EDGE VISION MODEL

PIPELINE DE MODELAGEM, AVALIAÇÃO E COMPARAÇÃO DE ARQUITETURAS DE DETECÇÃO DE OBJETOS

Relatório Técnico

Londrina, Dezembro de 2025

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 analítica de eventos de monitoramento provenientes de sistemas externos de detecção de risco.

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), e a documentação da API é gerada automaticamente por meio do padrão OpenAPI.

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, auditoria, controle de acesso e disponibilização de dados para análise.

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 análise posterior.

2.2 Objetivos Específicos

- Implementar uma API REST para recebimento de eventos de monitoramento enviados por dispositivos edge.
- Persistir eventos de detecção e evidências visuais associadas em banco de dados relacional.
- Implementar autenticação e controle de acesso baseados em JSON Web Tokens (JWT).
- Garantir rastreabilidade das operações por meio de registros estruturados de auditoria.
- Disponibilizar endpoints para consulta e visualização analítica dos eventos registrados.
- Organizar o backend de forma modular, favorecendo manutenção, testes e evolução futura.
- Viabilizar integração direta e desacoplada com sistemas de inferência executados em computação de borda.

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 processamento de visão computacional.

A aplicação recebe eventos estruturados e evidências visuais enviados por sistemas externos de detecção, persistindo essas informações de forma auditável e disponibilizando-as para consulta analítica. Cada funcionalidade do backend é executada de forma explícita e independente, seguindo um fluxo bem definido.

O sistema está organizado em quatro responsabilidades principais:

- **Autenticação e controle de acesso:** gerenciamento de usuários e proteção dos endpoints por meio de tokens JWT.
- **Ingestão de eventos de monitoramento:** recebimento e validação dos eventos enviados pelos dispositivos edge.
- **Persistência e auditoria:** armazenamento dos eventos, evidências visuais e registros de log.
- **Consulta e visualização analítica:** disponibilização de dados consolidados para dashboards e aplicações cliente.

Essa separação garante baixo acoplamento, clareza operacional, segurança e facilidade de manutenção, permitindo evolução independente do backend em relação aos sistemas de inferência.

4. Arquitetura do Sistema

A arquitetura do Edge Monitor API reflete diretamente a organização real do repositório, sendo estruturada de forma modular e desacoplada, 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 análise, sem executar qualquer lógica de inferência ou processamento computacional pesado. Toda a responsabilidade de detecção e validação dos eventos ocorre externamente, em sistemas de computação de borda, cabendo à API exclusivamente o papel de suporte backend.

A seguir, são descritos os principais módulos que compõem a arquitetura do sistema, bem como as interfaces REST 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.

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

- POST /api/authentication/login/
- POST /api/authentication/renovate/
- POST /api/authentication/logout/

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/
- GET /api/user/
- GET /api/user/{id}/
- PUT /api/user/{id}/
- DELETE /api/user/{id}/

A separação deste módulo permite evolução independente das regras de negócio relacionadas a usuários e mantém a camada de controle de acesso claramente definida.

monitoring/

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.

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, classe detectada, data e hora da ocorrência e evidência visual associada, persistindo essas informações de forma auditável.

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/

Contém os 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 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 interfaces administrativas ou dashboards.

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 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.

5.2 Autenticação e Controle de Acesso

A autenticação do sistema é baseada em JSON Web Tokens (JWT), utilizando a biblioteca SimpleJWT. 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.

Todos os endpoints sensíveis da API exigem autenticação válida, garantindo que apenas usuários autorizados possam criar, consultar, atualizar ou excluir recursos do sistema. Esse mecanismo assegura controle de acesso consistente e alinhado às boas práticas de segurança em APIs REST.

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. Cada evento representa uma detecção previamente confirmada 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.

Os dados são recebidos por meio de requisições estruturadas e passam por validação rigorosa antes da persistência, assegurando integridade, padronização e consistência das informações armazenadas no sistema.

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, mantendo vínculo direto e explícito com o respectivo evento registrado no banco de dados.

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

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.

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 um endpoint específico para consulta analítica dos eventos registrados. As consultas permitem filtrar eventos por intervalo de datas, retornando apenas os dados necessários para consumo por dashboards ou aplicações externas.

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

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 dos dados.

6. Registro e Monitoramento

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 pelo módulo de dashboard.
- 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 ou do tipo de operação realizada.

