RAFAEL MACHADO DE MIRANDA

MONITORAMENTO EM TEMPO REAL DE DESEMPENHO DE BANCOS DE DADOS

JI-PARANÁ
2023

RAFAEL MACHADO DE MIRANDA

MONITORAMENTO EM TEMPO REAL DE DESEMPENHO DE BANCOS DE DADOS

Monografia apresentada à Banca Examinadora do curso de Sistemas de Informação do Centro Universitário São Lucas Ji-Paraná, como requisito de aprovação para obtenção do Título de Bacharel em Sistemas de Informação.

Orientador: Prof. Esp. Wallison Storck Coelho

JI-PARANÁ
2023

Dados Internacionais de Catalogação na Publicação - CIP

M672m Miranda, Rafael Machado de.

Monitoramento em tempo real de desempenho de bancos de dados. / Rafael Machado de Miranda. – Ji-Paraná, 2023.
80 p.; il.

Monografia (Sistemas de Informação) – Centro Universitário São Lucas Ji-Paraná, 2023.

Orientador: Prof. Esp. Wallison Storck Coelho.

1. Monitoramento em tempo real. 2. Desempenho de bancos de dados. 3. MySQL. 4. Dashboard. I. Coelho, Wallison Storck. II. Título.

CDU 004.658

Ficha Catalográfica Elaborada pelo Bibliotecário Giordani Nunes da Silva CRB 11/1125

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida e por ter me dado condições para concluir este trabalho.

Aos meus pais, pelo amor, incentivo e apoio incondicional.

Aos meus amigos e professores.

Aos meus professores orientadores que me mostraram o caminho a ser seguido e pelas valiosas sugestões sempre objetivas e claras.

E a todos aqueles que colaboraram de alguma forma na elaboração deste trabalho e direta ou indiretamente fizeram parte da minha formação. A todos vocês o meu muito obrigado.

Dedico este trabalho à minha família,
presente de Deus na minha vida.

RESUMO

O profissional responsável por gerenciar bancos de dados possui múltiplas responsabilidades, entre elas garantir a estabilidade dos sistemas que dependem dessas tecnologias. Monitorar estes ambientes acaba sendo uma parte crítica de suas tarefas, demandando conhecimentos técnicos específicos e tempo excessivo. Por isso é importante ao DBA utilizar ferramentas de software que automatizem e facilitem o monitoramento. Este trabalho tem como objetivo apresentar uma solução para o monitoramento em tempo real de desempenho de bancos de dados MySQL e MariaDB. Para isso, foram realizadas pesquisas sobre as principais ferramentas e técnicas utilizadas para o monitoramento de bancos de dados e sobre as métricas mais relevantes para avaliar o desempenho de um banco de dados MySQL. Com base nesse levantamento, foi desenvolvido um *dashboard* utilizando as tecnologias Node.js, Vue e Vuetify, capaz de exibir de forma clara e objetiva as informações de desempenho do banco de dados em tempo real. O resultado obtido demonstrou a efetividade da solução proposta, permitindo ao DBA identificar problemas de desempenho e atuar de forma preventiva para evitar possíveis falhas.

Palavras-chaves: Monitoramento em tempo real, Desempenho de bancos de dados, MySQL, *Dashboard*.

ABSTRACT

The professional responsible for managing databases has multiple responsibilities, including ensuring the stability of systems that depend on these technologies. Monitoring these environments ends up being a critical part of their tasks, requiring specific technical knowledge and excessive time. Therefore, it is important for the DBA to use software tools that automate and facilitate monitoring. This paper aims to present a solution for real-time monitoring of performance of MySQL databases. To achieve this goal, research was conducted on the main tools and techniques used for database monitoring and on the most relevant metrics for evaluating MySQL and MariaDB database performance. Based on this survey, a dashboard was developed using Node.js, Vue, and Vuetify technologies, capable of displaying database performance information in real-time in a clear and objective way. The results obtained demonstrated the effectiveness of the proposed solution, allowing the DBA to identify performance issues and take preventive action to avoid possible failures.

Keywords: Real-time monitoring, Database performance, MySQL, Dashboard.

LISTA DE FIGURAS

LISTA DE SIGLAS

DBA	*Database Administrator* (Administrador de Banco de Dados)
HTTP	*Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)
JS	*JavaScript*
TI	Tecnologia da Informação
SGDB	Sistema de Gerenciamento de Banco de Dados
KPI	*Key Performance Indicator* (Indicador-Chave de Performance)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
SAAS	*Software as a Service* (Software como um Serviço)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
GPL	*General Public License* (Licença Pública Geral)
API	*Application Programming Interface* (Interface de Programação de Aplicação)
DOM	*Document Object Model* (Modelo de Objeto de Documento)
CSS	*Cascading Style Sheets* (Folhas de Estilo em Cascata)
ORM	*Object Relational Mapper* (Mapeamento Objeto-Relacional)
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)
CRUD	*Create, Read, Update, Delete* (Criar, Ler, Atualizar e Deletar)

SUMÁRIO

1 INTRODUÇÃO

Atualmente, o uso de bancos de dados em ambientes corporativos tem se tornado cada vez mais frequente, visto que esses sistemas são responsáveis por armazenar informações críticas para as empresas. Nesse contexto, é fundamental garantir que os bancos de dados apresentem um bom desempenho, a fim de evitar falhas e garantir a estabilidade dos sistemas que dependem dessas tecnologias.

Um DBA (Administrador de Banco de Dados) é o profissional responsável por gerenciar, configurar e otimizar bancos de dados, e o monitoramento é uma parte crítica de suas responsabilidades (DevMedia, s.d.).

No entanto, o monitoramento de desempenho de bancos de dados pode ser uma tarefa complexa, que demanda conhecimentos técnicos específicos. Além disso, muitas vezes, as informações de monitoramento são apresentadas de forma confusa e pouco intuitiva, dificultando a interpretação e tomada de decisões.

É importante investir em soluções efetivas para o monitoramento de desempenho de bancos de dados, a fim de garantir a estabilidade e a eficiência dos sistemas que dependem dessas tecnologias.

Portanto, este trabalho tem objetivo apresentar uma solução para o monitoramento em tempo real de desempenho de bancos de dados MySQL e MariaDB.

1.1 Problematização

Atualmente, quase todas as atividades desempenhadas nas empresas dependem de uma aplicação da TI. Por isso, quando o banco de dados não funciona como esperado, isso afeta diretamente o desempenho de quase todas as áreas (FWC, 2020).

No entanto, nem sempre o DBA dispõe de tempo suficiente para realizar um monitoramento eficaz do desempenho do banco de dados, pois precisa lidar com outras demandas urgentes ou rotineiras. Além disso, o monitoramento manual pode ser sujeito a

erros ou imprecisões. Por isso, é recomendável que o DBA utilize ferramentas de *software* que automatizem e facilitem o monitoramento de desempenho do banco de dados, fornecendo *insights* valiosos sobre status, performance e prevenção (QUEST, s.d.).

Essas atividades demandam tempo e atenção do DBA, que precisa estar sempre atento aos problemas que podem afetar o funcionamento dos bancos de dados. O monitoramento de desempenho é uma das atividades mais importantes e desafiadoras do DBA, pois envolve a avaliação do uso de recursos do banco de dados e a detecção de tendências e padrões ao longo do tempo. Essas informações podem ajudar o DBA a planejar e implementar estratégias de escalabilidade e capacidade, antecipando a necessidade de ajustes e melhorias no sistema (MICROSOFT LEARN, 2023).

Em se tratando dos bancos de dados *Open Source*, destaca-se neste trabalho a necessidade de empresas com menores recursos de trabalhar com os SGDB's MySQL e MariaDB. Estes possuem grande popularidade e aceitação no mercado, porém apresentam limitações e desafios, especialmente no que se refere à sua documentação. Muitas vezes vista como insuficiente e confusa por muitos usuários e desenvolvedores, a documentação sobre estes mecanismos de bancos de dados é controlada pela Oracle (no caso do MySQL), tem uma base de conhecimento ainda em crescimento (no caso do MariaDB) e ou possuem fontes de informação geradas pelas comunidades que podem ser dispersas, desorganizadas, desatualizadas ou contraditórias, dificultando a busca e a validação das informações.

Além da dificuldade da documentação, ou seja, de como extrair dados sobre o funcionamento do serviço de banco de dados, MySQL ou MariaDB não possuem ferramenta nativa para monitoramento de performance, salvo pelo MySQL Enterprise Monitor, que faz parte do pacote pago MySQL Enterprise Edition (MYSQL, s.d.).

1.2 Hipóteses

O monitoramento de desempenho de bancos de dados é uma atividade contínua e proativa, que requer o uso de ferramentas específicas e a interpretação de métricas e indicadores para identificar problemas e tomar ações corretivas de forma eficaz. Portanto, é fundamental que um DBA tenha habilidades sólidas de monitoramento de desempenho como parte integrante de suas responsabilidades profissionais (FWC, 2020).

Nesse sentido, o desenvolvimento de uma solução efetiva para visualização de desempenho de bancos de dados pode trazer benefícios significativos para as empresas que dependem desses sistemas. O monitoramento em tempo real de máquinas virtuais, bancos de dados e aplicativos oferece uma maneira eficiente de obter essa visibilidade, além de proporcionar *insights* acionáveis para a empresa para que ela possa melhorar a disponibilidade e a experiência do usuário (IBM, s.d.).

Diante desses desafios, surge a necessidade de desenvolvimento de uma nova ferramenta que atenda as demandas dos DBAs no dia a dia, oferecendo uma solução completa, integrada, segura e acessível para o monitoramento e a otimização de performance dos servidores MySQL/MariaDB através de *dashboards*, gráficos e listagens simples de fácil utilização pelos responsáveis em Banco de Dados.

1.3 Objetivos

1.3.1 Objetivo Geral

Disponibilizar para a comunidade de Administradores de Bancos de Dados que trabalhem com Bancos de Dados MySQL/MariaDB uma ferramenta gratuita que mostra a saúde do ambiente em tempo real, ajudando assim os DBAs a garantir a estabilidade e a eficiência dos sistemas e a se concentrar em tarefas mais estratégicas.

1.3.2 Objetivo Específico

- A criação de uma aplicação web que possa ser implantada em qualquer ambiente de Bancos de Dados já existente.
- Implantar funções que coletem os dados de performance, configurações e segurança dos bancos de dados monitorados.
- Apresentar as informações coletadas pela aplicação em um painel de visualização intuitivo e fácil de usar com KPIs (Indicadores-Chave de Desempenho).
- Fornecer avaliações sobre status, performance e prevenção, para o gerenciamento diário do DBA
- Desenvolver algum nível de segurança para acessar estas informações, como login com usuário e senha, perfis de acesso e *timeout* para desconexão.
- Disponibilizar de forma gratuita para toda a comunidade de Administradores de Bancos de Dados que trabalhem com Bancos de Dados MySQL/MariaDB.

1.4 Justificativa

A necessidade de monitorar e gerenciar os bancos de dados tem se tornado cada vez mais importante devido ao aumento da complexidade dos sistemas. Além disso, a falta de monitoramento pode levar a problemas como baixa performance, indisponibilidade de serviços e perda de dados.

O monitoramento permite identificar e solucionar problemas de desempenho em tempo real, como altas cargas de CPU, uso excessivo de memória, tempos de resposta lentos e outros indicadores de desempenho inadequado. O DBA deve certificar-se que o banco de dados é rápido e que a performance do servidor não afetará negativamente sua disponibilidade e usabilidade (DevMedia, s.d.).

Além disso, o monitoramento de desempenho também permite avaliar o uso de recursos do banco de dados e detectar tendências e padrões ao longo do tempo. Essas informações podem ajudar o DBA a planejar e implementar estratégias de escalabilidade

e capacidade, antecipando a necessidade de ajustes e melhorias no sistema (MICROSOFT LEARN, 2023).

Outra razão importante para o monitoramento de desempenho é a identificação de eventos e atividades que podem representar riscos para a segurança e integridade dos dados do banco de dados, ajudando a identificar tentativas de acesso não autorizado, como ataques de hackers ou violações de segurança, protegendo assim dados confidenciais. (Coradin, 2023).

Dessa forma, a motivação para a realização deste trabalho está na possibilidade de contribuir para o desenvolvimento de uma solução efetiva e acessível para o monitoramento em tempo real de desempenho de bancos de dados MySQL e MariaDB, possibilitando uma melhor gestão e tomada de decisões mais assertivas, com dashboards que permitam a visualização de informações relevantes de maneira clara e objetiva.

As ferramentas descritas de monitoramento de performance de bancos de dados MySQL/MariaDB apresentam pontos fracos que podem comprometer a qualidade e a eficácia do trabalho dos DBAs. Algumas dessas limitações são: a falta de detalhamento e de visualização das consultas e dos planos de execução, a dependência de acesso ao banco de dados, a incompatibilidade com versões mais recentes ou com diferentes mecanismos de armazenamento, a ausência de recursos gráficos ou interativos e o alto custo de aquisição ou manutenção (licenciamentos).

2 REFERENCIAL TEÓRICO

Um banco de dados é uma coleção organizada de informações que pode ser facilmente acessada, gerenciada e atualizada. Eles permitem que você mantenha o controle das informações e acesse-as de maneira rápida e fácil (BASIT, 2021). Os sistemas de banco de dados são essenciais para as empresas, pois eles comunicam informações relacionadas às transações de vendas, inventário de produtos, perfis de clientes, atividades de marketing etc.

MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto que é amplamente utilizado em todo o mundo. É conhecido por sua confiabilidade, escalabilidade e desempenho (AWS, s.d.).

MariaDB é um sistema de gerenciamento de banco de dados de código aberto que tem ganhado crescente popularidade nos últimos anos. Ele surgiu como um projeto *fork* do MySQL, outro conhecido sistema de banco de dados, e tem se destacado como uma opção robusta e confiável para aplicações de banco de dados. MariaDB foi desenvolvido pela comunidade de software livre e é mantido por uma fundação sem fins lucrativos, a MariaDB Foundation (MariaDB Foundation, s.d.). Desde a sua criação, o MariaDB tem evoluído rapidamente, ganhando novos recursos, aprimorando a performance e obtendo uma crescente base de usuários e desenvolvedores em todo o mundo graças à sua evolução ao longo dos anos e os recursos que o tornam uma alternativa viável aos sistemas de gerenciamento de banco de dados existentes.

Embora MariaDB tenha sido inicialmente criado como uma atualização aperfeiçoada do MySQL, com o tempo, os dois sistemas de gerenciamento de banco de dados se tornaram bastante diferentes. Por exemplo, MariaDB é totalmente licenciado com GPL, enquanto o MySQL tem uma abordagem de dupla licença. Além disso, cada um lida com *pools de thread* de maneira diferente e MariaDB suporta muitos mecanismos de armazenamento diferentes (Jankov, 2022).

MySQL e MariaDB são dois dos mais populares sistemas de gerenciamento de banco de dados de código aberto, ambas opções oferecendo uma ampla variedade de recursos e ferramentas para ajudar os usuários a gerenciar seus dados.

2.1 Ferramentas existentes

Um administrador de banco de dados (DBA) é o profissional responsável pela instalação, administração e suporte dos SGBDs, sempre assegurando segurança, disponibilidade e eficiência à base de dados (DevMedia, s.d.). Para prevenir problemas e garantir a eficiência do sistema, um DBA pode adotar várias metodologias.

A seguir, serão apresentadas algumas dessas ferramentas, destacando seus pontos fracos e a necessidade de desenvolvimento de uma nova ferramenta que atenda as demandas dos DBAs no dia a dia.

- MySQL Enterprise Monitor: O MySQL Enterprise Monitor é uma ferramenta paga, que requer a aquisição do MySQL Enterprise Edition, que tem um custo elevado em comparação com outras soluções de código aberto ou mais acessíveis (MYSQL, s.d.).

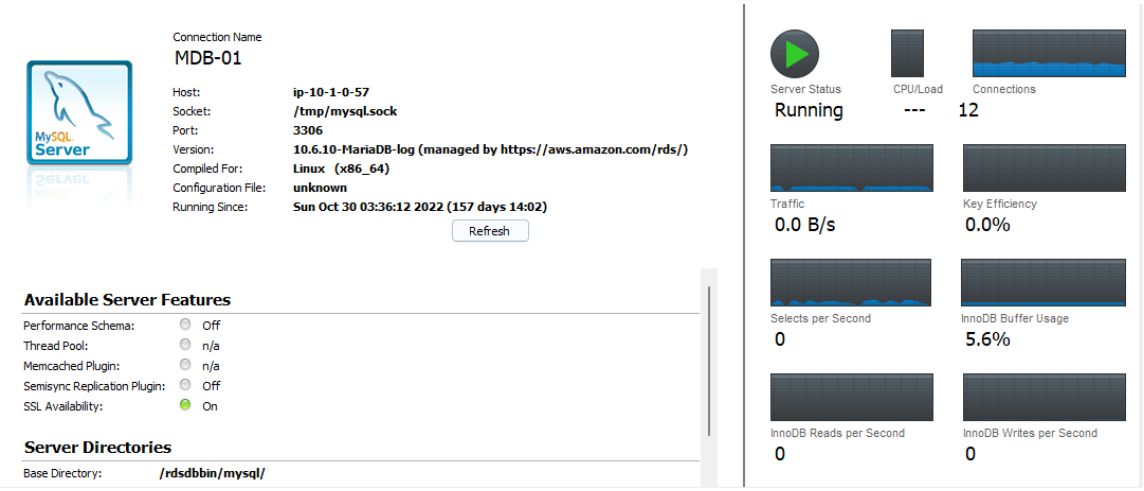
É uma ferramenta complexa, que requer uma instalação e configuração cuidadosas, além de um conhecimento prévio dos conceitos e recursos do MySQL. Só oferece capacidade para monitorar servidores MySQL e não oferece suporte a outros tipos de bancos de dados ou sistemas (MYSQL, s.d.).

