des dégradations mécaniques

CONCEPTS

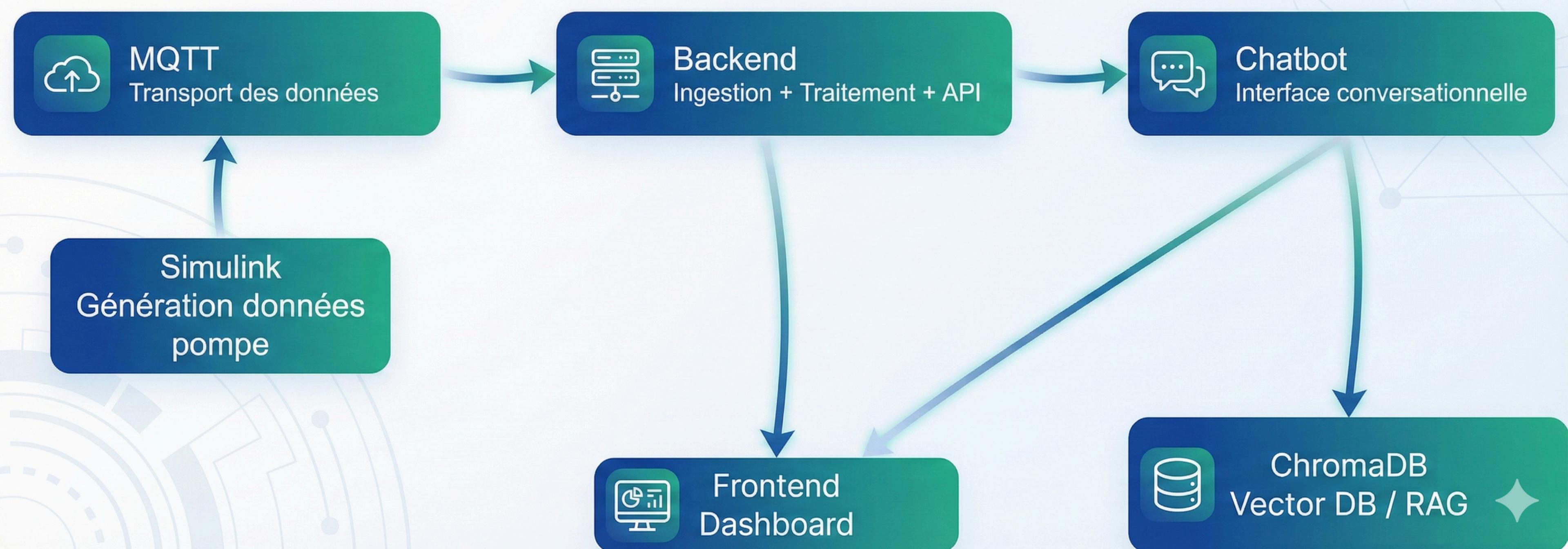
Study case : Machine étudiée





PIPELINE

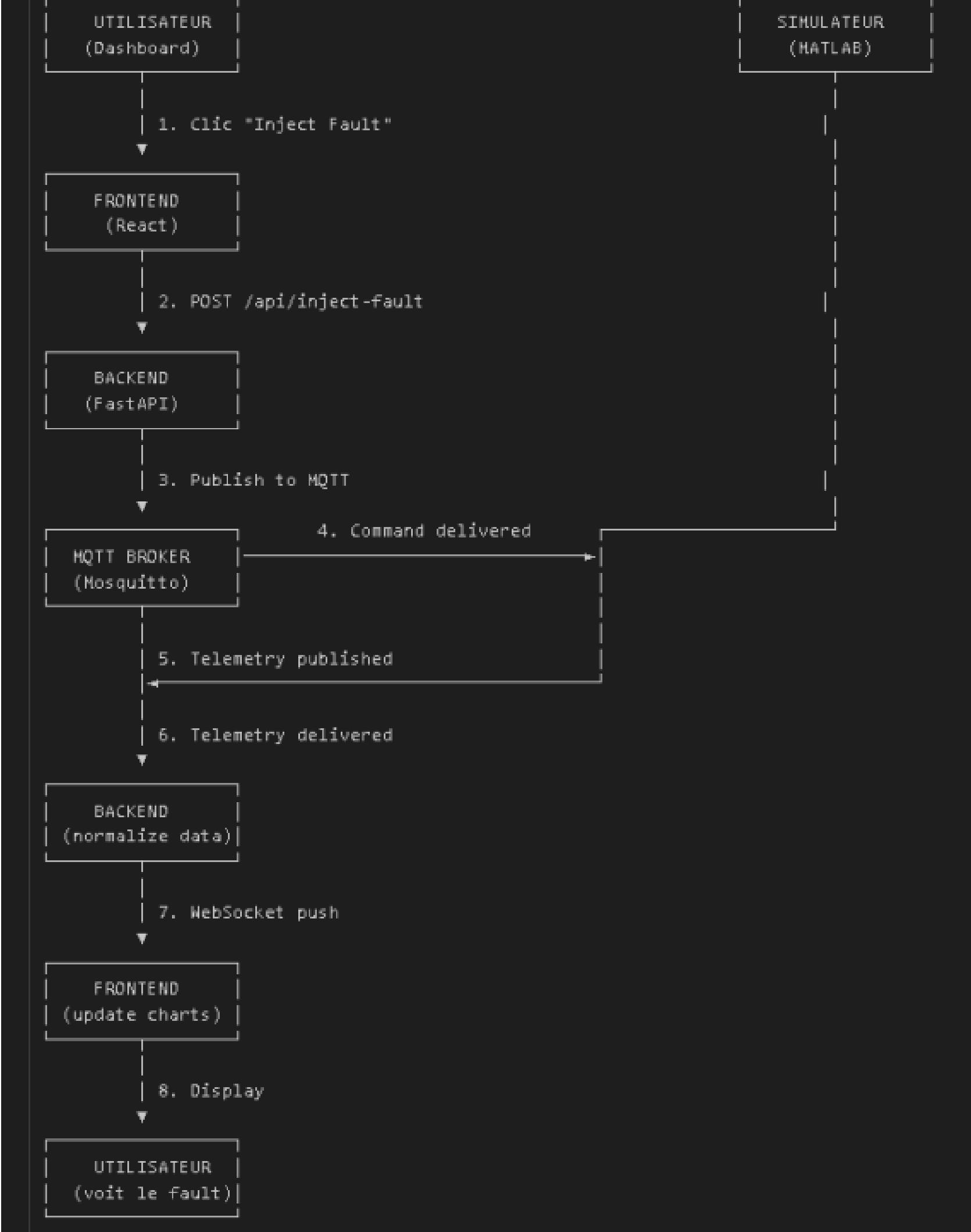
General pipeline



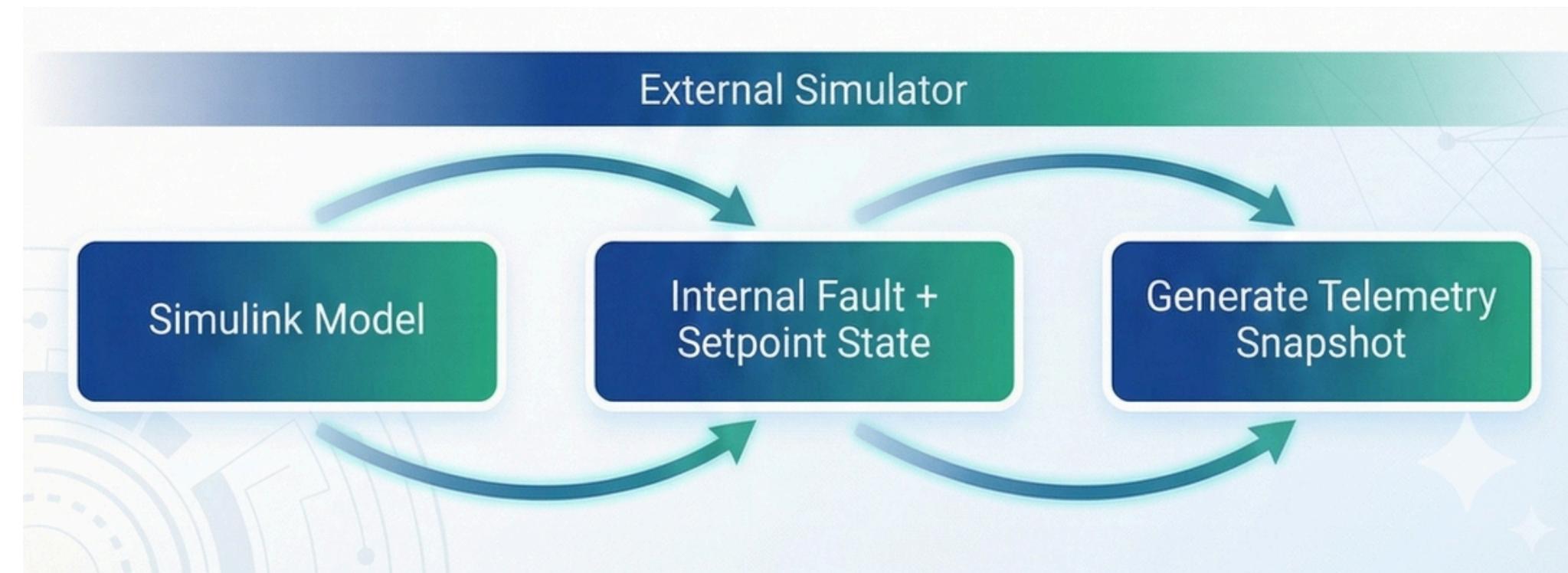
Simulation ↔ Backend

Architecture MQTT - Digital Twin Grundfos CR 15

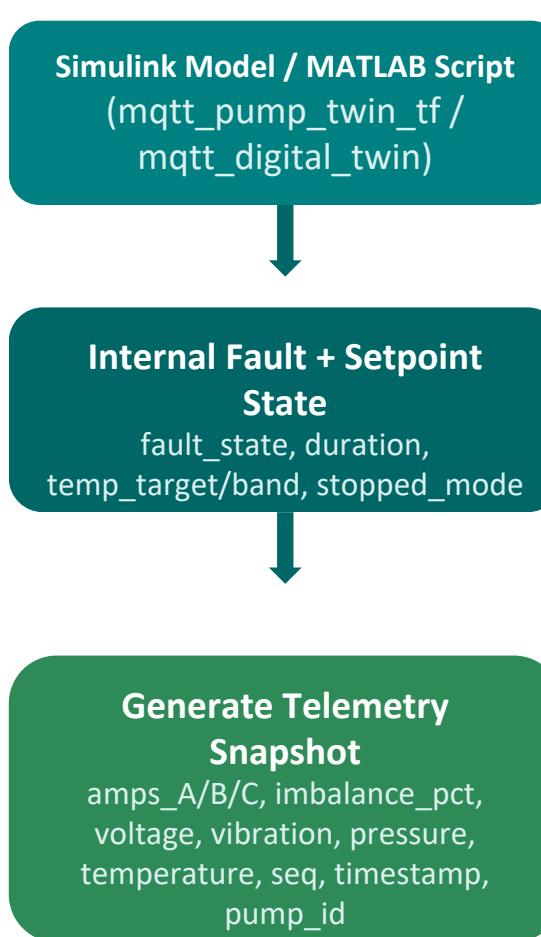
Exemple Scénario injection



1. External Simulator (MATLAB / Simulink)



1. External Simulator (MATLAB / Simulink)



Explications

Rôle du Simulateur

- Simule le comportement physique de la pompe CR 15
- Génère des données capteurs réalistes chaque seconde

