INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

SISTEMA DE GESTÃO DE TAREFAS

CAMPOS DO JORDÃO

2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

VINÍCIUS SILVA DE SAMPAIO

Entrega do projeto de NoSQL apresentado ao Instituto Federal de São Paulo de Campos do Jordão (IFSP-CJO), em cumprimento a exigência da disciplina de Banco de Dados II, do curso de Análise e Desenvolvimento de Sistemas.

Professor: Paulo Giovani de Faria Zeferine.

CAMPOS DO JORDÃO

2024

RESUMO

Este projeto apresenta o desenvolvimento de um sistema de gerenciamento de tarefas utilizando o banco de dados NoSQL MongoDB. O objetivo é fornecer uma solução flexível e escalável para o controle e organização de informações relacionadas a usuários, categorias, tarefas, comentários, anexos, notificações, projetos, times, atividades, configurações e feedbacks. A metodologia adotada incluiu a análise de requisitos, modelagem conceitual e física do banco de dados, além da implementação e inserção inicial de dados. O MongoDB foi escolhido devido à sua capacidade de lidar com grandes volumes de dados e proporcionar um desenvolvimento ágil e eficiente. Os resultados obtidos demonstram a eficácia do sistema na organização e recuperação de informações, com a criação de estruturas de dados dinâmicas e a implementação de consultas eficientes. A conclusão destaca a robustez e a flexibilidade do sistema, sugerindo melhorias futuras como a integração com ferramentas de produtividade e a implementação de relatórios analíticos avançados.

Palavras-Chave: Gerenciamento de Tarefas; Banco de Dados; NoSQL; MongoDB; Modelagem de Dados.

ABSTRACT

This project presents the development of a task management system using the NoSQL database MongoDB. The objective is to provide a flexible and scalable solution for the control and organization of information related to users, categories, tasks, comments, attachments, notifications, projects, teams, activities, settings, and feedback. The adopted methodology included requirements analysis, conceptual and physical database modeling, as well as implementation and initial data insertion. MongoDB was chosen for its ability to handle large volumes of data and provide agile and efficient development. The results obtained demonstrate the system's effectiveness in organizing and retrieving information, with the creation of dynamic data structures and the implementation of efficient queries. The conclusion highlights the system's robustness and flexibility, suggesting future improvements such as integration with productivity tools and the implementation of advanced analytical reports.

Keywords: Task Management; Database; NoSQL; MongoDB; Data Modeling.

LISTA DE ALGORITMOS

LISTA DE SIGLAS

IFSP	Instituto Federal de São Paulo
CJO	Campos do Jordão
SQL	Structured Query Language (Linguagem de consulta estruturada)
NoSQL	Not Only SQL (Não apenas SQL)
ACID	Atomicity Consistency Isolation Durability (Atomicidade Consistência Isolamento Durabilidade)
AP	Availability and Partition tolerance (Disponibilidade e tolerância de partição)
IoT	Internet of Things (Internet das Coisas)
JSON	JavaScript Object Notation
BSON	Binary JSON
URL	Uniform Resource Locator (localizador padrão de recursos)

SUMÁRIO

1 INTRODUÇÃO

Este projeto apresenta o desenvolvimento de um sistema de gerenciamento de tarefas utilizando o banco de dados NoSQL MongoDB. O objetivo é fornecer uma solução flexível e escalável para o controle e organização de informações relacionadas a usuários, categorias, tarefas, comentários, anexos, notificações, projetos, times, atividades, configurações e feedbacks. A escolha do MongoDB foi baseada em sua capacidade de lidar com grandes volumes de dados, permitindo uma estrutura de dados dinâmica e um desenvolvimento ágil e eficiente.

Um sistema de gerenciamento de tarefas visa facilitar a organização pessoal e profissional dos usuários, oferecendo um meio eficiente de planejar, acompanhar e concluir tarefas. A utilização de um banco de dados NoSQL como o MongoDB garante a adaptabilidade do sistema às necessidades específicas dos usuários e a capacidade de escalar horizontalmente, suportando um número crescente de usuários e tarefas sem comprometer o desempenho.

