

## Développement d'une Solution de Relevé de Données Process

Dans un environnement industriel de plus en plus connecté, disposer d'outils capables de récupérer et d'exploiter les données des process est devenu essentiel pour surveiller en temps réel, prévenir les pannes, optimiser la consommation énergétique, et maintenir les standards de qualité. Ces données contribuent ainsi à une meilleure performance globale et à la réduction des coûts.

Cependant, les outils de supervision actuels, souvent basés sur des technologies propriétaires, présentent plusieurs inconvénients :

- **Coût élevé** : Des licences et matériels onéreux qui impactent les budgets d'investissement.
- **Dépendance aux fournisseurs externes** : Un manque d'autonomie pour ajuster et personnaliser les outils en fonction des besoins spécifiques.
- **Rigidité** : Des systèmes peu flexibles, limitant leur adaptabilité aux évolutions rapides des besoins.

## Enjeux & Défis

Ce projet vise à créer une **solution de suivi indépendante et flexible**, permettant aux industriels de surveiller leurs process en toute autonomie et à moindre coût, tout en répondant à différents besoins. Cette solution permettrait :

- **Une indépendance technologique** : S'affranchir des solutions propriétaires pour éviter les coûts de licence et gagner en autonomie.
- **Adaptabilité et déploiement rapide** : Mettre en place les outils de suivi efficacement, notamment en cas de besoin urgent de visibilité sur les données.
- **Flexibilité et modularité** : Adapter les systèmes de suivi aux besoins spécifiques de chaque installation.

En s'appuyant sur des technologies open source, cette solution offre aux industriels un contrôle total sur leur suivi de données sans dépendance vis-à-vis de fournisseurs tiers. Elle est conçue pour être facilement déployable, modulable, et paramétrable afin que les techniciens puissent définir eux-mêmes les automates et variables à surveiller. Plus qu'une simple solution technique, il s'agit de développer un **outil stratégique pour l'industrie**, qui répond aux exigences d'autonomie et de résilience tout en restant accessible financièrement.

## Contexte du Hackathon

Pour répondre à ces enjeux, vous participerez à un **hackathon industriel**. Organisé en mode collaboratif, il vous place dans les conditions d'un besoin industriel. En équipes, vous allez concevoir une solution innovante et open source, capable de s'adapter aux nouvelles exigences des entreprises.

Pendant ce hackathon, vous devrez relever des défis concrets : déployer une application fonctionnelle sur un réseau local, collaborer en équipe, et produire une documentation pour rendre votre solution accessible et réutilisable. Ce projet est une occasion de renforcer vos compétences techniques tout en découvrant les exigences et les attentes du secteur industriel.

# HACKATHON



## Description de l'application

Votre mission est de concevoir un prototype de solution de suivi modulaire et portable, pensé pour s'adapter à divers environnements industriels. Cette application open source doit fournir aux industriels un outil de suivi indépendant et flexible, capable de surveiller en temps réel les variables critiques d'automates (par exemple, température, pression).

Elle doit permettre aux utilisateurs de configurer facilement les adresses IP des automates et les variables à surveiller via une interface intuitive. Les données collectées seront affichées sur un tableau de bord interactif pour faciliter l'analyse et la prise de décision. Pensée pour un déploiement rapide sur un réseau local, cette application représente une alternative aux solutions propriétaires, offrant une autonomie totale tout en minimisant les coûts.

## Objectifs de l'application

- Paramétrage des variables** : Une interface simple pour configurer les paramètres des automates et définir les variables à surveiller.
- Supervision en temps réel** : Un tableau de bord clair et interactif pour visualiser les variables monitorées.
- Utilisation de solutions open source** : Un développement basé sur des technologies libres pour garantir indépendance et maîtrise des coûts.
- Déploiement rapide** : Une architecture permettant une installation en quelques minutes, adaptée aux besoins urgents de supervision.

## Organisation

Chaque équipe sera composée de deux étudiants, qui devront choisir un nom d'équipe pour identifier leur projet. Les étudiants auront la possibilité de travailler soit sur leurs ordinateurs personnels, soit sur les postes informatiques de l'école. Les livrables devront être rendus en temps et en heure afin de garantir l'équité entre les équipes et de respecter le planning de l'évaluation.