État Interne (Variables)

- fault_state : Type de panne active (ou 'normal')
- duration : Temps écoulé depuis l'injection
- temp_target/band : Consigne température optionnelle
- stopped_mode : Flag arrêt d'urgence

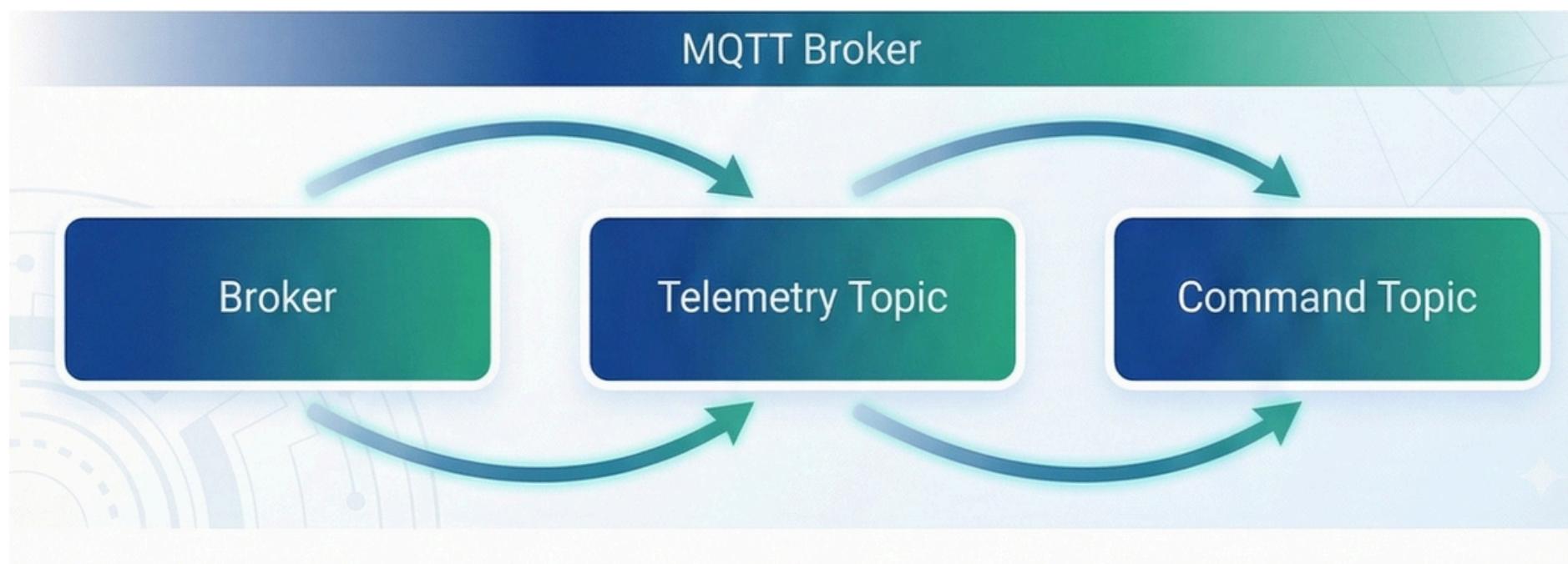
Télémetrie Générée (1/sec)

- amps_A, amps_B, amps_C : Courant triphasé
- imbalance_pct : Déséquilibre phases (%)
- voltage, vibration, pressure, temperature
- seq : Numéro de séquence
- timestamp : Horodatage ISO
- pump_id : Identifiant pompe



Le simulateur réagit aux commandes reçues via MQTT

2. MQTT Broker (Mosquitto)



2. MQTT Broker (Mosquitto)

Broker
localhost:1883

Telemetry Topic
digital_twin/{pump_id}/telemetry

Command Topic
digital_twin/{pump_id}/command

Explications

Qu'est-ce que MQTT ?

- Protocole léger de messagerie (Publish/Subscribe)
- Idéal pour IoT et communication temps réel
- Broker = serveur central qui route les messages

Topic Télémétrie (Vert)

→ Direction : Simulateur → Backend

- Le simulateur PUBLIE les données capteurs
- Le backend S'ABONNE pour recevoir
- Fréquence : 1 message/seconde

Topic Commandes (Violet)

→ Direction : Backend → Simulateur

- Le backend PUBLIE les commandes
- Le simulateur S'ABONNE pour exécuter
- Commandes : INJECT_FAULT, RESET, EMERGENCY_STOP

