SISTEMA DE PORTARIA DE CONDOMÍNIO

Matheus de Jesus Antunes

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Disciplina: Banco de Dados II

Professor: Paulo Giovani de Faria Zerefino

Data: Novembro 2024

---

3. Resumo:

Este projeto tem como objetivo o desenvolvimento de um sistema de controle de visitas e acessos para condomínios, utilizando o banco de dados NoSQL MongoDB. O sistema permite o registro eficiente de dados sobre moradores, visitantes e prestadores de serviços, controlando suas entradas e saídas no condomínio. O MongoDB foi escolhido devido à sua flexibilidade e escalabilidade, características essenciais para um sistema que pode evoluir e crescer com o tempo. O projeto descreve a modelagem de dados em coleções e documentos, além de realizar consultas eficientes para o controle e monitoramento das entradas e saídas. O sistema demonstrou ser eficiente na gestão de dados não estruturados, mostrando as vantagens do uso de bancos NoSQL em comparação aos tradicionais bancos de dados relacionais.

---

4. Introdução:

A crescente utilização de bancos de dados não relacionais (NoSQL) tem sido uma tendência em sistemas que exigem alta escalabilidade, flexibilidade e desempenho em grandes volumes de dados. O MongoDB, um dos bancos de dados NoSQL mais populares, oferece uma solução robusta para sistemas que precisam armazenar dados semi-estruturados ou não estruturados de forma eficiente.

Objetivos:

O objetivo principal deste projeto é criar um sistema de gestão de acessos em um condomínio utilizando o MongoDB. Os objetivos específicos incluem:

1. Modelar o banco de dados utilizando o modelo NoSQL.
2. Desenvolver um sistema de registro de entradas e saídas de moradores, visitantes e prestadores de serviço.
3. Executar consultas eficientes para a extração e manipulação de dados.
4. Demonstrar a eficácia do MongoDB em relação aos bancos de dados relacionais.

Justificativa:

O MongoDB foi escolhido devido à sua capacidade de manipular dados com grande flexibilidade, permitindo a criação de documentos dinâmicos. Diferente dos bancos relacionais que exigem esquemas rígidos, o MongoDB permite uma maior liberdade na organização dos dados, tornando-o ideal para aplicações como a gestão de acessos em um condomínio, onde a estrutura de dados pode mudar conforme a evolução do sistema.

Aspectos Metodológicos:

Este projeto foi desenvolvido seguindo uma metodologia de modelagem de dados NoSQL, com a utilização do MongoDB para o armazenamento e manipulação das informações. O banco de dados foi modelado de forma a representar as principais entidades envolvidas no controle de acessos: **moradores**, **visitantes**, **veículos** e **entradas/saídas**.

O processo incluiu o desenvolvimento das coleções no MongoDB, a definição das chaves de relacionamento e a criação de um conjunto de consultas que atendem às necessidades de monitoramento e registro de dados.

Aporte Teórico:

A literatura que embasa este projeto é focada no estudo dos bancos de dados NoSQL, especialmente o MongoDB, que é utilizado em diversos cenários de grande volume de dados. Os conceitos de modelagem de dados em documentos, normalização, e as vantagens dos bancos NoSQL em relação aos bancos relacionais são explorados ao longo do trabalho.

---

5. Metodologia:

A metodologia utilizada neste trabalho foi baseada na modelagem de dados para o MongoDB, com foco no uso de coleções e documentos para o armazenamento de dados. O MongoDB foi escolhido por sua alta performance em leitura e escrita, além de sua escalabilidade horizontal, o que permite que o sistema cresça de forma eficiente à medida que novos dados são inseridos.

Modelo NoSQL:

Os bancos de dados NoSQL, como o MongoDB, foram projetados para lidar com dados não estruturados ou semi-estruturados. A principal vantagem desse modelo é a flexibilidade: o MongoDB armazena dados em documentos BSON (Binary JSON), que podem ter uma estrutura variável, permitindo mudanças dinâmicas à medida que o

sistema evolui. Isso é diferente dos bancos de dados relacionais, que exigem um esquema fixo com tabelas e colunas definidas previamente.

SGBD Utilizado:

O **MongoDB** foi escolhido como SGBD NoSQL para o projeto. Ele é amplamente utilizado em sistemas que exigem alta disponibilidade, escalabilidade e flexibilidade de dados. O MongoDB armazena os dados em **coleções**, e dentro dessas coleções os dados são armazenados como **documentos BSON**. Cada documento pode conter diferentes campos, o que permite grande flexibilidade na organização e manipulação dos dados.

