****

****

**Projeto Kali - Documentação final Base de Dados – SQL**

**Autor: Edson Monteiro de Almeida**

**Instituição : CODERHOUSE**

**Treinamento : SQL**

**Professor : Levi Cruz**

**Data: 23/02/2024**

1. **Introdução:**

Este projeto tem como objetivo modelar um banco de dados relacional para centralizar e integrar informações de diversas ferramentas de monitoração (Zabbix, Grafana, Orion, SIEM), sistema de inventário e sistema ITSM, com foco em auxiliar o time de operação de uma empresa que oferece serviços de telecomunicações, cloud e segurança. A criação dessa base de dados visa solucionar os desafios da gestão de dados em silos, proporcionando uma visão unificada do ambiente de TI e facilitando a análise e correlação de informações relevantes para a operação dos serviços. resolvendo os desafios da gestão de dados distribuídos e melhorando a correlação de informações críticas.

1. **Objetivo:**

O objetivo principal é construir um modelo de banco de dados robusto e escalável, capaz de armazenar e organizar dados de diferentes fontes para a aplicação **Kali**, que está sendo desenvolvida com foco na **otimização da gestão da infraestrutura de TI** e na **garantia da qualidade dos serviços de telecomunicações, cloud e segurança**. O modelo de dados permitirá:

* **Controlar a disponibilidade do ambiente e serviços**, facilitando a identificação de falhas e a tomada de ações proativas.
* **Relacionar eventos de monitoração com seus respectivos ativos e contratos**, fornecendo informações relevantes para a gestão de SLAs e SLOs.
* **Fornece detalhes sobre a abertura de incidentes no ITSM**, permitindo uma análise completa do histórico de problemas e da performance da equipe de suporte.
* **Controlar SLAs e SLOs dos serviços fornecidos e da infraestrutura**, garantindo o cumprimento dos acordos de nível de serviço e a satisfação dos clientes.

Adicionalmente, o modelo de dados facilitará a **análise centralizada de informações** de monitoração, inventário e ITSM, permitindo a **identificação de tendências, gargalos e anomalias** que possam impactar os serviços. A integração dos dados visa **agilizar a resolução de problemas** e fornecer **insights para a tomada de decisão estratégica**, com base em informações completas e contextualizadas. Além disso, a aplicação **Kali** oferecerá um **portal interativo e dashboards dinâmicos**, permitindo que os usuários acompanhem em tempo real a saúde da infraestrutura e realizem **ações gerenciais baseadas nos dados**