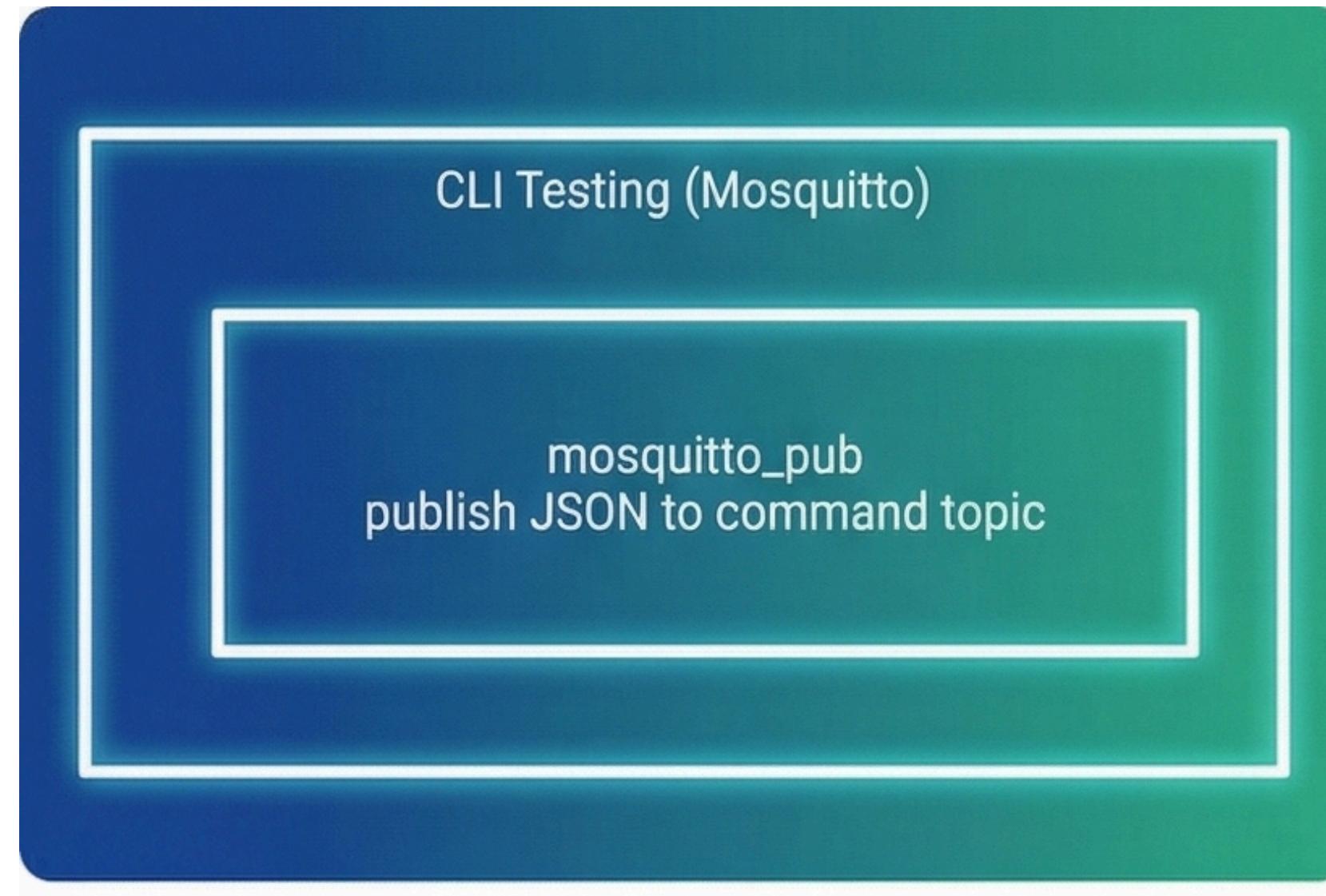
Avantage

- Découplage total entre producteur et consommateur



Mosquitto tourne sur localhost:1883 - pas d'auth pour le dev

2. CLI Testing (Mosquitto)



2. CLI Testing (Mosquitto)



```
# 1. Injecter une panne CAVITATION
mosquitto_pub -h localhost -p 1883 \
-t digital_twin/pump01/command \
-m '{"command":"INJECT_FAULT","fault_type":"CAVITATION"}'

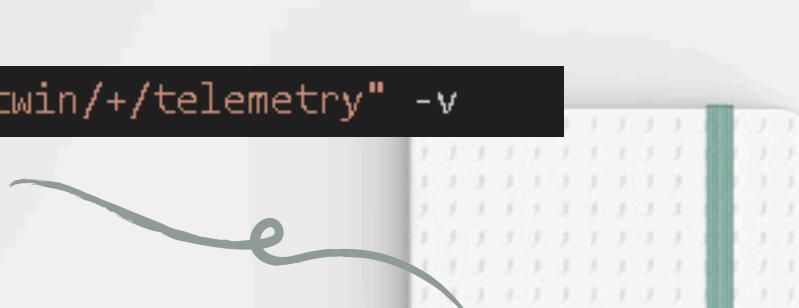
# 2. Injecter avec température verrouillée (90 ± 2°C)
mosquitto_pub -h localhost -p 1883 \
-t digital_twin/pump01/command \
-m '{"command":"INJECT_FAULT","fault_type":"WINDING_DEFECT","temperature_target":90,"temperature_band":2}'

# 3. Arrêt d'urgence
mosquitto_pub -h localhost -p 1883 \
-t digital_twin/pump01/command \
-m '{"command":"EMERGENCY_STOP"}'

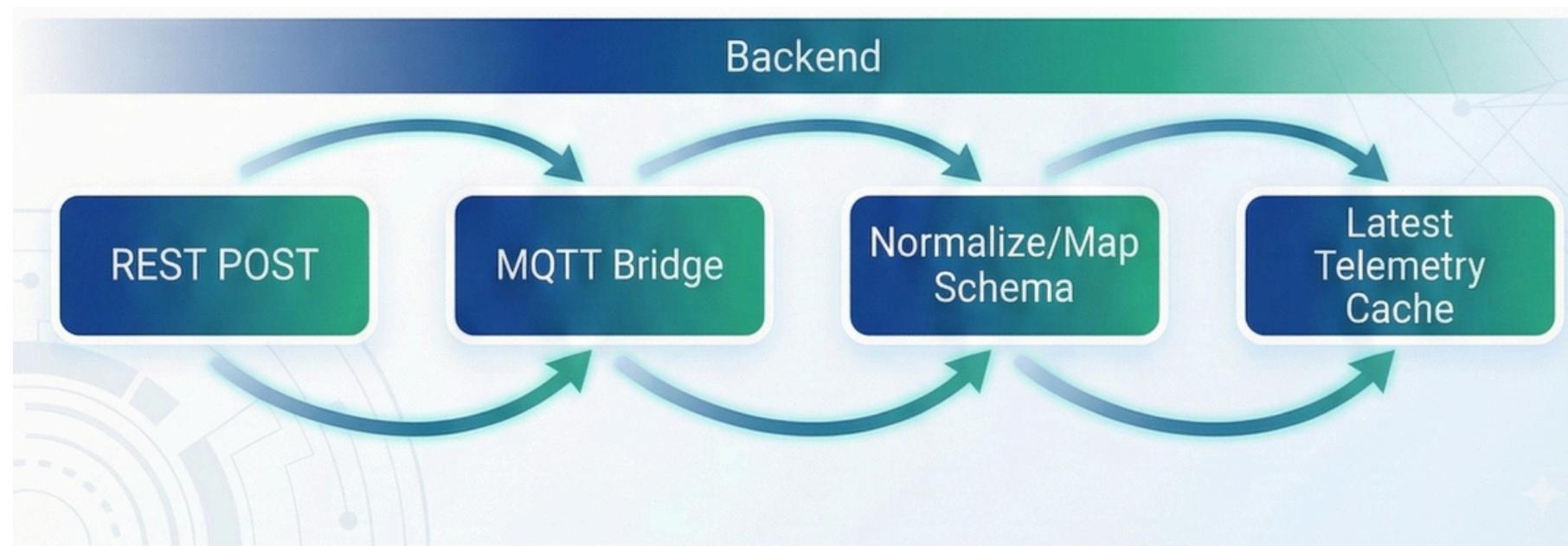
# 4. Reset
mosquitto_pub -h localhost -p 1883 \
-t digital_twin/pump01/command \
-m '{"command":"RESET"}'
```

Pour Ecouter la telemetry

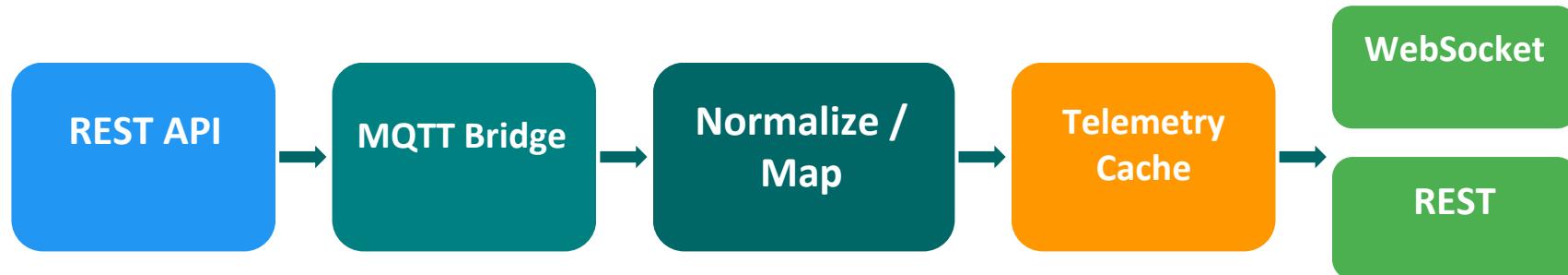
```
mosquitto_sub -h localhost -p 1883 -t "digital_twin/+/telemetry" -v
```



3. Backend (MQTT Mode) - FastAPI Python



3. Backend (MQTT Mode) - FastAPI Python



Explications

REST API (Bleu) - Point d'entrée commandes

- POST /api/inject-fault → Injecter une panne
- POST /api/emergency-stop → Arrêt d'urgence
- POST /api/reset → Réinitialiser

MQTT Bridge (Teal) - Pont bidirectionnel

- Subscribe : digital_twin/{pump_id}/telemetry
- Publish : digital_twin/{pump_id}/command
- Convertit les requêtes REST en messages MQTT

Normalize / Map Schema (Teal foncé)

- amps_A → phase_a, amps_B → phase_b, amps_C
- Calcule phase_average = (A+B+C)/3
- Pass-through : voltage, vibration, pressure, temperature

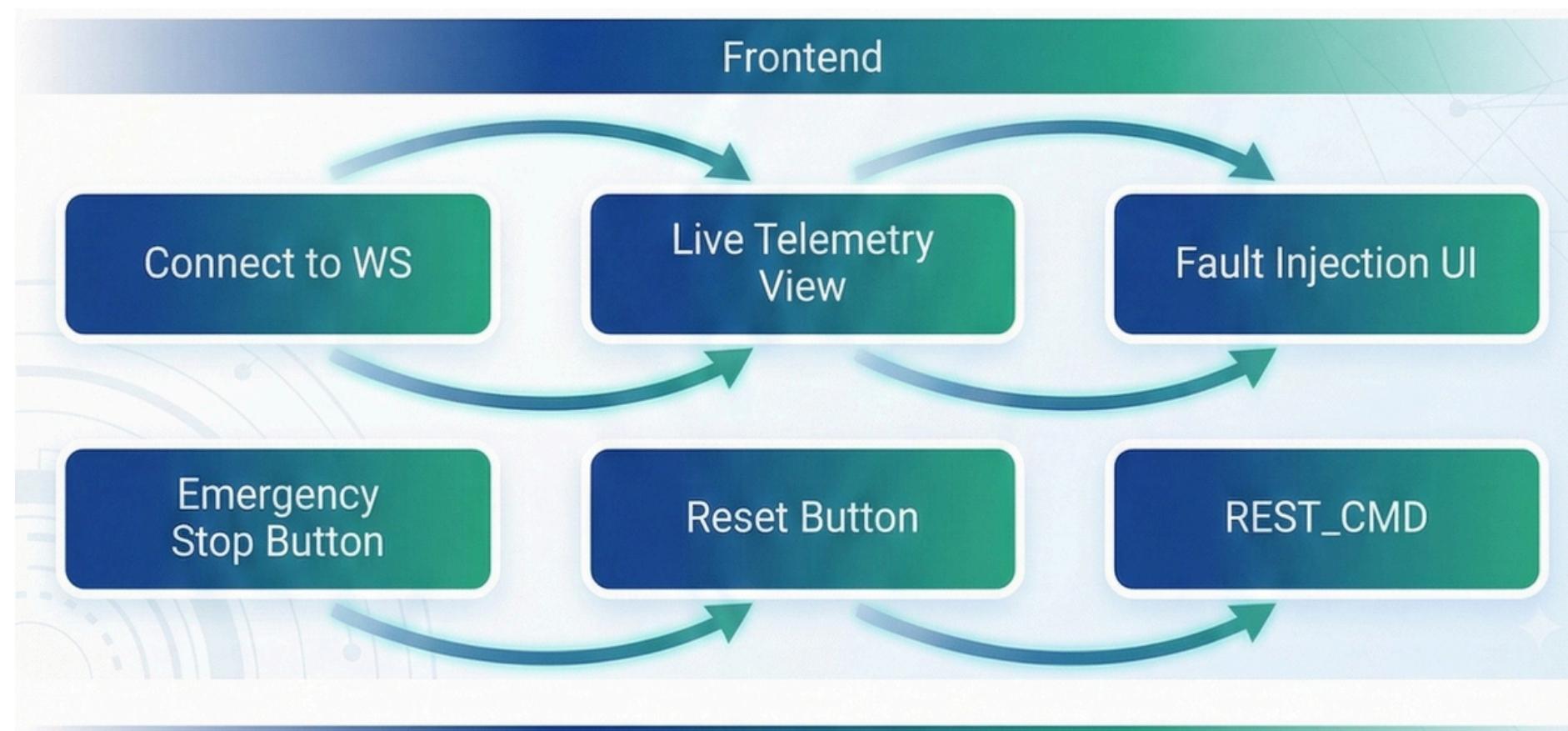
Cache + Sorties (Orange/Vert)

- Stocke le dernier snapshot pour réponses instantanées
- WebSocket : Push temps réel vers le frontend
- REST GET : Pour requêtes ponctuelles

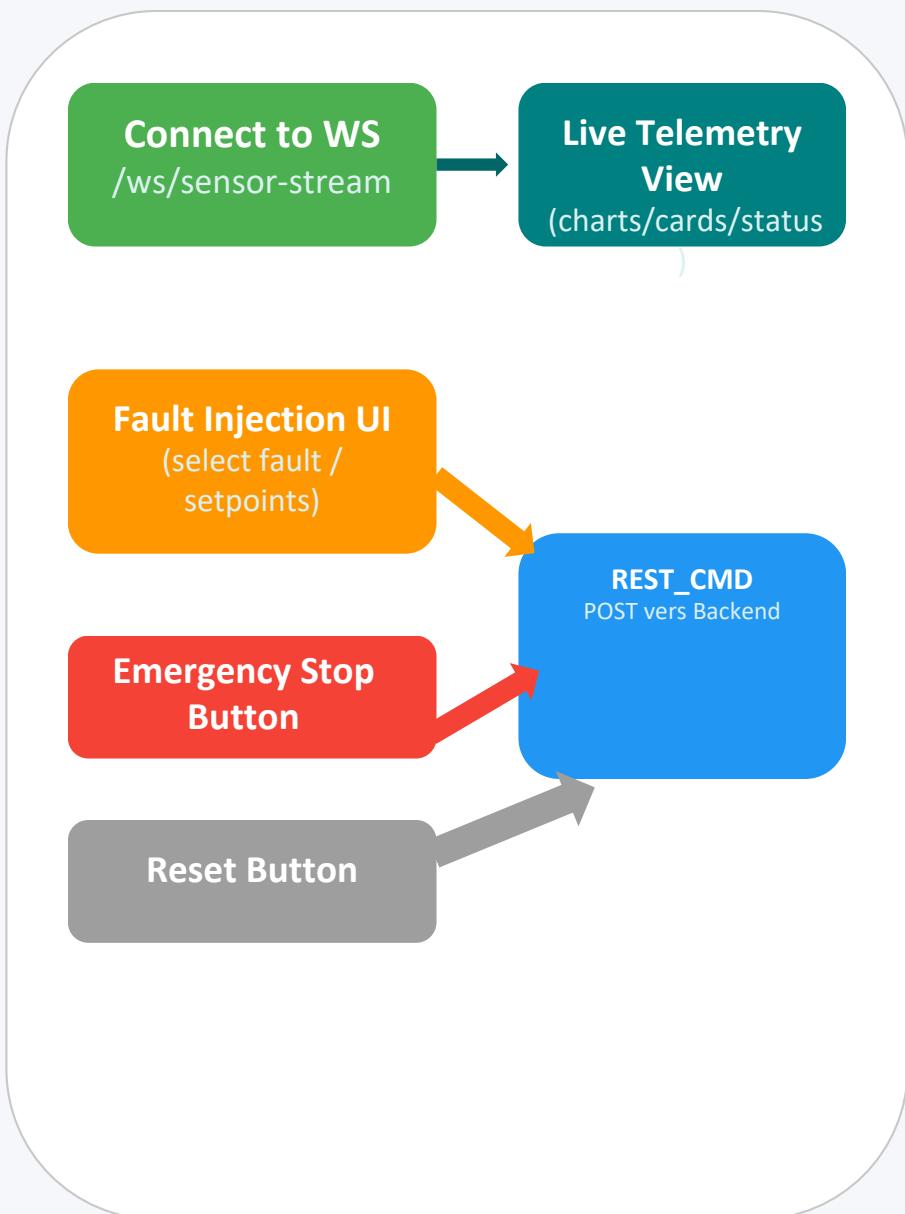


Le backend est le chef d'orchestre de toute l'architecture

4. Dashboard React



4. Dashboard React



Explications

Connexion WebSocket (Vert)