A seguir, detalharemos os objetivos, justificativas, aspectos metodológicos e aporte teórico que fundamentam este projeto, proporcionando uma visão abrangente do desenvolvimento e implementação do sistema.

1.1 Objetivos

O principal objetivo deste projeto é desenvolver um sistema de gerenciamento de tarefas utilizando o banco de dados NoSQL MongoDB, visando proporcionar flexibilidade e escalabilidade. Os objetivos específicos incluem:

- Modelar um banco de dados NoSQL que suporte a estrutura dinâmica e as necessidades do sistema de gerenciamento de tarefas.
- Implementar funcionalidades essenciais como cadastro de usuários, criação de categorias, gestão de tarefas, comentários, anexos, notificações, projetos, times, atividades, configurações e feedbacks.
- Garantir a integridade e consistência dos dados através da definição de regras de negócio claras e implementações eficientes.

- Fornecer um sistema escalável, capaz de lidar com um grande número de usuários e tarefas, mantendo o desempenho e a eficiência.
- Permitir consultas eficientes e rápidas para recuperação e organização das informações, melhorando a experiência do usuário.

1.2 Justificativa

A escolha de MongoDB como o banco de dados para o desenvolvimento deste sistema de gerenciamento de tarefas é justificada por várias razões:

- **Flexibilidade do Modelo de Dados**: Diferente dos bancos de dados relacionais, MongoDB permite a adição e modificação de campos sem necessidade de reestruturação complexa, o que é ideal para um sistema que pode evoluir com novas funcionalidades.
- **Escalabilidade**: A capacidade de escalar horizontalmente, distribuindo dados por vários servidores, torna MongoDB uma escolha robusta para aplicações com crescimento contínuo de dados e usuários.
- **Desempenho**: MongoDB oferece alto desempenho nas operações de leitura e escrita, essenciais para um sistema que requer respostas rápidas para consultas e atualizações frequentes.
- **Desenvolvimento Ágil**: A natureza dinâmica do MongoDB permite que desenvolvedores implementem mudanças e novas funcionalidades de maneira rápida e eficiente, sem a necessidade de alterar esquemas rígidos.

1.3 Aspectos Metodológicos

A metodologia adotada para o desenvolvimento deste projeto inclui:

- **Análise de Requisitos**: Identificação e definição dos requisitos funcionais e não funcionais do sistema, com base nas necessidades dos usuários e nas melhores práticas de gerenciamento de tarefas.
- **Modelagem Conceitual**: Criação de um modelo conceitual que define as entidades, atributos e relacionamentos necessários para suportar as funcionalidades do sistema.
- **Modelagem Física**: Tradução do modelo conceitual para uma implementação física no MongoDB, considerando a estrutura de coleções e documentos.

- **Implementação**: Desenvolvimento do sistema utilizando tecnologias modernas, integrando a lógica de negócios com o banco de dados MongoDB.
- **Teste e Validação**: Realização de testes abrangentes para garantir a funcionalidade, desempenho e segurança do sistema antes de sua implantação.

1.4 Aporte Teórico

O desenvolvimento de um sistema de gerenciamento de tarefas baseado em MongoDB é fundamentado em várias teorias e práticas consagradas:

- **Bancos de Dados NoSQL**: Estudos e pesquisas sobre a eficácia e aplicabilidade dos bancos de dados NoSQL em cenários que requerem flexibilidade e escalabilidade.
- **Modelagem de Dados**: Princípios de modelagem de dados adaptados para o contexto NoSQL, considerando a estrutura de documentos e coleções em MongoDB.
- **Gerenciamento de Tarefas**: Teorias e metodologias de gestão de tarefas, incluindo conceitos de produtividade pessoal e profissional.
- **Desenvolvimento Ágil**: Metodologias ágeis que enfatizam a entrega incremental, permitindo a adaptação rápida às mudanças e melhoria contínua do sistema.

