



Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Câmpus Campos do Jordão.

Barbara Letícia da Silva

Análise e implementação de um sistema de gestão com MongoDB em uma gráfica de placas personalizadas.

O presente trabalho tem como objetivo apresentar a análise e a implementação de um sistema de gestão para uma gráfica especializada na produção e entrega de placas personalizadas, utilizando o banco de dados não relacional MongoDB. A proposta surgiu da necessidade de adaptar um projeto originalmente desenvolvido com banco de dados relacional (MySQL) para uma abordagem NoSQL, mantendo o mesmo contexto e regras de negócio. O estudo discute os fundamentos dos bancos de dados não relacionais, seus modelos (documento, grafo, chave-valor e coluna larga) e, em especial, o funcionamento do MongoDB, que utiliza o modelo orientado a documentos. Através da modelagem de coleções e documentos, a estrutura anteriormente normalizada em tabelas foi reorganizada de forma flexível e escalável, mantendo a integridade lógica da aplicação. A aplicação do MongoDB permitiu uma visualização mais intuitiva dos dados personalizados da gráfica, além de simplificar operações como inserção e consulta. O relatório apresenta ainda exemplos práticos da estrutura dos dados em formato JSON, operações CRUD adaptadas e consultas específicas que refletem as rotinas administrativas, fiscais e comerciais da empresa. Ao final, o estudo evidencia as vantagens da adoção do MongoDB em contextos onde a flexibilidade e a agilidade no tratamento de dados são diferenciais relevantes.

CAMPOS DO JORDÃO

2025

Barbara Letícia da Silva

Análise e implementação de um sistema de gestão com MongoDB em uma gráfica de placas personalizadas.

Trabalho apresentado ao Curso
Tecnologia em Análise e
Desenvolvimento de Sistemas.

Banco de Dados 2-CJOBDD2

Professor: Paulo Giovani de
Faria Zeferino.

CAMPOS DO JORDÃO

2025

Projeto Final - NoSQL

- 1.Introdução.**
- 2.Metodologia.**
- 3.Resultados Obtidos.**
- 4.Conclusão.**
- 5.Referências Bibliográficas**

1. O uso de tecnologias de banco de dados é essencial para a gestão eficiente de informações em ambientes corporativos, especialmente em empresas que lidam com alto volume de dados personalizados e dinâmicos. Uma gráfica especializada na produção e entrega de placas personalizadas lida diariamente com uma grande diversidade de pedidos que variam quanto ao conteúdo textual, formato, cor, dimensões, prazos e formas de pagamento. Esse tipo de negócio exige não apenas agilidade operacional, mas também um sistema de gestão de dados que permita acompanhar as atividades fiscais, logísticas e comerciais de forma integrada e segura.

Este trabalho tem como proposta a reestruturação de um projeto previamente desenvolvido com base no modelo relacional (utilizando MySQL), adaptando-o à lógica de um banco de dados não relacional, com a adoção do sistema gerenciador MongoDB. O tema e o contexto do projeto original foram mantidos, garantindo a preservação das regras de negócio e das exigências da empresa gráfica simulada. A principal motivação para a mudança de abordagem está na necessidade de maior flexibilidade e escalabilidade, atributos que os bancos NoSQL oferecem de forma mais adequada em determinados cenários.

Os bancos de dados não relacionais, ou NoSQL (Not Only SQL), surgiram como alternativa aos bancos relacionais tradicionais, especialmente em contextos que exigem alta disponibilidade, grande volume de dados não estruturados ou semi-estruturados e maior agilidade na manipulação de dados. No universo dos bancos NoSQL, existem diferentes modelos de dados, como chave-valor, orientado a documentos, orientado a grafos e orientado a colunas. Entre esses modelos, o MongoDB se destaca por utilizar uma estrutura baseada em documentos no formato JSON (ou BSON), o que permite organizar os dados de maneira mais natural e aderente à realidade de muitas aplicações modernas.

A adoção do MongoDB neste projeto justifica-se pela sua capacidade de representar estruturas complexas de forma simples e direta, eliminando a necessidade de junções complicadas entre tabelas. Cada pedido da gráfica, por exemplo, pode ser representado como um único documento que agrupa todas as informações relacionadas ao cliente, à encomenda, às placas e aos detalhes fiscais e financeiros. Esse formato oferece vantagens em termos de desempenho e facilidade de desenvolvimento, principalmente quando se trabalha com dados altamente personalizados e variáveis.

A metodologia adotada neste trabalho compreende a revisão do projeto original, a conversão da modelagem relacional para uma modelagem orientada a documentos, a definição das coleções e documentos que compõem o banco de dados MongoDB e a implementação de operações básicas como inserção, consulta, atualização e exclusão (CRUD). Além disso, serão apresentadas consultas específicas que simulam rotinas de gestão da gráfica, com base nas regras de negócio já estabelecidas.

Por fim, o presente relatório visa demonstrar, na prática, como é possível aplicar os conceitos de banco de dados NoSQL em um cenário realista, avaliando os impactos da mudança de paradigma em termos de estrutura, performance e manutenção do sistema. A proposta busca não apenas validar a utilização do MongoDB no contexto da gráfica de placas personalizadas, mas também proporcionar uma compreensão mais ampla das vantagens e desafios associados aos bancos de dados não relacionais.