- Se connecte à /ws/sensor-stream au démarrage
- Reçoit les données en temps réel (1/sec)
- Pas de polling = performance optimale

Vue Télémétrie (Teal)

- Cards : Affichent valeurs actuelles
- Charts : Historique des 60 dernières secondes
- Status : Normal (vert), Fault (orange), Stopped (rouge)

Contrôles → REST_CMD (Bleu)

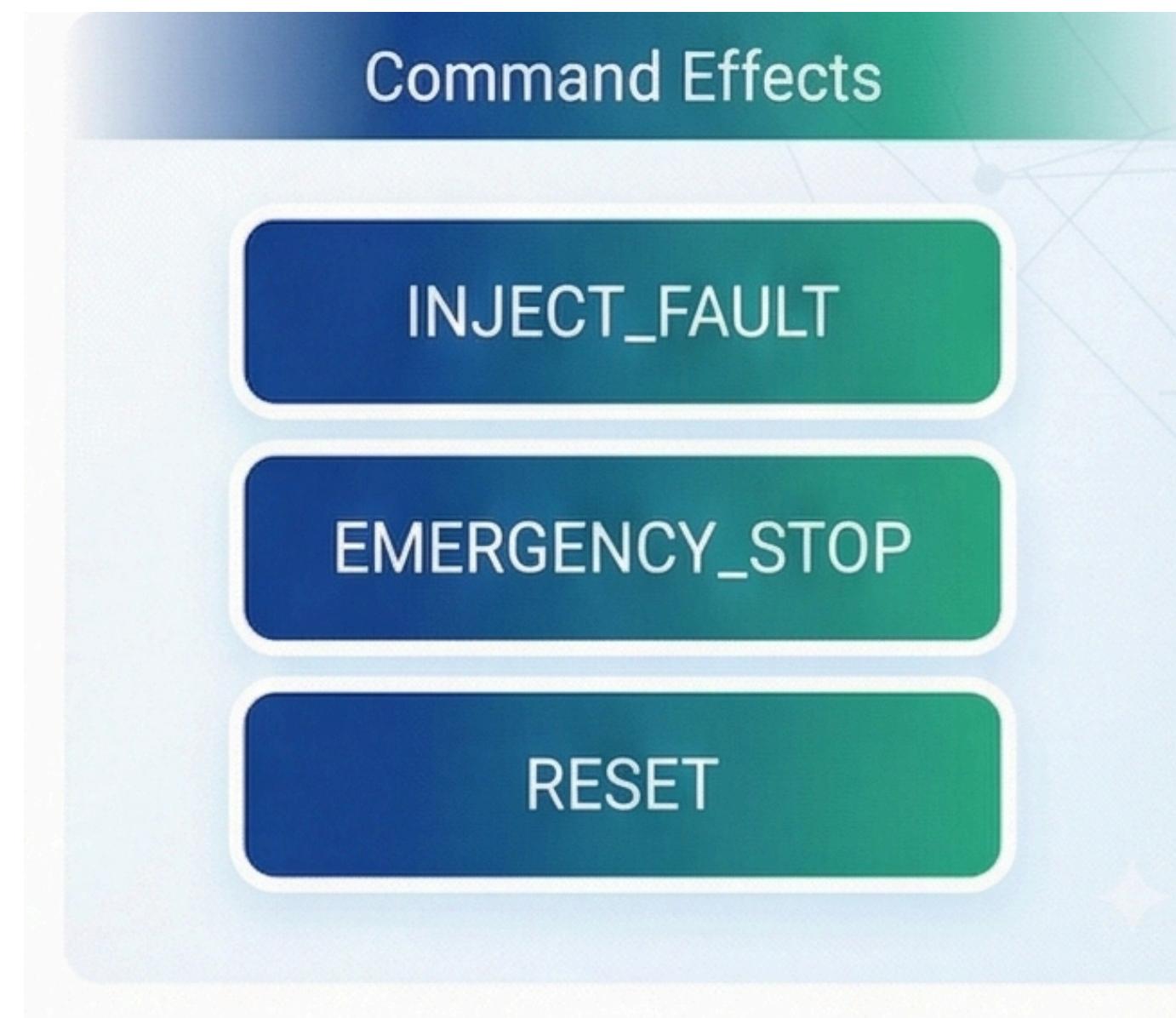
- Fault Injection UI : Sélection panne + setpoints
- Emergency Stop : Arrêt immédiat (bouton rouge)
- Reset : Retour état normal

Tous les boutons envoient des POST au backend



Le frontend ne communique jamais directement avec MQTT

5. Command Effects



5. Command Effects (dans le Simulateur)

INJECT_FAULT

- set fault_type
 - optional
- temperature_target/band
- start/continue duration

EMERGENCY_STOP

- enter stopped_mode
- publish exact zeros

RESET

- clear fault
- clear setpoints
- exit stopped_mode

Explications

INJECT_FAULT (Orange)

- Paramètre obligatoire : fault_type
(CAVITATION, BEARING_WEAR, WINDING_DEFECT, SUPPLY_FAULT, OVERLOAD)
- Paramètres optionnels :
 - temperature_target : Consigne °C
 - temperature_band : Tolérance ±°C
- Active le compteur duration

EMERGENCY_STOP (Rouge)

- Active stopped_mode = true
- TOUTES les valeurs capteurs = 0.0 exact
- Reste bloqué jusqu'à RESET
- Simule un arrêt physique de la pompe

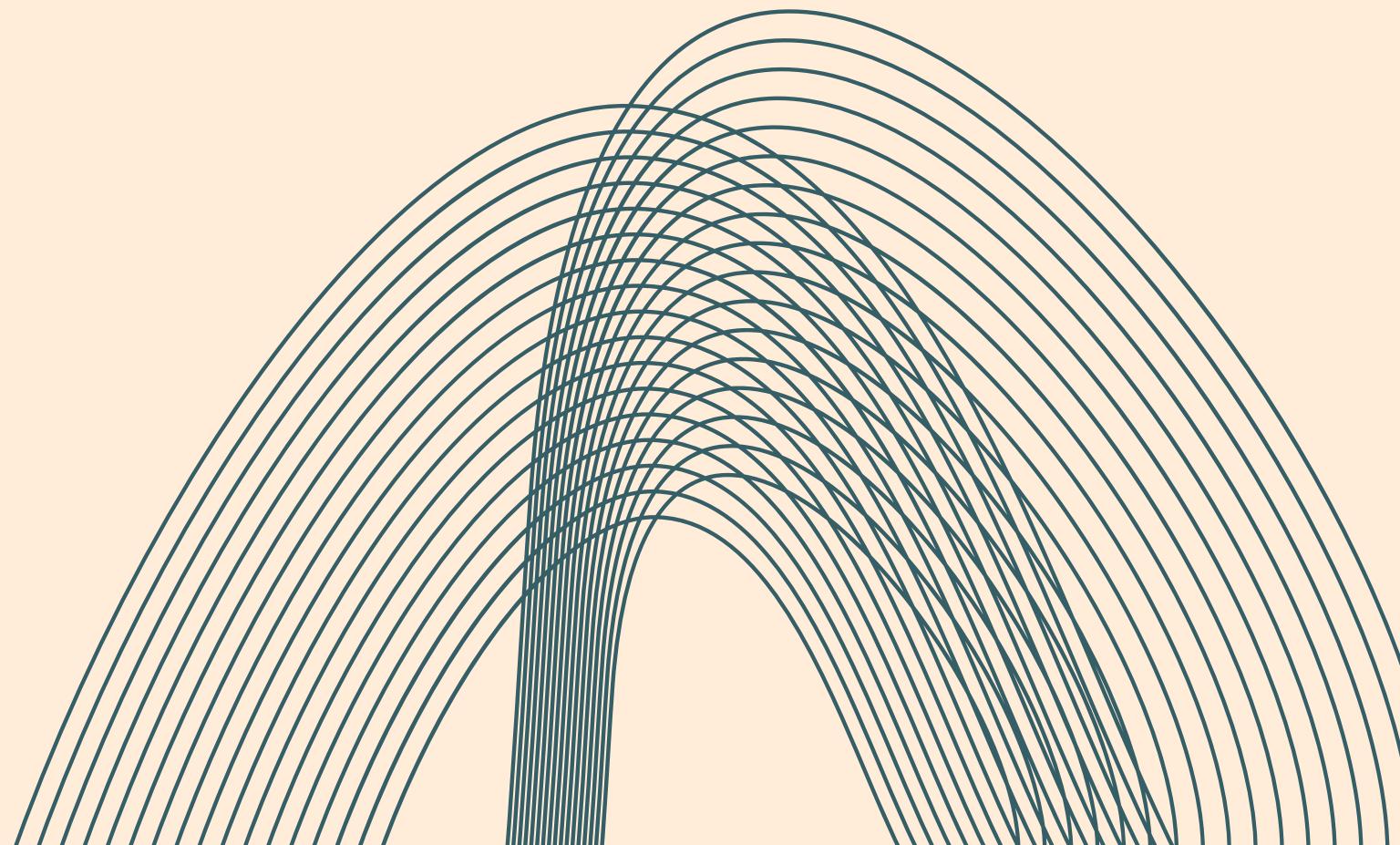
RESET (Vert)

- Efface fault_state → 'normal'
- Efface tous les setpoints
- Désactive stopped_mode
- Valeurs reviennent aux nominales

