EDGE MONITOR API

BACKEND PARA INGESTÃO, PERSISTÊNCIA E DISPONIBILIZAÇÃO DE EVENTOS DE MONITORAMENTO INTEGRADO A SISTEMAS DE COMPUTAÇÃO DE BORDA

Relatório Técnico

Londrina, Janeiro de 2026

1. Introdução

Sistemas modernos de monitoramento de risco em ambientes industriais exigem arquiteturas capazes de centralizar dados operacionais, garantir rastreabilidade das ocorrências, armazenar evidências associadas e disponibilizar informações consolidadas para análise, ao mesmo tempo em que se integram de forma eficiente a soluções distribuídas baseadas em computação de borda.

Nesse contexto, este relatório apresenta o desenvolvimento do Edge Monitor API, um backend projetado para atuar como camada central de ingestão, persistência, auditoria e disponibilização de eventos de monitoramento, provenientes de sistemas externos de detecção de risco executados na borda.

O sistema foi desenvolvido utilizando o framework Django, em conjunto com o Django REST Framework, adotando uma arquitetura modular, desacoplada e aderente a padrões REST. A segurança de acesso é garantida por meio de autenticação baseada em JSON Web Tokens (JWT), assegurando controle de acesso aos recursos da API. A documentação dos endpoints é gerada automaticamente por meio do padrão OpenAPI, facilitando integração, validação e manutenção do sistema.

O Edge Monitor API integra-se de forma direta a sistemas de inferência executados em computação de borda, como o projeto Edge Risk Monitor, responsável pela detecção de eventos em tempo real utilizando técnicas de visão computacional. Nesse arranjo arquitetural, o backend não executa inferência nem processamento intensivo, concentrando-se exclusivamente nas responsabilidades de persistência de eventos, armazenamento de evidências, auditoria de operações, controle de usuários e disponibilização de dados para consumo por aplicações clientes, como dashboards web e ferramentas analíticas.

Essa separação clara de responsabilidades permite que o sistema evolua de forma independente, suportando múltiplas fontes de eventos e diferentes consumidores de dados, mantendo a robustez, a rastreabilidade e a previsibilidade operacional do ambiente de monitoramento.

2. Objetivos

2.1 Objetivo Geral

Desenvolver um backend estruturado para ingestão, persistência, auditoria e disponibilização de eventos de monitoramento, permitindo centralizar dados enviados por sistemas de computação de borda e disponibilizá-los de forma segura, organizada e rastreável para consumo por aplicações clientes e análise posterior.

2.2 Objetivos Específicos

- Implementar uma API REST para recebimento de eventos de monitoramento enviados por dispositivos executando inferência em computação de borda.
- Persistir eventos de detecção e evidências visuais associadas em banco de dados relacional, garantindo integridade e rastreabilidade das informações.

- Implementar autenticação e controle de acesso baseados em JSON Web Tokens (JWT) para proteção dos endpoints da API.
- Garantir rastreabilidade das operações por meio de registros estruturados de auditoria, possibilitando diagnóstico e análise das ações realizadas no sistema.
- Disponibilizar endpoints de consulta para acesso e visualização analítica dos eventos registrados, permitindo consumo por dashboards web e outras aplicações clientes.
- Organizar o backend de forma modular e desacoplada, favorecendo manutenção, testes e evolução futura do sistema.
- Viabilizar integração direta e desacoplada com sistemas de inferência executados em computação de borda, mantendo clara separação de responsabilidades entre edge e backend.

3. Visão Geral do Sistema

O Edge Monitor API foi concebido como a camada central de backend de um sistema de monitoramento de risco baseado em computação de borda. O sistema opera exclusivamente após a detecção e validação dos eventos no ambiente edge, não realizando inferência, classificação ou qualquer processamento relacionado à visão computacional.

A aplicação é responsável por receber eventos estruturados e evidências visuais enviados por sistemas externos de detecção, persistir essas informações de forma auditável e disponibilizá-las para consulta analítica por aplicações clientes. Todas as funcionalidades do backend são executadas de forma explícita e independente, seguindo fluxos bem definidos e previsíveis.