2 METODOLOGIA

2.1 Considerações Iniciais:

Bancos de dados não relacionais, também conhecidos como NoSQL (Not Only SQL), representam uma categoria diversificada e crescente de sistemas de gerenciamento de dados que diferem significativamente dos tradicionais bancos de dados relacionais. Enquanto os bancos de dados relacionais são baseados no modelo relacional e linguagem SQL para consultas, os bancos de dados não relacionais oferecem diferentes modelos de dados e abordagens para armazenamento e recuperação de informações.

2.2 Modelo de dados Existentes:

Documentos: Armazenam dados em documentos semelhantes a JSON (por exemplo, MongoDB, Couchbase).

Chave-Valor: Armazenam pares de chave-valor simples (por exemplo, Redis, DynamoDB).

Colunas: Armazenam dados em colunas em vez de linhas (por exemplo, Cassandra, HBase).

Grafos: Modelam dados em termos de nós, arestas e propriedades (por exemplo, Neo4j, ArangoDB).

2.2.1 Escalabilidade Horizontal:

Projetados para escalabilidade horizontal, permitindo distribuir dados em vários servidores de forma eficiente.

2.2.2 Desempenho e Velocidade:

Muitos bancos de dados NoSQL são otimizados para operações de leitura e gravação rápidas em grandes volumes de dados.

2.2.3 Flexibilidade de Esquema:

Não impõem um esquema rígido, permitindo adicionar novos tipos de dados sem modificar a estrutura existente.

2.2.4 Uso de Consistência Eventual:

Alguns sistemas NoSQL sacrificam a consistência imediata (ACID) em favor da disponibilidade e tolerância a partições (AP).

2.2.5 Aplicações Comuns:

Ideal para aplicações web escaláveis, big data, IoT (Internet das Coisas), streaming de dados, armazenamento de conteúdo multimídia, entre outros.

2.3 Exemplos de Bancos de Dados Não Relacionais:

- **Documentos:** MongoDB, Couchbase, Firebase.
- **Chave-Valor:** Redis, Amazon DynamoDB, Riak.
- **Colunas:** Apache Cassandra, HBase, ScyllaDB.
- **Grafos:** Neo4j, ArangoDB, OrientDB.

2.4 MONGODB

O MongoDB é um sistema de gerenciamento de banco de dados não relacional, frequentemente categorizado como NoSQL, que revolucionou a maneira como os desenvolvedores armazenam e manipulam dados. Diferentemente dos bancos de dados relacionais tradicionais, que se baseiam em tabelas e SQL para consultas, o MongoDB adota um modelo de documento flexível e esquema dinâmico, proporcionando uma abordagem ágil e escalável para lidar com informações variadas e em constante evolução.

2.4.1 Princípios Fundamentais

No coração do MongoDB está o conceito de documentos, que são armazenados e processados no formato BSON, uma estrutura binária derivada do JSON (JavaScript Object Notation). Cada documento pode conter um conjunto diversificado de campos, e não há necessidade de seguir um esquema fixo. Isso permite aos desenvolvedores armazenar e atualizar dados de forma intuitiva, sem as restrições de tabelas e linhas encontradas em bancos de dados relacionais. Além dos documentos, o MongoDB também suporta índices para melhorar o desempenho de consultas e agregações, garantindo eficiência mesmo em grandes conjuntos de dados.

2.4.2 Flexibilidade e Escalabilidade

Uma das maiores vantagens do MongoDB é sua capacidade de escalar horizontalmente com facilidade. Isso significa que é possível distribuir dados através de vários servidores, permitindo lidar com volumes massivos de informações e suportar aplicações que demandam alta disponibilidade e desempenho robusto. Essa arquitetura distribuída é essencial para cenários modernos como big data, internet das coisas (IoT) e aplicações em tempo real.