- MySQLWorkbench: é uma ferramenta visual para arquitetos de banco de dados, desenvolvedores e DBAs. Fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes para configuração de servidor, administração de usuários e backup (MySQL, s.d.).

Possui painel de desempenho e relatórios que permitem visualizar o desempenho geral do servidor. Para desenvolvedores, o MySQL Workbench fornece visualizações para otimizar consultas e acesso a dados (MySQL, s.d.).

Entretanto, possui vários problemas ao trabalhar com MariaDB: Não é fácil conectar o MariaDB ao MySQL Workbench, a ferramenta fornecida pelo PhpMyAdmin para extrair o diagrama Entidade-Relacionamento do esquema SQL é muito básica, configurar o SSL no MariaDB para se conectar ao MySQL Workbench requer um procedimento complexo e ainda assim mostra um ‘warning’ ao tentar se conectar com SSL, e é necessário clicar em ‘Continue Anyway’ para prosseguir. Claramente o MySQL Workbench precisa ser atualizado para funcionar corretamente (Lo Duca, 2021).

Figura 1 – Tela de Server Status do MySQL Workbench. Solução que esta ferramenta e design de banco de dados visual oferece para monitoramento.



Fonte: print screen da aplicação do MySQL Workbench

- Mytop: é uma ferramenta que roda no console, inspirada no comando top do Linux, que exibe as estatísticas do servidor MySQL em formato tabular, como o total de consultas, requisições lentas, *uptime*, carga etc. É escrita em Perl e requer acesso ao banco de dados para funcionar. Um ponto fraco dessa ferramenta é que ela não oferece uma visão detalhada das consultas executadas nem dos planos de execução (Praciano, 2014).

- Mtop: é uma ferramenta similar ao Mytop, mas com algumas funcionalidades adicionais, como a possibilidade de pausar o monitoramento, filtrar as consultas por

usuário ou banco de dados e executar comandos SQL diretamente no console. Também é escrita em Perl e requer acesso ao banco de dados. Um ponto fraco dessa ferramenta é que ela não é mais mantida desde 2006 e pode apresentar problemas de compatibilidade com versões mais recentes do MySQL/MariaDB (Praciano, 2014).

- Innotop: é uma ferramenta que roda no console e é especializada em monitorar servidores MySQL/MariaDB que utilizam o mecanismo de armazenamento InnoDB. Ela exibe informações sobre as transações, os bloqueios, os *buffers*, os logs e outros aspectos do InnoDB. Também permite filtrar e ordenar os dados por diversos critérios e executar comandos SQL no console. É escrita em Perl e requer acesso ao banco de dados.

Um ponto fraco dessa ferramenta é que ela não monitora outros mecanismos de armazenamento além do InnoDB e pode não ser adequada para ambientes heterogêneos (Praciano, 2014).

- Mysqladmin: é uma ferramenta que faz parte do pacote oficial do MySQL/MariaDB e permite executar diversas operações administrativas no servidor, como criar ou remover bancos de dados, usuários e tabelas, alterar senhas, verificar o status do servidor, entre outras. Também pode ser usada para monitorar algumas métricas básicas do servidor, como o número de conexões ativas, o número de consultas por segundo e o uso da CPU e da memória. É escrita em C e requer acesso ao banco de dados.

Um ponto fraco dessa ferramenta é que ela não oferece uma visão detalhada das consultas executadas nem dos planos de execução e não possui recursos gráficos ou interativos (Praciano, 2014).

- SolarWinds Database Performance Monitor (DPM): é uma ferramenta baseada em software como serviço (SaaS) que permite monitorar e otimizar o desempenho dos servidores MySQL/MariaDB em tempo real. Ela mede diversas métricas sobre cada consulta executada, como latência, taxa de transferência, erros, avisos, uso de índice, planos de execução etc., em resolução de microssegundos (SolarWinds, s.d.).

Sendo uma solução baseada em SaaS, não é adequada para ser implantada *on-premise* (servidor local), limitando o controle e a personalização do usuário sobre o monitoramento e a análise de desempenho.

- Saveincloud: é uma plataforma que oferece soluções de hospedagem e gerenciamento de banco de dados para empresas e desenvolvedores que desejam melhorar a performance e a escalabilidade de seus sistemas.

Seu foco principal é no MySQL e no MariaDB, dois dos sistemas de gerenciamento de banco de dados mais populares do mundo. Através de técnicas de "tuning" e otimização de desempenho, a Saveincloud pode ajudar a aumentar a velocidade e a eficiência do banco de dados, reduzindo o tempo de resposta e garantindo uma melhor experiência para o usuário final. A plataforma também oferece serviços de backup, replicação e monitoramento para garantir a segurança e a disponibilidade dos dados (Saveincloud, 2021).

É uma solução focada em cloud escalável, que pode não atender às necessidades de empresas que preferem soluções *on-premises* ou híbridas para seus bancos de dados.

- Devtrends: A ferramenta DevTrends é uma solução de monitoramento e performance para bancos de dados SQL Server. Ela permite coletar métricas importantes do banco de dados, como uso de CPU, memória, E/S de disco, entre outras, e exibe essas informações em um painel de controle fácil de usar.

Com o DevTrends, os administradores de banco de dados podem identificar e solucionar problemas de performance rapidamente, garantindo a disponibilidade e eficiência do banco de dados. Além disso, a ferramenta oferece alertas proativos que avisam quando uma métrica crítica está fora dos limites estabelecidos, permitindo que os usuários tomem ações antes que ocorram problemas mais sérios (DevTrends, s.d.).

O Devtrends não possui uma plataforma própria de cloud escalável, mas utiliza a tecnologia da Virtuozzo, que pode limitar o controle e a personalização do usuário sobre o monitoramento e a análise de desempenho (DEVTRENDS, s.d.).

2.2 Prevenção

Por exemplo, boas práticas de segurança de dados incluem técnicas de proteção de dados, como criptografia de dados, gerenciamento de chaves, edição de dados, subconjunto de dados e mascaramento de dados, bem como controles de acesso de usuário privilegiado, auditoria e monitoramento (Oracle, s.d.).

Além disso, o DBA deve monitorar a performance da base de dados e determinar os procedimentos de recuperação de dados. Isso permite identificar e solucionar problemas em tempo real e planejar estratégias para garantir a escalabilidade e capacidade do sistema (ESR, 2020).

2.3 Monitoramento

O objetivo do monitoramento de bancos de dados é avaliar o desempenho do servidor. Um monitoramento eficaz envolve a criação de instantâneos periódicos do desempenho atual para isolar processos que estão ocasionando problemas e a coleta contínua de dados para o controle das tendências de desempenho. (MICROSOFT LEARN, 2023).

Uma abordagem de monitoramento comum é o uso de ferramentas de monitoramento especializadas que permitem a coleta de métricas e indicadores do banco de dados em tempo real. Essas ferramentas podem fornecer informações detalhadas sobre a utilização de recursos do banco de dados, como CPU, memória, disco e rede, além de identificar consultas lentas, bloqueios, erros de configuração e outras anomalias de desempenho (QUEST, s.d.).

Outra forma importante de monitoramento é o desenvolvimento de painéis de controle com dados de monitoramento, que permitem ao DBA visualizar e interpretar as métricas de desempenho em formato gráfico e intuitivo. Esses painéis podem ser criados utilizando-se ferramentas de visualização de dados, como o Grafana, que possibilita a

criação de *dashboards* customizados com gráficos, tabelas e outros elementos visuais para monitorar o desempenho dos bancos de dados (Grafana Labs, s.d.).

Além das ferramentas de monitoramento e dos painéis de controle, o DBA também pode utilizar scripts e comandos SQL para coletar e analisar dados de desempenho do banco de dados. Por exemplo, é possível criar consultas SQL para identificar consultas lentas, analisar o plano de execução de consultas, verificar o uso de índices, entre outros aspectos que impactam o desempenho do banco de dados (MICROSOFT LEARN, 2022).

É importante ressaltar que o monitoramento de desempenho de bancos de dados é uma atividade contínua e proativa, que requer a interpretação correta dos dados coletados e ações corretivas eficazes. Portanto, é fundamental que um DBA tenha conhecimentos técnicos sólidos e habilidades de interpretação de dados para realizar um monitoramento efetivo do desempenho de bancos de dados (Jankov, 2022).

2.4 Performance

Existem várias metodologias para monitorar o desempenho de bancos de dados. Por exemplo, a Oracle oferece uma ferramenta para monitorar e avaliar o desempenho de bancos de dados Oracle em um compartimento ou grupo de bancos de dados. Os dados de desempenho podem ser exibidos em um mapa de árvore avançado de desempenho, onde é possível identificar facilmente o banco de dados que requer atenção em várias dimensões, como tempo do BD, CPU e E/S (Oracle, s.d.).

Outra ferramenta é o SolarWinds Database Performance Monitor (DPM), que oferece otimização e monitoramento de desempenho para bancos de dados tradicionais, de código aberto e nativos da nuvem. Ele fornece dados históricos e em tempo real para identificar problemas de desempenho (SolarWinds, s.d.).

A meta do monitoramento de bancos de dados é avaliar o desempenho do servidor. Um monitoramento eficaz envolve a criação de instantâneos periódicos do

desempenho atual para isolar processos que estão ocasionando problemas e a coleta contínua de dados para o controle das tendências de desempenho (SolarWinds, s.d.).

3 MATERIAIS E MÉTODOS

3.1 Materiais

3.1.1 MySQL/MariaDB

Ao monitorar um sistema ou servidor, não basta apenas mostrar dados em tempo real (uso de memória atual, por exemplo). É preciso também manter um histórico do monitoramento de performance, ou seja, armazenar e consultar os dados coletados ao longo do tempo. Isso permite identificar tendências de comportamento, realizar comparações de períodos de uso e detectar problemas ou quedas no serviço.

Neste trabalho, independente do objetivo de monitorar bancos de Dados MySQL e MariaDB, podemos utilizar a própria instância de banco para armazenar os dados de monitoramento, facilitando a implantação e gerenciamento pelos DBA's.

3.1.2 Github

Toda a aplicação será disponibilizada de maneira pública por um repositório do GitHub, de forma que possa continuar sendo desenvolvida pela comunidade.

GitHub é um serviço de hospedagem de código-fonte com controle de versão usando o Git. Ele foi fundado em 2008 como Logical Awesome e lançou seu site em abril do mesmo ano. Desde então, o GitHub tem crescido continuamente e lançado vários produtos, incluindo o GitHub Enterprise, Redcarpet e Hubot (Timeline of GitHub, s.d.).

Hoje em dia, o GitHub é amplamente utilizado por desenvolvedores de todo o mundo para hospedar e compartilhar código-fonte. Ele também oferece recursos como controle de acesso, rastreamento de bugs, solicitações de recursos de software, gerenciamento de tarefas, integração contínua e wikis para cada projeto (KINSTA, 2022).

O sistema de monitoramento desenvolvido será armazenado e disponibilizado através do GitHub, permitindo que outros desenvolvedores tenham acesso ao código-fonte e possam utilizar e contribuir para o projeto.

3.1.3 NodeJS

O *back-end* da aplicação será desenvolvida com o serviço web do Node.js e com o *framework* Express.js para fornecer os recursos de rotas e requisições (NPM, s.d.).

O Node.js é uma plataforma de código aberto baseada em JavaScript que permite a execução de código JavaScript no lado do servidor. Ele foi criado em 2009 por Ryan Dahl e é mantido pela Node.js Foundation. O Node.js foi construído sobre o motor de JavaScript V8 do Google Chrome, o que o torna altamente eficiente e escalável para aplicações de rede e servidor (Melo, 2021).

A história do Node.js remonta a uma apresentação feita por Ryan Dahl no JSConf EU em 2009, intitulada "Construindo servidores HTTP escaláveis com Node.js" (Dahl, 2009). A partir dessa apresentação, o Node.js começou a ganhar popularidade rapidamente, principalmente entre os desenvolvedores de JavaScript que buscavam uma forma eficiente de criar aplicações de servidor em tempo real e escaláveis. Desde então, o Node.js tem evoluído constantemente, com o apoio de uma grande comunidade de desenvolvedores e empresas que contribuem para seu desenvolvimento.

Uma das principais características do Node.js é seu modelo de E/S (entrada/saída) não bloqueante e orientado a eventos, que permite a construção de aplicações de alto desempenho e escaláveis (Node.js Foundation, s.d.). Além disso, o Node.js é conhecido por seu ecossistema de pacotes gerenciados pelo npm (Node Package Manager), que oferece uma vasta gama de módulos e bibliotecas para auxiliar no desenvolvimento de aplicações.

O Node.js tem sido amplamente adotado por empresas e desenvolvedores em todo o mundo para a construção de aplicações web, APIs, aplicações em tempo real,

microserviços e muitos outros tipos de projetos. Sua popularidade e adoção crescente têm contribuído para a evolução constante da plataforma, com atualizações regulares e suporte ativo da comunidade de desenvolvedores.

3.1.4 VueJS

O visual da aplicação, o *front-end*, será desenvolvido com o Vue.js, devido ser um framework amigável que possibilita desenvolver interfaces de usuário interativas, ainda mais ao utilizar os componentes disponíveis do Vuetify,js, e pode ser integrado com outros frameworks e bibliotecas.

Vue.js é um framework JavaScript de código aberto para a construção de interfaces de usuário interativas e reativas. Ele foi criado em 2014 por Evan You e é mantido por uma comunidade ativa de desenvolvedores e empresas (Ninja do Linux, 2020). O Vue.js foi projetado para ser fácil de adotar e integrar em projetos existentes, permitindo que os desenvolvedores construam interfaces de usuário eficientes e reativas com facilidade.

A história do Vue.js começou com a visão de Evan You de criar um framework JavaScript que combinasse o melhor de outros frameworks populares, como Angular e React, mas com uma abordagem mais simples e flexível (Vue.js, s.d.). Ele lançou o Vue.js inicialmente como um projeto de código aberto em 2014 e, desde então, tem sido constantemente aprimorado e atualizado pela comunidade de desenvolvedores.

Uma das principais características do Vue.js é sua abordagem de componentes reativos, que permite a criação de interfaces de usuário altamente responsivas e dinâmicas. O Vue.js também oferece uma ampla gama de recursos, como suporte a templates, diretivas, reatividade, gerenciamento de estado e manipulação do DOM, tornando-o uma opção popular para o desenvolvimento de aplicações web modernas (Vue.js, s.d.)

O Vue.js tem sido amplamente adotado por empresas e desenvolvedores em todo o mundo para a construção de aplicações web, *single-page applications* (SPAs), aplicações de página múltipla e muitos outros tipos de projetos. Sua popularidade tem crescido rapidamente nos últimos anos devido à sua flexibilidade, desempenho e curva de aprendizado suave para os desenvolvedores.

3.1.5 Vuetify

Vuetify é um framework de design de código aberto para a construção de interfaces de usuário responsivas em aplicações web utilizando o Vue.js. Ele foi criado por John Leider e lançado em 2016 como uma biblioteca de componentes Vue.js com foco em design e usabilidade (Vuetify, s.d.). O Vuetify é mantido por uma comunidade ativa de desenvolvedores e designers, e é amplamente utilizado em projetos Vue.js em todo o mundo.

A história do Vuetify começou com a ideia de John Leider de criar uma biblioteca de componentes Vue.js que seguisse os princípios do ‘Material Design’, o guia de design do Google, para a criação de interfaces de usuário modernas e atraentes (Vuetify, s.d). Ele lançou o Vuetify inicialmente como um projeto de código aberto e, desde então, tem sido constantemente aprimorado e atualizado pela comunidade.

Uma das principais características do Vuetify é sua ampla gama de componentes prontos para uso, como botões, formulários, grids, cards, entre outros, que seguem as diretrizes do ‘Material Design’. Além disso, o Vuetify também oferece suporte a temas e estilos personalizáveis, permitindo que os desenvolvedores criem interfaces de usuário consistentes e estilizadas de forma fácil e rápida (Vuetify, s.d.).

O Vuetify tem sido amplamente adotado por desenvolvedores Vue.js para a construção de interfaces de usuário modernas e atraentes em aplicações web. Sua popularidade tem crescido rapidamente nos últimos anos devido ao seu design responsivo, usabilidade e facilidade de uso.

3.1.6 Visual Studio Code

O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Ele inclui suporte para depuração, controle de versionamento Git incorporado, realce de sintaxe, complementação inteligente de código, snippets e refatoração de código. É criado com Electron, uma ferramenta criada pelo GitHub que permite a criação de softwares Desktop com HTML, CSS e JavaScript (MICROSOFT, s.d.).

O Visual Studio Code é altamente personalizável e possui uma grande loja de extensões que permite adicionar funcionalidades adicionais ao editor (HANASHIRO, 2021).

3.2 Métodos

No desenvolvimento desta aplicação, o papel de ‘cliente’ foi desempenhado pelo próprio desenvolvedor, que mesmo enquanto atuava como DBA, desenvolveu uma ferramenta para solucionar os problemas que enfrentava no dia a dia. Isso permitiu que o desenvolvedor tivesse uma compreensão profunda das necessidades e desafios enfrentados por outros DBAs e pudesse criar uma solução eficaz para atender a essas necessidades.

