WebScrepStatusQlik: Monitoramento Automatizado de Tarefas do Qlik Sense com Envio de Alertas via WhatsApp

Wagner Filho wagner.helio@discente.ufg.br
Carlos Miqui carlosmiqui@discente.ufg.br
Pedro Koziel pedrokoziel@discente.ufg.br
Marcos Medeiros marcos.medeiros@discente.ufg.br
Hailton Lemos haiilton@discente.ufg.br

Instituto de Pós-Graduação, Universidade Federal de Goiás

***Abstract*—Este artigo apresenta o WebScrepStatusQlik, um sistema para a coleta, transformação e notificação automatizada de falhas na execução de tarefas do Qlik Sense. Em organizações públicas e privadas, a confiabilidade das plataformas de Business Intelligence (BI) é de suma importância, entretanto, o Qlik Sense carece de mecanismos nativos eficientes para a detecção e comunicação automatizada de falhas operacionais. Essa deficiência resulta, frequentemente, em uma latência significativa para a identificação de erros críticos, dificultando uma governança de dados proativa. A solução proposta aborda essa lacuna ao integrar a extração dinâmica de dados (web scraping), por meio do Selenium, com o processamento de informações e a geração padronizada de relatórios. Adicionalmente, o sistema utiliza a EvolutionAPI para a disseminação automatizada de alertas via WhatsApp, possibilitando notificações em tempo real. O trabalho delineia a arquitetura técnica, aborda os aspectos éticos e legais envolvidos no uso de dados e apresenta diretrizes para a reprodutibilidade do projeto. A validação em ambiente de produção demonstrou uma redução substancial no tempo médio de resposta, o que confirma a eficácia do sistema em reforçar a resiliência operacional dos ecossistemas de BI.**

I. Introdução

A crescente dependência de sistemas de Business Intelligence (BI) impõe desafios significativos quanto à confiabilidade e continuidade dos processos de análise de dados [1]. Em plataformas como o Qlik Sense, falhas na execução de tarefas de ETL (Extract, Transform, Load) podem comprometer diretamente a consistência dos dashboards analíticos e impactar negativamente a tomada de decisão baseada em dados.

Apesar de seu valor como ferramenta de visualização e modelagem, o Qlik Sense não oferece mecanismos nativos eficazes para detecção automatizada de falhas em tempo real. Em ambientes client-managed, por exemplo, não há suporte integrado para notificações de erros em recargas de apps, e as alternativas existentes são limitadas a e-mails básicos sem personalização ou estruturação. Ferramentas comerciais como SenseOps Alerting ampliam essas capacidades com alertas acionados por condições em nível de dados e envio via WhatsApp, e-mail e Teams [2]. Porém, ainda são necessarios projetos open source que permitam auxiliar na monitoração unificada de recargas em Qlik Sense (on-premise e Cloud), com templates de alertas configuráveis para diversos meios e logs de script para facilitar o diagnóstico da causa de falhas. [3].

Nesse contexto, o projeto **WebScrepStatusQlik** propõe uma solução híbrida, automatizada e de rápida implementação para monitoramento contínuo desse ecossistemas. Utilizando-se de técnicas de *web scraping* com Selenium para inspeção da interface web e integração com APIs de mensageria como Evolution API (compatível para envio de mensagens com WhatsApp), além da geração de relatórios dinâmicos. O sistema permite identificar problemas e realizar notificações para dispositivos diversos, reduzindo significativamente a latência operacional.

Essa abordagem alinha-se aos princípios do DataOps, que combinam automação, integração contínua e monitoramento sistemático ao longo do ciclo de vida dos dados e eventos [4], [5]. Ferramentas modernas de observabilidade em pipelines de BI apontam a importância de detectar anomalias ou falhas em tempo real, apoiando intervenções proativas [6]. Em particular, frameworks como DQSOps permitem obter pontuações de qualidade de dados dentro de workflows de DataOps [7], enquanto abordagens como Auto-Validate by History detectam automaticamente problemas de integridade em pipelines recorrentes de BI/ML [8].

A justificativa técnica do WebScrepStatusQlik reside na necessidade de mecanismos escaláveis e adaptáveis para monitorar múltiplas instâncias QMC (Qlik Management Console) e módulos como NPrinting, eliminando a leitura manual de logs e dashboards internos. Isso resulta em redução do tempo médio de restauração e maior rastreabilidade de eventos críticos. Por fim, esta pesquisa contribui para preencher lacunas em soluções comerciais e acadêmicas, ao propor uma arquitetura de baixo custo, com foco no ecossistema Qlik e sendo uma proposta aplicável em ambientes de produção reais.