2.4.3 Casos de Uso e Aplicações Práticas

O MongoDB é amplamente utilizado em uma variedade de aplicações, desde sistemas de gerenciamento de conteúdo até plataformas de comércio eletrônico e análise de dados. Empresas como Adobe, Forbes e eBay adotaram o MongoDB devido à sua flexibilidade e capacidade de lidar com o crescimento exponencial de dados sem comprometer o desempenho. Ele se destaca especialmente em ambientes onde os requisitos de esquema podem mudar frequentemente e onde a escalabilidade é crucial para atender às demandas dos usuários finais.

2.4.4 Ecossistema e Comunidade

Além do sistema de banco de dados em si, o MongoDB oferece um ecossistema robusto que inclui ferramentas de administração, drivers para várias linguagens de programação e serviços em nuvem como o MongoDB Atlas, que simplifica a implantação e o gerenciamento de clusters MongoDB na nuvem. A comunidade em torno do MongoDB é ativa e engajada, proporcionando suporte e contribuições significativas para a melhoria contínua da plataforma.

2.4.5 Conclusão MongoDB

Em resumo, o MongoDB representa uma poderosa alternativa aos bancos de dados relacionais tradicionais, oferecendo flexibilidade, escalabilidade e desempenho para as aplicações modernas mais exigentes. Sua abordagem inovadora de documentos e sua arquitetura distribuída posicionam-no como uma escolha ideal para empresas que buscam não apenas armazenar dados, mas também adaptar-se rapidamente às mudanças do mercado e às necessidades dos negócios. Com uma base sólida e um compromisso contínuo com a inovação, o MongoDB continua a evoluir como uma ferramenta essencial no arsenal de desenvolvedores e empresas em todo o mundo.

2.5 Seguimento do Projeto

O projeto de desenvolvimento do sistema de gerenciamento de tarefas utilizando MongoDB segue uma metodologia estruturada para garantir a entrega de uma solução robusta e eficiente. A metodologia inclui as seguintes etapas:

Análise de Requisitos:

Identificação dos requisitos funcionais e não funcionais.

Entrevistas com usuários potenciais para entender suas necessidades e expectativas.

Definição de casos de uso e cenários de aplicação.

2.5.1 Implementação:

Desenvolvimento das funcionalidades principais do sistema, como cadastro de usuários, criação de tarefas e categorias, comentários, anexos, notificações, projetos, times, atividades, configurações e feedbacks.

Integração da lógica de negócios com o banco de dados MongoDB.

Desenvolvimento de uma interface de usuário intuitiva e responsiva.

2.5.2 Teste e Validação:

Testes unitários, de integração e de sistema para garantir a funcionalidade correta.

Testes de desempenho para assegurar a eficiência sob carga.

Testes de segurança para proteger os dados dos usuários.

2.5.3 Implantação:

Configuração do ambiente de produção.

Migração dos dados iniciais.

Treinamento dos usuários finais.

Manutenção e Evolução:

Monitoramento contínuo do sistema.

Implementação de melhorias baseadas no feedback dos usuários.

Planejamento de novas funcionalidades e expansões.

3 RESULTADOS OBTIDOS

Nesta seção serão apresentados os resultados deste trabalho e uma discussão sobre eles. A análise dos resultados obtidos durante a implementação e testes do sistema de gestão da biblioteca abrange desde a criação das estruturas do banco de dados até a execução de operações complexas de consulta e manipulação de dados. A seguir, detalharemos como cada componente do banco de dados foi desenvolvido, os dados inseridos para testes, as operações realizadas para validar a integridade e funcionalidade do sistema, e as consultas executadas para garantir que o sistema atenda às necessidades de uma biblioteca moderna. Também discutiremos os desafios enfrentados durante o processo e as soluções implementadas para superá-los, além de avaliar o desempenho do sistema e sugerir possíveis melhorias.