Em termos de metodologia, a que mais se aplicou do desenvolvimento foi a XP (Extreme Programming). Essa metodologia enfatiza a colaboração entre os membros da equipe e a entrega contínua de pequenas melhorias no software. Isso permitiu que o desenvolvimento fosse ágil e flexível, permitindo que a ferramenta evoluísse de acordo com as necessidades do usuário (ROMBAOA, 2021).

Dessa forma, os requisitos foram sendo criados e ajustados ao longo do processo de desenvolvimento, sem a necessidade de um levantamento prévio de documentação e funcionalidades, permitindo que o desenvolvimento mais rápido e simplificado.

À posterior, podemos estabelecer quais foram os requisitos para o desenvolvimento da aplicação. Segundo ROSSETTI (2021), um software possui um conjunto de requisitos divididos entre funcionais: que dizem respeito às funções e informações que o software deve possuir, ou seja, ao seu comportamento; e não funcionais, que se referem aos critérios que qualificam os requisitos funcionais: estão relacionados a qualidades específicas e restrições que o software deve atender, ou seja, não se referem a funcionalidades em si, mas fazem parte do escopo do produto.

Como requisitos funcionais, se destacam:

- Possuir autenticação com Login e senha;
- Cada usuário deve ter um perfil de acesso, controlado pelo administrador;
- A aplicação deve fornecer um *dashboard* com dados do monitoramento;
- A aplicação deve fornecer listagens de usuários, eventos e variáveis da instância de Banco de Dados;

E como requisitos não-funcionais, se destacam:

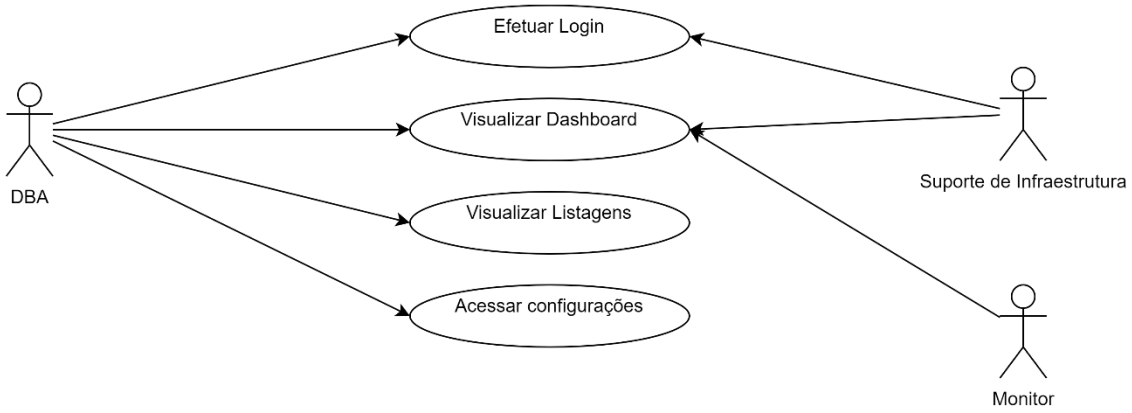
- A aplicação deve ser executada via navegador (web);
- A aplicação pode ser implantada em qualquer ambiente de Bancos de Dados já existente;
- Deve ser executada em segundo plano, não dependendo de comandos do DBA;
- Apresentar as informações coletadas pela aplicação em um painel de visualização do estado de funcionamento dos Bancos de Dados gerenciados;
- Fornecer avaliações sobre status, performance e prevenção dos Bancos de Dados.
- O código da aplicação deve ser disponibilizado em repositório público;

Para definir a interação com a aplicação, definem-se 3 atores, que de acordo com CREATELY (2021) são quaisquer entidades que desempenham um papel em um determinado sistema, podendo ser uma pessoa, organização ou um sistema externo.

Estes são:

- DBA: é o administrador responsável por controlar a aplicação. Possui acesso total e pode criar usuário e definir perfis de acesso dentro da aplicação.
- Suporte em Infraestrutura: responsável pelo funcionamento dos *hardwares* de servidores de bancos e aplicações dentro da organização. Deve possuir acesso para monitorar os bancos de dados no evento de queda dos serviços, reinicialização não programada, entre outras falhas técnicas.
- Monitor: no modelo proposto, o monitor é definido como uma tela que exibe constantemente o *dashboard* de monitoramento dos bancos de dados. Este pode ser um segundo monitor na estação de trabalho do DBA ou uma TV posicionada de forma a ser facilmente vista pelo DBA e outros colaboradores. Dessa forma, mesmo que o foco esteja em outra tarefa, qualquer mudança nos contadores do *dashboard* chamará a atenção e permitirá que problemas sejam identificados e resolvidos rapidamente.

Figura 2 – Diagrama de Caso de Uso



Fonte: elaborado pelo autor.

4 DESENVOLVIMENTO

A criação de um Dashboard de Monitoramento de Performance de um Banco de Dados MariaDB pode ser dividida em várias etapas. Primeiramente, é necessário realizar a configuração e o setup do ambiente de desenvolvimento utilizando o Node.js, um ambiente de execução de JavaScript amplamente utilizado no desenvolvimento de aplicações web (Melo, 2021). Para isso, é possível consultar a documentação oficial do Node.js para obter informações detalhadas sobre como instalar e configurar o ambiente de desenvolvimento (Node.js Foundation, s.d.).

Em seguida, é preciso utilizar o Vue.js, um framework JavaScript popular para a construção de interfaces de usuário reativas, para desenvolver a parte *front-end* do *dashboard*. O Vue.js permite a criação de componentes reutilizáveis e interativos, o que facilita a construção de interfaces de usuário dinâmicas e responsivas. Para isso, pode-se consultar a documentação oficial do Vue.js para aprender sobre os conceitos básicos e avançados do framework e como utilizá-lo para construir interfaces de usuário (Vue.js, s.d.).

Além disso, é possível utilizar o Vuetify, um framework de componentes de interface de usuário para Vue.js, para acelerar o desenvolvimento do *dashboard*. O Vuetify fornece uma ampla variedade de componentes pré-projetados, como botões, tabelas, gráficos e outros elementos de interface de usuário, que podem ser facilmente integrados ao projeto Vue.js (Vuetify, s.d.). É possível consultar a documentação oficial do Vuetify para aprender como utilizar seus componentes e personalizar o estilo do *dashboard*.

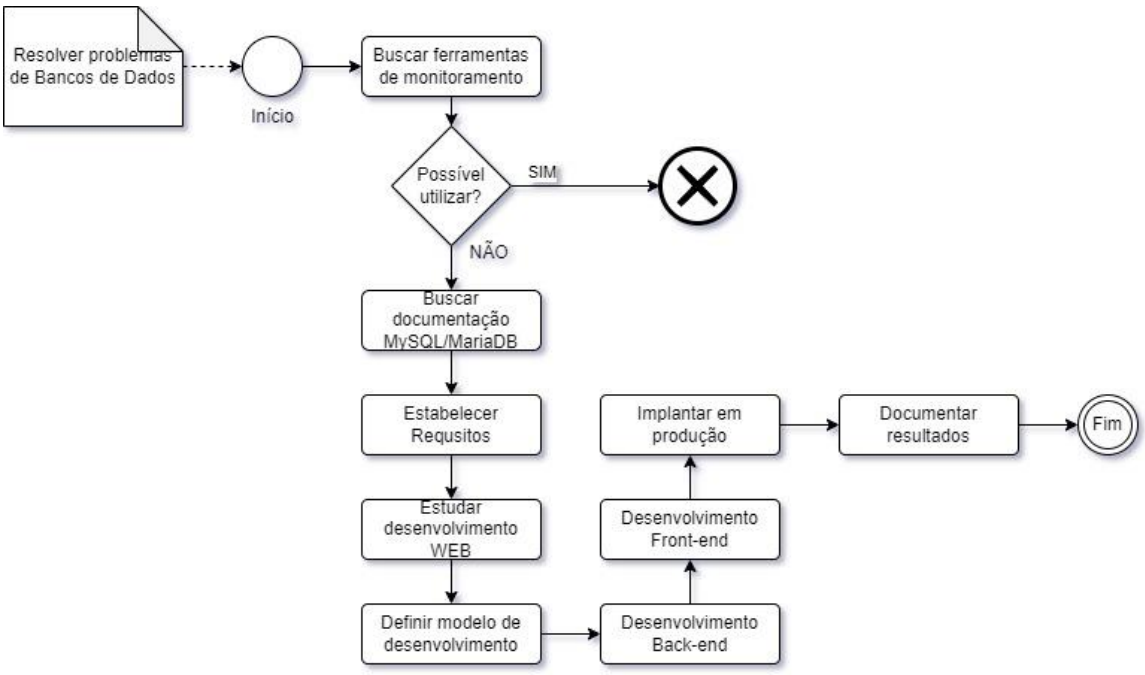
No desenvolvimento do *dashboard*, é importante também integrar a aplicação com o MariaDB, um sistema de gerenciamento de banco de dados relacional, para coletar e exibir as métricas de desempenho do banco de dados. Para isso, é possível utilizar bibliotecas e drivers JavaScript específicos para a conexão com o MariaDB, como o MariaDB Connector/Node.js, que permite a interação com o banco de dados utilizando

JavaScript (MariaDB, s.d.). É possível consultar a documentação oficial do MariaDB Connector/Node.js para aprender como realizar a conexão e a interação com o banco de dados MariaDB.

Por fim, é importante garantir a segurança da aplicação, implementando práticas de segurança recomendadas, como proteção contra ataques de injeção de SQL, autenticação e autorização adequadas, entre outras medidas de segurança relevantes para a aplicação web. É possível consultar boas práticas de segurança para desenvolvimento web em fontes confiáveis, como a OWASP (*Open Web Application Security Project*) (OWASP Foundation, s.d.).

Para auxiliar no desenvolvimento, foi elaborado um diagrama de atividade para ajudar a unir todas as áreas e entender o processo:

Figura 3 – Diagrama de Atividades para o desenvolvimento do projeto.



Fonte: elaborado pelo autor.

Para criar uma aplicação CRUD com Vue.js, Node.js, Express e MySQL, seguiremos os seguintes passos:

1. Crie um servidor *back-end* usando Node.js e Express para exportar APIs REST.
2. Configure o banco de dados MySQL e use um ORM como o Sequelize ou Knex para interagir com ele.
3. Crie um cliente front-end usando Vue.js com Vue Router e axios para enviar solicitações HTTP e recuperar respostas HTTP.
4. Use os componentes Vue e Vuetify para consumir os dados e exibi-los na interface do usuário.

Podem ser encontrados tutoriais mais detalhados sobre como criar uma aplicação CRUD com essas tecnologias em sites como BezKoder (BEZKODER, 2022) e LuizTools (DUARTE, 2020).

4.1 Back-End

4.1.1 Instalação de Recursos

É necessário acessar o site nodejs.org para realizar o download do arquivo de instalação apropriado para o sistema operacional. Após, instalar seguindo as instruções, e verificar se o Node foi instalado corretamente através do comando “node -v” no terminal ou prompt de comando (Node.js Foundation, sd.).

Para instalar o Yarn, é recomendável instalá-lo através do gerenciador de pacotes npm, que vem incluído com o Node.js quando você o instala em seu sistema. Depois de ter o npm instalado, você pode executar o seguinte comando para instalar e atualizar o Yarn: `npm install --global yarn` (Yarn, s.d.).

Yarn é um gerenciador de pacotes para Node.js que se concentra em velocidade, segurança e consistência. Ele foi criado originalmente para resolver alguns problemas com o popular gerenciador de pacotes npm. Embora os dois gerenciadores de pacotes

tenham convergido em termos de desempenho e recursos, o Yarn continua popular, especialmente no mundo do desenvolvimento React (BOUCHERON, 2021).

Por fim, criar um diretório e inicializar o Node.js App com o comando: “npm init”. Após isso será gerado o arquivo package.json no diretório (BEZKODER, 2022).