II. Fundamentação Teórica

O monitoramento contínuo de pipelines de dados tem se tornado uma prática essencial na garantia da qualidade, resiliência e confiabilidade de sistemas analíticos. Estudos recentes destacam a importância de incorporar práticas de *observabilidade de dados* em ambientes que utilizam ETL

(Extract, Transform, Load), visando reduzir custos operacionais e detectar anomalias com agilidade [9], [10].

A observabilidade de dados difere do simples monitoramento por envolver a coleta, agregação e análise contínua de logs, métricas e traços distribuídos, promovendo uma visão unificada e preditiva do estado dos pipelines [11]. Essas abordagens são cada vez mais adotadas em arquiteturas orientadas a eventos, microserviços e sistemas distribuídos.

No contexto do Qlik Sense, essa necessidade se intensifica pela ausência de mecanismos nativos para alertas e diagnósticos automáticos. O sistema *WebScrepStatusQlik* é embasado em pilares teóricos reconhecidos na automação de coleta de dados, monitoramento de pipelines, mensageria e arquitetura modular de dados.

A. Web Scraping e Automação

O *web scraping* é uma técnica empregada para a extração de dados de páginas web, especialmente útil quando APIs nativas são inexistentes ou oferecem acesso limitado [12]. Ferramentas e frameworks como Selenium WebDriver e Playwright permitem a automação de interações humanas — como cliques, autenticação e navegação — viabilizando a coleta de informações a partir de interfaces dinâmicas, como o Qlik Management Console (QMC) ou o NPrinting, mesmo quando essas plataformas são protegidas por mecanismos de autenticação.

O uso dessas técnicas como solução para o monitoramento de dashboards e consoles administrativos tem se mostrado eficaz em cenários onde a integração não é viável. Essa abordagem possibilita a observação direta da interface gráfica de administração, permitindo transformar comportamentos relevantes em alertas estruturados e acionáveis [13].

B. Arquitetura de BI e Qlik Sense

O Qlik Sense é uma plataforma de Business Intelligence que opera com uma arquitetura associativa, permitindo o consumo interativo de dados por meio de dashboards visuais. Seu ecossistema é composto por vários componentes chave que, juntos, orquestram a análise de dados, mas que também apresentam desafios de monitoramento em ambientes de produção. Os elementos centrais deste ecossistema incluem:

- **QMC (Qlik Management Console)**: É a interface administrativa central utilizada para orquestrar tarefas de atualização de dados, como os processos de ETL. Além da orquestração, o QMC é responsável pelo gerenciamento de usuários e pelo controle de segurança da plataforma. Em ambientes corporativos, é comum a existência de múltiplas instâncias especializadas, como, por exemplo, um QMC para tarefas de estatística (QAP) e outro para a gestão de painéis (HUB).
- **NPrinting**: Corresponde ao módulo destinado à criação e distribuição automatizada de relatórios personalizados. Sua operação é baseada em tarefas agendadas, permitindo a entrega recorrente de informações consolidadas para diferentes stakeholders.
- **Arquitetura de Tarefas**: Refere-se ao sistema subjacente que coordena os ciclos internos de ETL (Extract, Transform, Load). Esta arquitetura é fundamental, pois define as dependências e a sequência de execução entre as diversas cargas de dados, garantindo a consistência das informações apresentadas nos painéis.

Apesar da robustez da plataforma para visualização e análise de dados, seus recursos nativos de monitoramento e alerta são considerados limitados, especialmente quando comparados a soluções modernas de ITSM (IT Service Management). Essa carência de mecanismos de alerta automatizados dificulta a detecção proativa de falhas, constituindo um risco operacional para a governança dos dados.

C. Padrões de Projeto

A arquitetura de soluções para monitoramento de dados pode se beneficiar do uso de padrões de projeto (*design patterns*), que auxiliam na modularização, reusabilidade e manutenibilidade dos componentes do sistema [14]. Esses padrões têm sido aplicados com sucesso em plataformas que visam escalar a captura de logs, observação de execução de jobs e encaminhamento de falhas para canais de notificação adequados. A aplicação de padrões como Observer e Scheduler é utilizado de forma estratégica em arquiteturas de monitoramento. O Observer Pattern, por exemplo, permite notificar múltiplos subscritores sobre mudanças de estado, promovendo acoplamento frouxo entre componentes. Já o Scheduler Pattern garante execuções periódicas de coleta, mantendo vigilância contínua sem intervenção humana. Esses padrões promovem modularidade, governança e escalabilidade em soluções distribuídas.