O sistema está organizado em quatro responsabilidades principais:

- **Autenticação e controle de acesso:** Gerenciamento de usuários e proteção dos endpoints por meio de autenticação baseada em JSON Web Tokens (JWT), garantindo acesso controlado aos recursos da API.
- **Ingestão de eventos de monitoramento:** Recebimento, validação e registro de eventos enviados por dispositivos edge, incluindo metadados estruturados e evidências visuais associadas.
- **Persistência e auditoria:** Armazenamento dos eventos, arquivos de evidência e registros de log, assegurando rastreabilidade, integridade e possibilidade de auditoria das operações realizadas.
- **Consulta e visualização analítica:** Disponibilização de dados consolidados por meio de endpoints de leitura, permitindo consumo por dashboards web, aplicações cliente e ferramentas analíticas externas.

Essa separação clara de responsabilidades garante baixo acoplamento, clareza operacional, segurança e facilidade de manutenção, além de permitir a evolução independente do backend em relação aos sistemas de inferência executados na borda.

4. Arquitetura do Sistema

A arquitetura do Edge Monitor API reflete diretamente a organização real do repositório, sendo estruturada de forma modular, desacoplada e orientada a responsabilidades bem definidas, com o objetivo de garantir clareza estrutural, segurança, rastreabilidade das operações e facilidade de manutenção.

O sistema foi projetado para atuar como backend central de ingestão, persistência e disponibilização analítica de eventos, não executando qualquer lógica de inferência, classificação ou processamento computacional pesado. Toda a responsabilidade de detecção e validação dos eventos ocorre externamente, em sistemas de computação de borda (edge), cabendo à API exclusivamente o papel de suporte backend, persistência e governança dos dados.

A seguir, são descritos os principais módulos que compõem a arquitetura do sistema, bem como as interfaces REST efetivamente expostas por cada um deles.

auth/

Responsável pelos mecanismos de autenticação e gerenciamento de tokens JWT.

Este módulo concentra exclusivamente a lógica relacionada à segurança de acesso à API, incluindo:

- Serialização dos dados de autenticação
- Emissão de tokens de acesso (access token)
- Renovação de tokens por meio de refresh token
- Invalidação de tokens no processo de logout
- Recuperação e redefinição segura de senha por e-mail

As funcionalidades deste módulo são expostas por meio dos seguintes endpoints REST:

- **POST /api/authentication/login/**
  Autenticação do usuário e emissão de tokens JWT (access e refresh).

- **POST /api/authentication/renovate/**
  Renovação do access token a partir de um refresh token válido.

- **POST /api/authentication/logout/**
  Invalidação do refresh token, encerrando a sessão do usuário.

- **POST /api/authentication/password-reset/**
  Inicia o processo de recuperação de senha por e-mail, utilizando token temporário seguro.

- **POST /api/authentication/password-reset-confirm/**
  Confirma a redefinição de senha a partir de UID e token válidos.

Toda a lógica de autenticação é isolada neste módulo, garantindo separação clara de responsabilidades e facilitando a manutenção e evolução dos mecanismos de segurança.

users/

Responsável pela gestão de usuários do sistema.

Este módulo contempla operações administrativas e de gerenciamento, incluindo:

- Criação de usuários.
- Listagem e consulta individual de usuários.
- Atualização de dados cadastrais.
- Exclusão de usuários, respeitando regras de integridade.

As operações de gerenciamento de usuários são disponibilizadas por meio dos seguintes endpoints REST:

- **POST /api/user/**
  Criação de usuários (restrito a usuários administrativos).

- **GET /api/user/**
  Listagem de usuários.

- **GET /api/user/{id}/**
  Consulta individual de usuário.

- **PUT /api/user/{id}/**
  Atualização de dados cadastrais.

- **DELETE /api/user/{id}/**
  Exclusão de usuário, respeitando regras de permissão.

Todas as rotas deste módulo são protegidas por autenticação JWT e aplicam controle de acesso baseado no perfil do usuário autenticado, garantindo segurança e governança sobre as operações administrativas.

monitoring/

Módulo responsável pela ingestão e persistência dos eventos de monitoramento enviados pelos dispositivos edge.

Este módulo define:

- O modelo de dados dos eventos de monitoramento.
- A validação estrutural dos dados recebidos.
- O armazenamento das evidências visuais associadas aos eventos.
- A proteção dos endpoints por autenticação JWT.
- A disponibilização dos eventos persistidos para consumo por aplicações cliente.