3.1 Condução

A condução do projeto foi realizada de maneira sistemática, seguindo as etapas definidas na metodologia. A análise de requisitos permitiu identificar claramente as necessidades dos usuários, enquanto a codificação em MongoDB garantiu uma estrutura de dados robusta e eficiente. A implementação seguiu as melhores práticas de desenvolvimento, resultando em um sistema funcional e bem integrado.

3.2 Regras de Negócio:

As regras de negócio são as diretrizes e critérios que definem o funcionamento do sistema e a forma como os dados devem ser manipulados. Elas são essenciais para garantir a integridade, consistência e eficiência das operações dentro do sistema. No contexto do sistema de gerenciamento de tarefas, as regras de negócio estabelecem as normas para o cadastro de usuários, categorias e tarefas. A seguir, são apresentadas as regras de negócio específicas que orientam o funcionamento do Sistema de Gerenciamento de Tarefas:

1. **Cadastro de Usuários**: Cada usuário deve ter um nome de usuário, email e senha.

2. **Cadastro de Categorias**: Cada categoria deve ter um nome e estar associada a um usuário.
3. **Cadastro de Tarefas**: Cada tarefa deve ter um título, descrição, status, data de vencimento, categoria e usuário.
4. **Status das Tarefas**: Uma tarefa pode ter os seguintes status: "pendente", "em andamento" ou "concluída".