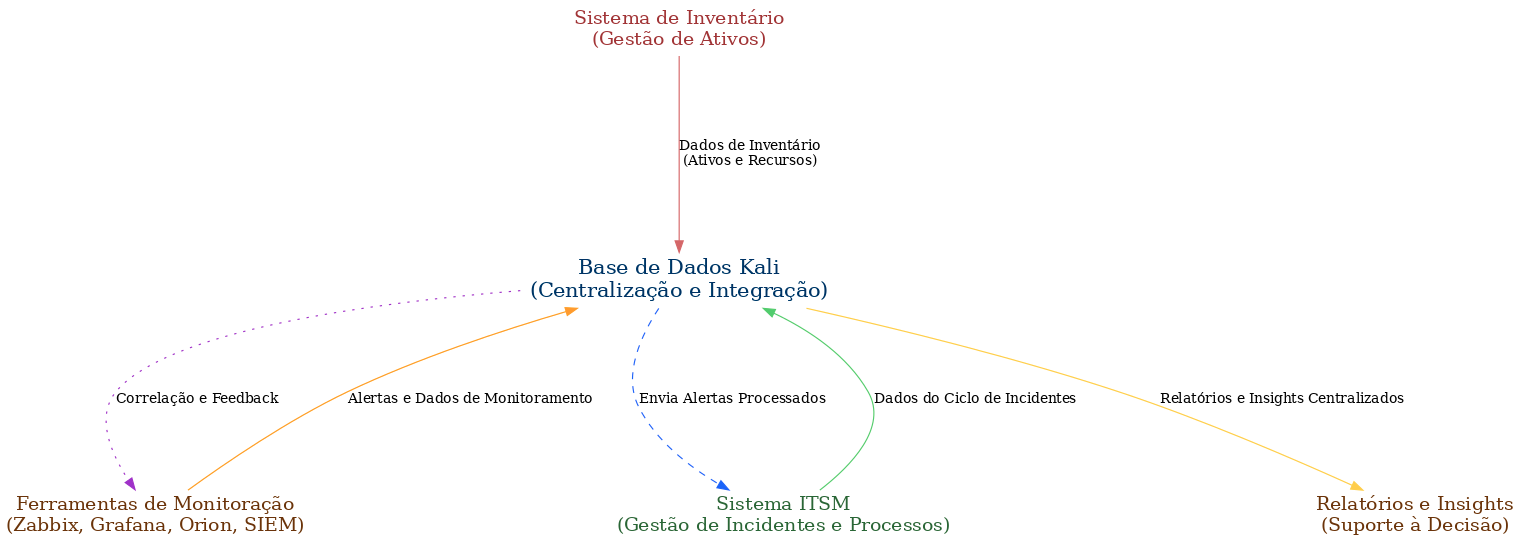
1. **Situação Problemática:**

Atualmente, a empresa enfrenta desafios na gestão de sua infraestrutura de TI e na garantia da qualidade dos serviços, devido à falta de uma solução centralizada para armazenar e analisar os dados gerados por suas diversas ferramentas de monitoração.Essa falta de integração resulta em:

* **Dados em silos:** As informações ficam dispersas em diferentes sistemas, dificultando a obtenção de uma visão unificada do ambiente e a correlação de eventos.
* **Dificuldade na análise de dados:** A análise de informações relevantes para a tomada de decisão torna-se complexa e trabalhosa, demandando tempo e recursos.
* **Gargalos na resolução de problemas:** A identificação da causa raiz de problemas e a tomada de ações corretivas são prejudicadas pela falta de visibilidade e acesso rápido às informações.
* **Gestão de SLAs e SLOs:** A falta de controle e visibilidade sobre a disponibilidade dos serviços e a performance da infraestrutura aumenta o risco de não cumprimento dos acordos de nível de serviço.

1. **Modelos de negócio:**

O modelo de dados desenvolvido é focado na **integração entre monitoração, inventário e gestão de incidentes ITSM**. O banco permitirá: **Registro centralizado de alertas e incidentes**, **monitoramento em tempo real da infraestrutura**, **Correlação entre falhas, ativos e SLAs**, melhorando a gestão operacional e um **Portal interativo para usuários**, permitindo a visualização e gerenciamento dos alertas e incidentes em um ambiente acessível via dashboard.