Ingestão de eventos

O ponto central de integração com os sistemas de inferência em computação de borda é realizado por meio do seguinte endpoint:

- **POST /api/monitoring/**

Este endpoint recebe eventos estruturados contendo:

- Identificador do dispositivo (MAC address).
- Classe detectada.
- Data e hora da ocorrência.
- Evidência visual associada (imagem).

Os dados são validados, persistidos em banco de dados relacional e registrados de forma auditável, garantindo rastreabilidade completa das ocorrências detectadas no ambiente edge.

Consulta básica de eventos

Além da ingestão, o módulo também disponibiliza os eventos persistidos para consumo por aplicações cliente, como o frontend do sistema, por meio do endpoint:

- **GET /api/monitoring/**

Este endpoint permite a listagem dos eventos de monitoramento registrados, retornando dados estruturados que incluem:

- Identificador do dispositivo.
- Classe detectada.
- Data e hora da ocorrência.
- Referência à evidência visual armazenada.

A consulta é protegida por autenticação JWT e tem como objetivo fornecer uma visão operacional básica dos eventos, sendo utilizada principalmente pelo frontend para listagem inicial e validação do fluxo de integração.

Consultas analíticas mais avançadas são tratadas de forma separada pelo módulo dashboard/, mantendo a separação clara entre ingestão, consulta operacional e análise.

dashboard/

Responsável pela disponibilização dos dados de monitoramento para análise e visualização.

Este módulo implementa:

- Filtros temporais baseados em intervalo de datas.
- Serialização específica para consumo por dashboards.
- Consulta eficiente dos eventos persistidos.

A interface REST de consulta analítica é exposta por meio do endpoint:

- **GET /api/dashboard?start_date=YYYY-MM-DD&end_date=YYYY-MM-DD**

A lógica de consulta é mantida separada da ingestão, garantindo clareza, organização e eficiência na recuperação dos dados.

core/

Módulo responsável pelos componentes transversais do sistema.

Este módulo concentra funcionalidades compartilhadas por toda a aplicação, incluindo:

- Modelo de auditoria (LogSystem) para registro de operações.
- Função utilitária de logging (report_log) utilizada de forma padronizada em todas as views.

Essa centralização garante rastreabilidade completa das operações executadas no sistema e padronização dos registros de auditoria.

project/

Responsável pela configuração global do sistema.

Este diretório contém:

- **settings.py:** configurações centrais da aplicação, autenticação, banco de dados, e-mail e mídia
- **urls.py:** roteamento principal da API e definição das rotas públicas.
- **wsgi.py:** interface de execução do servidor.

As configurações são mantidas centralizadas e desacopladas da lógica de negócio, facilitando manutenção e ajustes de ambiente.

media/

Diretório destinado ao armazenamento das evidências visuais enviadas pelos dispositivos edge.

As imagens persistidas neste diretório garantem auditoria visual, rastreabilidade das ocorrências e suporte à análise posterior por meio de dashboards e interfaces administrativas.

Essa arquitetura garante baixo acoplamento, alta coesão, segurança, auditabilidade e facilidade de evolução, estando alinhada às boas práticas de desenvolvimento de APIs REST para sistemas distribuídos e integrados a soluções de computação de borda.

5. Metodologia de Desenvolvimento

A metodologia adotada no Edge Monitor API foi definida com foco em confiabilidade, rastreabilidade, segurança e integração com sistemas externos, garantindo que todos os eventos recebidos dos dispositivos edge sejam corretamente persistidos, auditáveis e disponibilizados para consulta analítica de forma padronizada.

Todas as etapas descritas a seguir foram implementadas diretamente no código do projeto, respeitando princípios de separação de responsabilidades, baixo acoplamento e aderência

ao padrão REST. Não há procedimentos manuais, fluxos implícitos ou comportamentos não documentados fora do escopo desta metodologia.

5.1 Arquitetura da API

O Edge Monitor API foi desenvolvido seguindo uma arquitetura RESTful, utilizando Django e Django REST Framework como base. A organização do sistema é orientada a recursos bem definidos, com responsabilidades claras para cada aplicação interna.

A API atua exclusivamente como camada backend, não realizando qualquer tipo de inferência, processamento de visão computacional ou decisão de risco. Todo o processamento intensivo ocorre nos sistemas de computação de borda, cabendo ao backend apenas a ingestão, persistência, auditoria e consulta estruturada dos dados recebidos.

