Status page Grafana pour MessMass

Rapport d'étudiant en ingénierie
Stage de fin d'études
Année académique 2024-2025

Présenté par :
Mathis Roubille

Encadrant académique :
Renaud Chicoisne

Tuteur de stage :
Yann Segaud

Durée du stage : 5 mois

August 20, 2025

Remerciements

Je tiens à exprimer ma profonde gratitude envers les personnes qui m'ont accompagné tout au long de ce stage et qui ont largement contribué à la réussite de cette expérience professionnelle et personnelle.

Je remercie tout d'abord M. Yann Segaud, mon tuteur d'entreprise, pour sa disponibilité, sa confiance et la qualité de son encadrement. Son expertise, sa pédagogie et sa vision produit ont été déterminantes dans la réussite du projet de status page. Ses conseils et son exigence m'ont permis de progresser techniquement et méthodologiquement.

Je souhaite exprimer ma reconnaissance à M. Christophe Emotte, manager de l'équipe HIP, l'équipe dans laquelle j'ai été intégré durant mon stage, pour m'avoir accueilli et offert l'opportunité de participer à un projet stratégique. Son soutien constant et sa vision d'ensemble ont été précieux pour comprendre les enjeux de l'observabilité au sein de Michelin.

Je souhaite aussi remercier mon tuteur académique, M. Renaud Chicoisne, pour son accompagnement tout au long du stage et pour ses retours constructifs.

Enfin, je remercie l'ensemble des membres des squads HIP avec lesquels j'ai eu l'occasion de collaborer, ainsi que tous ceux qui ont contribué, de près ou de loin, à l'avancement de ce projet.

Contents

List of Figures

Résumé

Ce rapport retrace le développement d'une status page au sein de l'équipe MessMass de Michelin, dans le cadre du programme OneSystem IT Platforms. Ce projet, mené sur 24 semaines à partir d'avril 2025, avait pour but de proposer une solution de monitoring temps réel des middlewares critiques, en s'appuyant sur Grafana et Prometheus. L'interface mise en place permet de visualiser en un coup d'œil l'état de fonctionnement de quinze middlewares internes et externes, avec des indicateurs visuels, des historiques d'incidents, SLIs et un drill-down. Un accent particulier a été mis sur la documentation technique et utilisateur, la standardisation graphique et la maintenabilité. Des pistes sont actuellement explorées pour permettre un accès public à la status page, qui reste aujourd'hui réservée aux utilisateurs disposant d'un accès Grafana. La status page est désormais considérée comme un modèle pour les futures initiatives d'observabilité au sein de la direction DOTI.

Mots-clés : Status Page, Grafana, Middleware, Monitoring, K6, Prometheus, DOTI, Observabilité, Michelin, OneSystem IT Platforms, MessMass, Documentation, SLI, SLO, Standardisation, Dashboard Public, Diagnostic Opérationnel.

Abstract

This report outlines the design and implementation of a status page within Michelin's MessMass team, under the OneSystem IT Platforms program. Conducted over a 24-week internship starting in April 2025, the project aimed to build a real-time observability tool using Grafana and Prometheus to monitor fifteen critical internal and external middlewares. The dashboard includes visual indicators, historical uptime data, SLIs, and drill-down access for root cause analysis. Focus was placed on both user-facing and technical documentation, visual standardization, and maintainability. Research is currently underway to enable public access to the dashboard, which is currently restricted to Grafana users. The resulting status page is now regarded as a reference model for future observability initiatives across DOTI.

Keywords: Status Page, Grafana, Middleware, Monitoring, K6, Prometheus, DOTI, Observability, Michelin, OneSystem IT Platforms, MessMass, Documentation, SLI, SLO, Public Dashboard, Operational Diagnosis, Technical Standard.

Glossary

ACE IBM App Connect Enterprise, un middleware d'intégration orienté flux et API, permettant de connecter, transformer et orchestrer des données entre systèmes hétérogènes. 16

Agile@Scale Approche d'extension des méthodes agiles à l'ensemble d'une organisation, permettant de coordonner plusieurs équipes autour de cycles courts, avec backlog unifié et synchronisation planifiée. 1, 5, 7, 8

API Application Programming Interface, interface de programmation permettant à deux logiciels de communiquer entre eux. 1, 2, 6–8, 10, 15, 24, 28, 36, 41, II, IV, V

Backlog Liste priorisée des tâches à réaliser par une équipe agile durant ses sprints. 6, 36

Blackbox Exporter Outil Prometheus permettant de vérifier la disponibilité de services en réalisant des tests de type ping ou HTTP sans authentification complexe. 19, 22, 29, 36, 37, 39, IV, V

Boomi Plateforme iPaaS (integration Platform as a Service), facilitant la création de workflows, APIs et intégrations B2B en low-code. 7, 8, 15, 29, 31, III, V

CDC Change Data Capture, mécanisme permettant de répliquer les modifications de données en quasi temps réel entre systèmes. 15

CI/CD Continuous Integration / Continuous Deployment, pratiques DevOps consistant à intégrer et déployer en continu du code dans un système de production. 5, 7, 19

DevOps Culture et ensemble de pratiques visant à unifier le développement logiciel (Dev) et les opérations informatiques (Ops). 5, 7

ETL Extract Transform Load, processus permettant d'extraire, transformer et charger des données entre systèmes. 6, 7, 15

Grafana Outil open-source de visualisation de données permettant de créer des dashboards personnalisés pour le monitoring en temps réel. 16–22, 25, 28, 30, 33, 36, 39, 44–46, II, III

HashiCorp Vault Outil de gestion de secrets et de contrôle d'accès aux données sensibles, intégré dans les pipelines CI/CD. 19

IBM International Business Machines, entreprise proposant notamment des middlewares tels que MQ, DataStage ou DataPower. 6, 8, 15, 16, 32

iPaaS Integration Platform as a Service, plateforme d'intégration cloud permettant de connecter applications et flux de données avec peu de code. 15, 31

K6 Outil open-source de test de performance permettant de simuler des charges utilisateur sur des APIs ou services web. 1, 2, 19–21, 31, 32, 46, III, VI

Kubernetes Plateforme open source d'orchestration de conteneurs permettant d'automatiser le déploiement, la mise à l'échelle, la gestion et la supervision d'applications conteneurisées. Développé initialement par Google, Kubernetes est aujourd'hui maintenu par la Cloud Native Computing Foundation (CNCF). 6, 37, 38, III

Loki Composant de la suite Grafana pour l'agrégation et l'exploration de logs. 20, 32, 33

low-code Approche de développement logiciel avec peu de code, via interfaces graphiques ou composants préfabriqués. 7, 15

MFT Managed File Transfer, technologie de transfert de fichiers sécurisé et automatisé entre systèmes informatiques, souvent utilisée pour les transferts critiques inter-entreprises. 16, 41

middleware Logiciel d'intermédiation permettant les échanges entre applications ou systèmes informatiques hétérogènes. 1, 5–11, 13–17, 24–27, 29, 32, 45, 46, II, VI

Mimir Composant de la suite Grafana pour le stockage scalable de métriques, successeur de Cortex. 19, 20

monitoring Surveillance technique des systèmes informatiques via des outils de métriques, logs et alertes. 2, 5, 9, 16, 17, 27–29, 31, 32, 34, 36, 37, 39, 46, III, VI

MVP Minimum Viable Product, version minimale d'un produit permettant de valider ses usages avec le moins d'effort possible. 6

observabilité Capacité à diagnostiquer le comportement interne d'un système à partir de ses sorties (logs, traces, métriques). 1, 2, 7, 9, 17, 19–21, 45, 46, I, III, VI

on-premise Système d'information ou application déployé sur l'infrastructure interne d'une entreprise, par opposition au cloud. 6

OpenTelemetry Framework open-source permettant la collecte unifiée de métriques, logs et traces pour l'observabilité. 1, 7, 17–21, 28, III, V

PoC Proof of Concept, prototype ou expérience destinée à valider la faisabilité d'une idée technique ou fonctionnelle. 2, 20, 28, 31, 39, 46, III–V

product-centric Approche organisationnelle où chaque équipe est responsable d'un produit complet et autonome, comme une IT Platform. 5

Prometheus Base de données temporelles open-source utilisée pour la collecte et la consultation de métriques dans les systèmes de monitoring. 1, 10, 17, 18, 20, 27, 30, 36, 46, VI

R&D Recherche et Développement, ensemble des activités visant à créer des innovations techniques ou scientifiques. 5

rill-down Fonctionnalité de navigation dans les dashboards permettant d'accéder à des vues plus détaillées à partir d'un indicateur général. 1, 25, 29, 30, 34–38, 44, V, VI

SaaS Software as a Service, modèle de distribution où les applications sont hébergées par un fournisseur et accessibles via Internet. 15

ServiceNow Outil ITSM (IT Service Management) pour la gestion des incidents, demandes et changements dans les services informatiques. 6, 25, 42, 44

SLI Service Level Indicator, métrique quantifiant un aspect d'un service (disponibilité, latence, erreurs, etc.) pour en évaluer la qualité. 1, 14, 24–27, 31, 32, 41, 42, 44, II, V, VI, IX

SLO Service Level Objective, objectif chiffré à atteindre pour une métrique SLI, souvent exprimé en pourcentage. 14, 25, 26, 32, 44, 46, II, V, VI, IX

Splunk Plateforme propriétaire d'analyse de données machine, spécialisée dans la collecte, l'indexation et la recherche en temps réel des journaux (*logs*) et événements applicatifs. Utilisée pour la supervision, l'alerting, la sécurité et la conformité, avec des fonctionnalités avancées de machine learning et de corrélation de données. 7, 9

SRE Site Reliability Engineering, discipline issue des pratiques de Google visant à appliquer des principes d'ingénierie logicielle à la gestion des systèmes en production, en s'appuyant sur des métriques comme les SLI et des objectifs tels que les SLOs pour améliorer la fiabilité et la disponibilité des services. 26

Tempo Composant de la suite Grafana pour la collecte et l'analyse des traces applicatives. 20

urn rate Indicateur estimant la vitesse à laquelle le budget d'erreur d'un SLO est consommé, utilisé dans les dashboards SRE. 25, 26, V

1 Introduction

Depuis le 1er avril 2025, j'ai eu l'opportunité d'effectuer mon stage de fin d'études au sein du groupe Michelin, dans l'équipe MessMass intégrée au programme OneSystem IT Platforms. Durant 24 semaines, j'ai été chargé de développer une interface stratégique de surveillance temps réel des services middleware internes et externes, connue sous le nom de "Status Page". Cette mission s'inscrivait dans un contexte de transformation numérique plus large du Groupe, qui vise à renforcer son efficacité opérationnelle tout en réduisant les coûts liés aux outils d'observabilité historiques.

Michelin, en tant qu'acteur industriel mondial, s'appuie sur une infrastructure technique complexe où les middlewares jouent un rôle central dans le bon fonctionnement des applications métier. L'absence de visibilité immédiate sur leur état de fonctionnement engendrait jusqu'alors des délais importants dans la détection et la résolution des incidents. C'est dans ce cadre que la création d'une status page dynamique, fiable, publique et centralisée est apparue comme un levier d'amélioration concret de la réactivité opérationnelle.

Ce projet a été développé avec l'outil Grafana, solution open-source, retenue pour sa flexibilité, sa compatibilité avec les outils internes (Prometheus, OpenTelemetry, K6) et sa capacité à fournir des tableaux de bord adaptés à différents profils d'utilisateurs. Ma mission consistait ainsi à prototyper, concevoir, standardiser et publier une interface accessible même aux utilisateurs non techniques, tout en garantissant un haut niveau de pertinence des données affichées.

Pour cela, j'ai commencé par une phase d'onboarding, incluant des formations internes (cybersécurité, outils internes, méthodologies Agile@Scale) et un apprentissage autonome de Grafana. Une exploration du code source open-source de Grafana m'a également permis de mieux comprendre ses capacités techniques, au-delà de la simple utilisation en surface.J'ai rapidement établi des points de contact avec les experts techniques des trois squads du HIP (Fusion, Atom, Integration Mastering), ce qui m'a permis de cerner les indicateurs critiques de chaque middleware : taux de disponibilité, erreurs de transferts, état des files d'attente, etc.

Un des enjeux majeurs était de proposer un standard graphique lisible, compréhensible immédiatement et pertinent pour un public métier comme pour les équipes techniques. J'ai donc mis en place un système d'indicateurs visuels (couleurs, icônes, jauges, SLIs sur 28 jours) avec une organisation en deux colonnes (MessMass et API). Chaque cellule de la page intègre un accès au "drill-down" pour détailler les causes d'un incident, ainsi qu'un historique temporel en tranches de 6 heures pour identifier les récurrences.

Ma position transverse au sein de l'équipe MessMass m'a permis d'interagir avec l'ensemble des équipes HIP. J'étais sous la responsabilité directe de Yann Segaud (Product Owner MessMass) et encadré hiérarchiquement par Christophe Emotte (Manager HIP). Je collaborais également avec Simon Dallet, alternant en charge de la partie API de la status page, avec qui j'ai co-défini l'architecture globale de la status page.

Plusieurs cas d'usage spécifiques ont ensuite structuré la suite du stage : supervision des moteurs IWS/DataStage, suivi des files MQ/WMQ, intégration de CFT et FTB via scripts K6 ou log monitoring, etc. Chaque problématique a nécessité une approche personnalisée, des tests, parfois des PoC, mais toujours avec la même rigueur : proposer une solution maintenable, validée par les experts et alignée avec la stratégie d'observabilité de Michelin.

Enfin, une attention particulière a été portée à la documentation du projet, tant pour assurer la continuité de l'outil que pour transmettre les bonnes pratiques aux équipes qui prendront la suite. Ce rapport de stage détaille les différentes étapes de la réalisation de ce projet, ses difficultés, ses réussites, et les perspectives qu'il ouvre pour l'avenir de l'observabilité chez Michelin.

2 Présentation de Michelin

2.1 Le Groupe Michelin et sa gouvernance

Le Groupe Michelin, fondé en 1889 à Clermont-Ferrand par André et Édouard Michelin, figure parmi les leaders mondiaux de la fabrication de pneumatiques, avec plus de 121 sites industriels et 9 centres de recherche et développement dans 26 pays. Il produit plus de 166 millions de pneus par an et emploie près de 132 500 collaborateurs [1]. Grâce à des innovations emblématiques du pneu démontable (1891) à la carcasse radiale (1946) il s'est affirmé comme un pionnier de la mobilité, couvrant non seulement le secteur automobile, mais aussi l'aéronautique, le ferroviaire et le spatial [1].

La gouvernance du groupe repose sur une structure de commandite par actions, dissociant gestion opérationnelle et supervision stratégique. La *Gérance*, formée de deux gérants (Florent Menegaux, Président de la Gérance et associé commandité, et Yves Chapot, gérant non-commandité), assure la direction au quotidien. Le Conseil de Surveillance, composé de onze membres élus pour quatre ans incluant représentants des actionnaires, du personnel et personnalités extérieures garantit le contrôle et la conformité des décisions stratégiques. Par ailleurs, la Société Auxiliaire de Gestion (SAGES) détient une part du capital, visant à préserver l'indépendance de la Gérance et l'équilibre des pouvoirs.