D. Arquitetura de Dados em Camadas

Trabalhos como o de Dachary [15] e Ogunwole et al. [16] reforçam a importância de sistemas de monitoramento adaptáveis, com suporte à orquestração distribuída e envio de notificações multicanal. Esses sistemas, além de reduzirem o tempo médio de recuperação, contribuem para a rastreabilidade de eventos críticos e promovem a governança de dados em tempo real.

No ecossistema atual de engenharia de dados, a construção de pipelines robustos não se resume à sua execução técnica. É fundamental assegurar que os processos operem de forma confiável ao longo do tempo, o que demanda um monitoramento inteligente, contextualizado e segmentado em camadas [17]:

- **Raw Layer**: captura inicial bruta de dados via scraping ou interface;
- **Clean Layer**: padronização, filtragem e validação da informação;
- **Analytics Layer**: consolidação de dados em relatórios e dashboards estruturados.

Esse modelo se alinha com arquiteturas de data lake centradas em múltiplas zonas (raw, cleaned, curated ou bronze, prata e ouro), promovendo separação de responsabilidades e rastreabilidade operacional.

Em contextos complexos, técnicas de *adaptive monitoring* permitem ajustar dinamicamente o comportamento de coleta e alerta conforme variações do ambiente, conforme mapeado em estudos sistemáticos. Ferramentas modernas de observabilidade recomendam monitoramento proativo baseado em métricas, logs e eventos para detecção de anomalias em tempo real. Além disso, arquiteturas em camadas como a *lambda architecture* combinam processamento em batch e em tempo real, balanceando precisão e latência — princípios que influenciam o modelo híbrido do WebScrepStatusQlik para reduzir latência sem comprometer confiabilidade dos dados.

E. Mensageria com API para WhatsApp

A integração com APIs de mensageria, como a Evolution API, Z-API e outras, possibilita comunicação automatizada por meio do WhatsApp e que oferece envio de mensagens, imagens, PDFs, logs e suporte a sessões simultâneas via WebSocket ou WebHook. O uso de APIs viabiliza notificações instantâneas e escaláveis, com indicadores de entregabilidade, leitura e engajamento. Ademais, a eficiência operacional e satisfação do usuário são ganhos que ocorrem quando implementadas outras automações, incluindo chatbots AI e workflows integrados com CRM.

F. Síntese Teórica

A solução proposta sintetiza esses fundamentos em uma abordagem prática e organizada da seguinte forma:

- Web scraping automatizado para preencher lacunas de APIs proprietárias;
- Arquitetura modular de dados que assegura integridade e rastreabilidade;
- Observabilidade contínua de pipelines para permitir alertas inteligentes e diagnósticos;
- Mensageria via API para o WhatsApp que habilita comunicação ágil com stakeholders.

Em suma, há um consenso crescente na literatura de que ferramentas de observabilidade e automação são peças-chave para garantir a integridade de ecossistemas de BI. Soluções como a deste trabalho, se alinham com as recomendações para pipelines e democratização do acesso a alertas operacionais [9], [10], [15]. Essa combinação confere ao sistema capacidade de operar com múltiplas instâncias Qlik Sense, minimizar tempo de resposta a falhas e oferecer relatórios estruturados e acionáveis em ambientes corporativos.

III. Metodologia

Esta seção descreve os componentes, fluxos e estratégias de implementação utilizadas.

A. Definição do Problema

A ausência de recursos nativos no Qlik Sense para alertas sobre falhas de execução compromete a governança dos dados e a confiabilidade dos painéis de decisão. Os principais desafios enfrentados incluem: latência elevada na detecção de falhas, falta de notificações em tempo real e inexistência de histórico sistemático das execuções.

B. Objetivos da Solução

O sistema foi projetado com os seguintes objetivos: coletar periodicamente o status das tarefas em diferentes instâncias QMC e NPrinting; identificar falhas e gerar logs de erro; estruturar essas informações em relatórios formais em HTML e PDF; e distribuir os relatórios e alertas por meio de mensagens via WhatsApp.