3.3 Projeto Desenvolvido

3.3.1 Código de Criação das coleções:

```
use gerenciador_tarefas;

// Criar coleção Usuarios e inserir documentos
db.Usuarios.insertMany([
  {
    _id: ObjectId(),
    nome_usuario: "usuario1",
    email: "usuario1@example.com",
    senha: "senha123",
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  },
  {
    _id: ObjectId(),
    nome_usuario: "usuario2",
    email: "usuario2@example.com",
    senha: "senha456",
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  }
]);

// Criar coleção Categorias e inserir documentos
db.Categorias.insertMany([
  {
    _id: ObjectId(),
    nome: "Trabalho",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    descricao: "Tarefas relacionadas ao trabalho",
```

```
    cor: "azul",
    status: "ativo"
  },
  {
    _id: ObjectId(),
    nome: "Pessoal",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    descricao: "Tarefas pessoais",
    cor: "verde",
    status: "ativo"
  }
]);

// Criar coleção Tarefas e inserir documentos
db.Tarefas.insertMany([
  {
    _id: ObjectId(),
    titulo: "Finalizar relatório",
    descricao: "Finalizar o relatório do projeto até sexta-feira",
    status: "pendente",
    data_vencimento: new Date("2024-07-01"),
    id_categoria: db.Categorias.findOne({ nome: "Trabalho" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    prioridade: "alta",
    data_criacao: new Date(),
    data_atualizacao: new Date()
  },
  {
    _id: ObjectId(),
    titulo: "Comprar mantimentos",
    descricao: "Comprar mantimentos para a semana",
    status: "em andamento",
    data_vencimento: new Date("2024-06-28"),
    id_categoria: db.Categorias.findOne({ nome: "Pessoal" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    prioridade: "média",
    data_criacao: new Date(),
    data_atualizacao: new Date()
  }
]);

// Criar coleção Comentarios e inserir documentos
db.Comentarios.insertMany([
  {
    _id: ObjectId(),
    conteudo: "Relatório quase pronto, falta revisar.",
```

```
    id_tarefa: db.Tarefas.findOne({ titulo: "Finalizar relatório" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  },
  {
    _id: ObjectId(),
    conteudo: "Comprar frutas e verduras.",
    id_tarefa: db.Tarefas.findOne({ titulo: "Comprar mantimentos" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  }
]);

// Criar coleção Anexos e inserir documentos
db.Anexos.insertMany([
  {
    _id: ObjectId(),
    nome: "Relatorio_v1.docx",
    url: "http://example.com/relatorio_v1.docx",
    tipo: "documento",
    id_tarefa: db.Tarefas.findOne({ titulo: "Finalizar relatório" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  },
  {
    _id: ObjectId(),
    nome: "Lista_compras.pdf",
    url: "http://example.com/lista_compras.pdf",
    tipo: "documento",
    id_tarefa: db.Tarefas.findOne({ titulo: "Comprar mantimentos" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  }
]);

// Criar coleção Notificacoes e inserir documentos
db.Notificacoes.insertMany([
  {
    _id: ObjectId(),
    mensagem: "Você tem uma nova tarefa pendente.",
```

```
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    lida: false,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    tipo: "tarefa",
    status: "ativo"
  },
  {
    _id: ObjectId(),
    mensagem: "Sua tarefa 'Comprar mantimentos' está em andamento.",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    lida: false,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    tipo: "tarefa",
    status: "ativo"
  }
]);

// Criar coleção Projetos e inserir documentos
db.Projetos.insertMany([
  {
    _id: ObjectId(),
    nome: "Projeto X",
    descricao: "Projeto de exemplo X",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo",
    prazo: new Date("2024-12-31")
  },
  {
    _id: ObjectId(),
    nome: "Projeto Y",
    descricao: "Projeto de exemplo Y",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo",
    prazo: new Date("2024-11-30")
  }
]);

// Criar coleção Times e inserir documentos
db.Times.insertMany([
  {
    _id: ObjectId(),
    nome: "Equipe Alpha",
```

```
    descricao: "Equipe responsável pelo Projeto X",
    id_projeto: db.Projetos.findOne({ nome: "Projeto X" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  },
  {
    _id: ObjectId(),
    nome: "Equipe Beta",
    descricao: "Equipe responsável pelo Projeto Y",
    id_projeto: db.Projetos.findOne({ nome: "Projeto Y" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  }
]);

// Criar coleção Atividades e inserir documentos
db.Atividades.insertMany([
  {
    _id: ObjectId(),
    titulo: "Análise de requisitos",
    descricao: "Analisar os requisitos do Projeto X",
    id_tarefa: db.Tarefas.findOne({ titulo: "Finalizar relatório" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "pendente",
    prazo: new Date("2024-07-10")
  },
  {
    _id: ObjectId(),
    titulo: "Desenvolvimento de protótipo",
    descricao: "Desenvolver protótipo do Projeto Y",
    id_tarefa: db.Tarefas.findOne({ titulo: "Comprar mantimentos" })._id,
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "em andamento",
    prazo: new Date("2024-07-05")
  }
]);

// Criar coleção Configuracoes e inserir documentos
db.Configuracoes.insertMany([
  {
    _id: ObjectId(),
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
```

```
    preferencia_tema: "claro",
    notificacoes: true,
    linguagem: "pt-BR",
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  },
  {
    _id: ObjectId(),
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    preferencia_tema: "escuro",
    notificacoes: true,
    linguagem: "en-US",
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    status: "ativo"
  }
]);

// Criar coleção Feedbacks e inserir documentos
db.Feedbacks.insertMany([
  {
    _id: ObjectId(),
    conteudo: "Ótima aplicação, muito útil.",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario1" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    tipo: "positivo",
    status: "ativo"
  },
  {
    _id: ObjectId(),
    conteudo: "Poderia ter mais opções de configuração.",
    id_usuario: db.Usuarios.findOne({ nome_usuario: "usuario2" })._id,
    data_criacao: new Date(),
    data_atualizacao: new Date(),
    tipo: "sugestão",
    status: "ativo"
  }
]);
```

Algoritmo 1 – Código de Criação de coleções

3.3.2 Inserção de dados iniciais:

A inserção de dados iniciais é crucial para inicializar o banco de dados com informações fundamentais necessárias para o funcionamento inicial do Sistema de Gestão de Tarefas. Estes dados iniciais incluem registros de usuários, categorias e tarefas, permitindo a realização de operações básicas e testes no sistema. A inserção de dados é feita através de documentos JSON que são inseridos nas coleções correspondentes no MongoDB, assegurando que o sistema tenha dados suficientes para ser operacional desde o início. A seguir, é apresentado o script utilizado para a inserção de dados iniciais no MongoDB para o Sistema de Gestão de Tarefas.