Sur le plan social, Michelin investit plus de 240 millions d'euros chaque année dans la formation de ses collaborateurs, notamment à travers la Manufacture des Talents et le Hall 32, pour renforcer la mobilité interne et l'employabilité à long terme. Toutefois, dans un contexte économique mondialisé tendu, avec une concurrence asiatique prononcée, le groupe a annoncé un gel partiel des recrutements afin de privilégier la qualité des profils, et la fermeture programmée des usines de Cholet et Vannes (1 254 postes concernés) d'ici début 2026, avec mise en place de dispositifs de reclassement et d'indemnisation pour les salariés impactés [1; 2].

Au quotidien, Michelin fonctionne selon une organisation matricielle, alliant un pilotage centralisé par la direction de site et les fonctions Groupe avec un management de proximité au sein de chaque usine ou centre de service, où les responsables locaux coordonnent la performance opérationnelle, la sécurité et la qualité. Cette structure permet de conjuguer réactivité locale et cohérence stratégique, tout en intégrant les enjeux de digitalisation et de durabilité portés par les équipes DOTI et OneSystem IT Platforms.

2.2 Valeurs et engagements RSE de Michelin

La responsabilité sociétale de Michelin s'articule autour de sa raison d'être *All Sustainable*, qui vise l'équilibre entre performance économique, respect de l'environnement et développement des personnes [3]. Cet engagement se traduit par des trajectoires climatiques de long terme (neutralité carbone visée à l'horizon 2050) et par une ambition de circularité renforcée, notamment l'augmentation progressive de la part de matériaux renouvelables ou recyclés dans les pneumatiques, avec une cible de 100% de matériaux durables à horizon 2050, assortie d'étapes intermédiaires [1; 4]. Sur le plan social, le Groupe met en avant la santé et la sécurité, la formation continue et la mobilité interne, en cohérence avec sa culture industrielle et ses dispositifs de développement des compétences (Manufacture des Talents, Hall 32) [1]. Enfin, l'entreprise déploie des actions d'écoconception, de sobriété énergétique sur ses sites, et de partenariats dans l'économie circulaire (collecte, régénération de matières), éléments présentés et suivis dans ses publications extra-financières annuelles [5].

Si ces orientations traduisent une volonté claire de positionner Michelin comme acteur responsable, leur concrétisation dépendra fortement de la mise en place d'objectifs intermédiaires mesurables et audités. L'échéance de 2050, bien que mobilisatrice, reste lointaine et peut diluer l'urgence d'actions concrètes à court terme. Par ailleurs, les ambitions de circularité reposent sur des innovations technologiques et des filières d'approvisionnement encore en développement, dont la viabilité à grande échelle demeure incertaine. Enfin, les engagements sociaux affichés coexistent avec des décisions industrielles comme la fermeture de sites, ce qui invite à questionner l'équilibre réel entre impératifs économiques et responsabilités sociétales.

2.3 La Direction Opérationnelle de la Transformation Digitale et du système d'Information (DOTI)

Le département DOTI pilote la transformation numérique du Groupe et l'intégration des systèmes d'information. DOTI regroupe environ 4 800 collaborateurs répartis dans plus de 20 pays, avec des antennes majeures à Clermont-Ferrand, Lyon et Paris en France, Greenville et Charlotte aux États-Unis, Magog au Canada, Pune en Inde et Shanghai en Chine.

Organisé en cinq entités principales, DOTI couvre l'ensemble des besoins IT du Groupe. L'entité *Infrastructure* (INFRA) gère les centres de données, les clouds, les réseaux et les opérations globales pour garantir la fiabilité et la sécurité des environnements IT. La *Data & Artificial Intelligence* (AI) conçoit des solutions analytiques et d'IA destinées à optimiser la R&D, la production et la chaîne logistique de Michelin. L'*Enterprise Architecture* (EA) définit les référentiels et l'urbanisation du système d'information, en s'appuyant notamment sur la méthodologie Forrester pour aligner ses roadmaps et ses KPI [6]. Enfin, l'entité *Digital Experience* (DXD) conçoit les interfaces, portails et plateformes collaboratives destinés à améliorer l'expérience utilisateur et soutenir la stratégie digitale du Groupe.

Au-delà de ces entités, DOTI porte de nombreux programmes transverses, cybersécurité, mobilité, DevOps, CI/CD et promeut la culture du *software craftsmanship*. DOTI participe à des conférences telles que Devoxx ou Volcamp pour échanger avec la communauté et recruter les meilleurs talents.

2.4 Le programme OneSystem IT Platforms

Le programme *OneSystem IT Platforms* du département DOTI, chargé de concevoir, déployer et maintenir les plateformes applicatives et les services numériques destinés aux différentes directions métiers du Groupe Michelin.

Depuis 2018, face à la complexité croissante et au vieillissement de son système d'information, Michelin a initié le programme *Evergreen IT Platforms* afin de mettre en place une modernisation continue et durable des fondations technologiques de son SI [7]. Ce programme repose sur le constat qu'un système en moyenne vieux de 12 ans et comptant près de 2 000 applications accumulées constitue un frein majeur à la réactivité et à la sécurité du Groupe [7].

Au cœur de ce programme, quinze *IT Platforms* sont pilotées par des équipes long-life structurées selon une approche *Agile@Scale*, c'est-à-dire l'application des principes agiles à l'échelle de l'organisation pour orchestrer efficacement les travaux entre plusieurs équipes [8]. Ces équipes sont responsables du cycle complet *plan-build-run* de capacités technologiques (bases de données, middlewares d'intégration, infrastructure-as-code, conteneurs, monitoring) et opèrent selon un modèle *product-centric* qui fait de chaque plateforme un véritable produit interne à maintenir et à faire évoluer [7].

Chaque plateforme délivre un *Minimum Viable Product* (MVP) bimensuel, garantissant un rafraîchissement régulier des technologies sans interruption majeure de service et une coordination via un *Program Increment Planning*, événement systématique de synchronisation des feuilles de route sur deux mois. La consolidation des Backlogs individuels dans un Backlog unifié, piloté par un Chief Product Owner, permet de prioriser les investissements technologiques au regard de leur valeur métier et des dépendances systémiques, conformément aux meilleures pratiques du Scaled Agile Framework (SAFe).

Une *IT Platform* est définie comme "une fondation de self-service APIs, outils, services, savoir-faire et support organisée en tant que produit interne", fournissant aux équipes de développement les briques nécessaires pour accélérer la livraison de fonctionnalités tout en réduisant les besoins de coordination et de montée en compétences sur des sujets complexes. Cette définition, inspirée des travaux de ThoughtWorks, souligne l'importance de traiter la plateforme comme un produit à part entière, avec un Product Owner dédié et des cérémonies (sprint planning, démos, rétrospectives) identiques à celles des squads métiers.

La mise en place de ces équipes a permis de casser les silos traditionnels (infrastructure, applications, méthodes & qualité) et d'appliquer un "inverse Conway maneuver" en alignant l'architecture organisationnelle sur les produits à délivrer plutôt que sur les fonctions techniques. Parmi les bénéfices constatés figurent la réduction significative du lead time pour mettre à jour les clusters Kubernetes (mise à jour automatisée quatre fois par an sans downtime), l'amélioration de la fiabilité des services via une automatisation accrue du déploiement, et une meilleure visibilité des priorités grâce à la tenue régulière de démos et de reporting consolidé [7].

2.5 L'équipe Hybrid Integration Platform (HIP)

L'équipe *Hybrid Integration Platform* (HIP) constitue le socle middleware de l'intégration au sein du programme OneSystem IT Platforms, offrant une couche d'abstraction pour orchestrer, transformer et superviser les échanges de données entre applications on-premise et services cloud. En centralisant les fonctionnalités de messaging, orchestration, ETL et API management, HIP permet à Michelin de casser les silos applicatifs, d'accélérer la mise en œuvre de nouveaux services et d'assurer la cohérence des processus métiers à grande échelle.

Le volet *Messaging & Mass Processing* est porté par l'équipe MessMass, qui industrialise et maintient des middlewares tels qu'IBM MQ, Kafka, CFT, Filehub et Datapower pour garantir une communication fiable et performante entre systèmes hétérogènes. Cette squad long-life assure le support via ServiceNow, la gestion des incidents, la mise à jour continue des versions et l'optimisation des performances, réduisant le temps de résolution et améliorant la continuité de service.

Pour l'*Orchestration de processus*, l'équipe HIP s'appuie sur des bus d'intégration et des orchestrateurs low-code qui enchaînent appels API et transformations de messages selon des scénarios métiers, garantissant résilience, traçabilité et auditabilité des flux. La qualité et la transformation des données sont gérées par des plateformes ETL comme DataStage et Boomi, permettant des mappings complexes, des validations et des règles de transformation afin d'assurer l'intégrité des données échangées.

Le pilier *API Management* complète l'équipe HIP en sécurisant, versionnant et monitorant les interfaces exposées aux équipes internes et aux partenaires externes. Les API sont publiées sur un portail collaboratif, avec des mécanismes de contrôle d'accès, de quotas et de reporting intégrés, favorisant la découverte et la réutilisation des services.

Cette organisation modulaire est déclinée en trois squads spécialisées, Fusion, Atom et Integration Mastering, chacune responsable du cycle complet plan-build-run de ses produits middleware selon les principes de l'Agile@Scale. Ce modèle de squads long-life permet de regrouper au sein de chaque équipe les compétences fonctionnelles, techniques et DevOps nécessaires et d'automatiser les pipelines CI/CD pour accélérer les déploiements et réduire les risques.

L'implémentation de l'équipe HIP chez Michelin a permis de réduire de 30 % le délai moyen de déploiement de nouveaux middlewares, d'améliorer la fiabilité des échanges via des tests automatisés et des rollbacks rapides, ainsi que d'enrichir la supervision centralisée grâce à l'intégration native avec la plateforme observabilité (Grafana, Splunk, OpenTelemetry).

2.6 Mon positionnement dans cette structure

Mon stage s'est déroulé au sein de l'équipe *MessMass*, du 1er avril 2025 au 29 août 2025. En tant que stagiaire développeur transverse, je n'étais rattaché à aucune squad middleware spécifique, ce qui m'a permis d'intervenir sur l'ensemble des briques de MessMass (messagerie, orchestration et ETL).

J'étais placé sous la responsabilité fonctionnelle de mon tuteur de stage, Yann Segaud, Product Owner MessMass et sous l'autorité hiérarchique de mon manager, Christophe Emotte, Manager HIP. Avec Yann, nous avions des points réguliers pour suivre l'avancement du projet et ajuster les priorités.

Durant ces cinq mois, j'ai travaillé en étroite collaboration avec chacune des trois squads du HIP: la Fusion Squad (IBM MQ, CFT, Filehub), l'Atom Squad (Kafka, DataStage, Boomi) et l'Integration Mastering Squad (M2I, E2E, WBI). Par ailleurs, j'ai collaboré avec Simon Dallet, alternant au sein de l'équipe API Platform responsable de la partie API de la status page, pour définir ensemble sa structure et ses normes.

Cette position transverse m'a permis de développer des compétences à l'intersection du middleware, de l'API management et de la supervision, tout en adoptant les bonnes pratiques de l'Agile@Scale, où la réussite dépend autant de la qualité technique que de la coordination entre les acteurs.

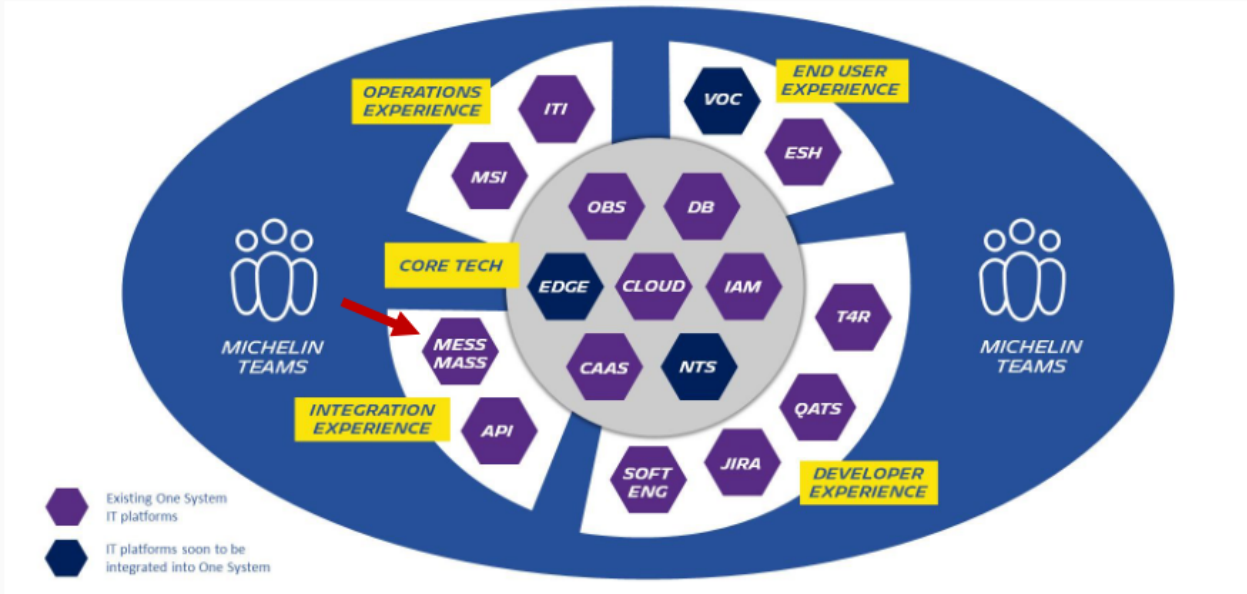

Figure 1: OneSystem IT Platforms

3 Objectifs du stage

L'objectif principal de mon stage consistait à concevoir, développer et déployer une status page globale destinée aux utilisateurs des quinze middlewares gérés par l'équipe MessMass au sein du groupe Michelin. Cette interface, destinée à environ 600 utilisateurs internes répartis entre les équipes techniques et les métiers, devait offrir une visibilité en temps réel sur l'état des services, ainsi qu'un diagnostic rapide des incidents.

Cette réalisation représentait une opportunité significative d'améliorer la visibilité et l'efficacité opérationnelle des équipes métier, tout en répondant à un besoin croissant de surveillance temps réel et de fiabilité des échanges. Dans ce contexte, ma mission comprenait plusieurs volets majeurs détaillés ci-dessous.

3.1 Migration vers Grafana : Contexte et enjeux

Michelin avait précédemment investi dans la plateforme Splunk pour gérer l'observabilité des systèmes et applications critiques. Toutefois, le coût élevé des licences et des infrastructures nécessaires a conduit à une stratégie de migration vers Grafana, une solution open-source reconnue pour sa flexibilité, sa simplicité d'utilisation et ses coûts maîtrisés [9]. Bien que l'objectif soit de réduire l'utilisation de Splunk au profit de Grafana, il n'est pas prévu de le retirer complètement, certaines fonctionnalités spécifiques restant pertinentes pour des cas d'usage avancés.