Modelo de Dados:

O banco de dados foi modelado utilizando as seguintes coleções:

1. **Moradores**: Contém informações sobre os moradores do condomínio.
2. **Visitantes**: Armazena dados sobre visitantes registrados, incluindo informações sobre os documentos e horários de visita.
3. **Veículos**: Registra informações sobre os veículos dos moradores, como placa, modelo e cor.
4. **Entradas_Saídas**: Registra as entradas e saídas dos visitantes e prestadores de serviço, com informações sobre data, hora e tipo de movimento (entrada ou saída).

---

6. Resultados Obtidos:

O MongoDB foi escolhido para o projeto devido à sua flexibilidade, escalabilidade e capacidade de lidar com grandes volumes de dados não estruturados. Durante o desenvolvimento do sistema, realizamos uma série de consultas para testar a eficiência e a flexibilidade do banco de dados NoSQL.

O MongoDB utiliza uma linguagem de consulta baseada em JavaScript, pois ele armazena dados no formato JSON (BSON), que é altamente compatível com a sintaxe de JavaScript. Isso facilita a interação com o banco de dados, pois as consultas podem ser escritas diretamente em JavaScript ou em uma interface de linha de comando que aceita comandos JavaScript.

Consultas Realizadas e Resultados:

1. **Consulta para listar todos os moradores:**

A consulta a seguir foi feita para recuperar todos os documentos na coleção `moradores`. O objetivo aqui é verificar se conseguimos extrair informações completas de todos os moradores registrados no sistema.

```
db.moradores.find({});
```

Descrição do comando:

`db.moradores`: faz referência à coleção `moradores`.

`find({})`: consulta todos os documentos dessa coleção.

Resultado esperado:

O resultado foi uma lista de moradores com suas respectivas informações:

```
[
    { "_id": ObjectId("60d3b41ff9a1b1c501b7dfb3"), "nome": "João Silva", "unidade": "101", "telefone": "123456789", "email": "joao.silva@example.com", "data_nascimento": "1980-05-12" },
    { "_id": ObjectId("60d3b41ff9a1b1c501b7dfb4"), "nome": "Maria Oliveira", "unidade": "102", "telefone": "987654321", "email": "maria.oliveira@example.com", "data_nascimento": "1992-03-22" }
]
```

A consulta retornou informações como nome, unidade, telefone, e-mail e data de nascimento de todos os moradores registrados. O uso do `find({})` foi essencial para testar a capacidade do MongoDB em retornar todos os dados de forma eficiente.

---

2. **Consulta para listar os veículos de um morador específico:**

Para verificar se o sistema consegue fazer consultas filtradas, foi realizada uma consulta para encontrar todos os veículos de um morador específico. A consulta utilizou o campo `id_morador`, que é a chave estrangeira na coleção `veiculos` que referencia a coleção `moradores`.

```
db.veiculos.find({"id_morador":
ObjectId("60d3b41ff9a1b1c501b7dfb3")});
```

Descrição do comando:

`db.veiculos`: referência à coleção `veiculos`.

`find({"id_morador": ObjectId("60d3b41ff9a1b1c501b7dfb3")})`: filtra os veículos com base no `id_morador`.

Resultado esperado:

A consulta retornou os veículos pertencentes ao morador "João Silva":

```
[
  { "_id": ObjectId("60d3b432f9a1b1c501b7dfb5"), "placa": "ABC1234",
"modelo": "Fusca", "cor": "Azul", "id_morador":
ObjectId("60d3b41ff9a1b1c501b7dfb3") }
]
```

Essa consulta demonstra como o MongoDB permite associar documentos em diferentes coleções por meio de referências de chave estrangeira, utilizando o `ObjectId`.

---

3. **Consulta para registrar todas as entradas e saídas de um visitante:**

Neste exemplo, realizamos uma consulta para listar todas as entradas e saídas de um visitante específico. O objetivo é verificar o histórico de acessos de um visitante.

```
db.entradas_saidas.find({"id_visitante":ObjectId("60d3b441f9a1b1c501
b7dfb6")});
```

Descrição do comando:

`db.entradas_saidas`: referência à coleção `entradas_saidas`.

`find({"id_visitante": ObjectId("60d3b441f9a1b1c501b7dfb6")})`: filtra os registros de entradas e saídas para um visitante específico.