Figura 4 – Inicialização do projeto, nomeado de ‘backend-monitor’, com o comando ‘npm init’.

```
Press ^C at any time to quit.
package name: (backend-monitor)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to G:\GIT\backend-monitor\package.json:

{
  "name": "backend-monitor",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Fonte: elaborado pelo autor.

4.1.2 Instalação de Pacotes

Os pacotes necessários neste ponto serão ‘body-parser’, ‘consign’, ‘cors’, ‘dotenv’, ‘express’, ‘jsonwebtoken’, ‘knex’, ‘mysql’, ‘nodemon’ e ‘validate.js’. Podemos executar o seguinte comando: “npm install express knex mysql body-parser cors consign dotenv jsonwebtoken nodemon validate.js --save” (BEZKODER, 2022)

- **body-parser**: é um middleware de análise de corpo de requisição para Node.js. Ele analisa os corpos das requisições recebidas em um *middleware* antes de seus manipuladores, disponível sob a propriedade req.body. Como a forma do req.body é

baseada na entrada controlada pelo usuário, todas as propriedades e valores neste objeto são não confiáveis e devem ser validados antes de confiar (NPM, s.d.).

- **consign**: é um módulo que permite o carregamento automático de scripts. Ele foi criado para tornar os aplicativos mais fáceis de desenvolver com separação lógica de arquivos e carregamento automático de scripts. O consign pode ser usado para carregar automaticamente modelos, rotas, esquemas, configurações, controladores, mapas de objetos etc. (NPM, s.d.).
- **cors**: é um pacote Node.js que fornece um *middleware* Connect/Express que pode ser usado para habilitar o CORS com várias opções. CORS (Cross-Origin Resource Sharing) é um mecanismo que permite que muitos recursos em uma página da web sejam solicitados de outro domínio fora do domínio de origem (NPM, s.d.).
- **dotenv**: é um módulo sem dependências que carrega variáveis de ambiente de um arquivo '.env' para 'process.env'. Armazenar a configuração no ambiente separado do código é baseado na metodologia 'The Twelve-Factor App' (NPM, s.d.).
- **express**: é um framework web rápido, não opinativo e minimalista para Node.js. Ele fornece um conjunto robusto de recursos para aplicativos web e móveis, incluindo roteamento robusto, suporte a middleware e uma API HTTP simples (NPM, s.d.).
- **jsonwebtoken**: é uma implementação de JSON Web Tokens. Ele fornece uma maneira de criar e verificar tokens que podem ser usados para autenticar usuários ou transmitir informações de forma segura entre partes (NPM, s.d.).
- **knex**: é um construtor de consultas SQL para PostgreSQL, CockroachDB, MSSQL, MySQL, MariaDB, SQLite3, Better-SQLite3, Oracle e Amazon Redshift. Ele foi projetado para ser flexível, portátil e divertido de usar. Ele possui uma interface de promessa para um controle de fluxo assíncrono mais limpo, uma interface de fluxo, construtores de consulta e esquema completos, suporte a transações (com pontos de

salvamento) e respostas padronizadas entre diferentes clientes e dialetos de consulta (NPM, s.d.).

- **mysql**: é um módulo Node.js que permite a interação com bancos de dados MySQL. Ele fornece uma interface simples para executar consultas e gerenciar conexões com o banco de dados (NPM, s.d.).
- **nodemon**: é uma ferramenta que ajuda a desenvolver aplicativos baseados em Node.js reiniciando automaticamente o aplicativo node.js quando são detectadas alterações de arquivo no diretório (NPM, s.d.).
- **validate.js**: é uma biblioteca de validação de formulários leve em JavaScript inspirada pelo CodeIgniter. Ele fornece uma maneira simples de validar formulários em navegadores e pode ser usado com browserify (NPM, s.d.).

Após a instalação das bibliotecas, é necessário criar o arquivo ‘index.js’ para configurar o servidor web Express (BEZKODER, 2022).

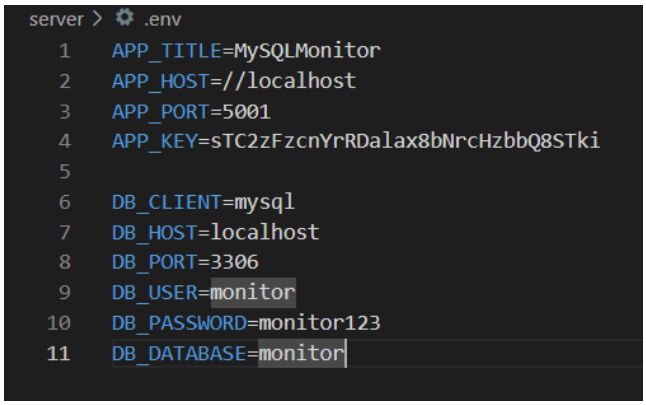
Figura 5 – Arquivo index.js para configuração do servidor web.

Fonte: elaborado pelo autor.

Neste arquivo fazemos a importação das bibliotecas (express, cors, etc), criamos uma constante app do Express e então adicionamos o ‘bodyParser’ e ‘cors’ pelo método ‘app.user()’. Criamos uma instância do Knex com a configuração especificada e a adicionamos à propriedade ‘db’ do app Express. Por fim, usamos o ‘consign’ para carregar rotas, controles e modelos na aplicação, e definimos em qual porta a aplicação vai receber requisições.

Uma vez instalada a biblioteca ‘dotenv’, vamos criar o arquivo ‘.env’ com as variáveis de ambiente:

Figura 6 – Arquivo de variáveis de ambiente.

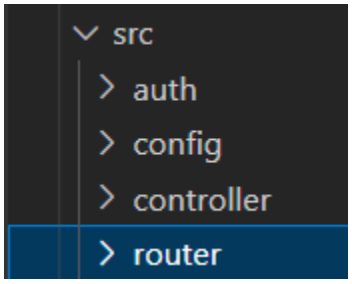


Fonte: elaborado pelo autor.

Um arquivo .env é um arquivo de texto simples que contém pares de chave e valor que representam as variáveis de ambiente. Cada linha do arquivo .env tem o formato VAR=VAL, onde VAR é o nome da variável e VAL é o seu valor (MOTDOTLA, s.d.). Utilizamos para armazenar os dados de host e porta do serviço web, de conexão com o banco de dados, entre outros.

Próximo passo é criar os diretórios onde ficarão os arquivos de autenticação, configuração do banco de dados, rotas e controles:

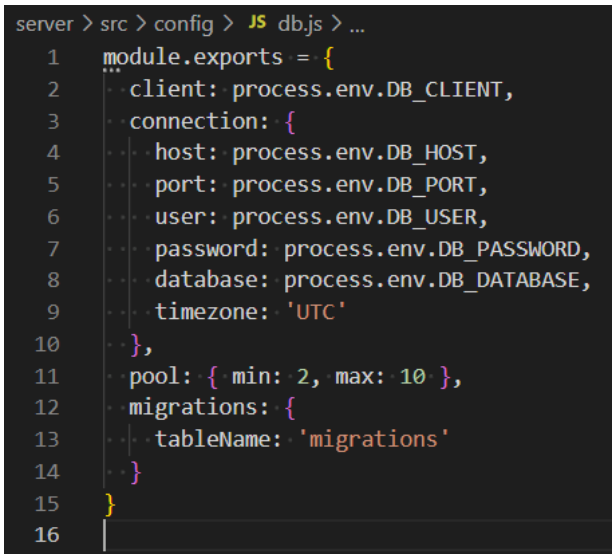
Figura 7 – Diretórios da aplicação.



Fonte: elaborado pelo autor.

Após isto, deve ser criada a configuração de Banco de Dados para uso da instância Knex. Criamos dentro do diretório ‘src’ o diretório ‘config’ e o arquivo ‘db.js’:

Figura 8 – Configuração do arquivo db.js.



Fonte: elaborado pelo autor.

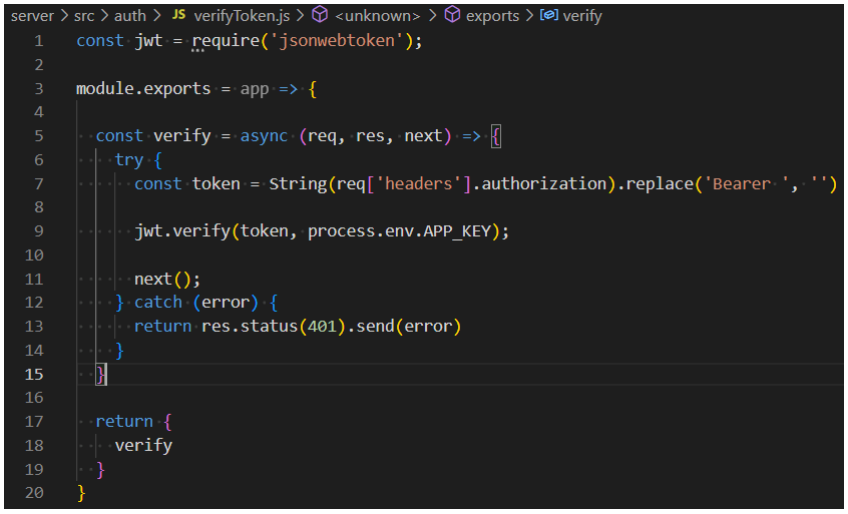
Neste arquivo exportamos um objeto que contém a configuração para uma instância Knex, com a propriedades: ‘client’ é o SGBD que será utilizado, ‘connection’ possui os objetos necessários para realizar a conexão com o banco (porta, host, usuário, senha, etc), ‘pool’ é utilizado para especificar o número de conexões e ‘migrations’ o objeto que especifica uma tabela para armazenar a estrutura do banco no caso de uma migração de servidor. Veja que as propriedades estão sendo carregadas com os dados incluídos no arquivo .env, possibilitando fácil alteração.

Como mostrado na Figura 5, também dentro da pasta ‘src’ criamos os diretórios ‘auth’, para funções de login, ‘controller’ para todas as funções da aplicação e ‘router’ para configuração das rotas.

Dentro do diretório ‘auth’, criamos o arquivo ‘login.js’, onde exportamos a função para validar se o usuário existe registrado no sistema e gerar um token via ‘jsonwebtoken’ de autenticação para realizar as requisições do front-end com o back-end. Também são criadas as funções iniciais como carregar os menus cadastrados para o usuário logado.

No mesmo diretório criamos o arquivo ‘verifyToken.js’, exportando apenas uma função ‘verify’, para utilizarmos a biblioteca ‘jsonwebtoken’ com o modo de autenticação ‘bearer’ para validar as requisições ao *back-end* (JWT.IO, s.d.). É usada uma chave específica da aplicação, armazenada no arquivo ‘.env’ para validar o token (MOTDOTLA, s.d.).

Figura 9 – Configuração do arquivo verifyToken.js.

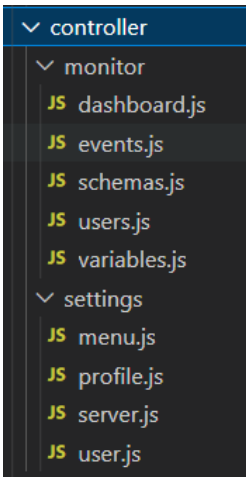


Fonte: elaborado pelo autor.

No diretório ‘router’, dever ser criado o arquivo ‘index.js’ onde ficarão definidas todas as requisições, usando métodos do objeto app do Express.js que correspondem aos métodos HTTP, como app.get(), app.post(), app.put(), etc (EXPRESS.JS, s.d.).

E no diretório ‘controller’, podemos dividir em mais dois diretórios as funções de configuração do sistema (usuário, menu, perfil etc.) e as funções que serão utilizadas para carregar os dados de monitoramento de bancos de dados.

Figura 10 – Arquivos do diretório ‘controller’.



Fonte: elaborado pelo autor

Com isto, podemos iniciar o serviço da aplicação web *back-end* executando o arquivo principal ‘index.js’com o comando “yarn start”. Este comando do gerenciador de pacotes Yarn é um atalho para executar o script definido no arquivo ‘package.json’ (gerado pelo comando “npm init”) (YARN, s.d.).

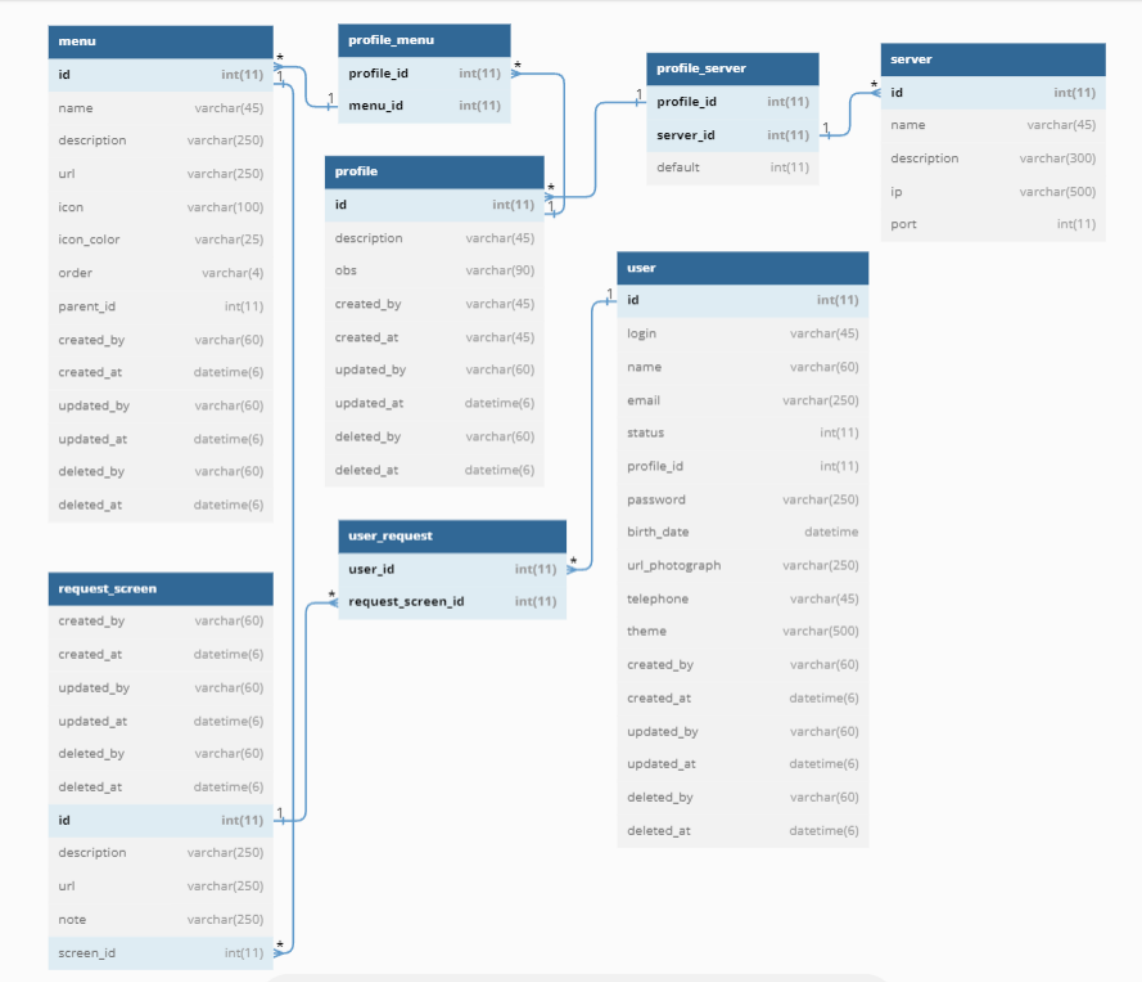
Como configurado no arquivo index.js do servidor web, ao gerar a instância do app, veremos a mensagem no console “Rodando na porta 5001”.

4.2 Banco de Dados

Será necessário a criação de 2 bancos de dados:

- Monitor: controla a autenticação de usuários no serviço de monitoramento, menus, perfis e cadastro de servidores monitorados. Pode estar em um servidor diferente do alvo do monitoramento.

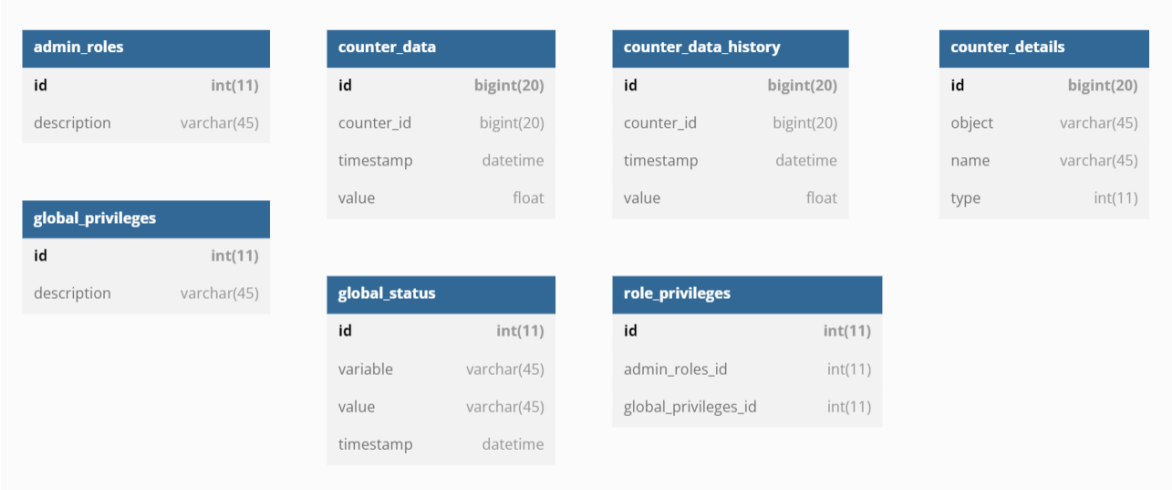
Figura 11 – Diagrama de entidade-relacionamento do Banco ‘monitor’



Fonte: elaborado pelo autor

- Audit: possui as tabelas que armazenam os dados captados pelo monitoramento e funções e procedures para gravação destes dados. Deve ser criado nos servidores alvos do monitoramento.

Figura 12 – Diagrama de entidade-relacionamento do Banco ‘audit’



Fonte: elaborado pelo autor

Também é necessário a criação de um usuário do banco, neste caso criaremos o usuário ‘monitor’, com direitos somente de acesso aos Schemas ‘monitor’ e ‘audit’.

Figura 13 – Comando de criação de um usuário do Banco de Dados para a aplicação.

```
CREATE USER 'monitor'@'%' IDENTIFIED BY 'vertmonitor23';

GRANT ALL PRIVILEGES ON audit.* to 'monitor'@'%' identified by "password" WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON monitor.* to 'monitor'@'%' identified by "password" WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

Fonte: elaborado pelo autor

Também é necessário criar procedures e funções para consultar e armazenar os dados do banco monitorado. Para isso devem ser consultas as documentações oficiais do MySQL, pode ser encontrada no site da Oracle em https://dev.mysql.com/doc/, e do MariaDB pode ser encontrada em https://mariadb.com/kb/en/. Nestas documentações existem referências para comandos SQL e funções.

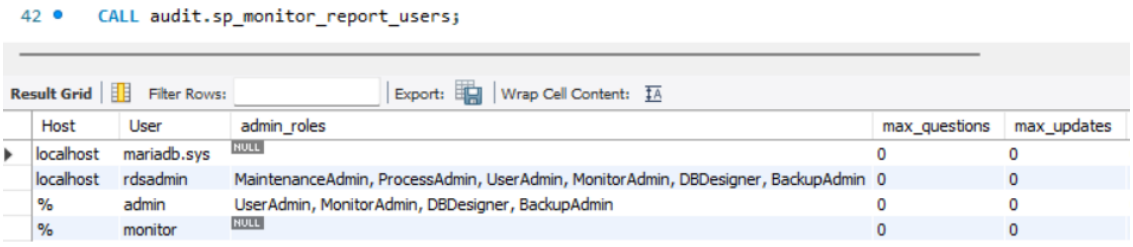
Existem 3 listagens que serão carregadas na aplicação para consulta dos DBAs responsáveis: usuários, eventos e variáveis.

4.2.1 Usuários

A tabela ‘mysql.user’ contém informações sobre usuários que têm permissão para acessar o servidor MySQL ou MariaDB e seus privilégios globais. No entanto, a partir do MariaDB 10.4, a tabela ‘mysql.global_priv’ substituiu a tabela ‘mysql.user’, e a tabela ‘mysql.user’ deve ser considerada obsoleta. Ela agora é uma visão da tabela ‘mysql.global_priv’ criada para compatibilidade com aplicativos e scripts de monitoramento mais antigos. Novas ferramentas devem usar as tabelas do ‘information_schema’ (MARIADB, s.d.).

Foi criado a procedure ‘sp_monitor_report_users’, que faz query na tabela ‘mysql.user’ para apresentar as colunas ‘user’, ‘host’ e ‘expired’ para identificar usuários com senha vencida. As informações de ‘roles’ do usuário (perfis de permissões) são consultadas através da função ‘fn_monitor_admin_roles’, que recebe parâmetros ‘user’ e ‘host’ para retornar dados da tabela ‘information_schema.USER_PRIVILEGES’.

Figura 14 – Output da procedure sp_monitor_report_users.



Fonte: print screen do MySQL Workbench após executar a procedure sp_monitor_report_users

4.2.2 Eventos

O banco de sistema ‘information_schema’ é um banco de dados virtual que contém informações sobre todos os bancos de dados em um servidor MySQL ou

MariaDB. Ele fornece acesso a metadados sobre os objetos do banco de dados, como tabelas, colunas e índices (MARIADB, s.d.).

A tabela EVENTS do ‘information_schema’ armazena informações sobre eventos no servidor. Você pode obter uma lista de eventos atuais com o comando SHOW EVENTS. Isso mostra apenas o nome e o intervalo do evento – os detalhes completos do evento, incluindo o SQL, podem ser vistos consultando a tabela EVENTS do ‘information_schema’ ou com o comando SHOW CREATE EVENT (MARIADB, s.d.).

Foi criado a procedure ‘sp_monitor_events’, que faz query na tabela ‘information_schema.events’ para apresentar as informações sobre os eventos cadastrados no Banco, tais como nome, tipo, intervalo, próxima execução, última execução e status.

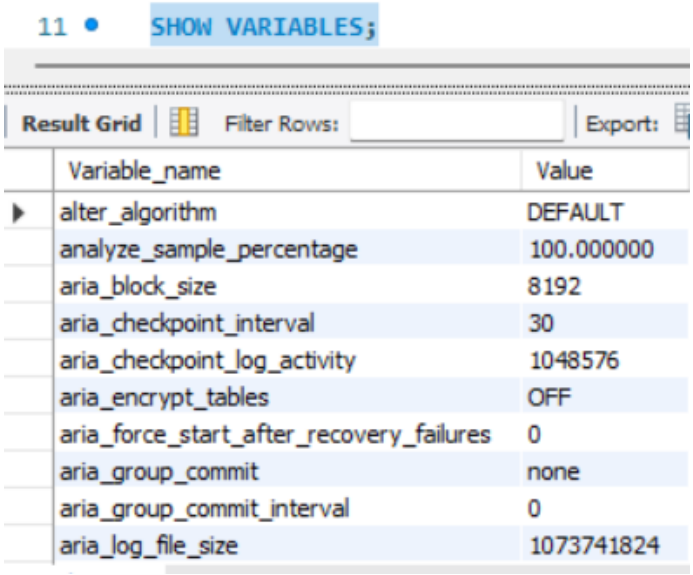
4.2.3 Variáveis

A tabela SYSTEM_VARIABLES do banco ‘information_schema’ mostra os valores atuais e vários metadados de todas as variáveis do sistema (MARIADB, s.d.).