Ce changement de paradigme représentait non seulement une transition technologique mais aussi culturelle. Il était essentiel que la nouvelle status page développée soit facile d'accès, intuitive et adaptée à des utilisateurs possédant ou non des compétences techniques avancées. Par conséquent, l'objectif était de créer une interface claire, rapidement compréhensible et accessible publiquement, même pour les utilisateurs ne disposant pas directement d'un compte Grafana.

3.2 Recueil des besoins et spécifications techniques

La première étape de mon travail consistait à identifier précisément les indicateurs critiques nécessaires à l'évaluation de l'état des middlewares. Pour ce faire, j'ai organisé et mené plusieurs réunions et entretiens avec les différentes squads (Fusion, Atom, Integration Mastering) responsables des middlewares concernés. Ces échanges m'ont permis de comprendre en profondeur leurs attentes, leurs pratiques actuelles de monitoring et les données spécifiques devant être affichées dans la status page.

Parmi les indicateurs clés identifiés figuraient notamment le taux de disponibilité (uptime), la latence des requêtes et des transferts, le nombre et l'état des files d'attente, ainsi que le statut général des services (*OK, KO, Dégradé*). Ces spécifications initiales ont constitué le socle sur lequel s'est construit le projet et ont permis d'établir une cohérence entre les attentes des utilisateurs et les possibilités techniques offertes par Grafana.

3.3 Apprentissage et développement des compétences sur Grafana

L'un des défis majeurs de ce stage était la prise en main rapide de Grafana, un outil nouveau pour moi. Afin d'être opérationnel le plus vite possible, j'ai suivi plusieurs formations internes et externes sur Grafana, complétées par des exercices pratiques réalisés sur des environnements de test.

Cet apprentissage m'a permis de maîtriser des aspects techniques essentiels tels que la création et la gestion de dashboards Grafana, l'intégration avec Prometheus pour la collecte de métriques en temps réel, la configuration de panels spécifiques (*Bar Gauge, Stat, Heatmap*) pour une visualisation efficace des données, ainsi que le paramétrage d'alertes automatisées basées sur les seuils critiques définis avec les experts middleware. Cette montée en compétence rapide fut cruciale pour la réussite du projet, car elle m'a permis de proposer rapidement un prototype et de l'adapter progressivement en fonction des retours utilisateurs et des contraintes techniques.

3.4 Prototypage et collaboration avec l'équipe API

Un autre élément clé de ma mission consistait à collaborer étroitement avec Simon Dallet, alternant au sein de l'équipe API Platform. Ensemble, nous avons ébauché un prototype initial de la status page globale, en définissant notamment des normes communes telles que les conventions de nommage, les codes couleur pour le statut des middlewares (vert, jaune, rouge) et le niveau de détail nécessaire à chaque type d'utilisateur (expert technique versus utilisateur métier).

Cette collaboration transversale a permis d'assurer une cohérence entre la partie middleware (gérée par moi-même) et la partie API (prise en charge par Simon), garantissant ainsi une expérience utilisateur fluide et homogène.

3.5 Accès public et facilité d'utilisation

Enfin, un des objectifs était de rendre la status page accessible publiquement, même aux utilisateurs ne disposant pas de compte Grafana. Pour ce faire, j'ai travaillé à la mise en place d'une interface simplifiée, exposée publiquement via une URL sécurisée et adaptée à une consultation rapide et efficace.

3.6 Synthèse du diagramme de Gantt prévisionnel

Le diagramme de Gantt prévisionnel présenté, en Figure 2, offre une vue détaillée et structurée des différentes étapes planifiées durant mon stage. Il se divise en plusieurs phases clés, à commencer par la phase initiale d'arrivée et d'intégration à l'équipe MessMass, incluant la prise en main des outils et la familiarisation avec les processus internes. Cette première étape couvre également la compréhension globale du sujet et l'identification précise des tâches à réaliser.

La seconde phase, essentielle à la bonne réalisation du projet, concerne la formation technique approfondie. Elle comprend plusieurs sessions ciblées sur Grafana et l'administration système, afin de garantir une autonomie suffisante pour mener à bien les objectifs fixés.

La troisième phase correspond à la collecte des spécifications techniques auprès des experts middleware. Elle se décline en plusieurs sous-tâches dédiées à chaque middleware, assurant une méthodologie rigoureuse et exhaustive dans le recueil des besoins et des métriques à afficher.

La quatrième phase concerne la création de la status page finale. Elle inclut des travaux de planification et de normalisation, le développement du prototype initial et la recherche d'une méthode d'affichage publique optimale.

Enfin, la dernière phase du projet englobe la validation finale par les membres de l'équipe MessMass, la rédaction de la documentation complète du projet, ainsi que la rédaction et la soumission du rapport de stage final. Le diagramme de Gantt permet ainsi de suivre rigoureusement l'avancement et de gérer efficacement les ressources et les échéances tout au long du projet.

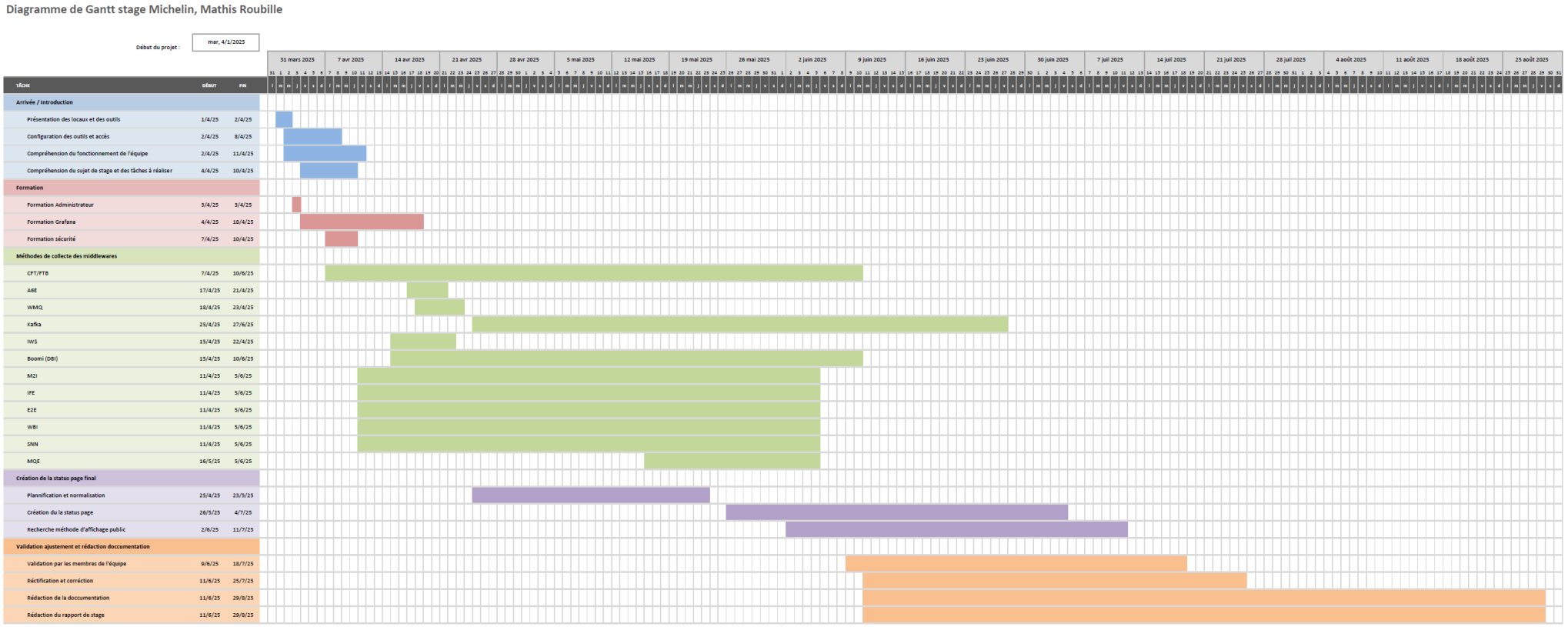

Figure 2: Diagramme de Gantt prévisionnel du stage

4 Analyse du besoin

Dès le début du stage et tout au long de son déroulement, une étape fondamentale a été la compréhension approfondie et évolutive du besoin à satisfaire. Ce besoin ne se limitait pas à une simple demande technique, mais reposait sur plusieurs enjeux distincts et complémentaires qu'il a fallu identifier, analyser et intégrer dans la conception de la status page.

4.1 Un besoin de visibilité et de réactivité opérationnelle

L'enjeu principal qui a motivé le développement de la status page Grafana était d'offrir aux utilisateurs finaux une capacité immédiate de diagnostic et de compréhension des problèmes rencontrés au quotidien. Avant cette initiative, les utilisateurs, confrontés à un problème technique, avaient des difficultés à déterminer rapidement si le problème venait de leur propre infrastructure, de leur application métier, ou d'un middleware spécifique.

Ce manque de visibilité immédiate engendrait des délais conséquents dans le processus d'identification et de résolution des incidents. L'objectif clair du projet était donc de permettre aux utilisateurs d'obtenir instantanément un statut précis et fiable sur la disponibilité et la performance des différents middlewares qu'ils utilisent quotidiennement. Ce besoin était crucial pour améliorer non seulement la réactivité opérationnelle des équipes métiers, mais également pour réduire les temps de réponse et le volume de sollicitations inutiles vers les équipes techniques, qui étaient souvent surchargées par des demandes redondantes ou mal ciblées.

4.2 Interface utilisateur : simplicité et lisibilité

Un des aspects majeurs du besoin, identifié très tôt au cours du stage grâce aux échanges avec les différents acteurs impliqués (experts middleware, utilisateurs métier, responsables opérationnels), concernait l'ergonomie et la simplicité d'utilisation de la future status page.

Les utilisateurs cibles de la status page étant diversifiés, allant des experts techniques aux utilisateurs métiers, il était impératif de concevoir une interface simple, claire et immédiatement compréhensible par tous. Ainsi, il fallait éviter toute surcharge d'informations inutiles ou trop techniques, tout en offrant suffisamment de détails pertinents pour permettre un diagnostic rapide.

4.3 Service Level Indicators (SLI) et discussions autour des Service Level Objectives (SLO)

La mise en place de cette status page a également servi de catalyseur pour introduire et structurer des discussions importantes autour de la définition des Service Level Indicators (SLI) et des Service Level Objectives (SLO) au sein du groupe HIP. Avant le projet, ces concepts étaient rarement formalisés ou partagés clairement entre les équipes techniques et les utilisateurs finaux.

Les échanges avec les différentes équipes (Fusion, Atom, Integration Mastering) ont permis de définir et d'afficher clairement les SLIs des services surveillés. Ces indicateurs permettent désormais une meilleure compréhension de la performance réelle des services, avec une granularité précise (deux chiffres après la virgule). L'ajout de cette colonne a non seulement amélioré la transparence sur la disponibilité réelle des services mais également initié une réflexion plus stratégique sur l'éventuelle mise en place future de SLOs.

En effet, bien que les SLOs ne soient pas encore formellement définis ou affichés à ce stade du projet, les données recueillies via les SLIs sur une durée prolongée offrent une base solide pour de futures analyses et décisions stratégiques. L'affichage des SLIs représente ainsi une première étape vers une gestion plus proactive et quantitative de la qualité de service.

4.4 Méthodologie et interactions avec les équipes middleware

Le recueil des besoins a nécessité une méthodologie structurée et des interactions régulières avec les membres des différentes équipes middleware (Fusion, Atom, Integration Mastering). Plusieurs réunions et échanges ont été menés tout au long du projet, permettant ainsi de collecter des retours réguliers et d'ajuster continuellement le développement de la status page selon les spécifications détaillées des équipes.

Ces interactions ont été cruciales pour garantir l'adéquation du produit final avec les attentes réelles des utilisateurs finaux et des experts techniques. Elles ont aussi permis d'identifier rapidement les indicateurs essentiels à remonter dans Grafana, tout en assurant une cohérence globale et une uniformisation de l'interface utilisateur.

En résumé, l'analyse du besoin a été une activité centrale, constante et évolutive tout au long du stage, intégrant plusieurs dimensions techniques, ergonomiques et stratégiques pour répondre au mieux aux enjeux opérationnels du groupe Michelin. Ce travail préalable, fondé sur une compréhension fine et continue du contexte métier et technique, a été déterminant pour la réussite du projet et pour la satisfaction des utilisateurs finaux.

5 Panorama des middlewares gérés par MessMass

La direction HIP et plus particulièrement l'équipe MessMass, est en charge de la supervision, de la configuration et de l'intégration d'un ensemble de middlewares critiques au sein du système d'information de Michelin. Ces composants assurent la fiabilité des transferts de données, l'intégration entre applications, la surveillance des flux et la gestion des messages.

- **CFT (Axway Transfer CFT)** : Middleware de transfert de fichiers point-à-point hautement sécurisé, CFT permet d'assurer des transferts fiables entre applications internes et partenaires externes. Il est supporté sur différentes plateformes et gère la traçabilité ainsi que la reprise sur incident.
- **FTB (Filehub)** : Solution SaaS utilisée en mode SFTP, FTB est conçue pour les transferts de fichiers avec les partenaires externes. Elle est opérée par Files.com et garantit une haute disponibilité des transferts ainsi qu'une protection des données pendant le transit.
- **IIB** / **A6E (IBM App Connect Enterprise)** : Plateforme d'intégration orientée flux et API, permettant de connecter différentes applications de manière flexible. Elle repose sur une approche de configuration sans codage et gère la transformation et le routage de messages complexes.
- **WMQ (IBM MQ)** : Middleware de messagerie d'entreprise garantissant la livraison fiable des messages via un modèle asynchrone. Il prend en charge les communications en point-à-point ou en publish-subscribe.
- **Kafka (Apache** / **Confluent)** : Plateforme de streaming événementielle utilisée pour la transmission de grands volumes de données en temps réel. Kafka est central pour les pipelines de données et les cas d'usage temps réel.
- **DataStage (IBM InfoSphere DataStage)** : Outil ETL de traitement de données, employé pour la transformation et l'intégration entre systèmes hétérogènes. Il fonctionne via des jobs planifiés et s'insère dans les flux de traitement par lot.
- **Boomi** : Plateforme d'intégration cloud-native (iPaaS) permettant le développement rapide de workflows, d'API et d'intégrations B2B en low-code.
- **Attunity (Qlik Replicate)** : Solution de Change Data Capture (CDC) pour la réplication de données quasi temps réel entre systèmes sources et entrepôts de données.
- **DataPower (IBM DataPower Gateway)** : Appliance de sécurisation et de médiation d'API et de messages, jouant un rôle de passerelle de protection pour les systèmes exposés sur Internet.