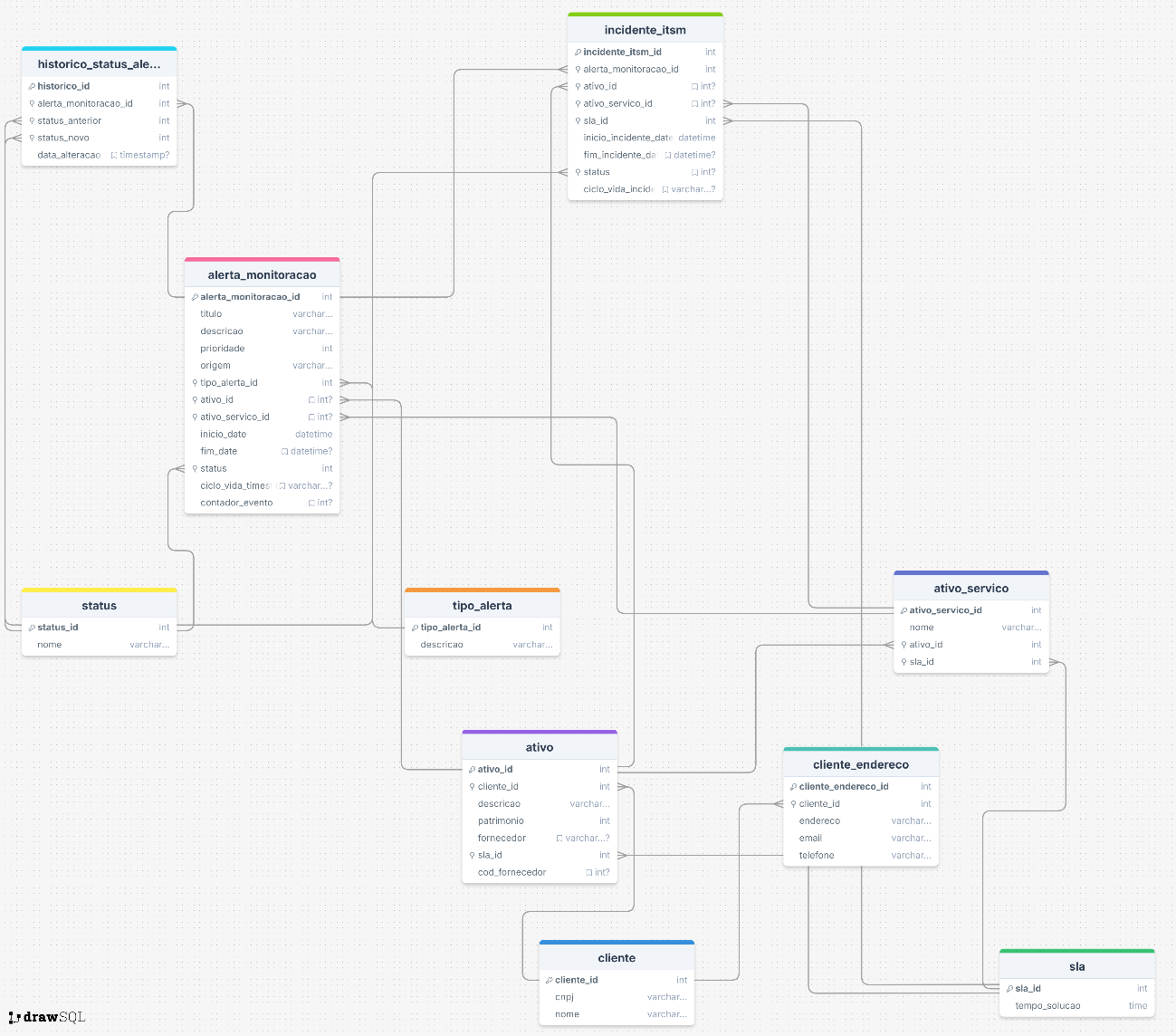


1. **Digrama da base dados:**

Diagrama

O conteúdo gerado por IA pode estar incorreto.

|  |  |
| --- | --- |
| **Tabela** | **Descrição** |
| Cliente | Registra os clientes do sistema. |
| cliente\_endereco | Endereços dos clientes. |
| Sla | Regras de Acordo de Nível de Serviço. |
| Ativo | Equipamentos e dispositivos monitorados. |
| ativo\_servico | Serviços associados a ativos. |
| tipo\_alerta | Classificação de alertas de monitoração. |
| alerta\_monitoracao | Registros de alertas ativos. |
| incidente\_itsm | Registros de incidentes associados a alertas. |
| historico\_status\_alerta | Armazena histórico de mudanças no status dos alertas. |
| Status | Status possíveis para alertas e incidentes. |

****

1. **Principais Relacionamentos**

**cliente → cliente\_endereco** *(1:N) :* Um **cliente** pode ter **vários endereços** cadastrados. Chave estrangeira: cliente\_endereco.cliente\_id → cliente.cliente\_id.

**cliente → sla** *(1:N) :* Um **cliente** pode ter **vários SLAs associados**, mas um SLA pertence a apenas um cliente. Chave estrangeira: sla.cliente\_id → cliente.cliente\_id.

**ativo → ativo\_servico** *(1:N):* Um **ativo** pode fornecer **vários serviços**. Chave estrangeira: ativo\_servico.ativo\_id → ativo.ativo\_id.