C. Arquitetura do Sistema

A arquitetura é composta por cinco módulos principais: agendamento, coleta de dados, gerenciamento de logs, geração de relatórios e envio de notificações. Cada módulo é implementado como um componente independente, permitindo substituições e evoluções sem impacto nos demais.

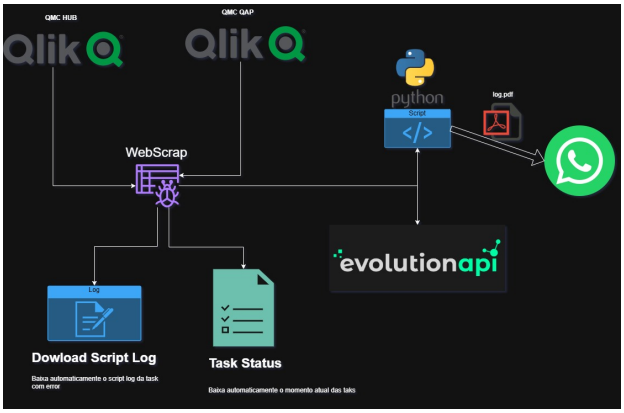

Fig. 1. Arquitetura do pipeline WebScrepStatusQlik

O módulo de agendamento executa periodicamente os coletores de status, garantindo a automação contínua sem necessidade de supervisão manual. O módulo de coleta utiliza Selenium para navegar pelas interfaces QMC e NPrinting, extrair o status das tarefas e identificar falhas.

Os dados extraídos são armazenados temporariamente e organizados por camada (raw e clean), permitindo rastreabilidade e controle de qualidade. O módulo de processamento de logs implementa lógica de detecção e recuperação de mensagens de erro, incluindo tentativa de reinício automático de tarefas falhadas.

Em seguida, os dados são transformados em relatórios HTML com estilos aplicados por classes CSS que indicam a criticidade (sucesso, falha, erro, etc.). Esses relatórios são então convertidos para PDF por meio da biblioteca *xhtml2pdf*, garantindo portabilidade e conformidade com padrões corporativos.

Por fim, o módulo de notificações coleta os relatórios gerados e envia as informações via WhatsApp para números e grupos previamente configurados, utilizando a Evolution API. Além de mensagens de texto com resumo das execuções, são enviados documentos em anexo, contendo os relatórios consolidados.

D. Execução Cíclica do Pipeline

O pipeline de execução é dividido em duas rotinas principais:

- A rotina horária realiza a coleta dos status das tarefas, classifica os resultados e, em caso de falhas, realiza o download dos logs de erro.
- A rotina diária, agendada para as 08:00, realiza o envio consolidado dos relatórios e mensagens de status, além da limpeza automática de arquivos temporários.

Esse ciclo garante monitoramento contínuo com mínima intervenção humana, além de gerar histórico estruturado para auditoria posterior.

E. Implementação Técnica e Reprodutibilidade

A implementação foi desenvolvida em `Python` como linguagem base, utilizando um conjunto de bibliotecas maduras e bem documentadas como **Selenium** para automação de navegador — indispensável para simular login, navegação e extração de dados de interfaces como QMC/NPrinting, **Jinja2** para geração de templates HTML/PDF dinâmicos com conteúdo parametrizado e **schedule** (ou alternativa similar como `APScheduler`) para agendamento de tarefas periódicas.

A integração com o WhatsApp é realizada por meio da *Evolution API*, utilizada via cliente HTTP em Python. Este cliente suporta envio de mensagens de texto e documentos em PDF por meio de endpoints REST ou WebSocket, permitindo comunicação com múltiplas sessões autenticadas. A documentação oficial destaca a facilidade de implantação via Docker e configuração por variáveis de ambiente, facilitando a escalabilidade da aplicação.

Cada instância monitorada — seja QMC ou NPrinting — é parametrizada por arquivos de ambiente, o que confere flexibilidade e segurança ao adicionar ou remover máquinas sem necessidade de alterar código. As credenciais e políticas de autenticação são lidas nesses arquivos, assegurando isolamento entre ambientes.

O repositório conta com estrutura modular clara, onde todos os scripts estão organizados em diretórios funcionais, com instruções detalhadas em arquivo `README.md`. A estrutura e as dependências gerenciadas via `requirements.txt` garantem que o ambiente possa ser reproduzido em outras máquinas ou servidores, inclusive via Docker, conforme planos de extensão do projeto.