5.1 Extensions internes et composants de pilotage

- **M2I** : Application interne historique pour l'inventaire des flux en production.
- **IFE (Integration Flows Extension)** : Remplaçant moderne de M2I, cette application permet de gérer l'inventaire et la traçabilité des flux sur une architecture améliorée.
- **E2E (End to End monitoring)** : Supervise les flux transitant par ACE, en les reconstituant de manière logique, pour un suivi complet des parcours de données.
- **WBI (Web-Based Interface)** : Interface frontale pour la supervision et l'administration des flux, avec base de données de référence.
- **SNN** : Application Web interne permettant le déploiement et la configuration des flux MFT.
- **MQE** : Extension interne développée pour l'inventaire, la cartographie et la surveillance des instances IBM MQ.
- **CGV (CFT Governance)** : Supervise les transferts CFT, en assurant un suivi unifié et la mise à jour centralisée des configurations.
- **HTB (HIP Toolbox)** : Ensemble d'outils internes de support et de diagnostic pour les opérations MessMass, incluant des scripts, planificateurs et générateurs de logs.
- **KFE** : Extensions pour Kafka, facilitant l'accès aux logs, la gestion des topics et l'automatisation des opérations.

L'ensemble de ces composants est connecté à la stack Grafana OSS, qui centralise les métriques, logs et alertes dans des dashboards construits pour répondre aux besoins de supervision et de diagnostic rapide. Ce paysage middleware est le socle sur lequel s'est construit le projet de status page.

6 Formation et montée en compétences techniques

6.1 Formations générales internes

Durant les premières semaines, j'ai suivi plusieurs formations internes obligatoires sur la plateforme *InTouch*, notamment sur la cybersécurité, les protocoles et standards internes à Michelin, ainsi que sur des sujets liés à la culture d'entreprise et à l'éthique professionnelle. Parmi celles-ci, la formation sur la cybersécurité était particulièrement importante car elle abordait des pratiques essentielles pour garantir la sécurité des données et des systèmes, comme la gestion sécurisée des accès, la prévention contre les cyberattaques et les procédures à suivre en cas d'incidents de sécurité. Cette sensibilisation a permis d'intégrer rapidement les bonnes pratiques de sécurité informatique nécessaires dans mon travail quotidien.

J'ai également suivi des modules de formation concernant les valeurs fondamentales de Michelin, incluant la prévention et la gestion du harcèlement au travail. Ces formations ont renforcé ma compréhension des attentes comportementales et professionnelles au sein du groupe, m'aidant à m'intégrer efficacement dans l'équipe tout en adoptant une attitude respectueuse et constructive.

6.2 Formation approfondie sur Grafana

Le cœur technique de mon projet étant la création d'une status page avec Grafana, une attention particulière a été consacrée à l'apprentissage approfondi de cet outil. Grafana est une plateforme open-source d'observabilité et de visualisation de données, largement utilisée pour le monitoring et l'analyse des systèmes et applications en temps réel. Cet outil permet la création de tableaux de bord interactifs en agrégeant des métriques provenant de sources variées comme Prometheus, InfluxDB, Elasticsearch, ou encore OpenTelemetry [10]. Des études récentes confirment que la combinaison Grafana-Prometheus s'impose comme une solution de référence pour le monitoring distribué et scalable dans les environnements microservices [11].

La formation dédiée à Grafana disponible sur *InTouch* comprenait plusieurs modules distincts, permettant une progression claire et structurée. J'ai débuté par une introduction générale, couvrant les concepts clés tels que les panels, les dashboards et les sources de données. Les panels Grafana sont des éléments de visualisation comme les graphiques, tableaux, ou jauges, chacun adapté à la présentation d'un type de données spécifique. J'ai exploré particulièrement les panels *Stat*, *Gauge* et *Time Series*, particulièrement adaptés pour représenter efficacement les métriques d'état des middlewares [12].

Par la suite, j'ai approfondi les systèmes de variables et d'alertes proposés par Grafana. Le système de variables est essentiel pour créer des dashboards interactifs et dynamiques, permettant aux utilisateurs de filtrer ou modifier l'affichage des données à la volée selon leurs besoins spécifiques [13]. L'apprentissage des alertes était tout aussi crucial, car Grafana propose une fonctionnalité puissante pour définir des seuils critiques et envoyer automatiquement des notifications aux équipes opérationnelles en cas d'anomalie ou d'indisponibilité des services surveillés [14].

En parallèle de ces formations théoriques, j'ai mis en pratique mes nouvelles connaissances en développant progressivement des dashboards de test. Cette approche pratique et expérimentale m'a permis de valider concrètement les acquis théoriques tout en gagnant rapidement en autonomie et en maîtrise des fonctionnalités avancées de Grafana.

6.3 Exploration du code source de Grafana

Animé par une curiosité personnelle et afin d'approfondir ma compréhension technique, j'ai consacré du temps à l'exploration du code source de Grafana, disponible en libre accès puisqu'il s'agit d'une solution open-source. Grafana est principalement développé en Golang (Go), un langage conçu par Google pour la performance, la simplicité et la scalabilité, particulièrement adapté aux applications système et réseau [15].

Bien que le Golang ne fasse pas partie de mes langages de programmation habituels, cette exploration m'a permis d'apprécier la qualité du design logiciel de Grafana, notamment son architecture modulaire et extensible. Le code source, structuré en plusieurs modules distincts et bien documentés, facilite grandement l'intégration de nouvelles fonctionnalités ou l'ajout de nouvelles sources de données. J'ai particulièrement étudié les modules liés au traitement des requêtes des utilisateurs, à la gestion des alertes et à l'intégration des sources de données telles que Prometheus ou OpenTelemetry [16].

Cette démarche, bien que complexe en raison de ma faible familiarité avec le Golang, a enrichi ma vision technique du projet et m'a permis de mieux comprendre les possibilités et limites intrinsèques de Grafana. Elle a également renforcé mes compétences en analyse et en compréhension de systèmes complexes, compétences précieuses dans mon parcours d'ingénieur informatique.

6.4 Documentation interne et support technique

Au-delà des formations officielles et de l'exploration autonome du code source, j'ai également bénéficié de l'accès à une documentation interne complète, spécifiquement élaborée par les équipes Observabilité de Michelin. Cette documentation comprenait des guides techniques détaillés, des procédures opérationnelles standardisées, ainsi que des exemples concrets de dashboards utilisés en production.

Par ailleurs, tout au long de cette phase d'apprentissage, j'ai pu compter sur le soutien continu et réactif de mes collègues de l'équipe Observabilité et MessMass. Leurs conseils et retours d'expérience m'ont permis d'avancer efficacement tout en bénéficiant de leur expertise approfondie sur Grafana et les pratiques internes de Michelin.

7 Performance tests et observabilité

En complément des formations initiales, j'ai approfondi mes compétences techniques sur les outils spécifiques utilisés pour l'observabilité à Michelin, en particulier K6 pour les tests de performances et OpenTelemetry pour la collecte des données.

7.1 Tests de performance et vérifications d'URL avec K6

K6 est un outil open-source de test de charge et de performance qui permet de vérifier des URL nécessitant une authentification complexe, contrairement à des solutions plus simples telles que le Blackbox Exporter. Pour intégrer les tests K6 à l'environnement Michelin, une solution complète a été mise en place à travers un dépôt GitLab dédié.

En suivant la documentation fournie, j'ai pu configurer rapidement mes tests K6 en clonant le dépôt *k6-common* et en éditant trois fichiers principaux :

- **Common.yml :** pour définir les variables globales nécessaires à l'exécution des scripts, incluant les endpoints vers Grafana et les identifiants pour HashiCorp Vault.
- **Exec.yml :** ce fichier définit l'environnement d'exécution des scripts K6, notamment les connexions vers le système de télémétrie Grafana via OpenTelemetry.
- **Runner.js :** contenant les scripts JavaScript exécutés par K6 pour réaliser les vérifications d'URL authentifiées.

Après la mise en place initiale, j'ai pu planifier ces tests pour une exécution récurrente via GitLab CI/CD, permettant ainsi une surveillance proactive des services critiques avec des métriques précises remontées directement vers Grafana Mimir.

7.1.1 Problématiques rencontrées avec K6

Malgré une configuration réussie des scripts K6 et plusieurs preuves de concept (PoC) réalisées, une difficulté majeure subsiste concernant les runners GitLab fournis par l'équipe Observabilité. En effet, ces runners ne permettent actuellement pas d'exécuter des tests sur des liens externes, ce qui constitue un blocage critique pour certains cas d'utilisation spécifiques. Ainsi, bien que les scripts soient prêts à être déployés, leur mise en production est actuellement suspendue jusqu'à ce que l'équipe Observabilité apporte une solution permettant de surveiller efficacement les liens externes. Cette situation a entraîné une réflexion continue et une communication régulière avec l'équipe Observabilité afin de lever cette contrainte technique dès que possible.

7.2 Observabilité avec OpenTelemetry

Selon un rapport publié par Grafana Labs en juin 2025, plus de 71% des entreprises utilisent aujourd'hui Prometheus ou OpenTelemetry en production et 74% placent la maîtrise des coûts comme critère principal dans le choix de leurs outils d'observabilité [17].

OpenTelemetry est au cœur de la stratégie d'observabilité de Michelin. Cet ensemble d'outils permet la collecte standardisée de données techniques : métriques, logs et traces applicatives. Ces données sont ensuite transmises vers la stack Grafana, composée notamment de Mimir pour les métriques, Loki pour les logs et Tempo pour les traces.

L'architecture OpenTelemetry repose sur deux niveaux principaux :

- **Collector Agent** : Installé sur les serveurs ou embarqué dans les applications, il collecte les données à la source.
- **Collector Gateway** : Point centralisé qui reçoit les données collectées, les transforme si nécessaire, puis les transmet aux outils de visualisation comme Grafana.

L'identification des données passe par des labels, notamment *service.name*, indispensable pour organiser les flux dans les différents tableaux de bord. Durant le stage, j'ai suivi les conventions de nommage définies par l'équipe Observabilité pour assurer l'interopérabilité entre les composants.

Au départ, des scripts Python avaient été envisagés pour envoyer manuellement des métriques vers Prometheus. Ces scripts fonctionnaient de façon satisfaisante dans un environnement de test, mais ont été abandonnés pour la solution finale. En effet, leur déploiement à grande échelle posait plusieurs problèmes : absence d'infrastructure claire pour leur hébergement, difficultés de maintenance et manque d'alignement avec les standards retenus par l'équipe Observabilité. Ces dernières ont donc préféré intégrer des collecteurs OpenTelemetry officiels, plus robustes et adaptés à l'architecture cible.

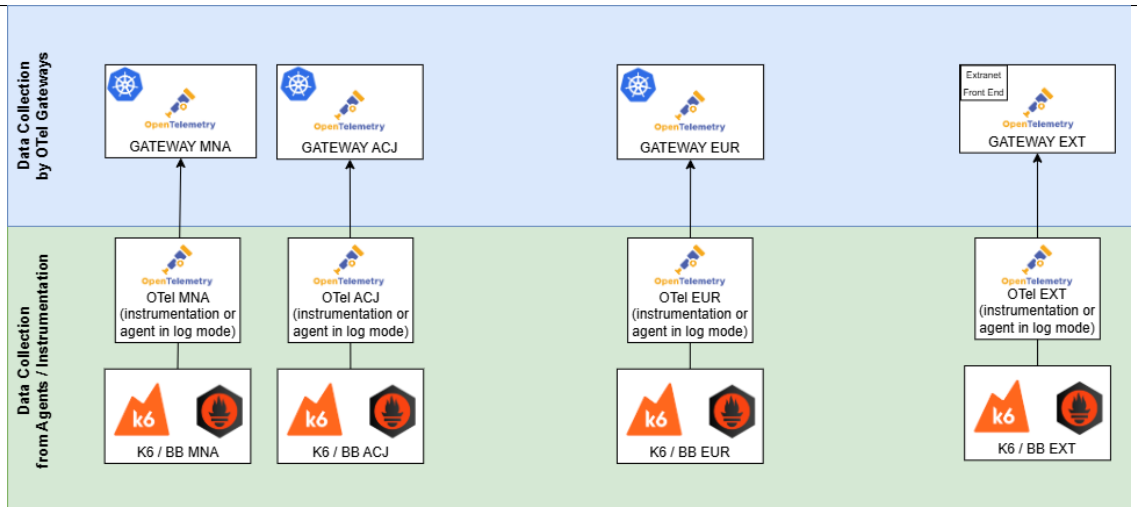

Figure 3: Architecture de collecte des données OpenTelemetry

7.3 Intégration pratique des outils d'observabilité

J'ai participé à la configuration et aux tests de ces agents OpenTelemetry sur des environnements de développement, en les adaptant progressivement aux contraintes de production. Cela a inclus le respect des paramètres de collecte propres à chaque région (Europe, Amérique, Asie), ainsi que la gestion des labels pour permettre une agrégation efficace des métriques.

Cette phase de mise en pratique m'a permis de comprendre l'importance de la standardisation dans la collecte de données d'observabilité, en facilitant la création de dashboards unifiés et réutilisables. J'ai également collaboré avec les membres de l'équipe Observabilité pour vérifier la compatibilité des tests K6 avec les agents de collecte OpenTelemetry, en vue d'un suivi plus complet de la disponibilité des services.

8 Exploration et Analyse des Dashboards Existants

Afin de réaliser efficacement la status page sur Grafana, j'ai débuté par une phase essentielle d'exploration approfondie des dashboards existants au sein de Michelin. Cette étape visait à identifier les bonnes pratiques déjà mises en œuvre, les limites potentielles des solutions en place et les éléments que je pouvais réutiliser directement ou adapter pour mon projet.

8.1 Méthode de l'analyse

Ma méthodologie d'analyse a été structurée autour de plusieurs critères précis :

- **Lisibilité et simplicité** : La capacité des dashboards existants à transmettre clairement les informations essentielles, même à un public non technique.
- **Pertinence des indicateurs** : Identifier les indicateurs déjà suivis et juger de leur pertinence par rapport aux besoins spécifiques de la status page.
- **Flexibilité et dynamique** : Évaluer la facilité de personnalisation et de mise à jour des dashboards existants, notamment via les systèmes de variables et d'alerting.
- **Adaptabilité** : Analyser la compatibilité et le comportement responsive des dashboards sur différents supports (écrans, résolutions).

8.2 Analyse des exemples existants

Parmi les dashboards existants, j'ai particulièrement porté attention à ceux proposés par l'équipe Observabilité ainsi qu'à ceux créés par Simon Dallet. Cependant, étant donné que Grafana est encore relativement récent au sein de Michelin, le nombre d'exemples disponibles était limité.

Observabilité offrait plusieurs templates utiles, notamment pour les dashboards utilisant l'outil *Blackbox Exporter*, qui permet de vérifier simplement la disponibilité d'URLs sans authentification complexe. Ces templates m'ont aidé à rapidement identifier une approche efficace pour afficher des états simples tels que OK ou KO, ainsi que pour suivre l'évolution de ces états dans le temps.

Toutefois, d'autres dashboards existants basés sur le panel Canvas (voir Figure 4) présentaient une complexité élevée et affichaient trop d'informations techniques, peu pertinentes pour des utilisateurs finaux. De plus, ces formats présentaient un problème majeur de non-responsivité : les utilisateurs ayant des tailles d'écran différentes pouvaient ne pas visualiser correctement certaines parties essentielles de la page. Cette contrainte de design nous a poussé à écarter le format Canvas, malgré ses qualités visuelles intéressantes.