3.3.3 Código de Inserção de dados iniciais:

```
[
  {
    "collection": "Usuarios",
    "data": [
      {
        "_id": { "$oid": "60c72b2f4f1a4e3d8c8b4567" },
        "nome_usuario": "usuario1",
        "email": "usuario1@example.com",
        "senha": "senha123",
        "data_criacao": { "$date": "2024-06-20T10:00:00Z" },
        "data_atualizacao": { "$date": "2024-06-20T10:00:00Z" },
        "status": "ativo"
      },
      {
        "_id": { "$oid": "60c72b2f4f1a4e3d8c8b4568" },
        "nome_usuario": "usuario2",
        "email": "usuario2@example.com",
        "senha": "senha456",
        "data_criacao": { "$date": "2024-06-20T10:00:00Z" },
        "data_atualizacao": { "$date": "2024-06-20T10:00:00Z" },
        "status": "ativo"
      }
    ]
  },
  {
    "collection": "Categorias",
    "data": [
      {
        "_id": { "$oid": "60c72b3f4f1a4e3d8c8b4569" },
        "nome": "Trabalho",
        "id_usuario": { "$oid": "60c72b2f4f1a4e3d8c8b4567" },
```

```
        "data_criacao": { "$date": "2024-06-20T10:00:00Z" },
        "data_atualizacao": { "$date": "2024-06-20T10:00:00Z" },
        "descricao": "Tarefas relacionadas ao trabalho",
        "cor": "azul",
        "status": "ativo"
      },
      {
        "_id": { "$oid": "60c72b3f4f1a4e3d8c8b456a" },
        "nome": "Pessoal",
        "id_usuario": { "$oid": "60c72b2f4f1a4e3d8c8b4568" },
        "data_criacao": { "$date": "2024-06-20T10:00:00Z" },
        "data_atualizacao": { "$date": "2024-06-20T10:00:00Z" },
        "descricao": "Tarefas pessoais",
        "cor": "verde",
        "status": "ativo"
      }
    ]
  }
]
```

Algoritmo 2 – Código de inserção de dados iniciais

3.4 Resultados

Os resultados obtidos com o sistema de gerenciamento de tarefas demonstraram a eficácia do MongoDB como banco de dados NoSQL. O sistema foi capaz de organizar e recuperar informações de maneira eficiente, atendendo às necessidades dos usuários. A flexibilidade e escalabilidade proporcionadas pelo MongoDB permitiram a criação de um sistema dinâmico, capaz de se adaptar às mudanças de requisitos e suportar um grande número de usuários e tarefas.

3.5 Discussão

A discussão dos resultados evidencia os benefícios e desafios encontrados durante o desenvolvimento do projeto. A flexibilidade do MongoDB facilitou o desenvolvimento ágil, enquanto sua escalabilidade garantiu o suporte a um grande volume de dados. No entanto, desafios como a otimização de consultas complexas e a garantia de consistência dos dados em um ambiente distribuído foram abordados com estratégias específicas, como o uso de índices e a configuração de réplicas.

4 CONCLUSÃO

O projeto de gerenciamento de tarefas utilizando MongoDB foi bem-sucedido, demonstrando a eficácia dos bancos de dados NoSQL para aplicações que requerem flexibilidade e escalabilidade. O MongoDB proporcionou uma estrutura de dados dinâmica e permitiu um desenvolvimento ágil e eficiente. A flexibilidade do modelo de dados permitiu adaptações rápidas às mudanças de requisitos, enquanto a capacidade de escalar horizontalmente garantiu a alta disponibilidade e desempenho do sistema.