Essa abordagem garante clareza arquitetural, previsibilidade operacional e facilita a integração com sistemas externos heterogêneos, preservando a simplicidade e a escalabilidade do backend.

5.2 Autenticação, Controle de Acesso e Recuperação de Credenciais

A autenticação do sistema é baseada em JSON Web Tokens (JWT), utilizando a biblioteca SimpleJWT, garantindo identidade segura e controle consistente de acesso aos recursos da API.

O fluxo de autenticação contempla:

- Autenticação de usuários por meio de credenciais válidas.
- Emissão de access token com tempo de vida reduzido.
- Renovação controlada de tokens por meio de refresh token.
- Invalidação explícita de tokens durante o processo de logout.

Após a autenticação, o acesso aos recursos é controlado por permissões explícitas, aplicadas diretamente nas views da API. Todos os endpoints sensíveis exigem autenticação válida, garantindo que apenas usuários autorizados possam criar, consultar, atualizar ou excluir recursos do sistema.

O sistema diferencia usuários por nível de acesso, aplicando regras claras de governança:

- Usuários autenticados podem consultar eventos de monitoramento e acessar dados analíticos por meio do dashboard;
- Usuários administrativos possuem permissões adicionais para criação, atualização e exclusão de usuários;
- A exclusão de usuários administrativos é explicitamente bloqueada, preservando a integridade da governança do sistema.

Adicionalmente, o Edge Monitor API implementa um fluxo seguro de recuperação e redefinição de senha, permitindo que usuários recuperem o acesso sem intervenção manual. Esse fluxo inclui:

- Solicitação de redefinição de senha via e-mail;
- Envio de token temporário de redefinição;
- Validação do token antes da alteração;
- Atualização segura da credencial no sistema.

Esse mecanismo assegura continuidade de acesso, aderência às boas práticas de segurança e separação clara entre autenticação e recuperação de credenciais.

5.3 Ingestão de Eventos de Monitoramento

A ingestão de eventos ocorre por meio de um endpoint dedicado, responsável por receber eventos enviados pelos dispositivos edge após confirmação local da detecção. Cada evento representa uma ocorrência já validada no ambiente de borda e contém informações estruturadas, tais como:

- Identificador lógico do dispositivo (MAC address).
- Classe do objeto detectado.
- Data e hora da detecção.
- Evidência visual associada ao evento.

O MAC address é utilizado como identificador lógico do dispositivo emissor, permitindo rastreabilidade, correlação de eventos e análise por dispositivo no dashboard.

Os dados são recebidos por meio de requisições multipart/form-data e passam por validação estrutural rigorosa antes da persistência, assegurando integridade, padronização e consistência das informações armazenadas no sistema.

A responsabilidade pela validação semântica da detecção permanece integralmente no ambiente edge, mantendo o backend focado exclusivamente em ingestão confiável e persistência dos eventos.

5.4 Persistência de Evidências

As evidências visuais associadas aos eventos de monitoramento são armazenadas utilizando o mecanismo nativo de gerenciamento de arquivos do Django. As imagens são persistidas em diretórios organizados no sistema de arquivos, mantendo vínculo direto e explícito com o respectivo evento registrado no banco de dados.

Esse mecanismo garante rastreabilidade completa das ocorrências, possibilitando auditoria posterior, verificação visual dos eventos e suporte à análise operacional por meio de interfaces administrativas ou dashboards web.

5.5 Auditoria e Registro de Operações

Todas as ações relevantes executadas no sistema são registradas por meio de um mecanismo centralizado de auditoria. O modelo de logs armazena informações como:

- Usuário responsável pela ação (quando aplicável).
- Tipo de operação executada.
- Status da operação.
- Mensagem descritiva do evento.
- Data e hora da ocorrência.

São auditadas, entre outras, operações de autenticação, logout, gestão de usuários, redefinição de senha e ingestão de eventos. Esse processo garante transparência operacional, suporte à análise de falhas e histórico completo das interações realizadas na API, independentemente da origem da requisição.

5.6 Consulta Analítica e Dashboard

O sistema disponibiliza endpoints específicos para consulta dos eventos registrados, permitindo o consumo estruturado dos dados por aplicações cliente, como o dashboard web do Edge Monitor.