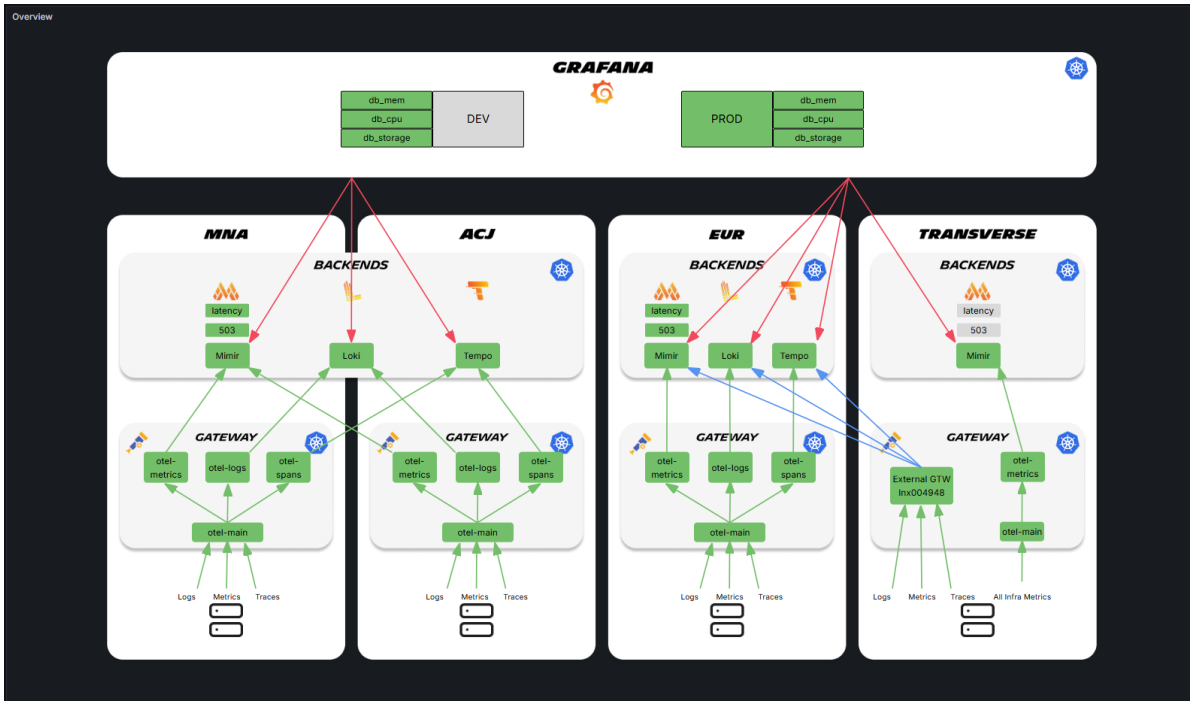

Figure 4: Exemple de Dashboard utilisant le panel Canvas

8.3 Résultats et éléments réutilisables

Malgré ces limitations, certains dashboards analysés m'ont permis d'identifier plusieurs bonnes pratiques :

- L'utilisation efficace du système de variables pour créer des dashboards dynamiques adaptables à différents contextes.
- La gestion avancée des alertes pour notifier rapidement les équipes en cas d'incidents majeurs.
- Une représentation claire et intuitive des états de services à l'aide de couleurs et d'indicateurs facilement compréhensibles.

Ces éléments réutilisables ont constitué un socle sur lequel j'ai construit les premières versions de la status page. Néanmoins, la spécificité de notre projet, visant une simplicité maximale tout en fournissant suffisamment de détails pour un diagnostic rapide, nous a amenés à créer notre propre standard de design et d'interaction.

8.4 Création d'un nouveau standard

Étant les premiers au sein de Michelin à développer une status page de ce type, Simon Dallet et moi-même avons dû établir des standards de référence totalement nouveaux. Après plusieurs itérations et discussions approfondies avec les équipes concernées, nous avons adopté une approche simplifiée avec une disposition en deux colonnes distinctes (MessMass et API) comme illustré sur la Figure 20.

Ce standard repose sur trois indicateurs clés pour chaque middleware :

- Un indicateur de statut en temps réel : il peut être de type binaire (OK/KO) ou indiquer un nombre de composants affectés (ex : 2/8 engines down).
- Un SLI (Service Level Indicator) calculé sur les 28 derniers jours[1], permettant d'évaluer la disponibilité moyenne du service.
- Un historique visuel affichant la disponibilité sur les dernières 72 heures, découpé en tranches de 6 heures et coloré selon la disponibilité mesurée :
  - Vert foncé : disponibilité $\geq 99.5\%$
  - Vert clair : disponibilité $99\% \leq x < 99.5\%$

[1] Une période de 28 jours correspond à quatre semaines complètes, ce qui permet de lisser les variations quotidiennes ou hebdomadaires tout en restant représentatif de l'état actuel du service. Elle est assez longue pour éviter qu'un incident ponctuel ne fausse les données, mais assez courte pour que les problèmes récents ne soient pas dilués dans des moyennes trop anciennes. De plus, cette durée inclut toujours exactement 4 week-ends et le même nombre de chaque jour de la semaine, ce qui n'est pas garanti avec 30 ou 31 jours.

- Jaune : disponibilité $90\% \leq x < 99\%$
- Rouge : disponibilité $< 90\%$

Ces trois indicateurs forment le socle du standard retenu. Ils sont généralisés à l'ensemble des middlewares suivis, renforçant ainsi la cohérence et la lisibilité de la page.

Chaque cellule est enrichie de boutons de *drill-down* personnalisés, permettant d'accéder rapidement à des dashboards plus détaillés dans Grafana. Cette personnalisation pallie la faible visibilité des boutons natifs Grafana.

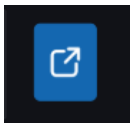

Figure 5: Bouton de drill-down

Pour les services ne pouvant pas être représentés par un simple statut binaire, comme MQ ou A6E, un indicateur numérique est directement affiché (ex : 6/67 MQ down).

Lien avec ServiceNow

Des liens vers les incidents et changements documentés dans ServiceNow sont également accessibles, à condition que l'utilisateur possède les droits adéquats. Sinon, les notifications critiques sont relayées via le canal Teams dédié à l'intégration.

Expérimentation SLO et SLI avancés

Certains dashboards intègrent une expérimentation de SLO simulés avec des jauges et courbes de burn rate, basés sur la méthodologie définie par Google [18]. Les SLI affichés sont calculés sur les 28 derniers jours, une durée correspondant à quatre semaines complètes. Ce choix permet de lisser les variations quotidiennes ou hebdomadaires tout en restant représentatif de l'état actuel du service. La période est suffisamment longue pour éviter qu'un incident ponctuel ne fausse la mesure, mais assez courte pour que les problèmes récents ne soient pas dilués dans des moyennes trop anciennes. Elle présente aussi l'avantage d'inclure exactement quatre week-ends et le même nombre de chaque jour de la semaine, contrairement à des périodes de 30 ou 31 jours où cette répartition varie. Le calcul typique du burn rate repose sur une formule dérivée :

$$\text{BurnRate} = \frac{100 - (\frac{\text{good events}}{\text{valid events}} \times 100)}{100 - SLO}$$

Ce calcul vise à estimer la rapidité avec laquelle le budget d'erreur SLO est consommé. Les valeurs affichées sont purement expérimentales, car les SLO officiels ne sont pas encore définis pour nos middlewares.

Figure 6: Tableau expérimental d'évaluation du SLI et burn rate SLO

Pour aller plus loin : vers une intégration complète des pratiques SRE

Au-delà de l'approche retenue dans le cadre de la status page MessMass, il serait possible d'inscrire cette initiative dans une démarche plus large inspirée des pratiques du SRE (Site Reliability Engineering), telles que définies par Google [18]. Dans ce modèle, les SLIs affichés ne constituent pas uniquement un indicateur visuel, mais deviennent la base pour définir des SLOs formels, avec un budget d'erreur explicite et partagé entre les équipes techniques et les métiers [19].

Certaines équipes chez Michelin ont déjà amorcé cette transition en associant leurs SLIs à des SLOs mesurés en continu et suivis dans le temps via des métriques de *burn rate*, concept détaillé dans la littérature SRE [20]. Le burn rate permet d'évaluer la vitesse de consommation du budget d'erreur et d'anticiper les actions correctives avant que les objectifs ne soient dépassés. Dans les implémentations les plus avancées, un burn rate élevé déclenche automatiquement des alertes ou des processus de mitigation [21].

Dans une *status page* plus évoluée, ces concepts pourraient être intégrés directement à l'interface, offrant ainsi une vision combinée de la disponibilité en temps réel, de la conformité aux SLOs et de la tendance de consommation du budget d'erreur. Une telle approche favoriserait une prise de décision proactive et un alignement fort entre objectifs de fiabilité et priorités opérationnelles.

Pour l'équipe HIP, cette évolution représenterait un levier stratégique : elle permettrait non seulement de renforcer la transparence et la prévisibilité vis-à-vis des équipes utilisatrices, mais aussi de structurer un langage commun autour de la fiabilité des services. Bien que cette intégration dépasse le périmètre initial du projet, elle constituerait une étape naturelle vers une maturité SRE accrue au sein du programme OneSystem IT Platforms.

Recording Rules : gain de performance et limitations

Pour garantir que la *status page* reste réactive, même lorsque le volume de données à traiter est important, une optimisation couramment utilisée dans l'écosystème Prometheus consiste à recourir aux **recording rules**. Celles-ci pré-calculent certaines métriques de manière régulière et les stockent pour un accès plus rapide [22].

Nous avons exploré cette approche afin de réduire la charge des requêtes PromQL et d'améliorer les temps d'affichage. Toutefois, nos tests ont révélé des écarts parfois significatifs entre les valeurs enregistrées par les rules et les mesures réelles, allant même jusqu'à produire des SLI supérieurs à 100%. Ce comportement, signe d'une mauvaise agrégation ou d'une instabilité temporaire, a conduit à ne pas intégrer ces *recording rules* dans la version actuelle du dashboard. Ce problème a été signalé à l'équipe Observabilité pour investigation, et ces rules restent une option à réévaluer dans de futures itérations, une fois les problèmes de fiabilité résolus.

9 Cas d'étude : CFT et FTB

L'un des aspects les plus complexes et critiques du projet de création de la status page concernait le monitoring des middlewares CFT (Cross File Transfer) et FTB (File Transfer Box). Ces deux services sont essentiels à Michelin, assurant des transferts de fichiers sécurisés et critiques pour les activités opérationnelles.

9.1 Contexte initial

Au début de mon stage, CFT et FTB étaient parmi les middlewares les plus problématiques à monitorer. En effet, CFT est composé de nombreux serveurs indépendants sans véritable solution de monitoring déjà implémentée, tandis que FTB, un service externe à Michelin, rencontrait des difficultés spécifiques liées au monitoring de liens externes.

9.2 Démarche et premières solutions

Pour aborder ces problèmes, j'ai organisé des réunions avec un des experts des middlewares FTB et CFT, afin d'identifier précisément les besoins et les contraintes techniques associées. Ces discussions ont confirmé la nécessité de trouver une solution efficace pour assurer un monitoring complet de ces services critiques.

Concernant CFT, ma première approche fut de créer un proof-of-concept (PoC) en Python, utilisant des appels API vers le **Flow Manager**, une interface de supervision fournie par Axway permettant d'exposer l'état opérationnel des nœuds CFT et de centraliser les journaux d'activité. Grâce à cet outil, j'ai pu récupérer dynamiquement les statuts des transferts et des composants CFT. Ces données étaient ensuite transmises à Grafana via OpenTelemetry. Ce PoC fonctionnait correctement, permettant un monitoring fiable.

Voici un exemple simplifié du code Python utilisé :

```python
import requests
from otel_metrics import push_cft_metrics

def obtenir_infos_environnement(host, username, password):
    # Authentification et récupération des données...
    return get_response.json()

# Utilisation du script pour les différents environnements

for env, host in environnements.items():
    produits = obtenir_infos_environnement(host, username, password)
if produits:
    total, status_dict = extraire_infos_produits(produits)
    started_count = len(status_dict.get("Started", []))
    push_cft_metrics(env.lower(), total, started_count)
```

Bien que ce PoC ait démontré la faisabilité technique du monitoring de CFT via le Flow Manager, il ne respectait pas les standards en vigueur définis par l'équipe Observabilité. En particulier, le script Python posait plusieurs problèmes : absence d'infrastructure d'hébergement pérenne, manque de supervision centralisée sur l'exécution du script et dépendance à un code personnalisé non validé.

Face à ces limitations, la responsabilité de concevoir une solution conforme aux standards a été transférée à la squad Fusion. Cette dernière travaille actuellement à l'intégration d'un monitoring complet sur l'ensemble des nœuds CFT, en exposant directement des métriques exploitables dans Grafana et ce via un mécanisme validé, scalable, maintenable et interopérable avec la stack OpenTelemetry.

9.3 Problématique spécifique de FTB

Le middleware FTB présentait des difficultés différentes. Il s'agissait d'un service externe nécessitant un monitoring par l'outil Blackbox Exporter fourni par l'équipe Observabilité. Malheureusement, un bug connu de l'outil empêche de monitorer les liens externes, rendant impossible un suivi efficace. Malgré plusieurs échanges et tests avec l'équipe Observabilité, aucune solution viable n'est encore disponible. Par conséquent, le monitoring de FTB a été mis temporairement de côté.

9.4 Résultats et perspectives

À la fin de mon stage, bien que la solution finale et optimale pour CFT et FTB n'ait pas été entièrement déployée, des progrès importants ont été réalisés :

- Pour CFT, une solution officielle a été amorcée grâce à mes recherches initiales et une implémentation complète par la squad Fusion est en cours.
- Pour FTB, même si le problème persiste, les scripts et méthodologies développés durant le stage (notamment via Blackbox Exporter) sont prêts à être mis en production dès que les contraintes techniques seront levées par l'équipe Observabilité.

Ces efforts constituent une avancée significative dans la gestion opérationnelle et la fiabilité de ces middlewares critiques et préparent le terrain à des solutions robustes futures.

10 Cas d'étude : DataStage (IWS) et Boomi

Le monitoring des middlewares DataStage (IWS) et Boomi présentait des défis spécifiques dus à la complexité des environnements et à l'absence préalable de solutions adaptées à ces cas précis.

10.1 Spécificités du monitoring de DataStage

DataStage présente une configuration particulière avec un grand nombre de moteurs indépendants : 48 moteurs en Europe et 8 moteurs en Amérique du Nord. Cette multiplicité rendait complexe la détermination d'un statut global simple (OK/KO). La solution retenue consistait à afficher un indicateur précis montrant le nombre exact de moteurs en panne, associé à une option de drill-down pour accéder aux détails sur les moteurs impactés Figure 7.