Repositório do Github - WebScrepStatusQlik: *[https://github.com/wagnerhelio/WebScrepStatusQlik/]*

IV. Resultados e Discussões

A validação do *WebScrepStatusQlik* foi realizada em ambiente de produção com múltiplas instâncias do Qlik Sense rodando em sistema operacional Windows 11. A seguir, apresentam-se os principais resultados técnicos, operacionais e as análises derivadas da aplicação prática da solução.

A. Implementação Técnica Validada

O sistema foi testado com sucesso nas seguintes instâncias do ecossistema Qlik:

- QMC Estatística (Tasks QAP)
- QMC Painéis (Tasks HUB)
- NPrinting (Relatórios)

Foram monitorados diversos tipos de tarefas, abrangendo execuções de ETL, atualizações periódicas de dados e geração de relatórios distribuídos. A implementação demonstrou compatibilidade com seletores CSS dinâmicos e robustez na autenticação automática.

B. Indicadores de Performance

Durante os testes, observou-se uma redução no tempo médio de resposta, com detecção de falhas inferior a uma hora, em contraste com janelas anteriores de 4 a 8 horas no processo manual. O tempo de envio de notificações via WhatsApp foi inferior a 5 segundos após a coleta e análise dos dados. A geração dos relatórios, anteriormente executada de forma manual e sujeita a erros, foi completamente automatizada, com produção de documentos PDF consistentes e visualmente padronizados.

C. Casos de Uso Observados

Três cenários principais foram validados:

1) **Monitoramento de tarefas ETL**: falhas foram detectadas automaticamente, logs de erro extraídos e alertas enviados aos responsáveis com detalhes contextualizados.
2) **Gestão de logs de erro**: o sistema baixou e enviou logs críticos de forma estruturada, promovendo maior agilidade na investigação de incidentes.
3) **Distribuição de relatórios**: PDFs de status foram enviados diariamente a grupos e indivíduos designados, assegurando comunicação eficaz e rastreável.

D. Impacto Operacional e Escalabilidade

Os ganhos operacionais observados foram:

- Redução do tempo entre falha e ação corretiva.
- Melhoria na visibilidade sobre execuções críticas.
- Adoção de práticas de monitoramento proativo.
- Diminuição da dependência de operações manuais.

Além disso, a implementação permitiu à equipe de dados concentrar esforços em análise estratégica, uma vez que o esforço anteriormente dedicado ao acompanhamento manual foi automatizado.

Comparado a soluções comerciais como Qlik Alerting, SenseOps e Splunk ITSI, o *WebScrepStatusQlik* apresenta vantagens como custo zero de licenciamento, integração nativa com WhatsApp e alto grau de personalização. Sua natureza modular e o uso de tecnologias amplamente disponíveis o tornam acessível para adoção em diferentes contextos organizacionais, mesmo com restrições de orçamento ou infraestrutura.

O sistema demonstrou capacidade de escalar para múltiplas instâncias simultaneamente e foi estruturado de forma a permitir integração futura com outras plataformas de BI, como Power BI e Tableau, por meio de adaptadores específicos.

Essa flexibilidade também se estende à possibilidade de uso de novos canais de notificação, como e-mail, Slack ou Microsoft Teams, mediante pequenas modificações na camada de notificações.

V. Reflexões Éticas e Limitações

A adoção de soluções de automação em ambientes corporativos exige a consideração cuidadosa de aspectos éticos, legais e operacionais. O *WebScrepStatusQlik* foi projetado respeitando os princípios da Lei Geral de Proteção de Dados (LGPD) e boas práticas de engenharia de software, sem negligenciar suas limitações técnicas e metodológicas.

A. Privacidade e Segurança da Informação

O sistema não coleta, armazena ou transmite dados pessoais ou sensíveis. Todas as informações manipuladas referem-se a metadados de execução de tarefas institucionais (nome da tarefa, status, horários de execução e logs de erro). A extração é realizada exclusivamente em servidores internos, com acesso controlado e autenticação configurada por meio de variáveis de ambiente.

Além disso, todas as credenciais são mantidas fora do código-fonte e protegidas por boas práticas de segurança, limitando o risco de exposição indevida de dados corporativos.