## Délai

Les équipes disposent de 29 heures planifiées dans l'emploi du temps pour concevoir et développer leur application. Une bonne gestion de ce temps sera essentielle pour mener à bien le projet. Lors de la dernière séance, chaque équipe devra présenter une démonstration fonctionnelle de sa solution. Afin de garantir l'équité entre les équipes, les livrables finaux, incluant le dépôt du code, la documentation, et la vidéo de présentation, devront être rendus au plus tard le dimanche 08 décembre à minuit, marquant la fin de la période école.

## Parcours utilisateur envisagé

Le parcours utilisateur met en lumière les fonctionnalités clés auxquelles les techniciens ou ingénieurs auront accès pour surveiller et exploiter les données en temps réel et historique.

1. **Accès à l'application :** L'application est accessible via un navigateur sur le réseau local de l'entreprise. L'utilisateur (technicien, ingénieur, etc.) peut accéder aux fonctionnalités de configuration et de supervision.
2. **Configuration des paramètres :** L'utilisateur peut ajouter ou modifier les paramètres des variables à surveiller. Pour chaque variable, il peut définir le nom, l'adresse IP de l'automate, la variable à lire, et la fréquence d'enregistrement. Ces paramètres sont sauvegardés dans la base de données pour être utilisés par le backend.
3. **Supervision en temps réel :** Pour chaque variable surveillée, un tableau de bord affiche les données en temps réel sous forme de graphiques. Ce graphique se met à jour automatiquement en fonction de la fréquence d'enregistrement configurée. Cette interface simple permet à l'utilisateur d'interpréter les résultats rapidement sans autre logiciel spécifique.
4. **Consultation des données historiques :** Les données des variables surveillées peuvent être exportées via une interface dotée d'un formulaire permettant de définir une plage de dates. L'export se fait au format CSV, afin de faciliter leur exploitation dans d'autres outils spécifiques ou leur partage avec d'autres équipes.

## Parcours Backend envisagé

Le parcours backend détaille les processus invisibles qui orchestrent la récupération des données des automates, leur stockage et leur mise à disposition pour le frontend.

1. **Initialisation et Chargement des Paramètres :** Au démarrage, le backend se connecte à la base de données pour récupérer la liste des variables à surveiller (IP des automates, registres à lire, fréquence, etc.). Il initialise les tâches périodiques de supervision en fonction des paramètres configurés.
2. **Surveillance et Lecture des Automates :** Le backend lit les variables automates à surveiller, à intervalles réguliers, en fonction de la fréquence définie pour chaque variable dans la base de données. Il gère les connexions Modbus et gère les erreurs en cas de défaillance de connexion ou de lecture.
3. **Enregistrement des Données en Base :** Après chaque lecture, les valeurs récupérées sont sauvegardées dans la base de données, avec un horodatage. Cette historisation permet la consultation et l'export des données par les utilisateurs.
4. **API REST pour le Frontend :** Le backend expose des routes pour permettre au frontend de notamment :
  - Ajouter, modifier ou supprimer des paramètres de surveillance.
  - Afficher les valeurs de la variable surveillée pour interprétation sous graphique depuis le frontend.
  - Récupérer les données historiques pour une période donnée et les mettre à disposition en téléchargement via un fichier CSV.
5. **Gestion des Tâches Périodiques :** Le backend surveille en continu les variables configurées grâce à des tâches périodiques (par exemple, avec node-cron ou un équivalent). Ces tâches s'adaptent dynamiquement : si des modifications sont apportées dans la configuration (par exemple, ajout d'une nouvelle variable), elles sont prises en compte sans nécessiter de redémarrage du serveur.
6. **Déploiement et Accessibilité :** Le backend est déployé dans un conteneur Docker et configuré pour écouter sur un port spécifique. Il est accessible depuis d'autres machines du réseau pour communiquer avec le frontend ou d'autres services.

# Livrables attendus & Technologies à utiliser

## 1. Développement de l'Application

- **Frontend** : Créer une interface utilisateur intuitive qui affiche en temps réel les données collectées et configurées par chaque équipe.
- **Backend** : Mettre en place une API ou un service backend pour gérer la logique métier, interagir avec la base de données, et traiter les requêtes du frontend.
- **Base de Données (BDD)** : Structurer une base de données pour stocker les informations essentielles, telles que les paramètres suivis, l'historique des données, et les configurations d'automates.