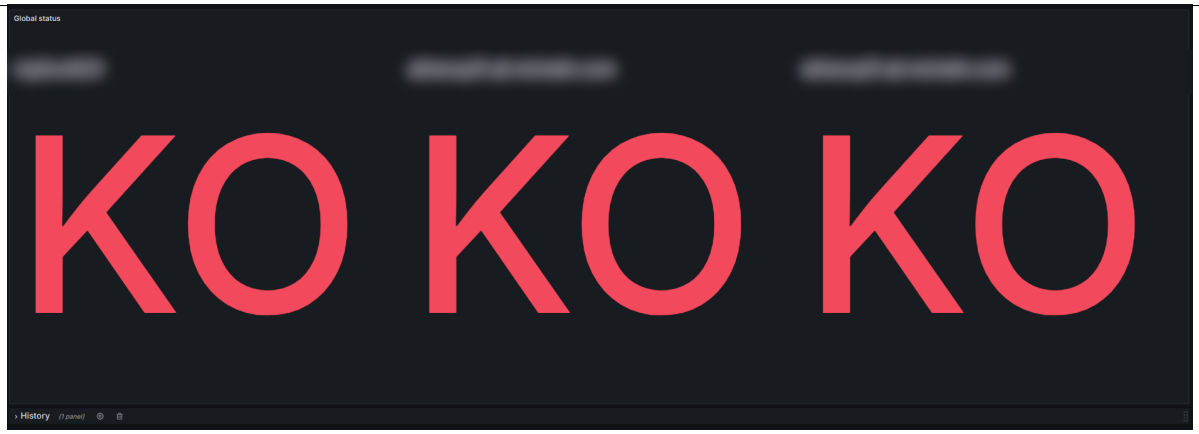

Figure 7: Page "drill-down" de IWS affichant spécifiquement les moteurs KO

Le principe de détection repose sur l'analyse, toutes les 5 minutes, de l'absence simultanée de certains processus critiques sur chaque nœud. Pour cela, Prometheus offre la fonction `absent_over_time`, qui permet de détecter les métriques manquantes sur une fenêtre temporelle définie.

Afin d'identifier les périodes de maintenance planifiée (le dimanche), une variable `is_sunday` a été introduite dans Grafana à l'aide de la formule suivante :

$$is\_sunday = \left\lfloor \frac{(\$\_\_to \bmod 604800000)}{86400000} \right\rfloor = 3 \tag{1}$$

Cette formule fonctionne comme suit :

- $__to représente le timestamp (en millisecondes) de fin d'intervalle.
- Le modulo `604800000` isole la partie écoulée de la semaine (7 jours).
- La division par `86400000` (durée d'un jour en ms) donne le jour courant dans la semaine.
- La fonction plancher (`floor`) convertit ce résultat en entier : 0 pour jeudi, 3 pour dimanche.
- En comparant le résultat à 3, on isole ainsi le dimanche.

Lorsque la valeur retournée est vraie, je donne une valeur arbitraire (par exemple 9999) pour la remplacer par "Backup" en orange pour signaler un état de maintenance, sans impacter les statistiques normales visible sur la Figure 8.

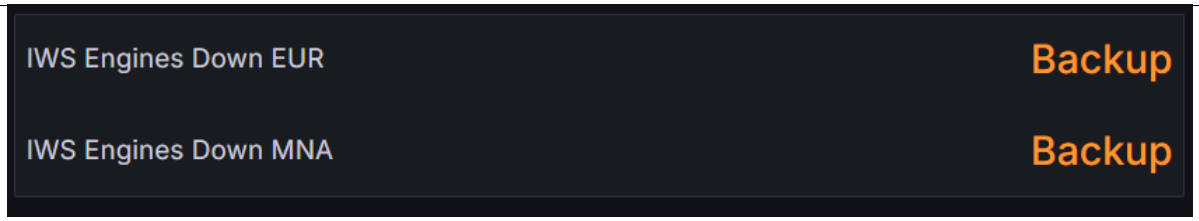

Figure 8: IWS en maintenance le Dimanche

10.2 Monitoring de Boomi : PoC et limitations

Boomi, étant une solution iPaaS externe, ne proposait pas de métriques internes exposées. Pour contourner cette absence, un PoC a été développé avec K6 (outil de test de charge), en interrogeant régulièrement des endpoints REST, des interfaces web conformes à l'architecture *Representational State Transfer*, qui permettent d'échanger des données via des requêtes HTTP standardisées, spécifiques à chaque atome Boomi.

Cependant, une limitation majeure empêchait la mise en production : les runners GitLab fournis par l'équipe Observabilité ne permettent pas d'exécuter des tests vers des URL externes, condition pourtant indispensable dans notre cas. En conséquence, ce PoC a été mis en pause en attendant la levée de cette restriction.

10.3 Solution alternative par monitoring des logs

À défaut d'une exécution active des requêtes, une solution temporaire a été mise en place : l'analyse des logs Boomi. Chaque **atome**, une unité d'exécution locale dans la plateforme Boomi, responsable du traitement et de l'exécution des intégrations dans un environnement donné, est configuré pour générer des logs d'exécution, dont la simple présence est indicative de l'état du service. Une absence prolongée de ces logs est interprétée comme une anomalie.

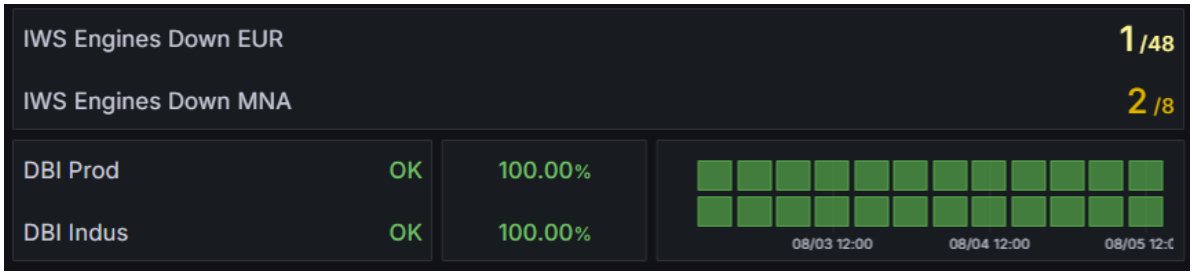

Figure 9: Panel combiné pour Boomi et IWS, illustrant indicateurs binaires, SLI et historique

10.4 Résultats et perspectives

À ce jour, les requêtes sont prêtes à être activées dès que les limitations techniques sur les runners seront levées. En attendant, le monitoring par logs assure un niveau minimal de supervision. À moyen terme, l'intégration complète de K6 permettra plus de précision et la mise en place de SLOs basés sur les SLI permettra une gouvernance rigoureuse de la qualité de service.

11 Cas d'étude : A6E et WMQ

Cette section s'intéresse à deux middlewares aux mécanismes de supervision spécifiques : les nœuds A6E et les Queue Managers IBM WMQ. Tous deux ont été sujets à une intégration personnalisée dans la Status Page, en raison de leurs architectures particulières et du volume de composants à superviser.

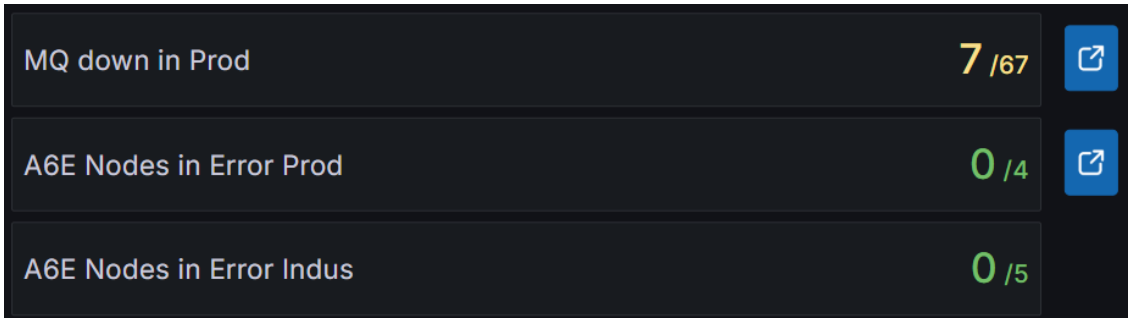

Figure 10: Cellules du statut des nœuds A6E et des Queue Managers

11.1 Contexte initial

Dans le cadre de l'intégration de A6E et WMQ à la Status Page, une réunion a été organisée avec l'expert des technologies de ces middlewares, en l'occurrence l'ingénieur responsable du suivi de ces systèmes. Lors de cet échange, deux axes clairs ont émergé :

- Pour A6E, il était d'ores et déjà possible de s'appuyer sur des logs existants pour récupérer l'état de chaque nœud de façon relativement fine.
- Pour WMQ, la supervision des 67 Queue Managers en production demandait une solution avec plusieurs niveaux d'abstraction, afin de ne pas surcharger visuellement la Status Page.

11.2 Monitoring des nœuds A6E

A6E est un système de traitement en batch, composé de plusieurs nœuds d'exécution autonomes. Ces nœuds transmettent leurs statuts via des logs, ce qui permet une intégration via Loki et une

exploitation simple dans Grafana. Le statut d'un nœud est défini par une variable `statuscode`, pouvant prendre quatre valeurs :

- `10` : OK
- `20` : STBY (Standby)
- `30` : WARN
- `40` : KO

Dans le contexte de la Status Page, les codes `10` et `20` sont considérés comme OK, tandis que `30` et `40` sont interprétés comme KO.

Un total de cinq requêtes Loki ont été définies, une par nœud supervisé, puis une cinquième agrégée permettant de détecter l'absence de logs pour un ou plusieurs nœuds pendant un intervalle de 15 minutes. Cette dernière est utilisée pour déclencher un alerting via l'opérateur PromQL `absent_over_time` :

```
count(
    absent_over_time(({service_name="ra6eeur0"} |= RNMPLL81)[15m]) or
    absent_over_time(({service_name="ra6eeur0"} |= RNMPLL82)[15m]) or
    absent_over_time(({service_name="ra6eeur0"} |= RNMPLL83)[15m]) or
    absent_over_time(({service_name="ra6eeur0"} |= RNMPLL84)[15m])
) * vector(30)
```

Un panneau centralise l'ensemble des statuts, affichant en temps réel la mention OK ou No value (absence de logs) et permet une supervision granulaire tout en offrant un bon niveau de synthèse.

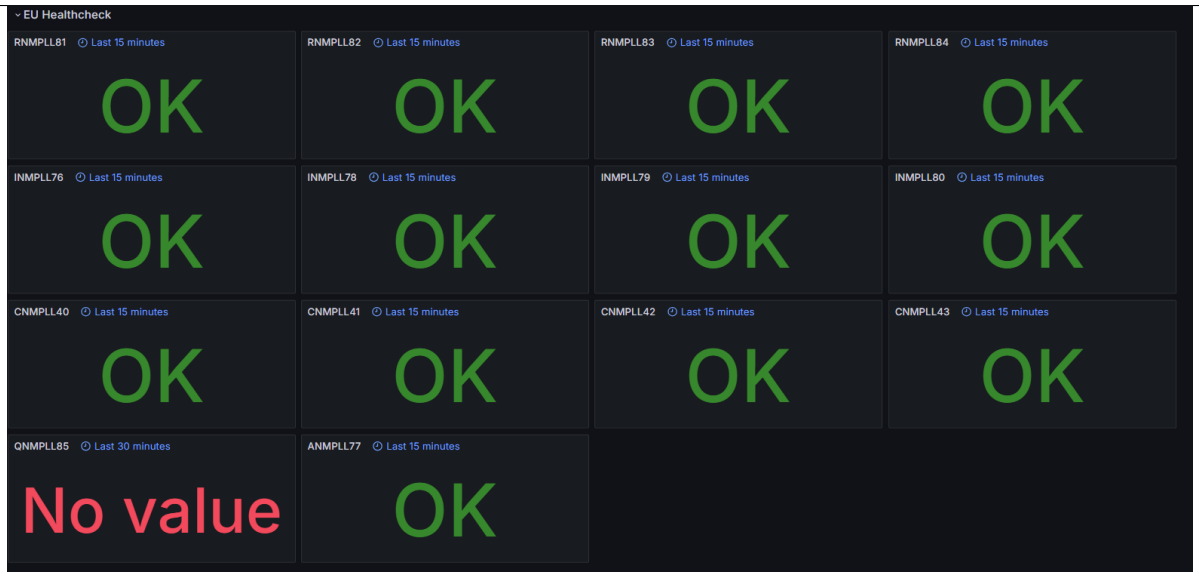

Figure 11: Vue drill-down A6E

11.3 Monitoring des Queue Managers WMQ

Le monitoring de WMQ a demandé une approche hiérarchique en raison du nombre important de Queue Managers : 67 en environnement de production. Chaque QM est un composant critique gérant les files de messages et une panne peut entraîner une rupture de service.

La solution adoptée se décompose comme suit :

- Un panneau de synthèse affichant le nombre de Queue Managers en erreur (MQ Down).
- Un panneau de drill-down listant tous les QMs en KO.
- Une variable dynamique permettant de sélectionner un QM à analyser individuellement.
- Une vue d'historique de l'état sur 3 jours, sous forme de barres temporelles (uptime).
- Un affichage des DLQ (Dead Letter Queues) lorsque leur taux de remplissage dépasse 50

Le niveau de détail fourni est l'un des plus riches du système, grâce à l'utilisation de variables, de panels multiples et d'un historique étendu. Cela permet une analyse fine et rapide de l'état d'un QM, y compris l'état de saturation de ses files DLQ.