Uma forma mais simples de listar as variáveis é através do comando ‘SHOW VARIABLES’, podendo ser filtrado (ex.: “SHOW VARIABLES LIKE '%maria%”) ou utilizar os modificadores ‘GLOBAL’ – para apresentar os valores utilizado para novas conexões, ou ‘SESSION’ – para os valores em efeito para a conexão atual.

As variáveis globais afetam o comportamento do servidor como um todo e são aplicadas a todas as conexões. Elas são inicializadas a partir das opções de configuração especificadas no arquivo de configuração do servidor ou na linha de comando ao iniciar o servidor. Já as variáveis de sessão afetam apenas a conexão atual e são inicializadas a partir dos valores das variáveis globais correspondentes quando a conexão for estabelecida (MARIADB, s.d.).

Figura 15 – Output do comando show variables.



Fonte: print screen do MySQL Workbench após executar o comando show variables

4.2.4 Funções do Dashboard

Para exibir dados no *dashboard* principal da aplicação, foram necessárias a criação de 9 procedures, que serão resumidas a seguir:

- sp_monitor_innodb_buffer_pool

Este procedure vai consultar as tabelas ‘information_schema.global_status’ para buscar os dados de ‘Innodb_buffer_pool’ e calcular o percentual de utilização de páginas de memória em tempo real.

O buffer pool do InnoDB é um componente chave para otimizar o MariaDB. Ele armazena dados e índices e, geralmente, você deseja que ele seja o maior possível para manter o máximo de dados e índices na memória, reduzindo a E/S de disco (MARIADB, s.d.).

- sp_monitor_host_info e sp_monitor_instance_info

Consultam a tabela ‘information_schema.global_variables’ para retornar dados do nome do *host*, versão do sistema operacional e da *engine* do Banco e tempo de execução do serviço.

- sp_monitor_connected_users_list

Consulta a tabela ‘information_schema.processlist’ para retornar a lista de usuários conectados no momento e o número de conexões abertas por cada um.

- sp_monitor_database_size

Calcula através da tabela ‘information_schema.tables’ o espaço em disco dos dados, índices e espaço livre de todos os bancos de dados no servidor.

- sp_monitor_process_list

Consulta a tabela ‘information_schema.processlist’ para retornar a lista de consultas sendo executadas na instância, retornando o tempo total de execução, usuário e comando sendo executado.

- sp_monitor_connections, sp_monitor_bytes e sp_monitor_statements

Apenas consultam as tabelas ‘audit.counter_data’ e ‘audit.counter_details’ para retornar as informações armazenadas de número de conexões, quantidade de bytes enviados e recebidos e número de consultas executadas.

As procedures ‘sp_counter_bytes’, ‘sp_counter_connections’ e ‘sp_counter_statements’ são executadas por eventos a cada determinado período para armazenar o histórico dos dados de bytes enviados/recebidos, conexões ativas e consultas realizadas.

4.3 Front-End

Para desenvolver uma aplicação web *front-end* com Vue.js, existem os seguintes passos:

1. Instalar o Vue.js e criar um projeto usando o Vue CLI.

2. Adicionar a biblioteca Vue-router para gerenciar a navegação entre as telas da aplicação.
3. Adicionar a biblioteca AXIOS ao projeto para fazer requisições HTTP para APIs e serviços web.
4. Adicionar o Vuetify.js para utilizar componentes de interface do usuário pré-construídas.

A instalação do Vue.js pode ser realizada pelo terminal ou prompt com o comando “npm install vue”. Para mais informações sobre a instalação, pode-se acessar o guia oficial em: https://br.vuejs.org/v2/guide/installation.html. Após isto, é necessário instalar o pacote Vue CLI, através do comando “npm install -g @vue/cli” para permitir executar comandos do Vue.js via terminal (VUE CLI, s.d.).

No diretório raiz do projeto, podemos executar o comando “vue create frontend-monitor” para gerar a estrutura da aplicação *front-end*.

É necessário instalar os demais pacotes que vamos utilizar, como Vue-router. É uma biblioteca que facilita a construção de aplicações de página única (*Single Page Applications*) com Vue.js. Algumas das características incluem mapeamento de rotas/visualizações aninhadas, configuração de roteador baseada em componentes e controle de navegação preciso (VUE ROUTER, s.d.). Executamos o comando no terminal: “npm install vue-router”.

Dentro do diretório ‘src’, vamos criar outro chamado ‘router’ e um arquivo ‘index.js’. Este será o router principal. Podemos ainda criar mais arquivos para separar as rotas das páginas de configurações e páginas do monitoramento de banco de dados.

No arquivo ‘index.js’, é importado e instalado o plugin do Vue-router, e dentro da nova instância são definidas as configurações das rotas principais de ‘login’, ‘início’ (tela inicial após o login) e erros 403 e 404 (acesso proibido e página não encontrada). Também na mesma instância VueRouter criada é adicionado um “gancho” global ‘beforeEach’ que é executado antes de cada navegação de rota para verificar se a rota é

permitida para a sessão atual e se existe um token armazenado válido gerado no login do usuário.

Figura 16 – Configuração do arquivo index.js do diretório ‘router’.

Fonte: elaborado pelo autor

Ainda é necessário importar o diretório ‘router’ no arquivo ‘src/main.js’ para que a instância do roteador possa ser usada na aplicação, tornando toda a aplicação ciente do roteador e permitindo que ele controle a navegação entre as rotas da aplicação (VUE ROUTER, s.d.).

A próxima instalação é do Axios, uma biblioteca cliente HTTP baseada em *Promises* que pode ser usada tanto em aplicações *front-end* quando em servidores Node.js. É uma maneira popular de fazer requisições HTTP para APIs e serviços web em aplicações Vue.js (VUEJS, s.d.). No terminal podemos executar o comando “npm install axios”.

Criaremos o diretório ‘src/plugins’, e o arquivo ‘axios.js’ para configurar a instância do Axios na aplicação. Definimos a URL base para todas as requisições que serão feitas, ou seja, o caminho para o serviço *back-end*. O token será sempre enviado nas requisições, dentro do cabeçalho de ‘Authorization’ do tipo ‘Bearer’. É importante implementar um interceptador de resposta para podermos tratar respostas e erros de requisições.

Figura 17 – Configuração do arquivo axios.js do diretório ‘plugins’.

client > src > plugins > JS axios.js > [@] Axios

```
import Vue from 'vue'
import axios from 'axios'
import router from '@/router'

const Axios = axios.create({
  baseURL: `${process.env.VUE_APP_HOST}:${process.env.VUE_APP_PORT}`,
  headers: {
    'Authorization': 'Bearer ' + localStorage.getItem('monitor-mysql:token')
  }
})

Axios.interceptors.response.use((response) => {
  if (response.data) {
    if (typeof response.data.mensagem === 'string') Vue.prototype.$notificacao(response.data.mensagem)
    else if (typeof response.data.erro === 'string') {
      if (response.data.tipo === 'Error') Vue.prototype.$notificacao(response.data.erro, 'erro')
      else Vue.prototype.$notificacao(response.data.erro, 'erro', true)
    } else if (typeof response.data.atencao === 'string') Vue.prototype.$notificacao(response.data.atenca
    else if (typeof response.data.informacao === 'string') Vue.prototype.$notificacao(response.data.infor
  }

  return response
}, (error) => {
  if (!error.response) {
    Vue.prototype.$notificacao('Erro ao processar requisição. Tente novamente!', 'erro')
  } else if (error.response.status === 401) {
    localStorage.removeItem('monitor-mysql:token')

    router.push('/login')
    Vue.prototype.$notificacao('Por Favor faça login novamente', 'atencao')
```

Fonte: elaborado pelo autor

Também vamos adicionar ao arquivo ‘src/main.js’ a importação do arquivo axios.js da seguinte forma: “import '@/plugins/axios'”. Assim, cada módulo da aplicação que precisar realizar requisições para o *back-end* poderá importar a configuração Axios que criamos utilizando: “import axios from '@/plugins/axios'”.

É necessário instalar a biblioteca Vuex.js para trabalharmos com um desenvolvimento modular, em que cada página da aplicação é tratada como um módulo independente, com seu próprio arquivo index.js e diretório store contendo os arquivos actions.js, mutations.js e state.js. Essa abordagem permite que cada página da aplicação gerencie seu próprio estado e lógica de forma independente, tornando o código mais organizado e fácil de manter (VUEX, s.d.).

Podemos usar o comando "npm install vuex@next –save" ou verificar a documentação oficial em https://vuex.vuejs.org/installation.html.

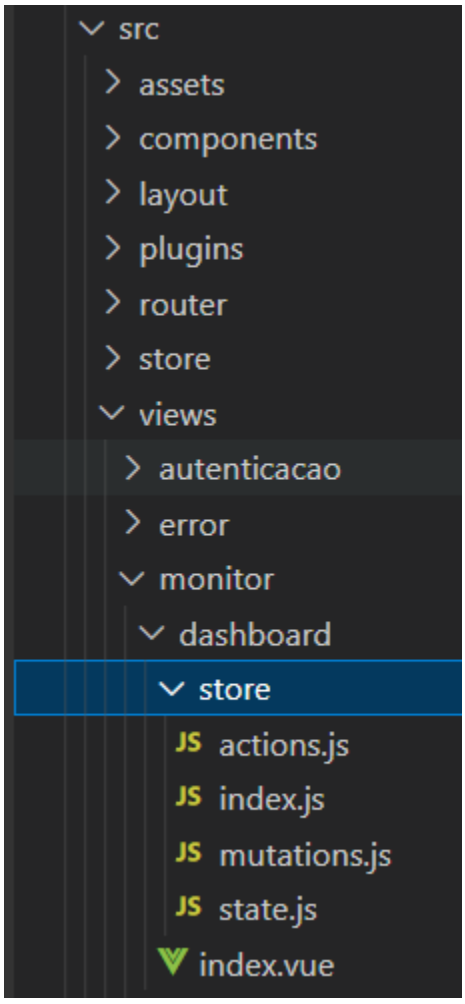
Criaremos a estrutura de pasta para os módulos da aplicação da seguinte forma:

- um diretório chamado 'src/views' para todas as páginas, tudo que é 'visual';
- dentro de views, separamos em 'autenticação' – páginas do login, 'monitor' – páginas para o monitoramento dos bancos de dados e 'settings' – páginas de configuração do sistema (usuários, acessos etc.);
- dentro de 'monitor' e 'settings', criamos um diretório para – agora sim – cada módulo do sistema, ex.: 'src/views/monitor/dashboard' para a página do *dashboard* principal do monitoramento, ou 'src/views/settings/user' para o cadastro de usuários da aplicação;
- dentro de cada módulo, teremos um arquivo 'index.js' para a página e um diretório 'store' para os arquivos 'actions.js', 'mutations.js' e 'state,js'.

O desenvolvimento modular com o Vuex permite gerenciar o estado da aplicação de forma centralizada, separando as responsabilidades e permitindo que o código seja mais organizado e fácil de entender, já que cada arquivo tem um propósito específico. O arquivo 'State' contém todas as informações que precisam ser compartilhadas entre os componentes do módulo. 'Mutations' define as funções síncronas que recebem o estado atual como primeiro argumento e qualquer payload adicional como segundo argumento, possibilitando alterar diretamente o estado do módulo. O arquivo 'Actions' define as ações que podem ser usadas para alterar o estado do módulo de forma assíncrona, como

por exemplo requisições ao *back-end* da aplicação, que chamam as mutações para alterar o estado do módulo. E o arquivo ‘index.js’ combina o estado, as mutações e as ações em um único módulo Vuex e exporta-o para ser usado na aplicação (VUEX, s.d.).

Figura 18 – Estrutura de diretórios do módulo ‘Dashboard’.



Fonte: elaborado pelo autor

Para configurar o módulo da página ‘dashboard’, é necessário configurar cada arquivo do diretório ‘store’. Começando pelo arquivo ‘actions.js’, configura-se as requisições do Axios à cada rota criada no *back-end* para trazer as informações para os componentes do *dashboard*.

Figura 19 – Arquivo actions.js do módulo dashboard, utilizando a instância Axios para fazer a requisição da rota 'monitor/dashboard/hostInfo'

client > src > views > monitor > dashboard > store > JS actions.js > ...

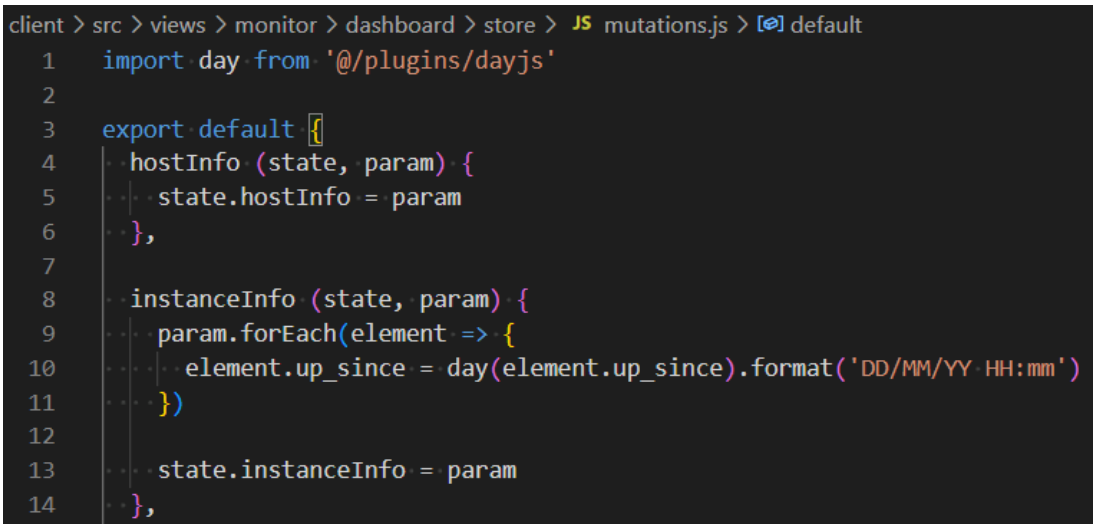
```
1  import axios from '@/plugins/axios'
2
3  export const listarHostInfo = async ({ commit }, filtro) => {
4    try {
5      const res = await axios.get('monitor/dashboard/hostInfo', {
6        params: {
7          ...filtro,
8          server_id: JSON.parse(window.atob(localStorage.getItem('monitor-mysql:server')))
9        }
10     })
11
12     if (!res.data.erro) {
13       commit('hostInfo', res.data)
14     }
15
16     return res.data
17   } catch (error) {
18     return error
19   }
20 }
```

Fonte: elaborado pelo autor

No arquivo 'index.js', combina-se 'state', 'mutations' e 'actions', importando cada arquivo e exportando como um único módulo Vuex.

No arquivo 'mutations.js', define as 'mutações' de cada estado que recebe o resultado da requisição feita no arquivo 'actions.js'. Podem ser realizadas formatações como por exemplo formatar uma data para outro padrão.

Figura 20 – Arquivo mutations.js do módulo dashboard



Fonte: elaborado pelo autor

Por último, o arquivo ‘state.js’ define as variáveis que conterão os dados das requisições com o *back-end* e que precisarão ser utilizadas entre os componentes do módulo e na página ‘dashboard’.

No arquivo ‘index.vue’, que é a página do módulo, é preciso importar as funções auxiliares ‘mapActions’, ‘mapState’ e o módulo ‘store’. Essas importações permitem que o componente acesse o estado e as ações do módulo Vuex “store” usando as funções auxiliares “mapState” e “mapActions” (VUEX, s.d.).

‘MapState’ será usada nas propriedades computadas para mapear as propriedades do estado do módulo Vuex para propriedades computadas do componente. E nos métodos do componente usa-se a função auxiliar ‘mapActions’ para mapear as ações do módulo Vuex para métodos do componente (VUEX, s.d.).

A estrutura do arquivo ‘index.vue’ que será usada será resumidamente (Vue.js, s.d.):

- Template: código HTML, onde serão construídos os componentes;
- Script: código JavaScript, dividido em:
    - Import: importa módulos ou valores para serem usados no componente.

- Export default: exporta um objeto como o componente Vue.js padrão, definindo o nome do componente.
- Data: define a função de dados do componente para retornar um objeto com as propriedades de dados iniciais.
- Computed: define as propriedades computadas do componente.
- Created: define um ‘gancho’ de ciclo de vida “created” que é executado quando o componente é criado.
- Methods: define os métodos do componente.

Repetindo o mesmo processo de criação de módulos, a aplicação terá módulos para controlar eventos, usuários e variáveis do Banco monitorado.

4.4 Componentes

Em uma aplicação web Vue.js, os componentes personalizados são blocos de construção reutilizáveis que podem ser criados para atender às necessidades específicas de um projeto. Eles podem ser registrados globalmente ou localmente e usados em qualquer lugar do aplicativo (VUEJS, s.d.).