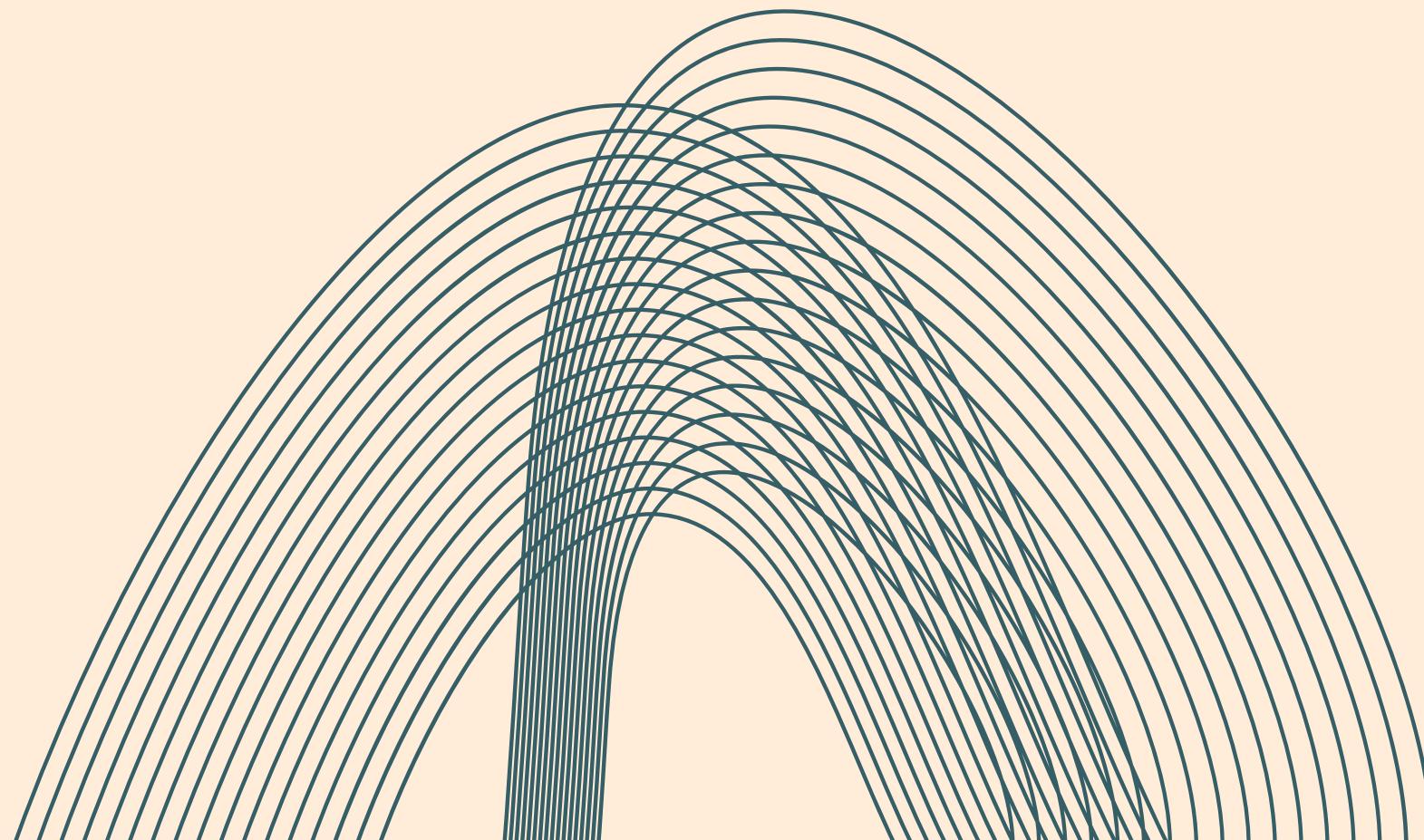


Les commandes sont reçues via le topic MQTT /command

Digital Twin : De la donnée simulée au rapport RAG

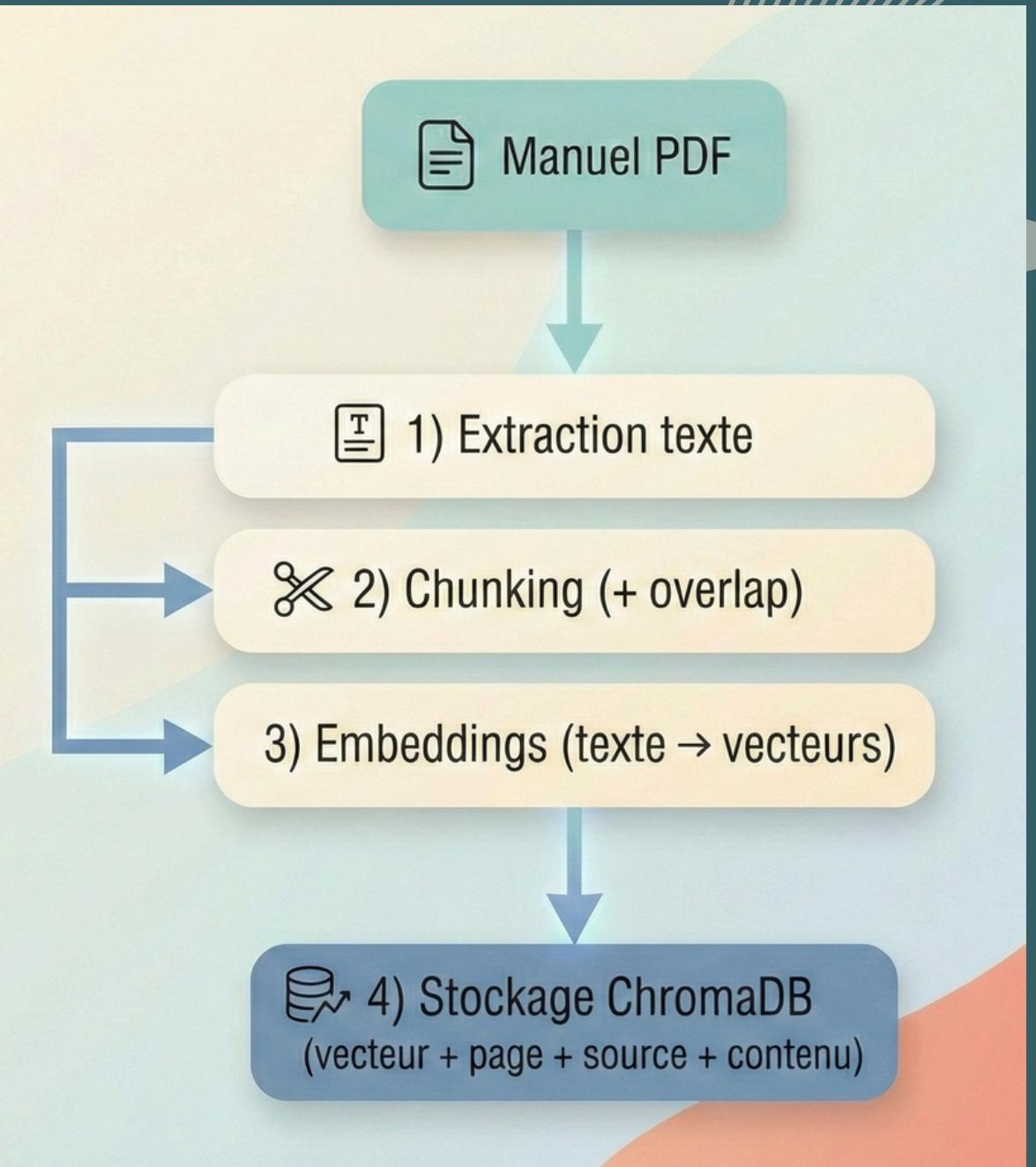


Concepts clés (définitions)



Ingestion

Processus automatisé qui transforme des données non structurées en format exploitable. Il se compose de quatre étapes clés : extraction du contenu brut, segmentation en chunks, vectorisation (embedding) et indexation dans la base de données vectorielle (ChromaDB).





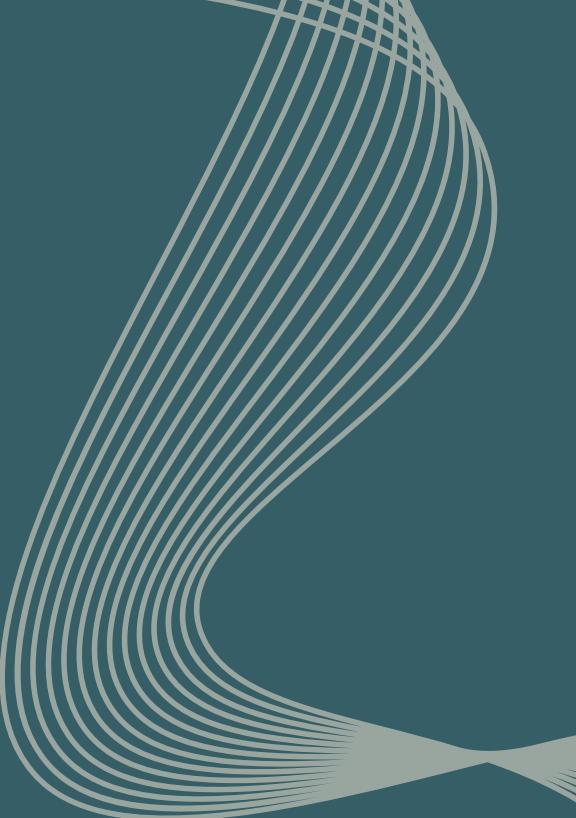
Chunk (Segment)

Unité logique de texte résultant du découpage du document original. Sa taille est calibrée pour maximiser la cohérence sémantique et respecter la fenêtre de contexte du LLM. Chaque chunk est associé à des métadonnées (source, numéro de page) pour assurer la traçabilité.



Embedding (Plongement vectoriel)

Représentation d'un texte sous forme d'un vecteur numérique de haute dimension. Cette transformation capture la signification sémantique du texte plutôt que ses mots-clés exacts, permettant de calculer mathématiquement la proximité de sens entre une requête et un document.



Vector DB (ChromaDB)

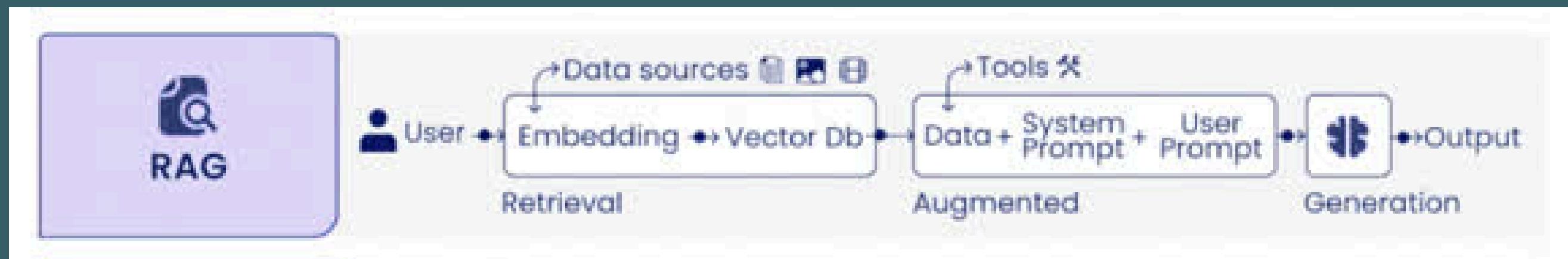
Base de données spécialisée dans le stockage et l'indexation de vecteurs. Elle utilise des algorithmes de recherche de plus proches voisins (comme ANN) pour identifier quasi-instantanément les segments les plus sémantiquement proches d'une requête utilisateur.