**ativo → alerta\_monitoracao** *(1:N) 1:* Um **ativo** pode gerar **vários alertas de monitoração** ao longo do tempo. Chave estrangeira: alerta\_monitoracao.ativo\_id → ativo.ativo\_id.

t**ipo\_alerta → alerta\_monitoracao** *(1:N) :* Um **tipo de alerta** pode estar associado a **múltiplos alertas registrados**. Chave estrangeira: alerta\_monitoracao.tipo\_alerta\_id → tipo\_alerta.tipo\_alerta\_id.

**alerta\_monitoracao → incidente\_itsm** *(1:1):* Um **alerta pode originar um único incidente no ITSM**, e cada incidente ITSM está vinculado a apenas um alerta. Chave estrangeira: incidente\_itsm.alerta\_monitoracao\_id → alerta\_monitoracao.alerta\_monitoracao\_id.

**sla → incidente\_itsm** *(1:N) :* Um **SLA** pode estar associado a **vários incidentes ITSM**, garantindo controle sobre tempos de resposta e resolução. Chave estrangeira: incidente\_itsm.sla\_id → sla.sla\_id.

**incidente\_itsm → historico\_status\_alerta** *(1:N) :* Um **incidente** pode ter **múltiplos registros de alteração de status**, criando um histórico completo. Chave estrangeira: historico\_status\_alerta.incidente\_itsm\_id → incidente\_itsm.incidente\_itsm\_id.

**status → historico\_status\_alerta e incidente\_itsm** *(1:N) :* A tabela **status** contém **os possíveis estados** para alertas e incidentes.

* Chave estrangeira: historico\_status\_alerta.status\_id → status.status\_id.
* Chave estrangeira: incidente\_itsm.status\_id → status.status\_id.

**ativo\_servico → sla** *(1:N) :* Um **SLA** pode ser associado a **vários serviços fornecidos por ativos**, garantindo controle sobre disponibilidade e desempenho. Chave estrangeira: sla.ativo\_servico\_id → ativo\_servico.ativo\_servico\_id.

1. Views

O banco de dados inclui diversas Views para facilitar a extração e análise de dados. Abaixo estão as principais Views e como validá-las:

1. vw\_alertas\_abertos : Exibe todos os alertas que ainda estão em aberto.

* Validação: SELECT \* FROM vw\_alertas\_abertos;

2. vw\_incidentes\_por\_cliente : Lista todos os incidentes ITSM associados a um cliente específico.

* Validação: SELECT \* FROM vw\_incidentes\_por\_cliente WHERE cliente\_id = 5;

3. vw\_sla\_vencidos : Exibe todos os incidentes onde o SLA foi ultrapassado.

* Validação: :SELECT \* FROM vw\_sla\_vencidos;

4. vw\_alertas\_por\_ativo : Lista todos os alertas associados a um ativo específico.

* Validação: SELECT \* FROM vw\_alertas\_por\_ativo WHERE ativo\_id = 10;

5. vw\_historico\_status\_alertas : Mostra todas as alterações de status de alertas registradas.

* Validação: SELECT \* FROM vw\_historico\_status\_alertas;

1. Store Procedures

7. Store Procedures e Validação

O banco de dados inclui diversas Stored Procedures para automatizar processos. Abaixo estão as principais e como validá-las:

1. sp\_abrir\_incidente\_para\_alerta

* Insere um novo incidente baseado em um alerta existente.
* Validação:

CALL sp\_abrir\_incidente\_para\_alerta(1, 1, 1, 2);

2. sp\_deletar\_alertas\_antigos

* Remove alertas antigos que não possuem mais relevância.
* Validação:

CALL sp\_deletar\_alertas\_antigos();

3. sp\_inserir\_alerta

* Insere um novo alerta ou atualiza um existente se já houver um similar.
* Validação:

CALL sp\_inserir\_alerta('Título do Alerta', 'Descrição do Alerta', 2, 5, 3, 1, 'Monitoramento');

4. sp\_concluir\_alerta

* Atualiza o status do alerta para "Fechado" e registra a data de conclusão.
* Validação:

CALL sp\_concluir\_alerta(1);

5. sp\_alterar\_status\_incidente

* Modifica o status de um incidente conforme sua evolução.
* Validação: CALL sp\_alterar\_status\_incidente(10, 3);