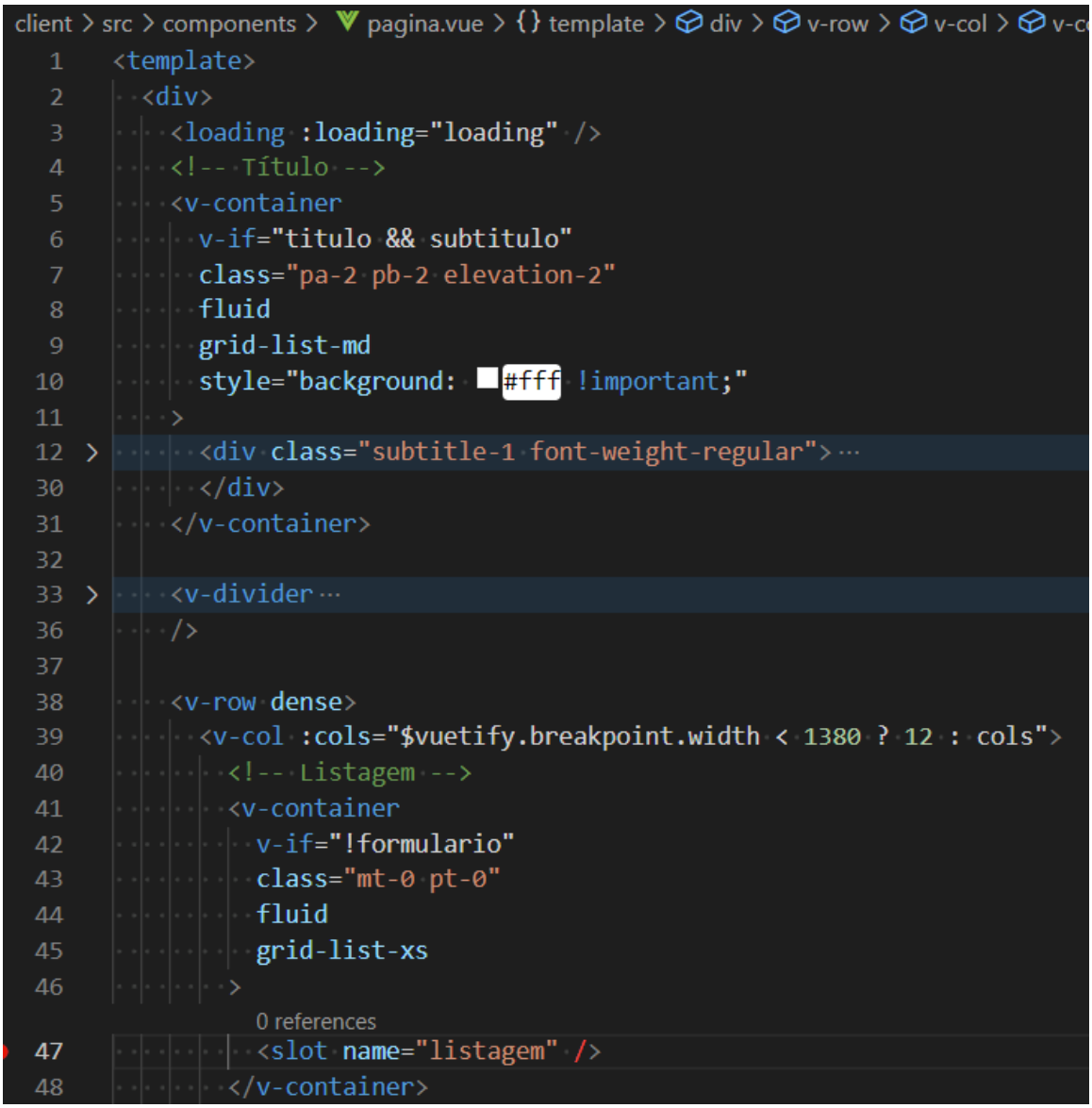
Ao utilizar um componente personalizado, podem ser configurados nele são atributos personalizáveis como título de um cabeçalho ou conteúdo de uma tabela, conhecidos como ‘props’. Quando um valor é passado para um atributo ‘prop’, ele torna-se uma propriedade daquela instância de componente (VUEJS, s.d.).

Outra forma de utilizar componentes é através de ‘slots’, que são uma maneira de distribuir conteúdo para componentes personalizados. Eles permitem que você crie um componente com espaços reservados para conteúdo personalizado, que pode ser preenchido pelo componente pai. Isso permite que você crie componentes mais flexíveis e reutilizáveis (VUEJS, s.d.).

Por exemplo, o componente ‘pagina.vue’ foi criado para ser usado em todos os formulários, criado o layout padrão de cabeçalho e rodapé. Este componente usa slots

para permitir que o componente pai forneça o conteúdo central da página web, que podem ser tabelas com dados para listagem, gráficos ou outros componentes personalizados.

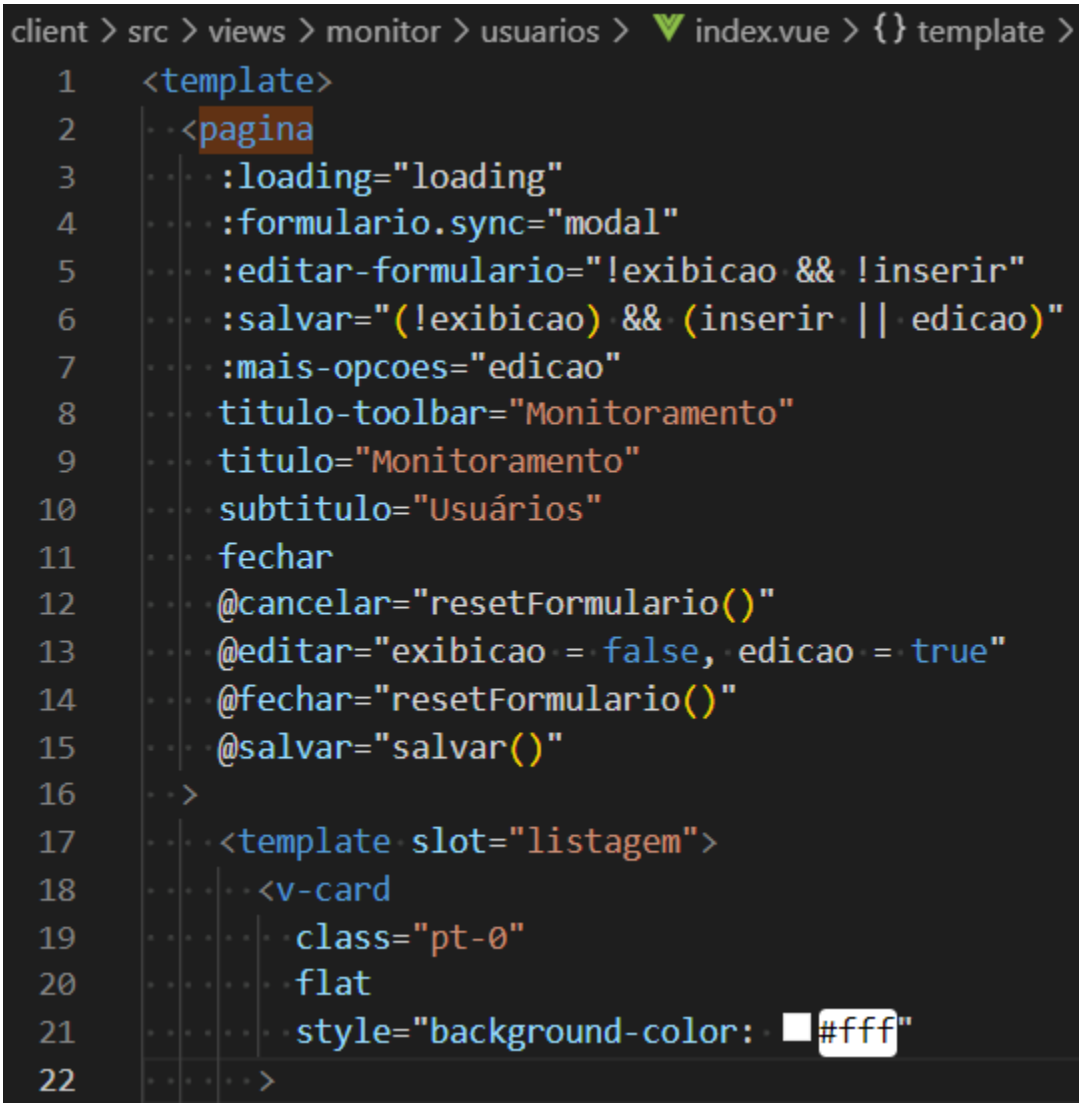
Figura 21 – Configuração do componente personalizado 'pagina.vue'



Fonte: elaborado pelo autor

Componentes personalizados podem ser importados localmente ou globalmente pelo arquivo ‘main.js’ com o seguinte código: “Vue.component('pagina', () => import('./components/pagina'))”. E para utilizar o componente, apenas utilize a tag html com o nome definido na importação e preencha com os ‘props’ e os ‘template slot’ definidos no componente.

Figura 22 – Utilização do componente ‘pagina’ no módulo de ‘usuarios’



Fonte: elaborado pelo autor

No módulo ‘dashboard’, além do componente página englobando o formulário, utiliza-se diversos componentes ‘v-card’ do Vuetify. Este é um componente versátil que pode ser usado para qualquer coisa, desde um painel até uma imagem estática. O componente card possui inúmeros componentes auxiliares para tornar a marcação tão fácil quanto possível. Um ‘v-card’ também pode conter vários componentes auxiliares, como ‘v-card-title’, ‘v-card-text’ e ‘v-card-actions’. Esses componentes ajudam a criar a estrutura do cartão e a organizar o conteúdo de maneira clara e legível (VUETIFYJS, s.d.).

Outro componente personalizado importante é o ‘tabela.vue’. É uma personalização do componente ‘v-data-table’ do Vuetify, que tem como função exibir dados tabulares (VUETIFYJS, s.d.). Ao personalizar, podem ser mudadas a aparência da tabela para se adequar ao estilo do aplicativo, novas funcionalidades podem ser adicionadas – como botões e caixas de seleção – e principalmente diminuir a quantidade de código que precisa ser reescrita para cada tabela na aplicação.

Para apresentação de gráficos, pode ser utilizado o ‘line-chart’ do Chart.js. É uma maneira de plotar pontos de dados em uma linha. É frequentemente usado para mostrar dados de tendência ou a comparação de dois conjuntos de dados (CHART.JS, s.d.). Foi utilizado nesta aplicação por ser altamente personalizável e oferecer várias opções para personalizar a aparência e o comportamento do gráfico.

Figura 23 – Utilização do componente personalizado ‘chartkick’ no módulo de ‘dashboard’

```
265            <v-col
266              id="Row2Col3"
267              cols="9"
268              lg="4"
269              md="5"
270              xs="6"
271            >
272              <chartkick
273                id="StatementsCount"
274                :dados="statementsCount"
275                :colors="colorsChart"
276                legend="top"
277                titulo="Statements"
278                altura="230px"
279                class="pa-0"
280              />
281            </v-col>
```

Fonte: elaborado pelo autor

4.5 Implantação

Para implantar a aplicação de monitoramento de Banco de Dados em um ambiente, é necessário um *host* de sistema Linux. Pode ter acesso à internet para acessar o repositório do GitHub, ou os arquivos serem movido manualmente. Você pode seguir os seguintes passos:

1. Assim como no desenvolvimento do *back-end*, efetue a instalação do Node.js.
2. É necessário clonar o repositório do GitHub em seu sistema Linux, então você precisará ter o Git instalado. Você pode instalar o Git com o comando ‘sudo apt-get install -y git’. Depois de instalar o Git, você pode clonar o repositório do GitHub em seu sistema Linux com o comando ‘git clone <URL_DO_REPOSITÓRIO>’ (GIT, s.d.).
3. Instale o ‘pm2’, gerenciador de processos para aplicações Node.js com um balanceador de carga integrado. Ele permite que você mantenha as aplicações

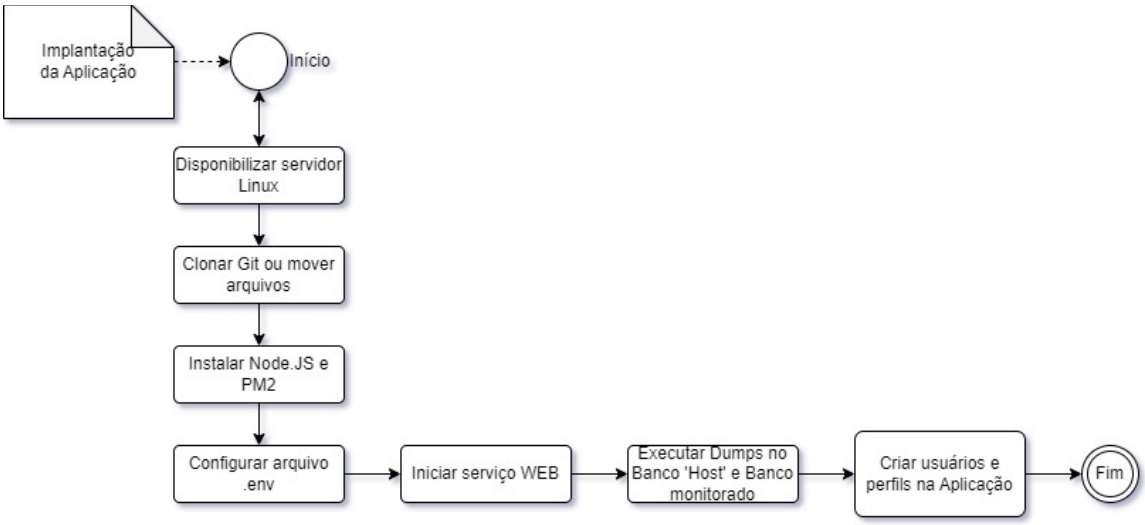
ativas para sempre, recarregue-as sem tempo de inatividade e facilite tarefas comuns de administração do sistema (NPM, s.d.).

   a. Para instalar o PM2, você pode usar o npm (gerenciador de pacotes do Node.js) com o seguinte comando: ‘npm install pm2 -g’. Isso instalará o PM2 globalmente em seu sistema (NPM, s.d.).
   b. Depois de instalado, você pode iniciar uma aplicação Node.js com o PM2 usando o comando ‘pm2 start app.js’, onde ‘app.js’ é o arquivo principal da sua aplicação. O PM2 cuidará de manter a aplicação ativa e reiniciá-la automaticamente em caso de falha (NPM, s.d.).

4. Configurar no arquivo ‘.env’ do diretório ‘server’ os dados corretos de *host* da aplicação, *host*, usuário e senha do Banco onde será armazenados os dados da aplicação (não confundir com o Banco de Dados que serão monitorados).
5. Utilizar os arquivos de *dump* disponibilizados no diretório do Github:
   a. ‘Audit – Dump.sql’ será executado nos servidores de Banco que serão monitorados
   b. ‘Monitor – Dump.sql’ será executado no servidor de Banco que vai controlar a aplicação de monitoramento.

A seguir pode ser visto em um diagrama de atividades o fluxo para implantação do sistema:

Figura 24 – Diagrama de atividades para implantação do sistema em um ambiente

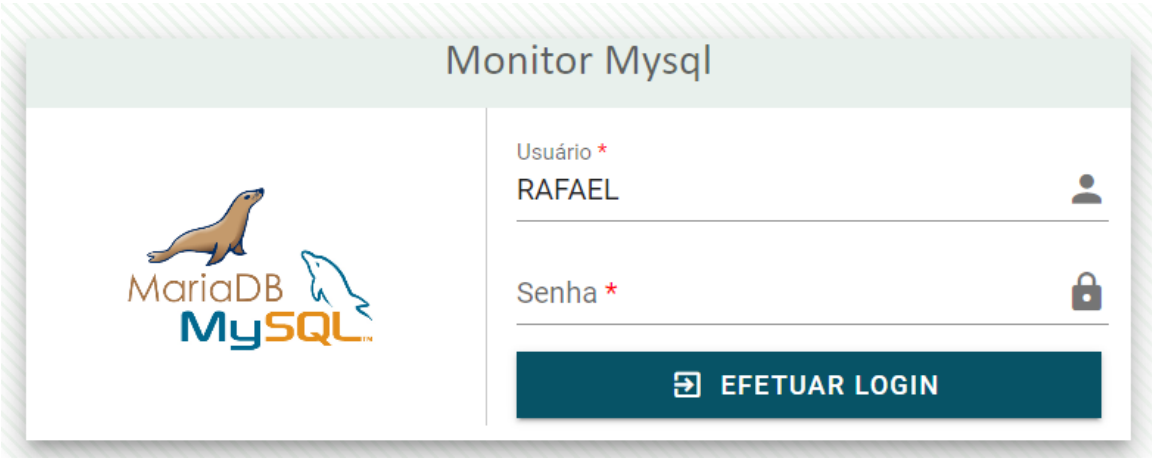


Fonte: elaborado pelo autor

5 RESULTADOS

A aplicação exige login para visualização dos dados de monitoramento do Banco, fornecendo um nível de segurança e permitindo que seja delegado a mais colaboradores o acesso aos dados de monitoramento dos Bancos (gerentes, responsáveis por suporte ou infraestrutura ou DBAs júnior).