RAG

Retrieval (Récupération) : L'utilisateur pose une question (ou le système détecte une panne). Le système effectue une recherche sémantique dans votre base de données vectorielle (ChromaDB) pour trouver les passages du manuel technique les plus pertinents.

Augmentation (Augmentation) : Le système prend la question de l'utilisateur et y "colle" les passages trouvés dans le manuel. On enrichit (augmente) le contexte.
Prompt envoyé au LLM : "Voici le contexte : [Extraits du manuel]. En utilisant ce contexte, réponds à la question : [Question utilisateur]".

Generation (Génération) : Le LLM reçoit ce prompt enrichi. Il génère une réponse en langage naturel en utilisant les faits précis qu'on vient de lui fournir, sans avoir besoin de les connaître par cœur.



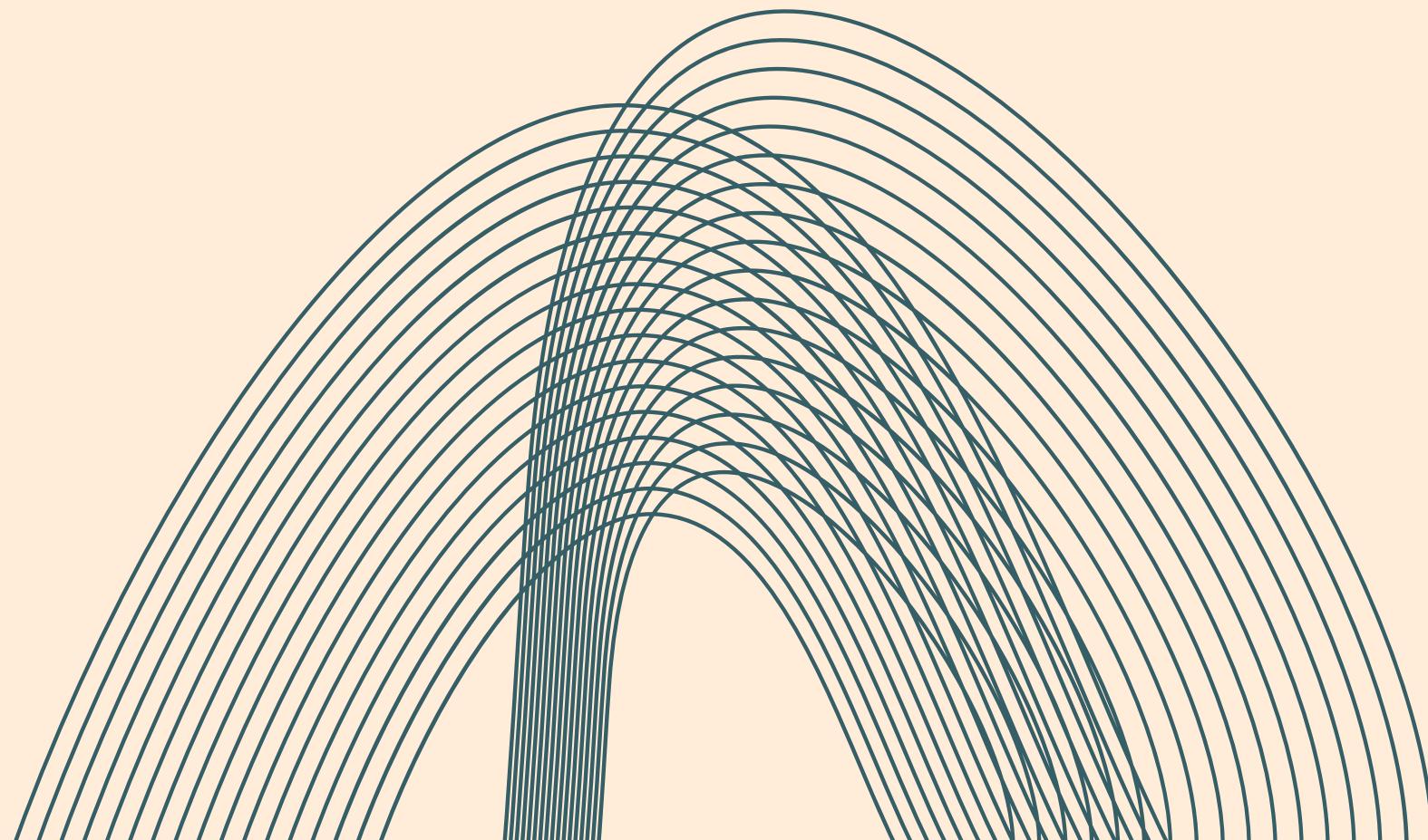


Autre Définitions :

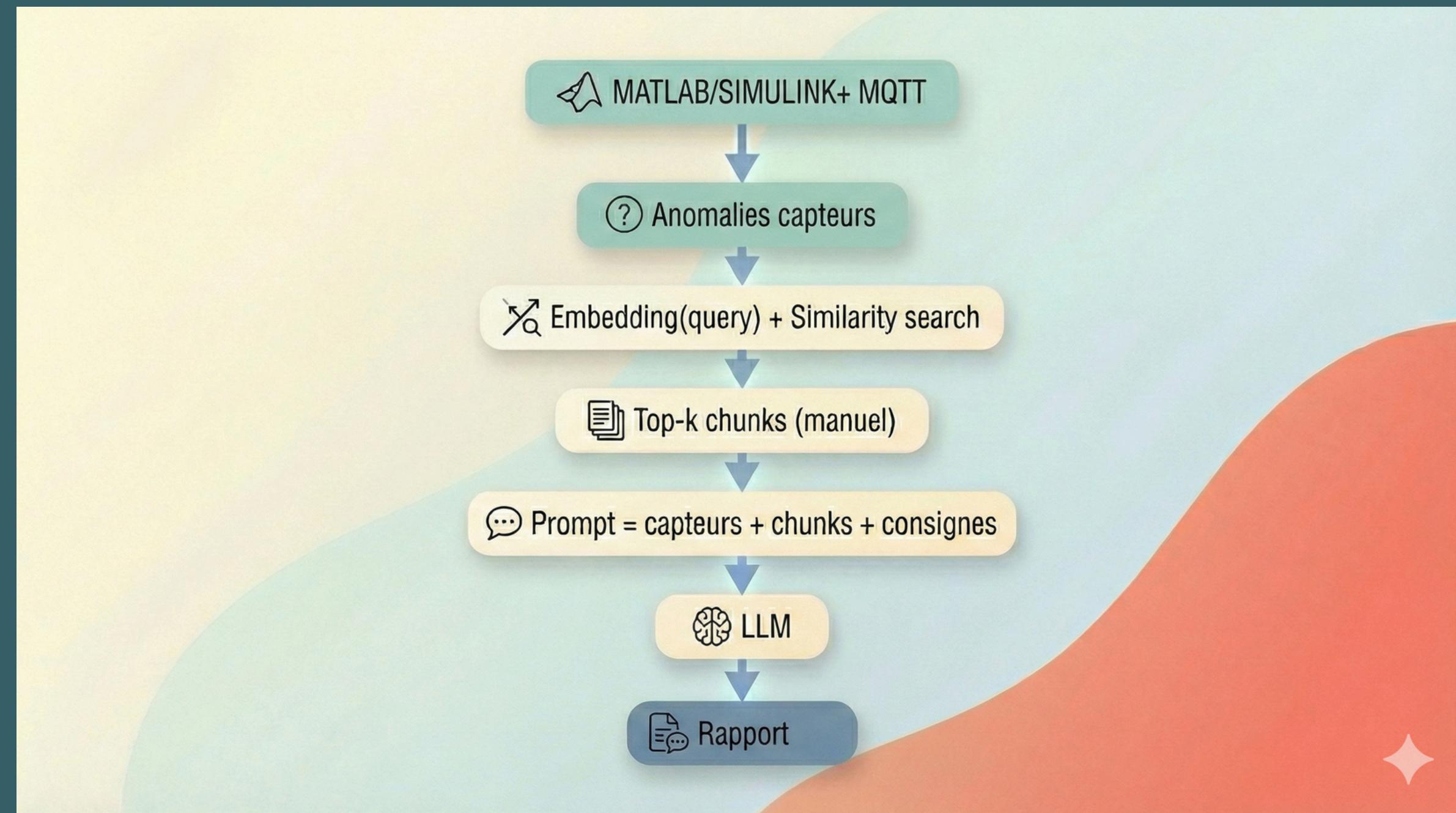
Prompt : entrée structurée du LLM (rôle + données + contexte + tâche).

Guardrails: filtres qui limitent le chatbot aux sujets maintenance.

Generation de rapport



Pipeline Rapport



1- Entrée du pipeline : données capteurs

- Source : MQTT (MATLAB/Simulink publie la télémétrie, le backend lit le dernier snapshot)
- Format : courant 3 phases + déséquilibre, tension, vibration, pression, température, fault_state + durée
- Ces données deviennent le « contexte temps réel » de l'IA

2- Transformation : capteurs → RAG query

Objectif : convertir des valeurs brutes en requête “texte” proche du vocabulaire du manuel.

- Exemple de règles (anomalies → mots-clés) : déséquilibre courant > 5% → `motor winding defect phase imbalance`

Résultat : une requête courte, informative, “search-friendly”.

3- Recherche dans ChromaDB (Vector Search)

On calcule l'**embedding** de la query (Google Embeddings)

- ChromaDB fait **une similarity search** dans la base du manuel
- On récupère top_k extraits (chunks) :
 - content (texte)
 - page (référence)
 - score (pertinence)

But : fournir au LLM des preuves / procédures issues du manuel.

4- Recherche dans ChromaDB (Vector Search)

On calcule l'**embedding** de la query (Google Embeddings)

- ChromaDB fait **une similarity search** dans la base du manuel
- On récupère top_k extraits (chunks) :
 - content (texte)
 - page (référence)
 - score (pertinence)

But : fournir au LLM des preuves / procédures issues du manuel.

4- Affichage Du Rapport



⚠ ATTENTION REQUISE

Continuer pour diagnostic, puis arrêter pour inspection

Avertissements:

- Temperature: **89.9°C** (seuil: 80°C)
- Phase Imbalance: **7.8%** (seuil: 5%)

■ Recommandation: Comme recommandé par Grundfos (Page 5): Ne pas arrêter immédiatement. Effectuer les mesures pendant le fonctionnement, puis arrêter pour correction.

● ARRÊTER APRÈS DIAGNOSTIC

PRIMARY DIAGNOSIS: Critical motor winding defect leading to severe overheating and significant phase current imbalance.