Cada entrada de log armazena informações como o usuário responsável (quando aplicável), a ação executada, o status da operação, uma mensagem descritiva e a data e hora do evento. Essa abordagem garante rastreabilidade temporal e fornece uma visão clara e consistente do comportamento da API ao longo de sua operação.

O mecanismo de registro é desacoplado da lógica principal das views, sendo acionado de forma padronizada por meio de um utilitário central. Essa estratégia assegura consistência na geração dos logs, baixo acoplamento entre os componentes do sistema e facilidade de manutenção e evolução do código.

Esse modelo de registro e monitoramento 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 de receber e processar eventos de monitoramento enviados por dispositivos edge. Os dados estruturados, incluindo identificador lógico do dispositivo, classe detectada e data e hora do evento, foram devidamente validados antes da persistência.

As evidências visuais associadas aos eventos foram recebidas e armazenadas conforme o fluxo definido, mantendo integridade 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 e Controle de Acesso

O mecanismo de autenticação baseado em JSON Web Tokens (JWT) apresentou funcionamento estável durante os 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.

Esse resultado confirmou a efetividade da camada de segurança implementada, assegurando controle de acesso adequado aos recursos expostos pelo sistema.

7.3 Gestão de Usuários

As funcionalidades de gestão de usuários permitiram a realização das operações de criação, consulta, atualização e exclusão de forma controlada e consistente. O sistema aplicou validações adequadas para evitar inconsistências, como duplicidade de dados e violações de integridade referencial.

Além disso, as respostas retornadas pelas operações seguiram os padrões REST definidos, favorecendo previsibilidade e facilitando a integração com aplicações cliente e sistemas externos.

7.4 Consulta Analítica via Dashboard

O módulo de dashboard possibilitou a recuperação eficiente de eventos de monitoramento por meio de filtros baseados em intervalo de datas. Os resultados retornados foram estruturados conforme 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 registrados.

7.5 Auditoria e Rastreabilidade

O mecanismo de auditoria registrou de forma consistente todas as operações relevantes executadas na API, incluindo autenticações, ingestão de eventos, manipulação de usuários e consultas ao dashboard.

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

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, auditoria e consulta analítica. O sistema apresentou comportamento estável, previsível e alinhado aos objetivos definidos, confirmando sua adequação como componente backend em uma arquitetura de monitoramento baseada em computação de borda.

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, como balanceamento de carga, replicação automática de serviços ou tolerância a falhas distribuídas.

Essa delimitação permitiu concentrar o desenvolvimento na robustez funcional da API e na organização arquitetural, evitando 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), sem diferenciação de perfis, papéis ou políticas de autorização mais granulares.

Essa abordagem garante segurança básica adequada ao escopo do sistema, ao mesmo tempo em que mantém simplicidade operacional. A introdução de mecanismos avançados de autorização, como controle baseado em papéis (RBAC), pode ser considerada como evolução futura.

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 e ao contexto do sistema, essa estratégia não contempla, nesta etapa, soluções de armazenamento distribuído ou serviços externos de object storage, os quais podem ser considerados em cenários de maior escala ou alta disponibilidade.

8.4 Processamento Síncrono dos Eventos

O processamento dos eventos de monitoramento é realizado de forma síncrona, garantindo previsibilidade e simplicidade no fluxo de ingestão. Essa abordagem mostrou-se adequada para o volume de eventos esperado e para os objetivos do sistema.

Entretanto, em cenários de alta taxa de eventos ou necessidade de maior escalabilidade, mecanismos assíncronos ou filas de processamento poderiam 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.

Análises semânticas mais complexas, correlações entre eventos ou validações de contexto não fazem parte do escopo atual do sistema, podendo ser incorporadas em evoluções futuras conforme necessidades analíticas adicionais.

8.6 Considerações de Segurança

Aspectos avançados de segurança, como políticas 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 e pelo uso de JWT, mantendo equilíbrio entre robustez e simplicidade arquitetural.

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, 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) e documentação automática por meio de OpenAPI proporcionou previsibilidade operacional, segurança adequada ao 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 implementados possibilitaram o registro consistente das ações realizadas no sistema, assegurando rastreabilidade, transparência operacional e suporte à análise de falhas. Adicionalmente, a disponibilização de dados consolidados por meio de consultas filtradas por intervalo de datas demonstrou a capacidade do backend em atender demandas analíticas e servir como base para dashboards e integrações futuras.

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.