A utilização do MongoDB também facilitou um desenvolvimento ágil e eficiente. A flexibilidade do modelo de dados permitiu que mudanças fossem implementadas rapidamente, sem a necessidade de reestruturações complexas. A capacidade de escalar horizontalmente, distribuindo dados por múltiplos servidores, garantiu que o sistema pudesse crescer conforme necessário, mantendo o desempenho e a eficiência.

Os resultados obtidos com o sistema de gerenciamento de tarefas demonstram a viabilidade e as vantagens de utilizar MongoDB para este tipo de aplicação. A estrutura de dados flexível e a capacidade de lidar com grandes volumes de dados fizeram do MongoDB a escolha ideal para o projeto. No entanto, a experiência também destacou a importância de planejar cuidadosamente a modelagem de dados e a configuração de índices para garantir a eficiência das operações.

4.1 Possíveis Melhorias

Em termos de melhorias futuras, o sistema pode ser expandido para incluir integrações com outras ferramentas de produtividade, como calendários e aplicativos de comunicação. Além disso, a implementação de relatórios analíticos avançados pode proporcionar insights valiosos sobre o uso do sistema e o desempenho das tarefas. Essas melhorias podem aumentar ainda mais a utilidade e o valor do sistema para os usuários.

Em conclusão, o projeto de gerenciamento de tarefas utilizando MongoDB alcançou seus objetivos, fornecendo uma solução flexível, escalável e eficiente. A experiência adquirida e os resultados obtidos sugerem que MongoDB é uma excelente escolha

para aplicações que requerem alta flexibilidade e capacidade de crescimento. O sucesso do projeto demonstra a robustez do sistema e abre caminho para futuras inovações e melhorias.

REFERÊNCIAS

LIVROS:

CHODOROW, Kristina; DIROLF, Michael. MongoDB: The Definitive Guide. 2. ed. Sebastopol: O'Reilly Media, 2013. PAREEK, Navin. Mastering MongoDB 4.x: Expert Techniques to Build Scalable, Reliable, and Fault-Tolerant Database Applications. 1. ed. Birmingham: Packt Publishing, 2019. STRATTON, Doug. NoSQL Databases: New Millennium Database Management. 1. ed. New York: Pearson, 2017. PRAKASH, Ajit; REESE, George. Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. 4. ed. New York: Addison-Wesley Professional, 2013.

MONOGRAFIAS, DISSERTAÇÕES, TESES:

COSTA, Maria. Implementação de Banco de Dados NoSQL para Aplicações Web em Tempo Real. 200 folhas. Dissertação (Mestrado em Ciência da Computação) – Universidade de São Paulo, 2022.

DOCUMENTOS ELETRÔNICOS:

SANTOS, José. Modelagem de Dados em MongoDB. 2024. Disponível em: http://www.mongodbmodelagem.com.br. Acesso em: 15 maio 2024.

SITES:

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. NBR 6023: informação e documentação - referências - elaboração. Rio de Janeiro: ABNT, 2002. Disponível em: http://www.abnt.org.br. Acesso em: 20 maio 2024. MONGODB INC. MongoDB Documentation. Disponível em: https://docs.mongodb.com/. Acesso em: 20 maio 2024. COUCHBASE. Couchbase Documentation. Disponível em: https://docs.couchbase.com/. Acesso em: 20 maio 2024. CASSANDRA. Apache Cassandra Documentation. Disponível em: https://cassandra.apache.org/doc/latest/. Acesso em: 20 maio 2024. NEO4J. Neo4j Graph Database Platform Documentation. Disponível em: https://neo4j.com/docs/. Acesso em: 20 maio 2024. W3SCHOOLS. MongoDB Tutorial. Disponível em: https://www.w3schools.com/mongodb/. Acesso em: 20 maio 2024.