1. Tigger

O banco de dados inclui diversas Triggers para garantir a integridade dos dados e automatizar processos. Abaixo estão as principais Triggers e como validá-las:

1. trg\_registrar\_mudanca\_status

* Registra automaticamente mudanças no status de alertas na tabela historico\_status\_alerta.
* Validação:
* UPDATE alerta\_monitoracao SET status = 2 WHERE alerta\_monitoracao\_id = 1;

SELECT \* FROM historico\_status\_alerta WHERE alerta\_monitoracao\_id = 1;

2. trg\_atualizar\_ciclo\_vida\_alerta

* Atualiza o campo ciclo\_vida\_timestamp no momento do fechamento do alerta.
* Validação:
* UPDATE alerta\_monitoracao SET status = 7, fim\_date = NOW() WHERE alerta\_monitoracao\_id = 2;

SELECT ciclo\_vida\_timestamp FROM alerta\_monitoracao WHERE alerta\_monitoracao\_id = 2;

3. trg\_atualizar\_ciclo\_vida\_incidente

* Atualiza o campo ciclo\_vida\_incidente\_timestamp ao encerrar um incidente.
* Validação:
* UPDATE incidente\_itsm SET status = 6, fim\_incidente\_date = NOW() WHERE incidente\_itsm\_id = 3;

SELECT ciclo\_vida\_incidente FROM incidente\_itsm WHERE incidente\_itsm\_id = 3;

4. trg\_auto\_inserir\_historico\_status

* Insere automaticamente um novo registro na tabela historico\_status\_alerta sempre que um alerta tem seu status alterado.
* Validação:
* UPDATE alerta\_monitoracao SET status = 4 WHERE alerta\_monitoracao\_id = 5;

SELECT \* FROM historico\_status\_alerta WHERE alerta\_monitoracao\_id = 5;

1. Funções

O banco de dados inclui diversas Funções para facilitar a manipulação e consulta de dados. Abaixo estão as principais Funções e como validá-las:

1. fn\_calcular\_sla\_restante

* Retorna o tempo restante para a conclusão do SLA de um incidente.
* Validação:

SELECT fn\_calcular\_sla\_restante(10);

2. fn\_verificar\_incidente\_duplicado

* Verifica se um incidente já foi aberto para um alerta específico.
* Validação:

SELECT fn\_verificar\_incidente\_duplicado(5);

3. fn\_contar\_alertas\_por\_ativo

* Retorna o número total de alertas associados a um ativo.
* Validação:

SELECT fn\_contar\_alertas\_por\_ativo(3);

4. fn\_tempo\_medio\_resolucao

* Retorna o tempo médio de resolução dos incidentes fechados.
* Validação:

1. Inserção de dados pela Aplicação

A inserção de dados no banco de dados será realizada por meio da aplicação Kali, desenvolvida em Python e integrada com um API Gateway. Esse gateway possibilita a comunicação segura entre as ferramentas de monitoração, a aplicação e o banco de dados.

Linha do tempo

O conteúdo gerado por IA pode estar incorreto.

Fluxo de Inserção de Dados

1. As ferramentas de monitoração enviam eventos via POST para a aplicação.
2. A aplicação valida se o alerta já existe na base de dados:
   * Se o alerta já existir, ele é atualizado.
   * Se o alerta não existir, ele é inserido na tabela alerta\_monitoracao.
3. Se for um novo alerta, a aplicação:
   * Realiza um POST no ITSM para abertura de um incidente.
   * O ID do incidente retornado pelo ITSM é utilizado como chave primária na tabela incidente\_itsm.
4. Alterações no status dos alertas ou incidentes são feitas via chamadas PUT para garantir que o banco de dados esteja sempre atualizado.
5. Os usuários podem visualizar, gerenciar e interagir com os alertas e incidentes através de um dashboard interativo.