Figura 25 – Tela de Login ao acessar o endereço IP do serviço da aplicação



Fonte: elaborado pelo autor

A tela inicial possui atalhos para as funcionalidades da aplicação. Foi escolhido desta forma, invés de abrir diretamente o *dashboard* principal para evitar carregamento desnecessário de dados do servidor.

Figura 26 – Tela inicial após login no Portal de Monitoramento. Possui os menus disponíveis para o usuário.

Fonte: elaborado pelo autor

5.1 Telas de Auxílio para o DBA

Foram criadas 3 listagens de auxílio para visualização de dados do Banco: Usuários, Variáveis e Eventos. Estas não fazem parte de monitoramento de performance, mas permitem a visualização de dados sem a necessidade de utilizar uma ferramenta de consulta de banco de dados e ou scripts complexos salvos pelo DBA.

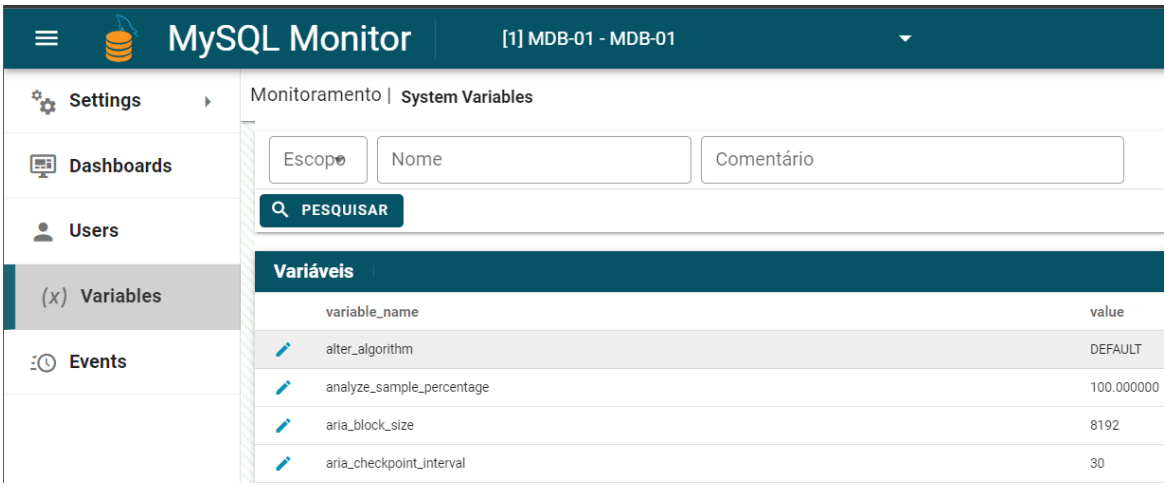
A listagem de usuários permite visualizar os usuários cadastrados na instância de banco de dados, quais são as '*roles*' atribuídas para este usuário e se o login não está expirado. Com esta informação fácil é possível notar problemas de segurança, como usuários que foram cadastrados sem permissão, ou possuem acessos superiores ao necessário, ou até mesmo identificar o motivo de alguma aplicação estar com problemas de acesso.

Figura 27 – Tela de listagem de Usuários cadastrados no Banco de Dados

Fonte: elaborado pelo autor

A listagem de variáveis do banco de dados, da mesma forma evita scripts e consultas manuais, e permite visualizar as configurações globais ou por sessão da instância. Estas variáveis são usadas para controlar o comportamento do servidor e o ajuste e ativação/desativação de variáveis fazem parte da configuração do serviço de banco de dados, seja na implementação inicial ou melhorias de performance.

Figura 28 – Tela de listagem de Variáveis da Instância de Banco de Dados

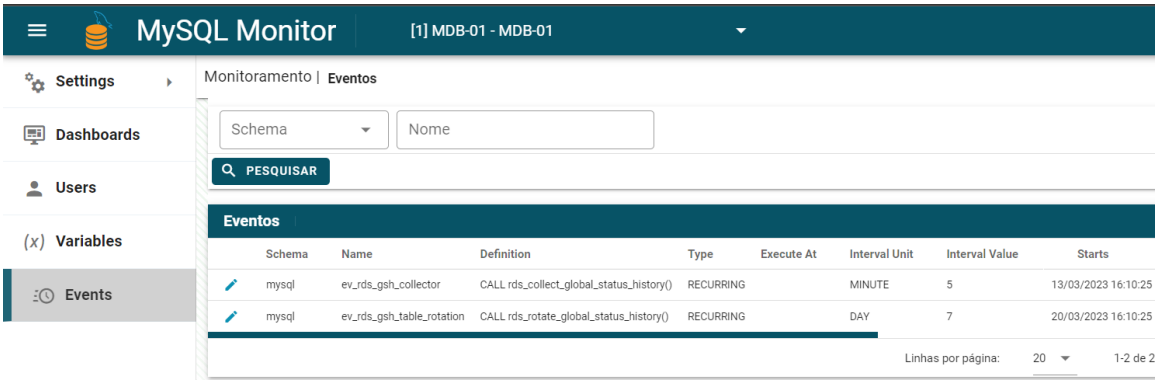


Fonte: elaborado pelo autor

Por fim, a listagem de eventos permite ao DBA visualizar as tarefas agendadas que são executadas em um horário específico ou em intervalos regulares. Elas são

semelhantes aos trabalhos do ‘cron’ no Unix ou às tarefas agendadas no Windows e podem ser usados para executar comandos SQL ou procedimentos armazenados em um horário específico. No caso de estarem desabilitadas, ou com horários conflitantes, o responsável pelo servidor pode identificar de maneira rápida e corrigir ou reestabelecer o evento.

Figura 29 – Tela de listagem de Eventos cadastrados na Instância de Banco de Dados

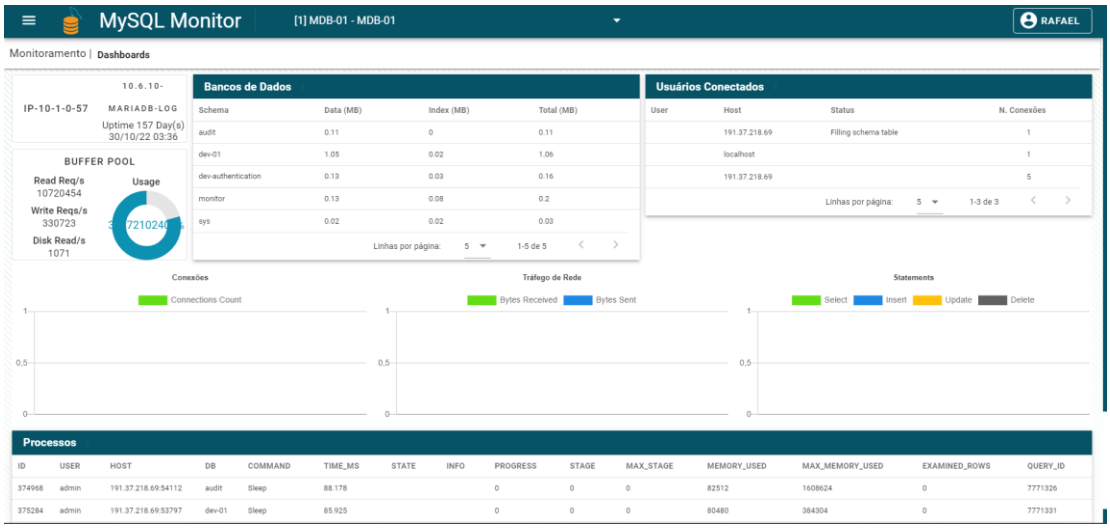


Fonte: elaborado pelo autor

5.2 Dashboard

Para o dashboard principal, foi adicionado vários quadros com informações sobre diversos aspectos do Banco de Dados monitorado.

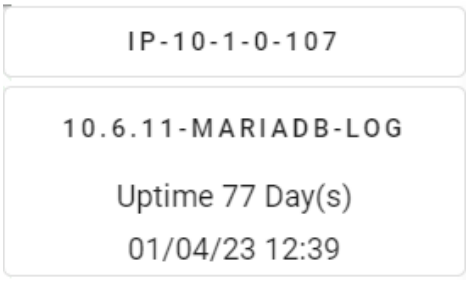
Figura 30 – Tela principal de Monitoramento do banco de dados, com todos os KPIs desenvolvidos.



Fonte: elaborado pelo autor

Para diferenciação entre possíveis múltiplos ambientes monitorados, o *hostname* do servidor está no primeiro quadro. Abaixo, existem as informações de Versão do Banco de Dados e *uptime*, destacando a data e hora em que o serviço do banco foi iniciado pela última vez.

Figura 31 – Cartões de 'hostname', versão e 'uptime' da Instância



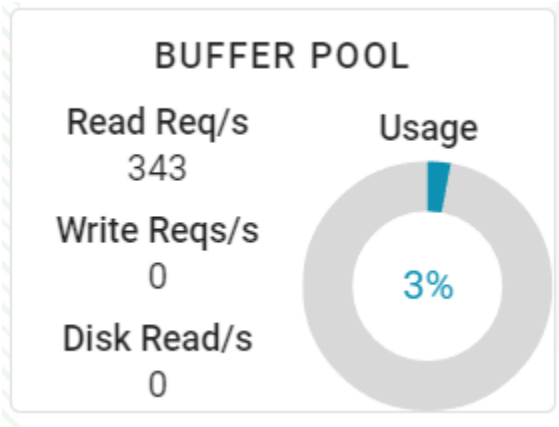
Fonte: elaborado pelo autor

O próximo quadro exibe uma visão geral do Buffer Pool do InnoDB e da atividade de disco gerada pelo mecanismo de armazenamento InnoDB. Possui o número

de solicitações de leitura e escrita lógica (por segundo) e o número de leituras lógicas que o InnoDB não pôde satisfazer a partir do buffer pool e teve que ser lido do disco.

Essas informações podem ajudar o DBA a entender como o mecanismo de armazenamento InnoDB está usando o buffer pool e interagindo com o disco, e podem fornecer *insights* sobre possíveis gargalos ou problemas de desempenho.

Figura 32 – Cartão de Uso do Buffer Pool da Instância



Fonte: elaborado pelo autor

O próximo cartão de informações é uma tabela com a listagem de bancos de dados da instância e o tamanho em disco do arquivo de dados, arquivo de índice e total, permitindo visualizar rapidamente o crescimento indevido de um ou mais bancos de dados.

Figura 33 – Cartão de espaço em disco dos Bancos de Dados da Instância

Bancos de Dados

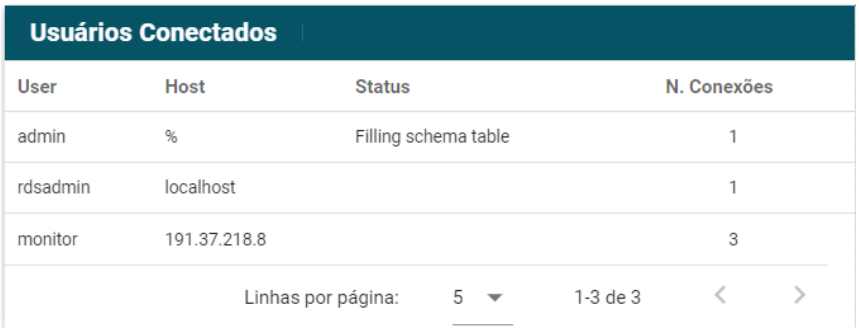
Schema	Data (MB)	Index (MB)	Total (MB)
audit	0.11	0	0.11
dev-01	1.47	0.05	1.52
dev-authentication	0.14	0.03	0.17
monitor	0.13	0.08	0.2
sys	0.02	0.02	0.03

Linhas por página: 5 1-5 de 5

Fonte: elaborado pelo autor

A próxima tabela mostra a listagem de usuários conectados no momento e o número de conexões abertas por cada um. Isto pode ajudar a entender alguma aplicação com multiplicas conexões precisando ser otimizada.

Figura 34 – Cartão de usuários conectados na Instância



Usuários Conectados

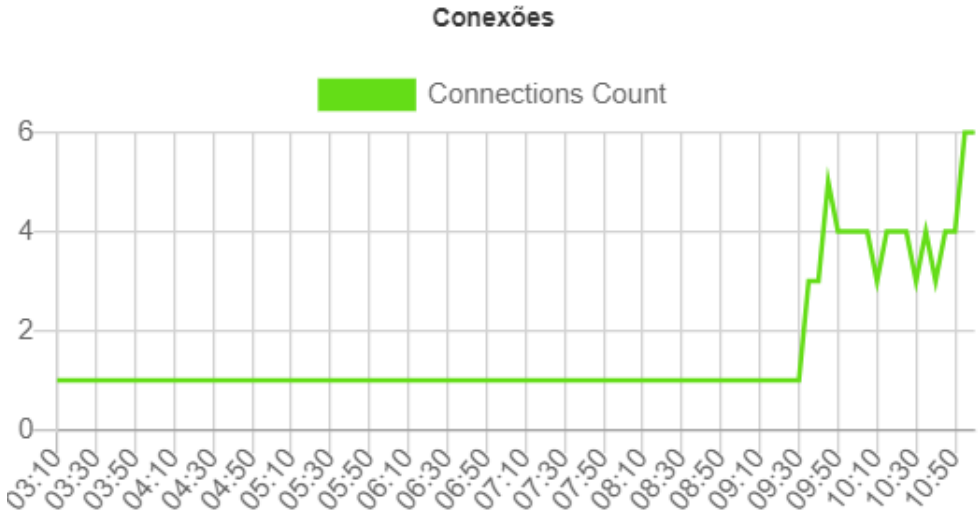
User	Host	Status	N. Conexões
admin	%	Filling schema table	1
rdsadmin	localhost		1
monitor	191.37.218.8		3

Linhas por página: 5 1-3 de 3

Fonte: elaborado pelo autor

Em seguida, foram adicionados três gráficos. No primeiro, é apresentado uma *timeline* do número de conexões de cliente atualmente abertas com o servidor. Esse valor pode ser útil para monitorar a carga do servidor e ajudar a identificar possíveis problemas de desempenho relacionados ao número de conexões.

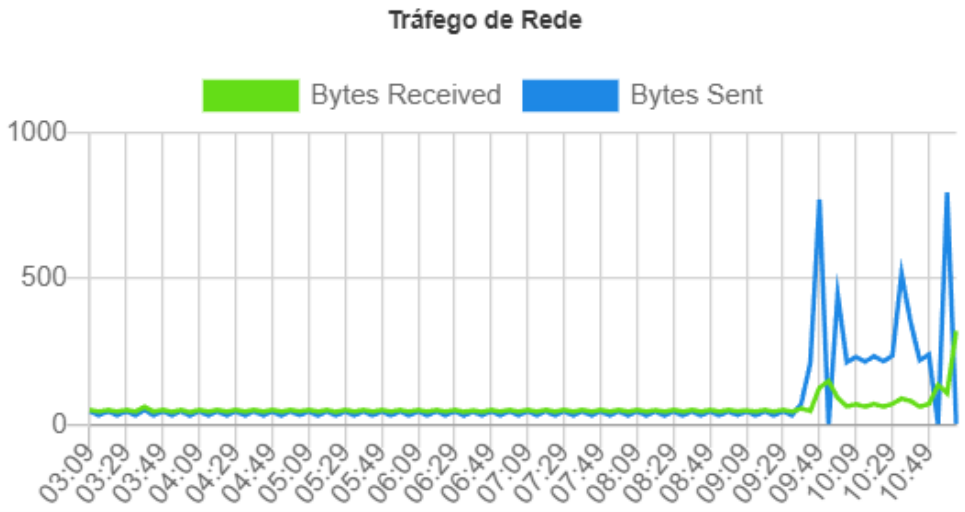
Figura 35 – Cartão de número de conexões na Instância



Fonte: elaborado pelo autor

No segundo gráfico, é indicado também em linha do tempo a quantidade total de dados recebidos e enviados pelo servidor em bytes, respectivamente. Esses valores podem ser úteis para monitorar o tráfego de rede do servidor e ajudar a identificar possíveis problemas de desempenho relacionados ao uso da rede.

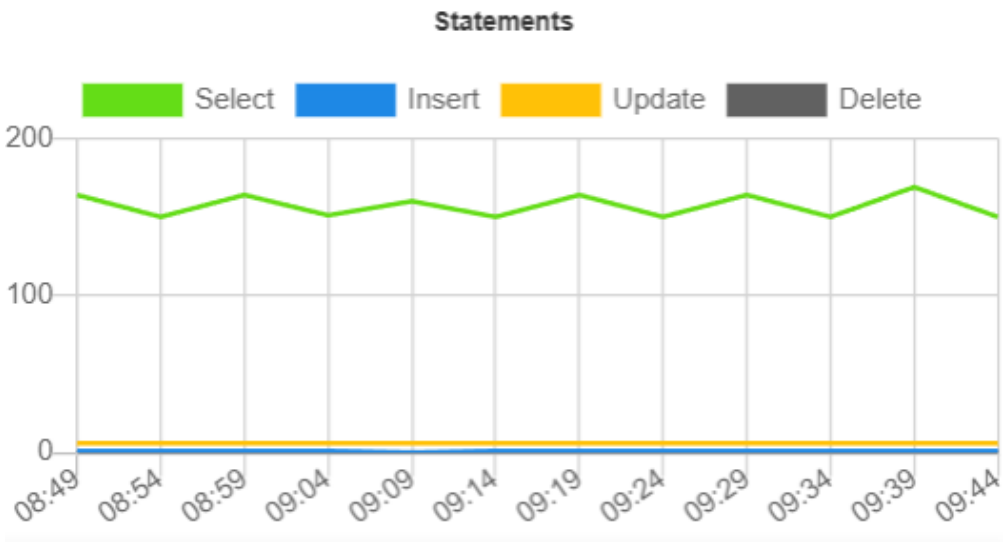
Figura 36 – Cartão de tráfego de rede, separado em 'bytes received' e 'bytes sent'



Fonte: elaborado pelo autor

O último gráfico apresenta o número total de comandos SELECT, INSERT, UPDATE e DELETE executados pelo servidor. Esses valores podem ser úteis para monitorar a atividade do servidor e ajudar a identificar execuções inesperadas de comandos em lote ou execuções indevidas de múltiplas alterações/exclusões de registros.