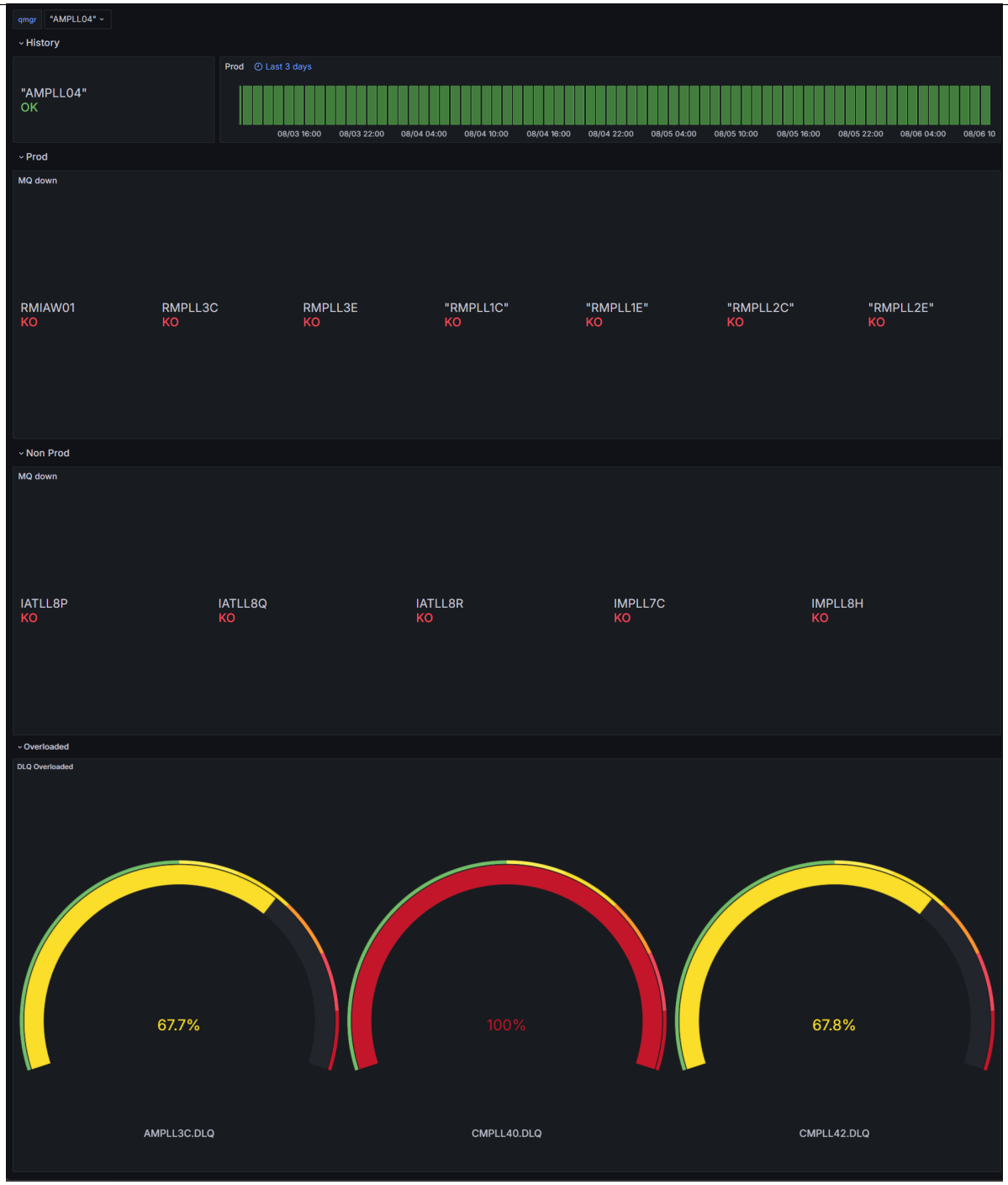

Figure 12: Vue drill-down WMQ

12 Cas d'étude : IFE et E2E

La supervision des applications IFE (Integration Flows Extension) et E2E (End-to-End) a été abordée dans un second temps lors d'une réunion dédiée avec l'expert sur ces sujets. Bien que les deux systèmes partagent certaines similarités en termes de mécanismes de supervision, leurs architectures internes ont nécessité une approche adaptée pour chacun.

12.1 Contexte technique et difficultés

12.1.1 E2E

Le système E2E est conçu pour monitorer plusieurs composants (API, client, listener) sur deux régions (EUR et MNA). La difficulté principale rencontrée fut l'absence de métriques de supervision pour le backend de la base MongoDB. En effet, seule la partie frontend était exposée via un Blackbox Exporter, ce qui restreignait le monitoring à une vision partielle de la disponibilité du service.

Pour remédier à cette limitation, une proposition de monitoring de la base MongoDB a été poussée en Backlog, en attente de l'exposition de métriques exploitables via un outil comme Prometheus ou Grafana Agent.

12.1.2 IFE

IFE est composé de trois briques : **frontend**, **backend** et **dataloader**. Lors des premiers tests, ces trois composants étaient monitorés de façon séparée, ce qui complexifiait la lecture sur la Status Page. À la suite de discussions, il a été décidé de définir un **statut composé**, qui retourne une erreur globale si un seul des composants est en panne. Cette agrégation simplifie la lecture tout en conservant un accès aux détails via une vue drill-down.

Figure 13: Panel de supervision de E2E : statut par région et composants

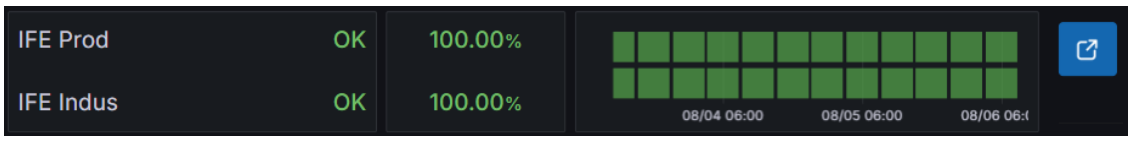

Figure 14: Panel de supervision de IFE : statut global des environnements Prod et Indus

12.2 Améliorations récentes et monitoring par Kubernetes

Une amélioration a été apportée au monitoring de ces deux applications, grâce à l'utilisation directe des métriques issues de Kubernetes. Cela a permis de contourner certaines limites du Blackbox Exporter et d'accéder à des métriques internes liées aux conteneurs, notamment les redémarrages anormaux.

Le principe repose sur une requête PromQL qui filtre les redémarrages par conteneur dans un namespace donné. Si un redémarrage est détecté, l'état est marqué KO dans le drill-down.

```
(
count by(container) (
        label_replace(
        k8s_container_restarts{namespace="re2eeur0"},
        "container",
        "$1",
        "container_name",
        "^[^-]+-(.*)$"
        )
    ) > bool 0
)
```

La requête est identique pour IFE, avec uniquement le namespace changé :

```
(
count by(container) (
        label_replace(
        k8s_container_restarts{namespace="rifegbl0"},
        "container",
        "$1",
        "container_name",
        "^[^-]+-(.*)$"
        )
    ) > bool 0
)
```

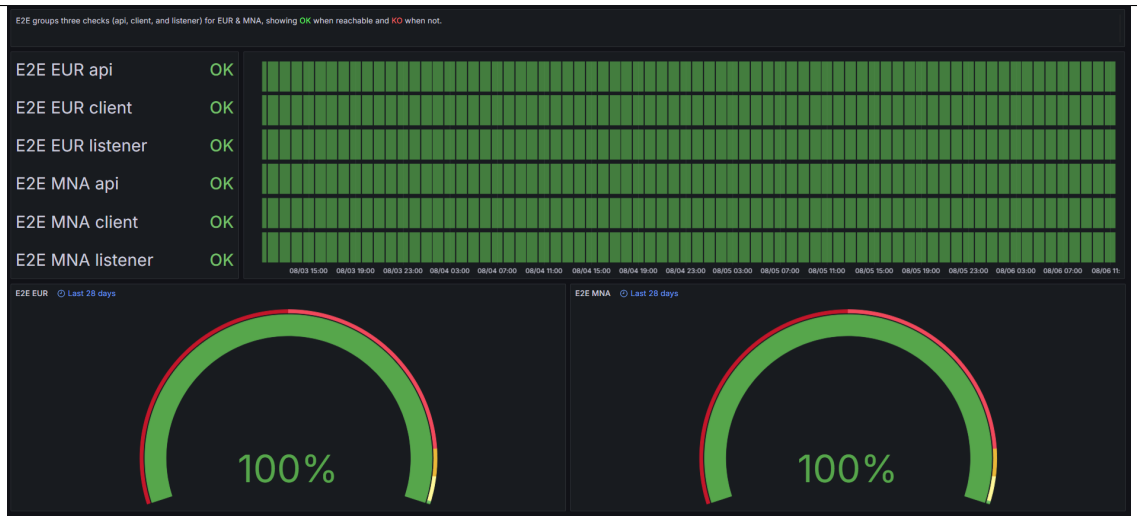

Figure 15: Drill-down E2E : répartition des statuts par conteneur

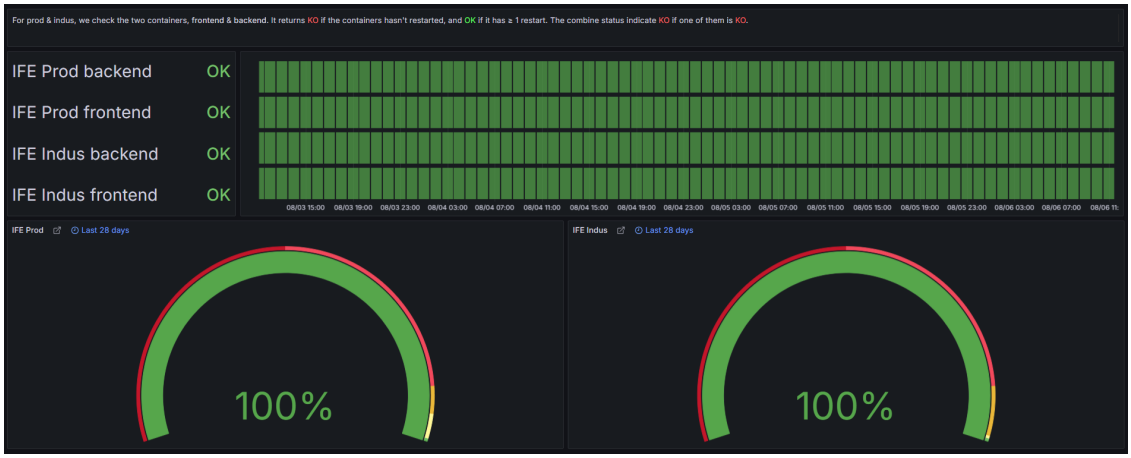

Figure 16: Drill-down IFE : analyse détaillée des composants

12.3 Bilan

L'utilisation des métriques Kubernetes a permis d'étendre la profondeur d'analyse sans alourdir la Status Page. En masquant la complexité sous-jacente, tout en rendant accessibles les informations critiques dans les vues drill-down, l'approche retenue combine clarté, maintenabilité et réactivité pour les équipes opérationnelles.

13 Evolution de la Status Page

La Status Page a connu plusieurs phases de maturation, depuis un premier prototype rudimentaire jusqu'à une version finalisée et validée par l'ensemble des équipes HIP. Cette section retrace l'évolution de l'outil, en mettant en évidence les améliorations techniques et esthétiques successives, ainsi que les retours utilisateurs.

13.1 Première version : PoC Blackbox Exporter

La première version de la Status Page reposait exclusivement sur l'utilisation du **Blackbox Exporter**, permettant uniquement un monitoring basique de disponibilité de frontends. Chaque service était présenté de manière isolée, sans structuration avancée, ni logique d'agrégation des statuts. L'arrangement visuel était encore désordonné, avec peu d'harmonisation entre les composants.

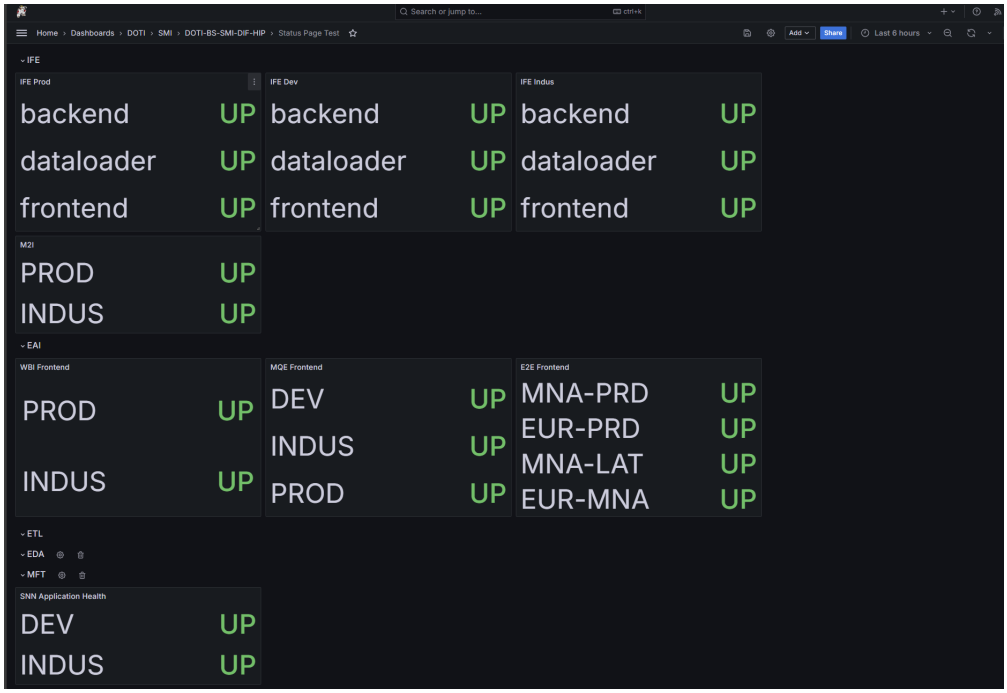

Figure 17: Première version de la Status Page (PoC avec Blackbox Exporter)

13.2 Deuxième version : Status Page Test avec nouveaux statuts

La deuxième version introduit des statuts plus détaillés, en explorant des visualisations plus riches comme les **canvas Grafana**. Certaines expérimentations ont été menées sur l'affichage de IFE avec des blocs dynamiques. Néanmoins, cette approche a été jugée trop complexe visuellement et a été finalement abandonnée.

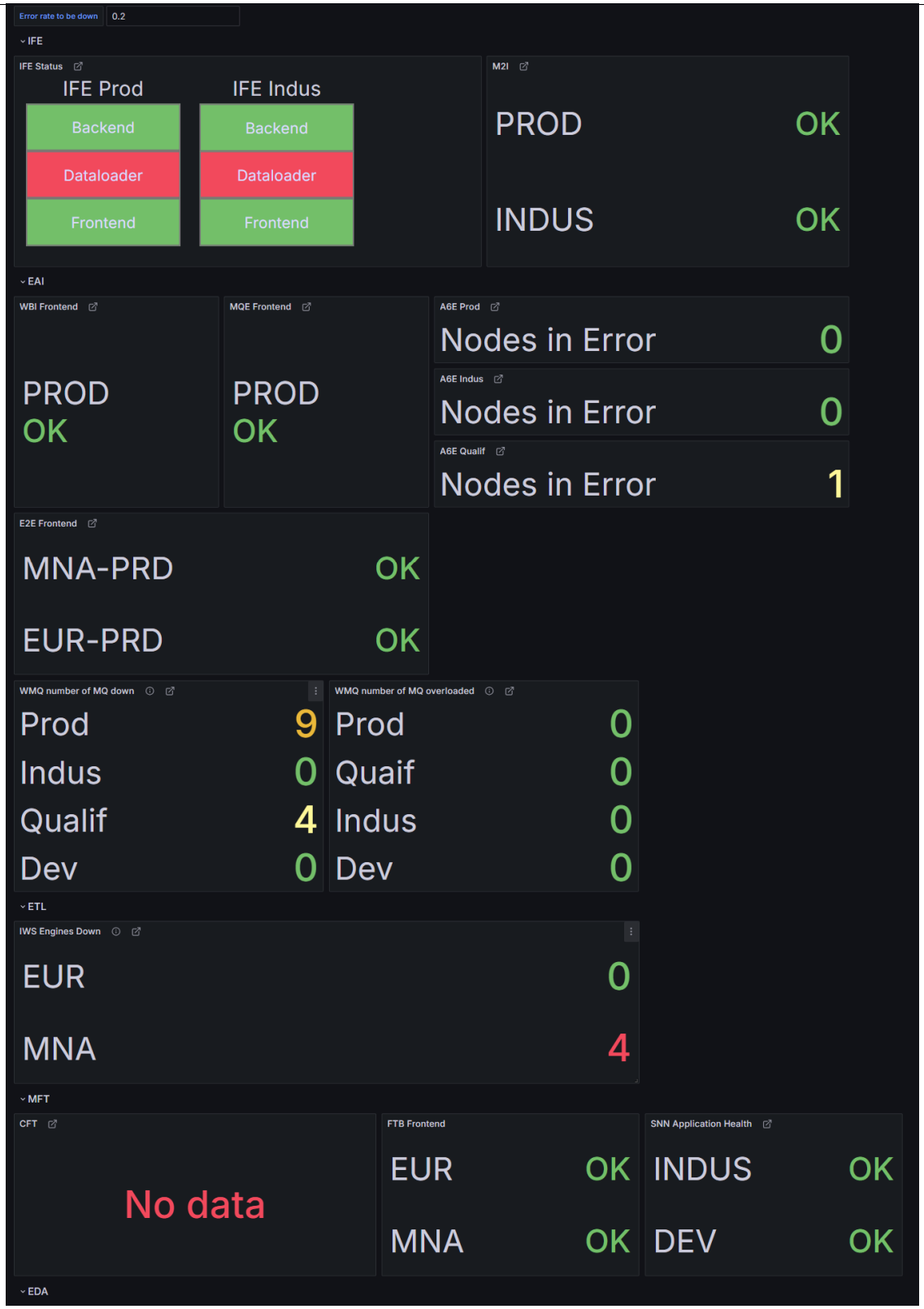

Figure 18: Deuxième version : Ajout de composants, mais disposition encore chaotique

13.3 Troisième version : Structuration par domaine et intégration API

Chaque domaine fonctionnel (EDA, MFT, EAI, IFE/M2I, ETL) dispose désormais de sa propre section clairement identifiée. Cette version est également la première à intégrer les informations issues de l'équipe API, posant ainsi les bases d'une version quasi finale de la page, tant sur le fond que sur la forme.