→ Outils / Technos à utiliser : HTML, CSS, Javascript, NodeJS, ExpressJS, MariaDB.

→ Librairies utiles : [Bootstrap](#), [ChartJS](#), [Nodemon](#), [Modul Serial](#), [Export-To-CSV](#), [Node-cron](#), [Dotenv](#).

## 2. Versionning et Travail en Équipe (GitHub)

1. **Branches et Workflows** : Utiliser un système de branches pour organiser le travail (ex. main, develop, etc.), permettant un développement structuré et collaboratif.
2. **Pull Requests (PR)** : Chaque équipe doit soumettre ses changements via des PRs, avec une revue par les coéquipiers avant fusion dans les branches principales, pour encourager les bonnes pratiques de revue de code.
3. **Commits Clairs et Organisés** : Assurer des commits descriptifs pour documenter chaque étape du développement.
4. **README et Documentation de Dépôt** : Inclure un fichier README complet, décrivant la configuration de l'application, les technologies utilisées, et les instructions pour le lancement et le test de l'application.

→ Outils / Technos à utiliser : Git, Github.

## 3. Déploiement en Réseau Local avec Docker

- **Dockerisation** : Encapsuler le frontend, le backend, et la base de données dans des conteneurs Docker, et configurer un fichier docker-compose.yml pour orchestrer les différents services.
- **Déploiement Local Simulé** : Configurer le déploiement sur un PC simulant un serveur, de manière à rendre l'application accessible depuis un autre ordinateur du réseau local.
- **Configuration Réseau** : Paramétriser le réseau Docker pour permettre la communication entre les conteneurs (backend, frontend, BDD).
- **Documentation de Déploiement** : Ajouter des instructions détaillées dans le README pour le déploiement et l'accès à l'application depuis un autre poste du réseau, avec des informations claires sur la configuration réseau et les accès.

→ Outils / Technos à utiliser : WSL, Docker : Frontend [image standard Nginx] + Backend [fichier Dockerfile spécifique au besoin] + Database [image standard MariaDB]

## Evaluation & Notation

La notation finale sera décomposée selon plusieurs catégories :

### 1. Démonstration en live [30%]

Lors de la dernière séance, chaque équipe présentera sa solution lors d'une démonstration en direct, où elle pourra présenter les fonctionnalités de l'application et répondre aux questions. Cette évaluation permettra de juger la capacité des participants à défendre leur travail et à s'adapter aux retours.

- Critères : Fonctionnalités, fluidité, ergonomie, originalité.
- Echelle : Non fonctionnel, partiellement fonctionnel, fonctionnel et fluide.

### 2. Audit du code [30%]

Un audit sera réalisé via le dépôt GitHub de chaque équipe. Seront évalués l'organisation des branches, la clarté des commits, l'architecture du code, et l'utilisation des pull requests. Cet audit mettra l'accent sur les bonnes pratiques de développement et la collaboration en équipe.

- Critères : Travail d'équipe avec utilisation de Github (branches, commits, PR, etc.).
- Echelle : Non structuré, partiellement structuré, bien structuré et complet.

### 3. Documentation [10%]

La qualité de la documentation est essentielle pour la clarté et la réutilisabilité du projet. Les équipes devront fournir un README simple et efficace. Une documentation bien structurée et claire permet de rendre le projet accessible à d'autres personnes.

- Critères : README, bon équilibre de commentaires dans le code.
- Echelle : Incomplet, quelques éléments, exhaustif et équilibré.

### 4. Rapport technique [30%]

Chaque équipe complètera le rapport technique sous la trame défini. Ce rapport permettra d'évaluer différents critères :

- Rappel du contexte et objectif de l'application.
- Architecture technique, schéma de fonctionnement.
- Fonctionnalités implémentées.
- Conclusion et retour d'expérience.

### 5. Classement au Hackathon [bonus]

Un classement final sera établi pour le hackathon, prenant en compte l'ensemble des critères précédents et l'appréciation du jury. Des points bonus seront attribués aux meilleures équipes selon le classement ci-dessous :

1. + 2 points
2. + 1 point
3. + 0.5 point