Figura 37 – Cartão de número de requisições ao banco, separado por consultas, inserções, alterações e exclusões



Fonte: elaborado pelo autor

A última informação do dashboard é a listagem de processos atualmente em execução no servidor MySQL ou MariaDB. Cada linha da tabela representa um processo, com informações como o ID do processo, o usuário que iniciou o processo, o tempo de execução do processo e o comando atualmente em execução pelo processo.

Apresentar essa informação pode ser útil para monitorar a atividade do servidor e ajudar a identificar possíveis problemas de desempenho de consultas ou bloqueios de comandos em execução no servidor.

Figura 38 – Cartão de listagem de processos executados na instância

Processos

ID	USER	HOST	DB	COMMAND	TIME_MS	STATE	INFO	PROGRESS	STAGE	MAX_STAGE	MEMORY_USED
328898	event_scheduler	localhost		Daemon	43165.204	Waiting for next activation		0	0	0	44392
332413	admin	191.37.218.8:17044	audit	Sleep	221.507			0	0	0	80528
331551	admin	191.37.218.8:17039	audit	Sleep	221.303			0	0	0	80528
320165	rdsadmin	localhost	mysql	Sleep	12.608			0	0	0	81600
331553	monitor	191.37.218.8:16445	monitor	Sleep	0.773			0	0	0	80288

Fonte: elaborado pelo autor

6 CONCLUSÃO

Neste trabalho, foi apresentada uma solução efetiva para o monitoramento em tempo real de desempenho de bancos de dados MySQL, com o desenvolvimento de um *dashboard* que permite a visualização de informações relevantes de maneira clara e objetiva. Através das pesquisas realizadas, foi possível identificar as principais ferramentas e técnicas utilizadas para o monitoramento de bancos de dados e as métricas mais relevantes para avaliar o desempenho de um banco de dados MySQL.

A solução proposta possibilita ao DBA agir de forma preventiva, evitando possíveis falhas e garantindo a estabilidade do banco de dados. Além disso, o uso de tecnologias modernas como Node.js, Vue e Vuetify tornou possível o desenvolvimento de um *dashboard* interativo e responsivo, que pode ser facilmente personalizado e adaptado às necessidades específicas de cada empresa.

Em resumo, a solução apresentada neste trabalho demonstrou ser eficiente e capaz de atender às necessidades de monitoramento em tempo real de desempenho de bancos de dados MySQL, contribuindo para a melhoria da qualidade e estabilidade dos sistemas que dependem desses bancos de dados.

Trabalhos futuros poderiam incluir a expansão da solução para suportar outros sistemas de gerenciamento de banco de dados além do MySQL/MariaDB, como MSSQL e PostgreSQL. Outra possibilidade seria a implementação de recursos adicionais, como alertas e notificações em tempo real, para auxiliar o DBA na identificação e resolução rápida de problemas.

Além disso, como a aplicação vai ficar disponível em repositório público, mais desenvolvedores poderão alterar funcionalidades, corrigir falhas ou criar módulos de monitoramento, de acordo com suas necessidades. Essas melhorias e expansões têm o potencial de tornar a solução ainda mais eficiente e útil para o monitoramento em tempo real de desempenho de bancos de dados.

REFERÊNCIAS

MariaDB Foundation. (sd). MariaDB Foundation. Disponível em: https://mariadb.org/about. Acesso em: 27/02/2023.

Melo, D. (s.d.). O que é Node.js? [Guia para iniciantes]. Tecnoblog. Disponível em: https://tecnoblog.net/responde/o-que-e-node-js-guia-para-iniciantes/. Acesso em: 27/02/2023.

Node.js Foundation. (s.d.). Documentation. Node.js. Disponível em: https://nodejs.org/en/docs/. Acesso em: 27/02/2023.

Node.js Foundation. (s.d.). About. Node.js. Disponível em: https://nodejs.org/en/about/. Acesso em: 27/02/2023.

Dahl, R. (2009). Original Node.js presentation [Vídeo]. YouTube. Disponível em: https://www.youtube.com/watch?v=ztspvPYybIY. Acesso em: 27/02/2023.

Ninja do Linux. (2020). Vue.js - O framework leve | Introdução. Disponível em: http://ninjadolinux.com.br/vue-js/. Acesso em: 27/02/2023.

Vue.js. (s.d.). Introduction. Disponível em: https://vuejs.org/guide/introduction.html. Acesso em: 27/02/2023.

Vue.js. (s.d.). Wikipedia. Disponível em: https://en.wikipedia.org/wiki/Vue.js. Acesso em: 28/02/2023.

Vuetify. (s.d.). Meet the team. Disponível em: https://vuetifyjs.com/en/introduction/meet-the-team/. Acesso em: 28/02/2023.

Vuetify. (s.d.). Components. Disponível em: https://vuetifyjs.com/en/components/all/. Acesso em: 28/02/2023.

DevMedia. (s.d.). Administrador de Banco de Dados (DBA). Disponível em: https://www.devmedia.com.br/administrador-de-banco-de-dados-dba/6897. Acesso em: 04/03/2023.

Felipe Coradin. (2023). 10 motivos para monitorar o ambiente de Banco de Dados. Disponível em: https://horusinfo.com.br/10-motivos-para-monitorar-o-banco-de-dados/. Acesso em: 04/03/2023.

FWC. Gestão proativa de banco de dados: qual importância para a eficiência da sua empresa? (2020). Disponível em: https://www.fwc.com.br/blog/banco-de-dados/monitoramento-de-banco-de-dados-qual-importancia-para-a-eficiencia-da-sua-empresa/. Acesso em: 05/04/2023.

Grafana Labs. (s.d.). Grafana. Recuperado de https://grafana.com/. Acesso em: 05/04/2023.

MICROSOFT LEARN. (2022). Troubleshoot slow-running queries on SQL Server. Disponível em: https://learn.microsoft.com/en-us/troubleshoot/sql/database-engine/performance/troubleshoot-slow-running-queries/. Acesso em: 05/04/2023.

Jankov, T. (s.d.). MariaDB vs MySQL: Um Resumos de Tecnologias de Base de Dados. Kinsta. Disponível em: https://kinsta.com/pt/blog/mariadb-vs-mysql/. Acesso em: 08/04/2023.

MariaDB. (s.d.). Getting Started With the Node.js Connector. Disponível em: https://mariadb.com/kb/en/getting-started-with-the-nodejs-connector/. Acesso em: 08/04/2023.

OWASP Foundation (s.d.). Disponível em: https://owasp.org/about/. Acesso em: 08/04/2023.

Elias Praciano (2014). MySQL: ferramentas de monitoramento. Disponível em: https://elias.praciano.com/2014/03/mysql-ferramentas-de-monitoramento/. Acesso em: 08/04/2023.

MySQL. (s.d.). MySQL Workbench: Performance. Disponível em: https://www.mysql.com/products/workbench/performance/. Acesso em: 08/04/2023.

MySQL. (s.d.). MySQL Workbench. Disponível em: https://www.mysql.com/products/workbench/. Acesso em: 08/04/2023.

Lo Duca, A. (s.d.). My Journey to Connect MariaDB to MySQL Workbench. Towards Data Science. Disponível em: https://towardsdatascience.com/my-journey-to-connect-mariadb-to-mysql-workbench-2d7e599a8a26/. Acesso em: 08/04/2023.

SolarWinds. (s.d.). MySQL Monitor - Ferramenta de monitoramento do desempenho do MySQL. Disponível em: https://www.solarwinds.com/pt/database-performance-monitor/integrations/mysql-monitoring. Acesso em: 17/04/2023.

Saveincloud. (2021). Otimização, replicação e tuning dos bancos MySql e MariaDB. Disponível em: https://saveincloud.com/pt/blog/banco-de-dados/tuning-e-performance-do-mysql-mariadb. Acesso em: 17/04/2023.

DevTrends. (s.d.). SQL: Monitoramento e Performance. Disponível em: https://dtrends.com.br/sql-monitoramento-e-performance/. Acesso em: 17/04/2023.

Oracle. (s.d.). Avaliar o Desempenho de Seus Bancos de Dados Instantaneamente. Disponível em: https://docs.oracle.com/pt-br/iaas/database-management/doc/assess-performance-your-databases-glance.html. Acesso em: 17/04/2023.

SolarWinds. (s.d.). Database Performance Monitor (DPM). Disponível em: https://www.solarwinds.com/pt/database-performance-monitor. Acesso em: 17/04/2023.

IBM. (s.d.). O que é monitoramento de cloud? Disponível em: https://www.ibm.com/br-pt/topics/cloud-monitoring. Acesso em: 17/04/2023.

Oracle. (s.d.). O que é Segurança de Dados? Disponível em: https://www.oracle.com/br/security/database-security/what-is-data-security/. Acesso em: 17/04/2023.

ESR. (2020). 10 dicas de como fazer uma boa administração de banco de dados. Disponível em: https://esr.rnp.br/desenvolvimento-de-sistemas/open-2/. Acesso em: 22/04/2023.

Timeline of GitHub. (s.d.). Wikipédia. Disponível em: https://en.wikipedia.org/wiki/Timeline_of_GitHub. Acesso em: 22/04/2023.

KINSTA. What is GitHub? Disponível em: https://kinsta.com/knowledgebase/what-is-github/. Acesso em: 22/04/2023.

MICROSOFT. Visual Studio Code. Disponível em: https://code.visualstudio.com/learn. Acesso em: 22/04/2023.

HANASHIRO, Akira. (2021). VS Code - O que é e por que você deve usar? Disponível em: https://www.treinaweb.com.br/blog/vs-code-o-que-e-e-por-que-voce-deve-usar/. Acesso em: 22/04/2023.

BOUCHERON, Brian. (2021). DigitalOcean. How To Install and Use the Yarn Package Manager for Node.js. Disponível em: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-the-yarn-package-manager-for-node-js. Acesso em: 22/04/2023.

YARN. (s.d.). Instalation. Disponível em: https://classic.yarnpkg.com/lang/en/docs/install/. Acesso em: 22/04/2023.

BEZKODER. (2022). Vue.js + Node.js + Express + MySQL example: Build a full-stack CRUD Application. Disponível em: https://www.bezkoder.com/vue-js-node-js-express-mysql-crud-example/. Acesso em: 05/05/2023.

NPM. (s.d.). body-parser. Disponível em: https://www.npmjs.com/package/body-parser. Acesso em: 05/05/2023.

NPM. (s.d.). consign. Disponível em: https://www.npmjs.com/package/consign. Acesso em: 05/05/2023.

NPM. (s.d.). cors. Disponível em: https://www.npmjs.com/package/cors. Acesso em: 05/05/2023.

NPM. (s.d.). dotenv. Disponível em: https://www.npmjs.com/package/dotenv. Acesso em: 05/05/2023.

NPM. (s.d.). express. Disponível em: https://www.npmjs.com/package/express. Acesso em: 05/05/2023.

NPM. (s.d.). jsonwebtoken. Disponível em: https://www.npmjs.com/package/jsonwebtoken. Acesso em: 05/05/2023.

NPM. (s.d.). knex. Disponível em: https://www.npmjs.com/package/knex. Acesso em: 05/05/2023.

NPM. (s.d.). mysql. Disponível em: https://www.npmjs.com/package/mysql. Acesso em: 05/05/2023.

NPM. (s.d.). nodemon. Disponível em: https://www.npmjs.com/package/nodemon. Acesso em: 05/05/2023.

NPM. (s.d.). validate-js. Disponível em: https://www.npmjs.com/package/validate-js. Acesso em: 05/05/2023.

QUEST. (s.d.). Monitoramento de desempenho do banco de dados. Disponível em: https://www.quest.com/br-pt/solutions/database-performance-monitoring/. Acesso em: 27/05/2023.

DEVTRENDS. (s.d.). Soluções reconhecidas mundialmente para uma abordagem Devops. Disponível em: https://dtrends.com.br/. Acesso em: 27/05/2023.

MYSQL. MySQL Enterprise Monitor. Disponível em: https://www.mysql.com/products/enterprise/monitor.html. Acesso em: 27/05/2023.

DUARTE, Luiz. (2020). LuizTools - Tutorial de CRUD com Node.js, Sequelize e MySQL. Disponível em: https://www.luiztools.com.br/post/tutorial-de-crud-com-node-js-sequelize-e-mysql/. Acesso em: 27/05/2023.

MOTDOTLA. (s.d.). dotenv. Disponível em: https://github.com/motdotla/dotenv. Acesso em: 27/05/2023.

JWT.IO. (s.d.). Introduction to JSON Web Tokens. Disponível em: https://jwt.io/introduction/. Acesso em: 27/05/2023.

EXPRESS.JS. (s.d.). Roteamento no Express. Disponível em: http://expressjs.com/pt-br/guide/routing.html. Acesso em: 27/05/2023.

YARN. (s.d.). Getting Started. Disponível em: https://classic.yarnpkg.com/lang/en/docs/getting-started/. Acesso em: 27/05/2023.

VUE CLI. (s.d.). Overview. Disponível em: https://cli.vuejs.org/guide/. Acesso em: 27/05/2023.

VUE ROUTER. (s.d.). Getting Started. Disponível em: https://router.vuejs.org/guide/. Acesso em: 27/05/2023.

VUEJS. (s.d.). Usando Axios para Consumir APIs. Disponível em: https://br.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html. Acesso em: 27/05/2023.

VUEX. (s.d.). What is Vuex? Disponível em: https://vuex.vuejs.org/. Acesso em: 27/05/2023.

NPM. (s.d.). PM2. Disponível em: https://www.npmjs.com/package/pm2/. Acesso em: 13/06/2023.

MICROSOFT LEARN, 2023. Monitorar e ajustar o desempenho - SQL Server. Disponível em: https://learn.microsoft.com/pt-br/sql/relational-databases/performance/monitor-and-tune-for-performance?view=sql-server-ver16. Acesso em: 13/06/2023.

MICROSOFT LEARN, 2023. Monitorar o desempenho do banco de dados com o Intelligent Insights. Disponível em: https://learn.microsoft.com/pt-br/azure/azure-sql/database/intelligent-insights-overview?view=azuresql. Acesso em: 16/06/2023.

GIT. (s.d.). Git. Disponível em: https://git-scm.com/. Acesso em: 15/06/2023.

MARIADB. (s.d.). Information Schema SYSTEM_VARIABLES Table. Disponível em: https://mariadb.com/kb/en/information-schema-system_variables-table/. Acesso em: 15/06/2023.

MARIADB. (s.d.). mysql.user Table. Disponível em: https://mariadb.com/kb/en/mysql-user-table/. Acesso em: 15/06/2023.

MARIADB. (s.d.). Events Overview. Disponível em: https://mariadb.com/kb/en/events/. Acesso em: 15/06/2023.

MARIADB. (s.d.). InnoDB Buffer Pool. Disponível em: https://mariadb.com/kb/en/innodb-buffer-pool/. Acesso em: 15/06/2023.

MARIADB. (s.d.). Server System Variables. Disponível em: https://mariadb.com/kb/en/server-system-variables/. Acesso em: 15/06/2023.

VUEJS. (s.d.). Básico sobre Componentes. Disponível em: https://br.vuejs.org/v2/guide/components.html. Acesso em: 16/06/2023.

VUEJS. (s.d.). Slots. Disponível em: https://br.vuejs.org/v2/guide/components-slots.html. Acesso em: 16/06/2023.

VUETIFYJS. (s.d.). Cards. Disponível em: https://v15.vuetifyjs.com/pt-BR/components/cards/. Acesso em: 16/06/2023.

VUETIFYJS. (s.d.). Data tables. https://v15.vuetifyjs.com/pt-BR/components/data-tables/. Acesso em: 16/06/2023.

CHART.JS. (s.d.). Gráficos de linha. Disponível em: https://www.chartjs.org/docs/latest/charts/line.html. Acesso em: 18/06/2023.

BASIT, Nimra. (2021). 15 Reasons Why Database is Important. Disponível em: https://curiousdesire.com/reasons-why-database-is-important/. Acesso em: 19/06/2023.

AWS. (s.d.). MySQL | Most Popular Open Source Relational Database | AWS. Recuperado em [data], de https://aws.amazon.com/rds/mysql/what-is-mysql/. Acesso em: 19/06/2023.

ROSSETI, Micaela L. (2021). Requisitos de Software: funcionais e não-funcionais. Disponível em: https://softdesign.com.br/blog/requisitos-de-software-funcionais-e-nao-funcionais/. Acesso em 19/06/2023.

CREATELY. (2021). Tutorial de Diagrama de Caso de Uso. Recuperado em [data], de https://creately.com/blog/pt/diagrama/tutorial-de-diagrama-de-caso-de-uso/. Acesso em 19/06/2023.

ROMBAOA, Kyle. (2021). DEWERKZ. Why Web Development Methodology is Important. Disponível em: https://www.devwerkz.com/blog/why-web-development-methodology-is-important/. Acesso em 19/06/2023.