B. Transparência e Auditoria

A lógica de classificação dos dados é completamente acessível, com código aberto e documentado, permitindo auditoria por pares e validação externa. O sistema também registra logs das operações realizadas, garantindo rastreabilidade das ações automatizadas e possibilitando a identificação de falhas ou abusos.

A automação proposta não visa substituir a análise humana, mas sim eliminar tarefas repetitivas e propensas a erro, como a verificação manual de status de execução. Com isso, libera-se tempo das equipes de BI para atividades analíticas de maior valor agregado. O sistema atua como suporte à decisão, mantendo o ser humano no centro do processo crítico.

C. Limitações Técnicas

O projeto apresenta algumas limitações inerentes às tecnologias utilizadas:

- **Dependência do DOM**: alterações na estrutura HTML/CSS das interfaces Qlik podem afetar a eficácia dos seletores de scraping, exigindo manutenção contínua.
- **Autenticação contínua**: o funcionamento do sistema depende de sessões autenticadas no QMC e NPrinting, o que pode demandar revalidação manual em alguns cenários.
- **API de comunicação**: a estabilidade da Evolution API é crucial para o envio de mensagens. Interrupções no serviço ou mudanças na API podem impactar o funcionamento do sistema.
- **Conectividade**: o scraping depende de conectividade estável com os servidores Qlik. Problemas de rede podem interromper a coleta.

D. Limitações Metodológicas

A validação do sistema foi realizada em ambientes específicos, não abrangendo todas as possíveis configurações organizacionais. Embora o sistema tenha se mostrado robusto, ainda não foram realizados testes de estresse com alto volume de tarefas simultâneas, tampouco avaliações formais de desempenho em ambientes com alta concorrência.

Além disso, a solução foi desenvolvida com foco no ecossistema Qlik Sense, o que limita sua portabilidade imediata para outras plataformas de BI, exigindo adaptações estruturais para novos contextos.

VI. Conclusão e Trabalhos Futuros

O desenvolvimento do *WebScrepStatusQlik* demonstrou a viabilidade técnica e operacional de uma solução automatizada e modular para monitoramento de tarefas críticas no ecossistema Qlik Sense. A combinação de técnicas de *web scraping*, geração de relatórios estruturados e integração com canais de comunicação modernos como o WhatsApp resultou em uma ferramenta eficaz para a detecção precoce de falhas e notificação proativa de stakeholders.

Entre os principais benefícios identificados destacam-se: a redução significativa no tempo médio de resposta a falhas, a melhoria da rastreabilidade dos processos de ETL e a eliminação de atividades manuais repetitivas associadas à verificação de tarefas em múltiplas instâncias QMC e NPrinting.

A arquitetura modular adotada — baseada em componentes especializados para coleta, análise, geração de relatórios e envio de notificações — permite a fácil adaptação da solução a diferentes ambientes organizacionais, mantendo um alto grau de reprodutibilidade e extensibilidade.

Do ponto de vista técnico e metodológico, o sistema contribui com:

- Uma arquitetura distribuída;
- Uma abordagem híbrida de monitoramento que combina scraping dinâmico com APIs REST;
- Um pipeline de processamento inteligente com lógica de classificação automatizada;
- Um sistema de notificações em tempo real utilizando um canal de alta penetração (WhatsApp).

Tais contribuições se aplicam tanto ao campo da engenharia de dados quanto à prática profissional em ambientes empresariais que dependem de alta disponibilidade e governança de sistemas BI.

A. Trabalhos Futuros

Como desdobramentos futuros, propõe-se:

- **Containerização com Docker**: para facilitar a implantação em diversos ambientes e nuvem, permitindo automação contínua e isolamento de dependências;

- **Integração com plataformas de monitoramento**: como Zabbix, Grafana ou Prometheus, para agregar métricas em tempo real e visualização integrada;
- **Desenvolvimento de dashboard web**: com interface gráfica interativa para acompanhamento dos relatórios, configuração de parâmetros e controle das execuções;
- **Fallback multicanal**: em caso de falha no envio via WhatsApp, prevê-se a implementação de rotas alternativas como e-mail ou dashboards internos;
- **Predição de falhas com Machine Learning**: utilizando padrões históricos de falhas e logs para antecipar comportamentos críticos e melhorar o SLA;
- **Suporte a múltiplas plataformas BI**: como Power BI, Tableau e Looker, por meio de extensões ou adaptação de extratores;
- **Otimizações de desempenho**: como paralelização de tarefas, cache inteligente e distribuição por microserviços;
- **Reforço de segurança**: incluindo autenticação OAuth2, criptografia de logs e registro de auditoria baseado em blockchain.