As consultas possibilitam a aplicação de filtros temporais e por dispositivo, retornando apenas os dados relevantes para análise operacional e auditoria visual.

Essa abordagem garante eficiência na recuperação das informações, clareza nos dados retornados e desacoplamento entre a camada de backend e as soluções de visualização, permitindo evolução independente de ambas.

Consultas analíticas mais avançadas e agregações permanecem separadas da ingestão, preservando a organização e a coesão arquitetural do sistema.

5.7 Integração com Sistemas Edge

A metodologia de desenvolvimento do Edge Monitor API foi concebida para integração direta com sistemas de computação de borda. O backend assume que os eventos recebidos já foram previamente validados e confirmados no ambiente edge, evitando processamento redundante e reduzindo a carga computacional do servidor central.

Essa separação de responsabilidades assegura simplicidade arquitetural, escalabilidade do sistema e facilidade de evolução futura, mantendo o backend focado exclusivamente em persistência, auditoria e disponibilização estruturada dos dados.

6. Registro de Logs e Auditoria

O Edge Monitor API utiliza um mecanismo centralizado de registro de logs e auditoria, projetado para garantir rastreabilidade completa das operações realizadas no sistema, bem como suporte à análise de falhas, auditoria técnica e monitoramento operacional.

O processo de registro contempla eventos relevantes associados à execução da API, incluindo:

- Autenticação de usuários (login, renovação de token e logout).
- Criação, consulta, atualização e exclusão de usuários.
- Ingestão de eventos de monitoramento enviados pelos dispositivos edge.
- Consultas realizadas pelos endpoints de visualização e análise.
- Ocorrência de erros, exceções e falhas operacionais.

Os registros são persistidos em banco de dados por meio de um modelo dedicado de auditoria, assegurando que todas as ações executadas no sistema possam ser rastreadas posteriormente, independentemente da origem da requisição, do tipo de usuário ou do módulo responsável pela operação.

Cada entrada de log armazena informações estruturadas, tais como:

- Usuário responsável pela ação, quando aplicável.
- Ação executada no sistema.
- Status da operação (sucesso, aviso ou erro).
- Mensagem descritiva contextualizando o evento.
- Data e hora da ocorrência.

Essa abordagem garante rastreabilidade temporal precisa e fornece uma visão clara e consistente do comportamento da API ao longo de sua operação, facilitando diagnóstico de falhas, análise de incidentes e auditorias técnicas.

O mecanismo de registro é desacoplado da lógica principal das views, sendo acionado de forma padronizada por meio de um utilitário central de logging. Essa estratégia assegura consistência na geração dos registros, reduz o acoplamento entre os componentes do sistema e facilita a manutenção e evolução futura do código.

Esse modelo de registro e auditoria fornece suporte tanto para análise técnica quanto para auditorias administrativas, reforçando a confiabilidade, a transparência operacional e a robustez do backend desenvolvido.

7. Resultados Obtidos

7.1 Ingestão de Eventos de Monitoramento

Durante os testes realizados, o Edge Monitor API demonstrou capacidade consistente e estável de receber, validar estruturalmente e processar eventos de monitoramento enviados por dispositivos de computação de borda.

Os dados estruturados recebidos — incluindo identificador lógico do dispositivo (MAC address), classe detectada e data e hora do evento — foram devidamente validados antes da persistência no banco de dados, assegurando integridade e padronização das informações armazenadas.

As evidências visuais associadas aos eventos foram recebidas corretamente por meio de requisições multipart/form-data e armazenadas conforme o fluxo definido, mantendo coerência entre os registros persistidos no banco de dados e os arquivos armazenados no sistema de arquivos. Esse comportamento confirmou a confiabilidade do mecanismo de ingestão e armazenamento adotado.

7.2 Autenticação, Controle de Acesso e Recuperação de Credenciais

O mecanismo de autenticação baseado em JSON Web Tokens (JWT) apresentou funcionamento estável e previsível ao longo dos testes realizados. Os fluxos de login, renovação de token e logout respeitaram corretamente o ciclo de vida dos tokens, garantindo que apenas usuários autenticados tivessem acesso aos endpoints protegidos da API.

Os controles de permissão foram aplicados corretamente, assegurando que operações administrativas e operações de consulta fossem executadas apenas por usuários com nível de acesso compatível.