ROOT CAUSE: The elevated current draw in Phase C (11.07 A) and the resulting high phase imbalance (7.76%) indicate a developing short or degradation within the motor's winding. This condition is causing excessive resistive heating, driving the motor temperature to a critical 89.93 °C, confirmed by the system's "WINDING_DEFECT" fault state. This exceeds the 5% imbalance threshold for healthy operation [Manual Reference 3 - Page 7].

IMMEDIATE ACTIONS:

1. **EMERGENCY SHUTDOWN:** Immediately de-energize and lock out the pump motor. Continuing operation risks catastrophic failure, fire, and extensive damage due to critical overheating.
2. Allow the motor to cool down completely before any further inspection or testing.

VERIFICATION STEPS:

1. **Visual Inspection:** Check motor terminals and wiring for loose connections, discoloration, or signs of arcing/overheating.
2. **Winding Resistance Test:** Perform an ohmmeter test across all three motor winding combinations (L1-L2, L2-L3, L1-L3) as detailed in [Manual Reference 2 - Page 9]. Compare readings to manufacturer specifications; expect an abnormal (likely lower) resistance value for the affected winding.
3. **Insulation Resistance Test:** Use a megohmmeter to check the insulation resistance of the motor windings to ground, following instructions in [Manual Reference 2 - Page 9], to confirm insulation breakdown.

Troubleshooting Checklist (dynamique)

Après le diagnostic, une **checklist de dépannage** est générée pour guider le technicien.

Principe :

1. Le backend appelle `/api/logigramme` avec le type de défaut + le diagnostic
2. L'agent fait du **RAG** sur le manuel (procédures de dépannage)
3. Le LLM génère **5–7 étapes actionnables** avec marquage `[CRITICAL]`
4. Le front affiche les étapes dans une checklist interactive

Troubleshooting Checklist

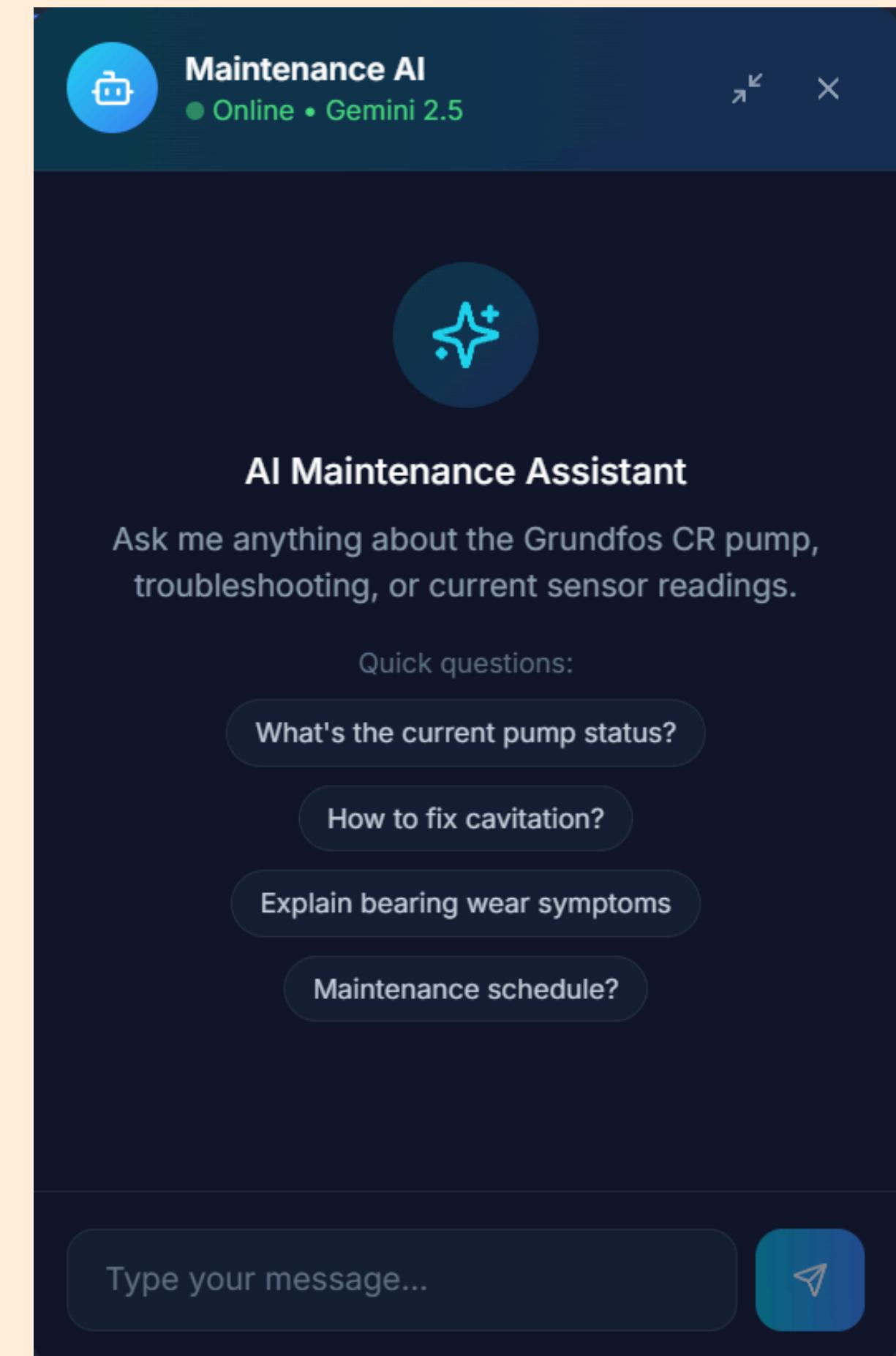
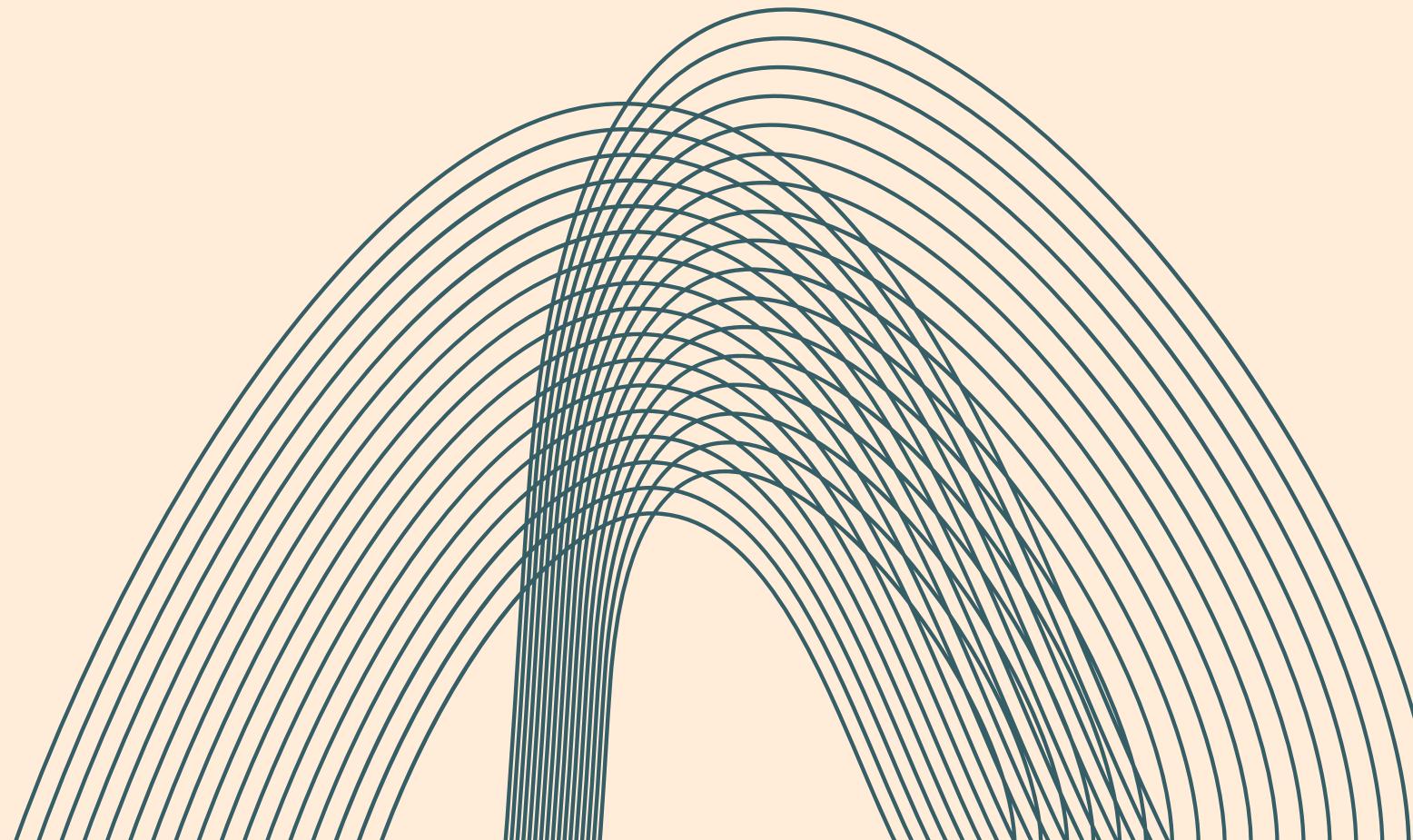
◆ Troubleshooting Checklist — Recommended actions for WINDING DEFECT

- 1 Isolate all power to motor. CRITICAL
- 2 Disconnect all electrical leads from motor.
- 3 Measure lead-to-lead winding resistance with ohmmeter.
- 4 Measure lead-to-ground insulation resistance with megohmmeter.
- 5 Compare all resistance readings to motor specifications.
- 6 If abnormal, remove and replace/repair the motor.