Chamadas de API e Procedimentos Envolvidos

A aplicação chama as Stored Procedures para garantir que a inserção e atualização de dados ocorram de forma eficiente:

* sp\_inserir\_alerta: Verifica a existência do alerta e o insere ou atualiza se necessário.
* sp\_abrir\_incidente\_para\_alerta: Cria um novo incidente ITSM vinculado a um alerta.
* sp\_concluir\_alerta: Fecha um alerta quando resolvido.
* sp\_alterar\_status\_incidente: Modifica o status do incidente ITSM conforme sua evolução.

Esse fluxo garante que os dados de monitoramento sejam registrados, processados e atualizados corretamente, mantendo a consistência e rastreabilidade das informações dentro do sistema Kali.

1. Análise e Relatorios analíticos

A extração e análise de informações do banco de dados foram realizadas utilizando ferramentas especializadas para fornecer insights estratégicos sobre a infraestrutura monitorada. O estudante utilizou Microsoft Excel, Tableau, Microsoft Power BI e outras aplicações para gerar relatórios detalhados.

1. Geração de Relatórios

* Os relatórios foram gerados a partir das Views e funçoes implementadas no banco de dados.
* A análise incluiu a avaliação da performance dos SLAs, a taxa de incidentes por ativo e tempo médio de resolução de problemas.
* Os resultados foram exportados para PDFs independentes, disponíveis junto com a documentação.

2. Validação dos Relatórios

* Relatório de SLAs vencidos:

SELECT \* FROM vw\_sla\_vencidos;

* Relatório de Tempo Médio de Resolução:

SELECT fn\_tempo\_medio\_resolucao();

* Relatório de Alertas por Ativo:

SELECT \* FROM vw\_alertas\_por\_ativo;

3. Exportação de Dados

* Os relatórios foram gerados automaticamente e exportados em formato CSV e PDF.
* A exportação dos dados permite a integração com ferramentas de BI para visualização avançada.

1. Backup e Alta Disponibilidade

Rotina de Backup para garantir a integridade e recuperação dos dados, foi implementado um processo automatizado de backup:

* Backup Diário: Um dump do banco de dados é gerado automaticamente todas as noites.
* Retenção: Backups são armazenados por um período de 30 dias.
* Armazenamento Seguro: Os backups são transferidos para um servidor remoto seguro e armazenados em múltiplos locais para redundância.

Comando de Backup:

mysqldump -u backup\_user -p --routines --triggers --single-transaction kali > /backups/backup\_kali\_$(date +%Y-%m-%d).sql

Estratégia de Alta Disponibilidade e garantir resiliência, o banco de dados conta com:

* Replicação Master-Slave: O banco de dados primário replica seus dados para um servidor secundário em tempo real.
* Failover Automático: Caso o servidor principal falhe, a aplicação se conecta automaticamente ao servidor secundário.
* Balanceamento de Carga: Uso de ProxySQL ou HAProxy para distribuir as requisições entre os servidores disponíveis.

Essa abordagem garante que o banco de dados permaneça acessível e minimiza os impactos de falhas no sistema.

14. Conclusão e Próximos Passos

primeira etapa do Projeto Kali foi concluída com sucesso, incluindo a modelagem do banco de dados, a definição da estrutura de tabelas, a implementação de Stored Procedures, Triggers, Views e Funções. Essa fase permite uma base sólida para a gestão eficiente de alertas, incidentes e SLAs, garantindo que as informações sejam registradas, validadas e processadas de forma otimizada.

O próximo passo do projeto está diretamente relacionado à trilha de desenvolvimento pessoal sobre Back-end e Front-end, com objetivo de expandir as funcionalidades da aplicação Kali para permitir uma experiência completa e interativa aos usuários. Os próximos focos incluem: Back-end: Desenvolvimento da API REST, Implementação de autenticação e controle de acesso para usuários e melhorias na automação de processos, incluindo notificações e integrações externas. Front-end: Criação de um portal com dashboard interativo, Implementação de gráficos e relatórios dinâmicos.

Essa evolução garantirá que o Projeto Kali se torne uma solução para a gestão de infraestrutura de TI, consolidando os dados e permitindo ações estratégicas para a melhoria contínua da operação.

**Links :**

**https://drawsql.app/teams/edson/diagrams/kali-date**