Adicionalmente, o fluxo de recuperação e redefinição de senha por e-mail funcionou conforme esperado durante os testes realizados. O sistema permitiu que usuários solicitassem a recuperação de acesso, recebendo por e-mail um token temporário de redefinição, validado pela API antes da alteração segura da credencial. Esse resultado confirmou a robustez do mecanismo de segurança e a aderência às boas práticas de autenticação.

7.3 Gestão de Usuários

As funcionalidades de gestão de usuários permitiram a execução controlada das operações de criação, consulta, atualização e exclusão, respeitando as regras de permissão definidas no sistema.

O backend aplicou validações adequadas para evitar inconsistências, como duplicidade de dados, operações não autorizadas e violação de integridade, incluindo a proteção contra exclusão de usuários administrativos.

As respostas retornadas pelas operações seguiram os padrões REST definidos, com códigos HTTP e mensagens coerentes, favorecendo previsibilidade no comportamento da API e facilitando a integração com aplicações cliente e sistemas externos.

7.4 Consulta Analítica via Dashboard

O módulo de dashboard possibilitou a recuperação eficiente dos eventos de monitoramento registrados, por meio de filtros baseados em intervalo de datas. Os dados retornados foram estruturados de acordo com o contrato da API, permitindo consumo direto por aplicações frontend, dashboards analíticos ou ferramentas externas de visualização.

Essa funcionalidade evidenciou o papel da API como fonte centralizada de dados para análise operacional e acompanhamento histórico dos eventos de monitoramento.

7.5 Auditoria e Rastreabilidade

O mecanismo de auditoria centralizada registrou de forma consistente todas as operações relevantes executadas na API, incluindo:

- Autenticações e encerramentos de sessão;
- Operações de gestão de usuários;
- Solicitações de recuperação e redefinição de senha;
- Ingestão de eventos de monitoramento;
- Consultas realizadas pelos endpoints analíticos;
- Ocorrência de erros e exceções.

Os registros gerados permitiram acompanhar de maneira clara, cronológica e estruturada o fluxo das operações, assegurando rastreabilidade completa, suporte à análise histórica, depuração técnica e auditoria administrativa.

7.6 Considerações Gerais sobre os Resultados

De forma geral, os resultados obtidos demonstram que o Edge Monitor API cumpre adequadamente seu papel como backend centralizador de eventos de monitoramento, controle de acesso, auditoria e consulta analítica.

O sistema apresentou comportamento estável, previsível e coerente com os objetivos definidos, confirmando sua adequação como componente backend em uma arquitetura de monitoramento baseada em computação de borda, com separação clara de responsabilidades entre edge e servidor central.

8. Limitações

As limitações identificadas no Edge Monitor API decorrem de decisões arquiteturais e de escopo adotadas durante o desenvolvimento do sistema. Tais limitações não representam falhas de implementação, mas sim restrições assumidas de forma consciente para manter clareza de responsabilidades, simplicidade arquitetural e alinhamento com os objetivos técnicos do projeto.

8.1 Escopo Funcional Delimitado

O Edge Monitor API foi projetado exclusivamente para atuar como backend de ingestão, persistência, auditoria e consulta de eventos de monitoramento. Não foram contemplados, nesta etapa, requisitos típicos de ambientes de alta disponibilidade, tais como balanceamento de carga, replicação automática de serviços ou mecanismos avançados de tolerância a falhas distribuídas.

Essa delimitação permitiu concentrar o desenvolvimento na robustez funcional da API, na organização arquitetural e na previsibilidade operacional, evitando a introdução de complexidade desnecessária no contexto avaliado.

8.2 Modelo Simplificado de Controle de Acesso

O controle de acesso foi implementado com base na autenticação de usuários por meio de JSON Web Tokens (JWT), aliado a um modelo básico de diferenciação de privilégios, distinguindo usuários comuns de usuários administrativos.

Esse modelo atende adequadamente ao escopo do sistema, garantindo controle de acesso consistente, proteção de operações administrativas e governança mínima do ambiente. Entretanto, não foram implementados mecanismos avançados de autorização, como controle baseado em papéis (RBAC) ou políticas específicas por recurso.

A adoção de modelos de autorização mais granulares pode ser considerada como evolução futura, caso surjam requisitos mais complexos de segregação de responsabilidades ou múltiplos perfis operacionais.