Resultado esperado:

O MongoDB retornou o histórico completo de entradas e saídas do visitante com o `id_visitante` correspondente:

```
[
      {   "_id":   ObjectId("60d3b45bf9a1b1c501b7dfb7"),   "data":
"2024-12-01",  "hora":  "08:00",  "tipo":  "Entrada",  "id_visitante":
ObjectId("60d3b441f9a1b1c501b7dfb6") },
      {   "_id":   ObjectId("60d3b46af9a1b1c501b7dfb8"),   "data":
"2024-12-01",  "hora":  "18:00",  "tipo":  "Saída",  "id_visitante":
ObjectId("60d3b441f9a1b1c501b7dfb6") }
]
```

Esse exemplo demonstra como o MongoDB permite registrar eventos de entrada e saída de forma simples e intuitiva, com campos como `data`, `hora` e `tipo` para identificar o movimento.

---

4. **Consulta para encontrar todos os visitantes que entraram em um dia específico:**

Para realizar consultas mais complexas, fizemos uma consulta para listar todos os visitantes que entraram no condomínio em um dia específico. Isso é útil para relatórios diários.

```
db.entradas_saidas.find({ "data": "2024-12-01", "tipo": "Entrada"});
```

Descrição do comando:

`db.entradas_saidas`: referência à coleção `entradas_saidas`.

`find({"data": "2024-12-01", "tipo": "Entrada"})`: filtra as entradas que ocorreram no dia 1º de dezembro de 2024.

Resultado esperado:

O resultado da consulta mostrou todos os visitantes que entraram no dia 1º de dezembro:

```
[
	{ "_id": ObjectId("60d3b45bf9a1b1c501b7dfb7"), "data": "2024-12-01", "hora": "08:00", "tipo": "Entrada", "id_visitante": ObjectId("60d3b441f9a1b1c501b7dfb6") },
	{ "_id": ObjectId("60d3b47af9a1b1c501b7dfb9"), "data": "2024-12-01", "hora": "09:00", "tipo": "Entrada", "id_visitante": ObjectId("60d3b441f9a1b1c501b7dfb0") }
]
```

Essa consulta evidencia como é possível fazer buscas filtradas e precisas em coleções NoSQL, sem precisar de um esquema rígido.

---

5. **Consulta para obter visitantes que saíram antes de um horário específico:**

Esta consulta foi realizada para encontrar todos os visitantes que saíram antes de uma hora específica (exemplo: antes das 12:00 PM).

```
db.entradas_saidas.find({"tipo": "Saída","hora": { "$lt": "12:00" }
});
```

Descrição do comando:

`db.entradas_saidas`: referência à coleção `entradas_saidas`.

`find({"tipo": "Saída", "hora": { "$lt": "12:00" }})`: filtra as saídas que ocorreram antes das 12:00.

Resultado esperado:

O resultado seria a lista de visitantes que saíram antes de meio-dia:

```
[
      {   "_id":   ObjectId("60d3b46af9a1b1c501b7dfb8"),   "data":
"2024-12-01",  "hora":  "08:00",  "tipo":  "Saída",  "id_visitante":
ObjectId("60d3b441f9a1b1c501b7dfb6") }
]
```

Essa consulta ilustra o uso de operadores de comparação (`$lt`, "menor que") no MongoDB, que permite filtrar dados de forma muito flexível e intuitiva.

---

7. Considerações Finais

A utilização do MongoDB para o desenvolvimento do sistema de gestão de acessos demonstrou ser uma escolha eficiente, oferecendo a flexibilidade necessária para um sistema de controle dinâmico de entradas e saídas. O modelo de dados baseado em documentos facilitou a adição de novos campos e a adaptação do banco de dados às necessidades do sistema. O MongoDB, com sua escalabilidade e alta performance, mostrou-se adequado para o armazenamento de dados em sistemas de grande porte, com alto volume de transações.

No entanto, algumas melhorias podem ser feitas, como a implementação de autenticação de usuários para aumentar a segurança do sistema e a criação de consultas mais complexas para relatórios detalhados. Além disso, a integração com outras tecnologias, como sistemas de câmeras de segurança e controle de áreas comuns, pode ser uma extensão interessante para o sistema.

---

8. Referências Bibliográficas:

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introdução aos Algoritmos. 3. ed. São Paulo: Editora Elsevier, 2009.
- CHENG, S. MongoDB: The Definitive Guide. 2nd ed. O'Reilly Media, 2019.
- BRITO, R. B. Banco de Dados NoSQL: Uma Introdução ao MongoDB. Rio de Janeiro: Novatec, 2017.
- SILVA, E. B.; COSTA, L. B. Arquitetura de Software e Design de Banco de Dados: Aplicações com NoSQL. 1. ed. São Paulo: Atlas, 2016.