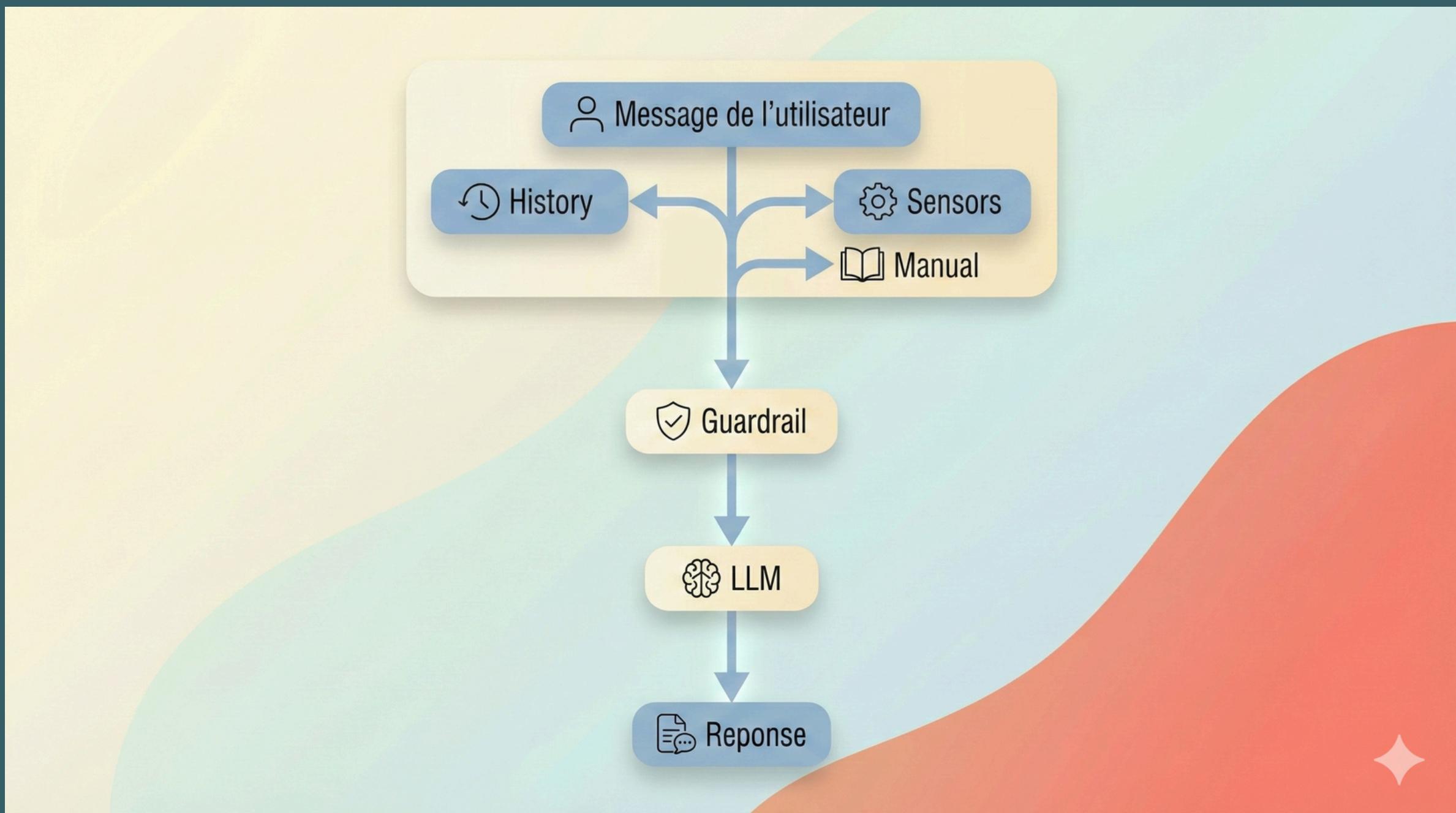
✓ Verify return to normal → Restart if OK

● Critical step ● Standard step ● Final validation

CHATBOT



Pipeline ChatBot



Pipeline ChatBot

Le Chatbot combine :

- Gardrails sous forme d'instruction
- Statut courant (snapshot capteurs)
- Contexte manuel (chunks RAG)
- Contexte événementiel (début du défaut)
- Mémoire long-terme (semantic memory : préférences/faits)

Guardrails : rester “maintenance”



- 1) Guardrail **au niveau backend** (on refuse avant d'appeler le LLM).
- 2) Guardrail **dans le prompt** (instructions strictes pour le style/règles).

Guardrails : rester “maintenance”

1) Guardrail au niveau backend :

Pourquoi filtrer avant le LLM ?

- Plus robuste qu'une simple consigne dans le prompt.
- Moins coûteux (pas d'appel LLM si hors sujet).
- Réduit les réponses “hors domaine” pendant la démo.

Comment on décide “maintenance” ? (principe)

- On détecte des : mots-clés liés à la pompe, capteurs, défauts, tests, actions, etc.
- Si aucun mots-clés : on refuse + on demande de reformuler.

Guardrails : rester “maintenance”

2) Guardrail dans le prompt (instructions strictes pour le style/règles).

Exemple :

```
prompt = f"""{self.system_prompt}"""
```

CHAT MODE INSTRUCTIONS (must follow):

- Reply in the same language as the user's question.
- Give a DIRECT answer to the user's question.
- Do NOT output headings like: DIAGNOSIS, ROOT CAUSE...
- If the safety evaluation indicates IMMEDIATE_SHUTDOWN, say to stop immediately.
- Only include the 1–2 most relevant verification checks.

```
"""
```

Mémoire de session (chat history)

Mémoire de session (chat history) — ce qu'on garde et comment:

garder le **contexte uniquement pendant la session de chat** (un seul chat), sans rien persister sur disque.

Ce que ça permet :

- Le chatbot “se souvient” de ce que l’utilisateur vient de dire (ex : “réponds en français”, “on parle de cavitation”).

Mémoire de session (chat history)

Comment c'est fait (principe).

1. Le front envoie un `session_id` avec chaque message.
2. Le backend garde une liste en mémoire RAM : derniers messages pour ce `session_id`.
3. À chaque question, on injecte cet historique dans le prompt sous {CHAT HISTORY}.

Perspectives & Comparaisons

Projet Actuel vs Pipeline Production + Modelica vs Simulink

Projet Actuel vs Pipeline Production

Ce qu'on a fait vs Ce qu'on pourrait améliorer

Architecture : Projet Actuel vs Production

❖ Projet Actuel (Ce qu'on présente)

Source de données

- ✓ Simulateur MATLAB/Simulink
- ✓ Génération de télémétrie synthétique
 - Pas de capteurs physiques réels

Communication

- ✓ MQTT Broker (Mosquitto)
- ✓ Topics telemetry / command
 - Sans TLS (développement)
 - Sans authentification

Backend

- ✓ FastAPI + WebSocket
- ✓ RAG avec ChromaDB
- ✓ Diagnostic IA (Gemini)

Frontend

- ✓ React Dashboard temps réel
- ✓ Injection de pannes manuelle

❖ Pipeline Production (Amélioration)

Source de données

- Capteurs réels (pression, vibration, T°...)
- PLC / VFD / DCS
- Edge Gateway + OPC UA

Communication

- MQTT avec TLS (MQTTSS)
- Certificats par device
- ACLs sur les topics
- QoS 1 pour commandes critiques

Backend

- Historian (PI, Maximo...)
- Intégration CMMS/EAM
- Détection d'anomalies ML

Sécurité

- DMZ OT/IT
- Firewall rules strictes



Le projet actuel est un POC fonctionnel, la production ajoute sécurité et intégrations

Rôle du Digital Twin : Actuel vs Production



Mode Actuel : Twin = Générateur

Pattern A : Twin publie télémétrie

- ✓ Simulink génère les données capteurs
- ✓ Backend consomme comme si c'était réel
- ✓ Idéal pour : Démo, Lab, HIL

Avantages

- Simple à mettre en place
- Pas besoin de pompe physique
- Scénarios de pannes contrôlés

Limitations

- Pas de comparaison attendu/mesuré
- Pas de détection d'écart



Mode Production : Twin = Estimateur

Pattern B : Twin consomme + compare

- Twin s'abonne à la télémétrie réelle
- Génère les valeurs attendues (modèle)
- Calcule les résidus :
 $r(t) = y_{\text{mesuré}}(t) - y_{\text{attendu}}(t)$

Avantages

- Détection de dérives réelles
- Diagnostic basé sur les écarts
- True 'Digital Twin' behavior

Implémentation

- Simulink subscribe + publish residuals
- Backend ingère mesuré + résidus



Le Pattern B est la vraie valeur ajoutée du Digital Twin en production

Perspectives d'Amélioration

1. Sécurité & Production

- Ajouter TLS sur le broker MQTT
- Implémenter l'authentification par certificats
- Configurer les ACLs par topic

2. Intégrations Industrielles

- Connecter un Historian (PI, InfluxDB)
- Intégration CMMS pour créer des ordres de travail
- API vers SCADA/HMI existants

3. Intelligence Artificielle

- Détection d'anomalies par ML (baseline par régime)
- Prédiction de durée de vie restante (RUL)
- Apprentissage continu sur données réelles

4. Digital Twin avancé

- Passer au Pattern B (Twin = Estimateur)
- Calcul de résidus en temps réel
- Fusion capteurs + modèle physique



Ces améliorations transformeraient le POC en solution industrielle

Modelica vs Simulink

Pourquoi avons-nous choisi Simulink ?

Modelica vs Simulink/MATLAB

Modelica (OpenModelica, Dymola)

Approche

- Langage de modélisation acausal
- Équations différentielles déclaratives
- Orienté composants physiques

Avantages

- ✓ Modèles très précis physiquement
- ✓ Réutilisabilité des composants
- ✓ Standard ouvert (MSL)

Inconvénients

- ✗ Courbe d'apprentissage élevée
- ✗ Connaissance requise :
 - Équations différentielles
 - Mécanique des fluides
 - Thermodynamique
- ✗ Debug plus complexe
- ✗ Moins d'exemples industriels

Simulink / MATLAB

Approche

- Modélisation par blocs (causal)
- Diagrammes visuels intuitifs
- Simscape pour la physique

Avantages

- ✓ Interface graphique intuitive
- ✓ Documentation abondante
- ✓ Exemples industriels nombreux
- ✓ Intégration MQTT native
- ✓ Toolboxes spécialisées
- ✓ Support MathWorks

Inconvénients

- ✗ Licence coûteuse
- ✗ Moins flexible que Modelica
- ✗ Propriétaire (vendor lock-in)



Pour un projet académique avec délai , Simulink était le choix pragmatique

Difficultés Rencontrées avec Modelica

1. Complexité Mathématique

- Nécessite une maîtrise des équations différentielles
 - Modélisation des fluides : Navier-Stokes, pertes de charge
 - Thermodynamique : transferts de chaleur, enthalpie
- Formation significative requise avant d'être productif

2. Debugging Difficile

- Erreurs de compilation peu explicites
- Systèmes d'équations sur/sous-déterminés
- Singularités numériques fréquentes

3. Intégration MQTT

- Pas de support natif comme Simulink
- Nécessite des wrappers C/Python externes
- Configuration plus laborieuse

4. Contrainte de Temps

- Projet académique avec deadline
- Simulink : productif en quelques jours
- Modelica : plusieurs semaines de formation



Modelica reste pertinent pour des modèles haute-fidélité à long terme

Récapitulatif des Choix Techniques

Ce qu'on a fait (Projet Actuel)

- ✓ Architecture MQTT découpée et fonctionnelle
- ✓ Simulink pour le Digital Twin (pragmatique)
- ✓ FastAPI + WebSocket pour le temps réel
- ✓ RAG + Gemini pour le diagnostic IA
- ✓ Dashboard React interactif

Ce qu'on pourrait améliorer (Perspectives)

- Sécurité : TLS, certificats, ACLs
- Intégrations : Historian, CMMS, SCADA
- Twin Pattern B : Estimateur avec résidus
- ML : Détection d'anomalies, prédition RUL

Questions ?