8.3 Estratégia de Armazenamento de Evidências

As evidências visuais associadas aos eventos de monitoramento são armazenadas localmente no sistema de arquivos do servidor, utilizando o mecanismo nativo de gerenciamento de mídia do Django.

Embora adequada ao volume de dados esperado e ao contexto atual do sistema, essa estratégia não contempla soluções de armazenamento distribuído ou serviços externos de object storage. Em cenários de maior escala, essas abordagens podem ser incorporadas de forma incremental, sem impacto estrutural na API.

8.4 Processamento Síncrono dos Eventos

O processamento dos eventos de monitoramento é realizado de forma síncrona, garantindo simplicidade, previsibilidade e controle direto do fluxo de ingestão.

Essa abordagem mostrou-se adequada para o volume de eventos esperado e para os objetivos do sistema. Contudo, em cenários de maior carga ou necessidade de escalabilidade horizontal, mecanismos assíncronos, filas de mensagens ou processamento em segundo plano podem ser incorporados como extensão natural da arquitetura.

8.5 Validação Orientada à Estrutura dos Dados

As validações realizadas pela API concentram-se na integridade estrutural e no formato dos dados recebidos, assegurando consistência, padronização e confiabilidade das informações persistidas.

Validações semânticas mais complexas, correlações entre eventos ou análises de contexto operacional permanecem fora do escopo atual, podendo ser incorporadas futuramente conforme necessidades analíticas adicionais.

8.6 Considerações de Segurança

Aspectos avançados de segurança, como políticas específicas de hardening, rotação automatizada de chaves, criptografia adicional de dados em repouso ou mecanismos avançados de detecção de intrusão, não foram aprofundados nesta etapa.

A segurança foi tratada de forma compatível com o papel do sistema e com os mecanismos nativos fornecidos pelo Django, aliados ao uso de autenticação baseada em JWT e recuperação de senha por e-mail, mantendo equilíbrio entre robustez, clareza arquitetural e simplicidade operacional.

9. Conclusão

O desenvolvimento do Edge Monitor API evidenciou a viabilidade de uma arquitetura backend dedicada à ingestão, persistência, auditoria e disponibilização analítica de eventos de monitoramento provenientes de dispositivos de computação de borda. A solução implementada atendeu plenamente aos objetivos definidos, validando tanto os aspectos funcionais quanto a organização arquitetural do sistema.

A separação clara de responsabilidades entre os módulos de autenticação, gestão de usuários, recuperação de credenciais, ingestão de eventos, auditoria e dashboard contribuiu para a construção de um backend coeso, modular e de fácil manutenção. A adoção de padrões REST, autenticação baseada em JSON Web Tokens (JWT), controle básico de permissões administrativas e documentação automática por meio de OpenAPI proporcionou previsibilidade operacional, segurança compatível com o contexto do sistema e facilidade de integração com aplicações externas.

A arquitetura adotada mostrou-se apropriada para cenários em que o processamento computacional intensivo ocorre na borda, enquanto o backend atua como elemento centralizador de dados, histórico e visualização. Essa abordagem reduz latência, minimiza a carga computacional no servidor central e favorece a escalabilidade do sistema, mantendo uma separação clara entre processamento de inferência e gestão de dados.

Os mecanismos de auditoria centralizada implementados possibilitaram o registro consistente das ações realizadas no sistema, assegurando rastreabilidade, transparência operacional e suporte à análise de falhas. Adicionalmente, o fluxo de recuperação e redefinição de senha por e-mail demonstrou-se funcional e seguro, reforçando a robustez dos mecanismos de controle de acesso e continuidade operacional.

A disponibilização de dados consolidados por meio de consultas filtradas, aliada à integração validada com uma aplicação frontend dedicada, evidenciou a capacidade do backend em atuar como fonte única e confiável de dados para visualização operacional, auditoria e gestão administrativa, reforçando a aderência da API a cenários reais de consumo por aplicações cliente.

De forma geral, o Edge Monitor API estabelece uma base técnica sólida para aplicações de monitoramento integradas a soluções de computação de borda, apresentando uma arquitetura organizada, extensível e alinhada às boas práticas de desenvolvimento de sistemas backend, servindo como fundação confiável para evoluções futuras do ecossistema.