B. Considerações Finais

O *WebScrepStatusQlik* representa uma solução concreta e replicável para os desafios de monitoramento automatizado em ambientes de Business Intelligence. Sua arquitetura aberta, documentação acessível e uso de tecnologias consolidadas tornam-no aplicável em diversos contextos, ao mesmo tempo em que contribui com novos paradigmas metodológicos para a engenharia de dados.

Em um cenário de transformação digital acelerada, soluções como esta reafirmam o papel da automação inteligente e da integração tecnológica como catalisadores da eficiência operacional, sem perder de vista os princípios éticos e legais que regem a gestão de dados.

REFERENCES

[1] H. Foidl, V. Golendukhina, R. Ramler, and M. Felderer, "Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers," *Journal of Systems and Software*, vol. 207, p. 111855, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223002509

[2] Senseops Inc, "Streamlining code management for qlik sense applications," https://www.senseops.com/, 2025.

[3] Qlik Community / Official Support, "The qlik sense monitoring applications for cloud and on premise," community.qlik.com, 2025.

[4] Qlik Blog / Qlik DataOps Guide, "What is dataops? how does it work?" https://www.qlik.com/us/dataops, 2025.

[5] P. Patterson, "Dataops: Modernizing bi with devops for data analytics," *DataOps Guide / DBTA Big Data Quarterly*, 2025. [Online]. Available: https://www.dbta.com/BigDataQuarterly/Articles/DataOps-Modernizing-BI-With-DevOps-for-Data-Analytics-131806.aspx

[6] S. Shankar and A. Parameswaran, "Towards observability for production machine learning pipelines," 2022. [Online]. Available: https://arxiv.org/abs/2108.13557

[7] F. Bayram, B. S. Ahmed, E. Hallin, and A. Engman, "Dqsops: Data quality scoring operations framework for data-driven applications," 2023. [Online]. Available: https://arxiv.org/abs/2303.15068

[8] D. Tu, Y. He, W. Cui, S. Ge, H. Zhang, H. Shi, D. Zhang, and S. Chaudhuri, "Auto-validate by-history: Auto-program data quality constraints to validate recurring data pipelines," 2023. [Online]. Available: https://arxiv.org/abs/2306.02421

[9] S. Narayanan, M. S, and P. Zephan, "Real-time monitoring of data pipelines: Exploring and experimentally proving that the continuous monitoring in data pipelines reduces cost and elevates quality," *EAI Endorsed Transactions on Scalable Information Systems*, 02 2024.

[10] K. Kanagarla, "Data observability: Ensuring trust in data pipelines," *International Journal of Social Impact*, vol. 9, 12 2024.

[11] S. Niedermaier, F. Koetter, A. Freymann, and S. Wagner, *On Observability and Monitoring of Distributed Systems – An Industry Interview Study*. Springer International Publishing, 2019, p. 36–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-33702-5_3

[12] R. Mitchell, *Web Scraping with Python: Collecting Data from the Modern Web*, 1st ed. O'Reilly Media, Inc., 2015.

[13] R. Koçi, X. Franch, P. Jovanovic, and A. Abelló, "Improving web api usage logging," 2021. [Online]. Available: https://arxiv.org/abs/2103.10811

[14] A. Pasinkova and O. Vikentyeva, "Application of design patterns in the development of the architecture of monitoring systems," *Proceedings of the Institute for System Programming of the RAS*, vol. 35, pp. 137–150, 01 2023.

[15] I. G. Dachary, "Design of a monitoring system for distributed data processing pipelines," *Master's thesis, University of Groningen*, 2023. [Online]. Available: https://fse.studenttheses.ub.rug.nl/id/eprint/31448

[16] O. Ogunwole, N. Sam-Bulya, M. Joel, and G. Achumie, "Optimizing automated pipelines for real-time data processing in digital media and e-commerce," *International Journal of Multidisciplinary Research and Growth Evaluation.*, vol. 3, pp. 112–120, 01 2022.

[17] A. Bonomi, "Data pipeline management and its importance in modern analytics," *Journal of Data Engineering*, vol. 7, no. 3, pp. 44–52, 2015.