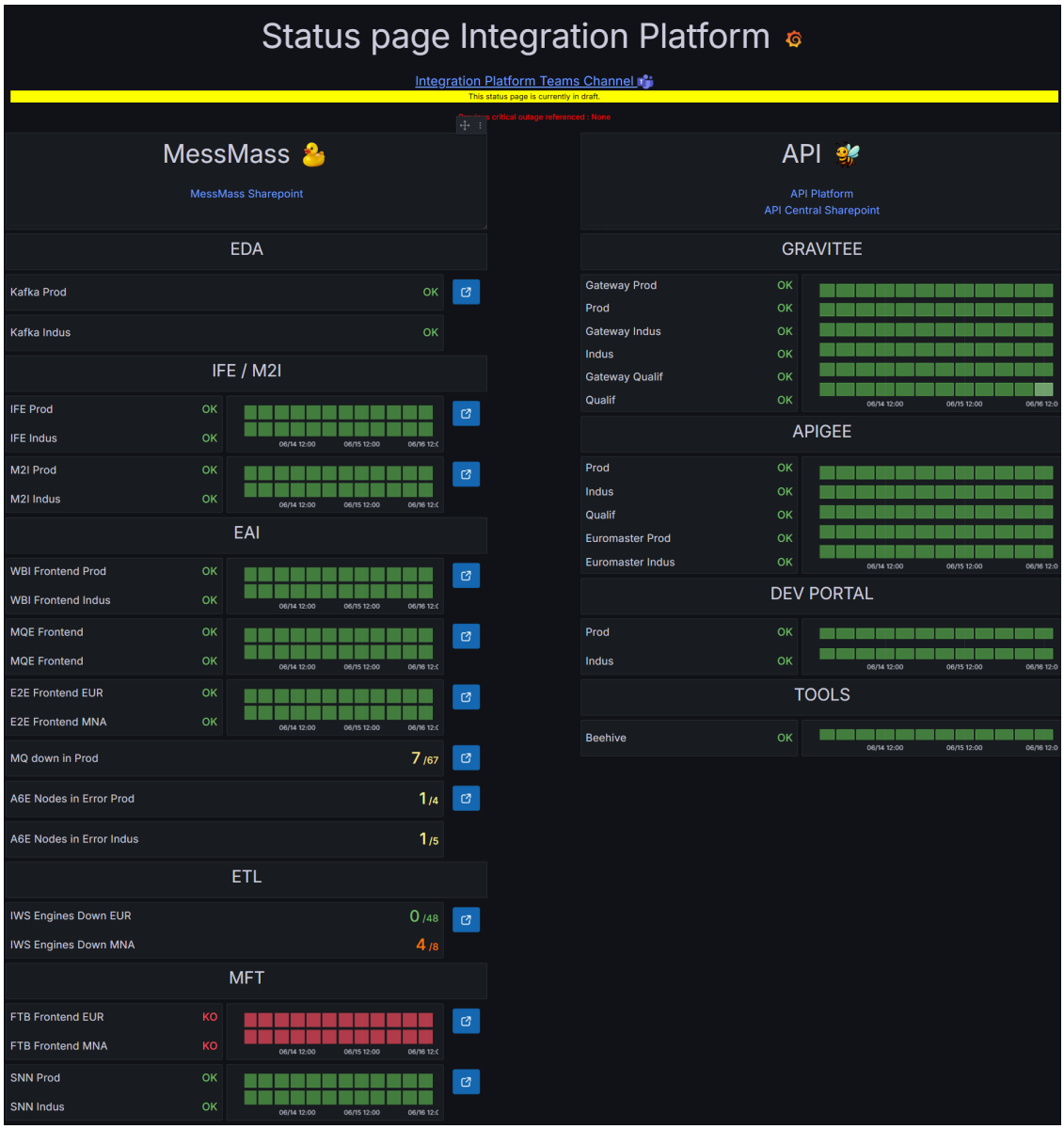

Figure 19: Troisième version : Organisation par domaine, ajout des SLI et de l'API

13.4 Version finale : Feedback utilisateurs et validation HIP

Une réunion avec tous les membres HIP a permis de présenter la Status Page dans sa version finalisée. Cette version propose une disposition harmonisée, un affichage synthétique (statut, historique, SLI) et des liens vers les sources (incidents ServiceNow, Sharepoints).

Figure 20: Version finale présentée à l'ensemble HIP

13.5 Analyse des retours utilisateurs

Un formulaire anonyme a permis de recueillir 15 retours sur la version finale. Les questions portaient sur la pertinence de la publication, le format des SLI, la gestion des liens vers ServiceNow et le bandeau d'information.

Question	Résultat majoritaire
Statut de préparation	4/5 (prêt ou presque)
Format des SLI	Flottant avec 2 décimales
Lien ServiceNow	Conserver
Message en haut de page	Conserver (majoritaire)

Table 1: Synthèse des avis utilisateurs

13.6 Conclusion

La Status Page a évolué d'un outil de test fragmentaire vers une solution centralisée et professionnelle, intégrant supervision technique et retour utilisateur. Les améliorations futures porteront sur l'actualisation du message d'en-tête, l'enrichissement des SLOs et la récupération automatique de messages depuis les canaux Teams.

14 Documentation et pérennisation

En parallèle du développement technique, j'ai également produit deux documentations complémentaires afin d'assurer la pérennité et l'accessibilité de la *status page* :

- **Une documentation technique interne**, à destination de l'équipe HIP. Elle regroupe l'ensemble des spécifications de mise en œuvre, les bonnes pratiques de développement dans Grafana, les règles de nommage, les logiques d'alerte, ainsi que les procédures pour modifier ou déployer les cellules. Elle a pour objectif de faciliter la reprise en main et la maintenance future de l'outil par les équipes internes.
- **Une documentation utilisateur**, plus synthétique et orientée grand public. Elle permet aux utilisateurs non techniques de comprendre la signification des indicateurs, la logique des cellules (stat, historique, drill-down, SLI) et les modalités d'accès aux informations de suivi (incidents, changements, état de la plateforme). L'objectif est ici d'améliorer l'adoption de l'outil tout en rendant son usage plus autonome.

15 Conclusion

Ce projet de status page marque une étape fondatrice dans l'évolution des pratiques d'observabilité chez Michelin. Il répond à un besoin critique de l'entreprise : disposer d'une vision claire, centralisée et en temps réel de l'état de ses services. L'outil conçu permet désormais de suivre les indicateurs clés de disponibilité et de performance pour un large panel de middlewares, en s'appuyant sur des technologies modernes, open-source et adaptées aux contraintes internes.

Au cours de ce stage, j'ai pu appréhender la complexité d'une architecture technique à grande échelle, en lien avec des enjeux opérationnels forts. J'ai également eu l'occasion de mettre en pratique une approche produit orientée utilisateurs, en cherchant constamment l'équilibre entre lisibilité, pertinence technique, maintenabilité et flexibilité. Les choix d'architecture (notamment l'utilisation de Grafana et des indicateurs visuels normalisés) ont été déterminants dans la qualité de l'interface finale.

Cependant, ce projet ne peut être considéré comme finalisé. Plusieurs contraintes actuelles réduisent encore son impact potentiel. En premier lieu, la restriction d'accès à Grafana empêche certains utilisateurs métier de consulter la status page. Or, pour remplir pleinement son rôle d'outil de communication transversale, celle-ci devra être accessible publiquement. La création d'une solution permettant de publier tout ou partie de la status page à destination d'utilisateurs non authentifiés dans Grafana constitue donc un axe prioritaire.

Un PoC allant dans ce sens a été réalisé. Il repose sur un compte bot capable d'extraire automatiquement les données de la status page interne, pour les injecter dans une page HTML externe, publiable sur un site web. Cette alternative temporaire ouvre une voie de contournement fonctionnelle, en attendant qu'une solution officielle soit mise en place par l'équipe Observabilité. Elle montre à la fois la faisabilité technique de l'exposition publique et la possibilité de décorréler la visualisation de Grafana lui-même.

Au-delà de l'accessibilité, plusieurs prochaines étapes clés devront être abordés :

- Intégrer un monitoring basique via K6 pour des middlewares simples, afin de détecter automatiquement les cas de rupture de service.
- Ajouter le support du middleware FTB, dès que les URLs publiques de test seront disponibles.
- Concevoir des scénarios de supervision complexes via K6, simulant des parcours réels d'utilisateurs (connexion, transfert de fichier, récupération) pour vérifier le bon fonctionnement global des services.
- Identifier les SLOs (Service Level Objectives) de chaque middleware, en collaboration avec les experts techniques et les intégrer comme référence dans la status page.
- Implémenter des recording rules dans Prometheus, permettant de pré-calculer certains indicateurs clés. Cela permettra d'alléger significativement la charge de calcul au niveau de Grafana, améliorant ainsi les temps de chargement et la stabilité de l'interface.

Ce projet a également mis en lumière l'importance de la documentation : deux guides ont été rédigés en parallèle, l'un technique pour les équipes HIP, l'autre à destination des utilisateurs métiers. Cette double documentation garantit une meilleure transmission des connaissances et une autonomisation progressive des utilisateurs.

D'un point de vue personnel, ce stage a été une expérience très enrichissante. Il m'a permis d'évoluer dans un environnement technique exigeant, de collaborer avec des experts aux compétences pointues et variées et de prendre des responsabilités dans la conception d'un outil critique. J'ai également pu mettre à l'œuvre des compétences transverses : communication, gestion de projet, découverte de méthodologies agiles et prise d'initiative.

En conclusion, cette status page pose les fondations d'une gouvernance plus proactive et transverse des services IT. Le chemin reste encore long, mais les bases sont solides. Ce projet ouvre aussi la voie vers une observabilité dite "2.0", qui combine logs, métriques, traces et corrélation contextuelle automatisée. Selon la CNCF, cette approche permettra d'anticiper les pannes critiques par analyse comportementale et détection proactive [23]. Le projet peut être vu comme un catalyseur, appelé à évoluer vers une plateforme d'observabilité plus large, au service d'une meilleure maîtrise de la performance applicative chez Michelin.

References

[1] GROUPE MICHELIN, Rapport annuel 2023. https://www.michelin.com/en/publications/regulated-information/financial-information-at-december-31-2023, 2023. Accédé le 10 juillet 2025.

[2] CGT63, En direct de michelin licenciements et restructurations. https://www.cgt63.fr/index.php/fr/en-direct-de-michelin, 2025. Accédé le 10 juillet 2025.

[3] GROUPE MICHELIN, Michelins purpose all sustainable. https://www.michelin.com/, 2025. Consulté en août 2025.

[4] GROUPE MICHELIN, Towards 100% sustainable materials by 2050. https://www.michelin.com/, 2025. Objectifs matériaux durables, consulté en août 2025.

[5] GROUPE MICHELIN, Document denregistrement universel 2024 performance extra-financière. https://www.michelin.com/, 2025. Sections ESG/RSE consolidées, consulté en août 2025.

[6] FORRESTER CLIENT STORIES, Michelin's ea org relies on forrester to hit its kpis. https://www.forrester.com/client-stories/michelin-enterprise-architecture/, 2023. Accédé le 10 juillet 2025.

[7] T. FRAUDET, A story on how michelin continuously modernizes its information system, *Michelin IT Engineering Blog*, 2021. Accédé le 10 juillet 2025.

[8] ATLASSIAN PTY LTD, What is agile at scale?. https://www.atlassian.com/agile/agile-at-scale, 2025. Accédé le 10 juillet 2025.

[9] A. BANDURCHIN, Grafana vs splunk - features, pricing, and performance compared. Uptrace website, 2024. Consulté en août 2025.

[10] Grafana official documentation. https://grafana.com/docs/, 2024. Accédé le 30 juillet 2025.

[11] R. SHARMA et N. KAPOOR, Unified monitoring for microservices: Implementing prometheus and grafana for scalable solutions, *International Journal of Software Engineering and Knowledge Engineering*, vol. 34, p. 53–71, march 2024.

[12] Grafana panel types and options. https://grafana.com/docs/grafana/latest/panels/, 2024. Accédé le 30 juillet 2025.

[13] Grafana variables. https://grafana.com/docs/grafana/latest/variables/, 2024. Accédé le 30 juillet 2025.

[14] Grafana alerting. https://grafana.com/docs/grafana/latest/alerting/, 2024. Accédé le 30 juillet 2025.

[15] The go programming language. https://go.dev/, 2024. Accédé le 30 juillet 2025.

[16] Grafana github repository. https://github.com/grafana/grafana, 2024. Accédé le 30 juillet 2025.

[17] G. LABS, Observability survey 2025: The rise of the open source stack, june 2025. Consulté en août 2025.

[18] B. BEYER, C. JONES, J. PETOFF et N. R. MURPHY, *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, 2016.

[19] G. CLOUD, Sli, slo, sla: Making sense of service level objectives, 2020. Présentation des définitions et relations entre SLI, SLO et SLA dans le cadre SRE.

[20] B. BEYER, N. R. MURPHY, J. PETOFF, K. RENSIN et S. T. KAWAHARA, *The Site Reliability Workbook: Practical Ways to Implement SRE*. O'Reilly Media, 2018.

[21] G. CLOUD, Slo burn rate alerts, 2022. Documentation technique Google Cloud sur la mise en place d'alertes basées sur le burn rate.

[22] PROMETHEUS AUTHORS, Prometheus recording rules. https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/, 2025. Accessed: 2025-08-06.

[23] CLOUD NATIVE COMPUTING FOUNDATION, What is observability 2.0?, january 2025. Consulté en août 2025.