2. Este projeto teve como base a reestruturação de um sistema originalmente desenvolvido com um banco de dados relacional (MySQL) para o paradigma NoSQL, utilizando o MongoDB como sistema gerenciador de banco de dados (SGBD). O objetivo foi manter o tema e o contexto do projeto original, uma gráfica especializada na produção e entrega de placas personalizadas, preservando as regras de negócio, enquanto se aplicavam os princípios e técnicas da modelagem orientada a documentos. A metodologia adotada envolveu a análise do modelo relacional, o estudo dos conceitos teóricos dos bancos de dados não relacionais, a definição de uma nova modelagem lógica compatível com o MongoDB e, por fim, a implementação de um conjunto de coleções e documentos que representassem a estrutura de dados da empresa.

Considerações iniciais sobre o projeto:

A gráfica simulada neste estudo trabalha com uma série de processos interligados: cadastramento de clientes, recebimento de encomendas personalizadas, produção e entrega de placas, emissão de notas fiscais e controle de faturas. No projeto original, essas informações estavam organizadas em tabelas distintas, relacionadas por meio de chaves primárias e estrangeiras. No entanto, essa abordagem mostrou-se menos adequada para a flexibilidade exigida por pedidos altamente personalizados e em constante mudança.

Com a evolução das aplicações web e mobile e a crescente necessidade de sistemas que processem grandes volumes de dados de forma ágil, o uso de bancos NoSQL vem se consolidando como alternativa viável. Nesse contexto, o presente trabalho se propôs a transformar a estrutura relacional em um modelo mais flexível e escalável, preservando a lógica do negócio da gráfica, mas reorganizando os dados sob uma nova arquitetura.

Modelo de banco NoSQL e classificações:

Os bancos de dados NoSQL surgiram como resposta à limitação dos bancos relacionais em lidar com dados não estruturados, escalabilidade horizontal e esquemas rígidos. O termo "NoSQL" refere-se a "Not Only SQL", e engloba uma variedade de modelos que fogem à lógica de tabelas e joins. Entre os principais modelos de bancos NoSQL, destacam-se:

- **Chave-Valor:** Armazena dados como pares únicos de chave e valor, sendo altamente performático para buscas simples e diretas.
- **Colunar (Wide Column Store):** Organiza dados por colunas, sendo eficiente em grandes volumes de leitura, muito utilizado em sistemas analíticos.
- **Orientado a Documentos:** Utiliza documentos no formato JSON ou BSON para representar dados complexos de maneira hierárquica.
- **Orientado a Grafos:** Especializado em armazenar e consultar relacionamentos complexos, sendo indicado para redes e estruturas de conexão.

Para o desenvolvimento do projeto da gráfica, foi adotado o modelo orientado a documentos por sua capacidade de lidar com estruturas de dados flexíveis e aninhadas. Essa abordagem se mostrou a mais adequada para representar os pedidos personalizados, que exigem um formato dinâmico e adaptável conforme as especificações de cada cliente.

Escolha do MongoDB e modelo orientado a documentos

O MongoDB foi o SGBD escolhido por sua compatibilidade com o modelo de documentos, sua maturidade no mercado, extensa documentação e suporte a operações complexas com desempenho satisfatório. Ele permite armazenar dados em documentos JSON (internamente convertidos para BSON), que podem conter arrays, subdocumentos e estruturas aninhadas, oferecendo grande flexibilidade na modelagem dos dados.

Diferentemente dos bancos relacionais, onde a estrutura é pré-definida e fortemente tipada, no MongoDB cada documento pode conter diferentes campos, mesmo dentro de uma mesma coleção. Isso permite evoluir o sistema de forma dinâmica, adaptando-se a novas necessidades sem impactar diretamente a estrutura já existente.

A modelagem baseada em documentos também favorece a agregação de dados em uma única unidade lógica. Por exemplo, uma encomenda da gráfica pode ser representada como um documento único que inclui os dados do cliente, os detalhes das placas encomendadas e informações sobre o pagamento, sem necessidade de múltiplas tabelas.

Modelagem de coleções e documentos

A estrutura de dados foi organizada em coleções, cada uma correspondendo a uma entidade lógica do negócio. As coleções principais definidas foram:

- empresas
- clientes
- produtos
- transportadoras
- notas_fiscais
- faturas
- impostos
- encomendas

Cada uma dessas coleções contém documentos que refletem a estrutura anterior do banco relacional. Em alguns casos, informações que antes estavam em tabelas auxiliares foram incorporadas diretamente aos documentos por meio de arrays ou subdocumentos. Por exemplo: Esse trecho abaixo é um documento JSON (JavaScript Object Notation), que representa os dados de uma encomenda em um banco de dados NoSQL, como o MongoDB.

O código irá representar a encomenda "E001" feita por Laura Mendes, com CPF 111.222.333-44, em 1º de maio de 2025. Ela pediu uma placa com a frase "Feliz Aniversário", tamanho 30x20, com frase preta e placa branca. O pagamento total é de R\$200,00, dos quais R\$50,00 foram pagos como sinal, e a forma de pagamento é dinheiro.

```
{
  "numero": "E001",
  "cliente": {
    "nome": "Laura Mendes",
    "cpf": "111.222.333-44"
  },
  "data_encomenda": "2025-05-01",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "tamanho": "30x20",
      "cor_frase": "Preto",
      "cor_placa": "Branco"
    }
  ],
  "pagamento": {
    "valor_servico": 200.00,
    "valor_sinal": 50.00,
    "forma": "Dinheiro"
  }
}
```

Regras de negócio aplicadas no modelo NoSQL

As regras de negócio previamente estabelecidas foram respeitadas na nova modelagem. Embora a estrutura física dos dados tenha mudado, a lógica do sistema foi mantida. Dentre as regras principais aplicadas:

- Um cliente pode ter várias encomendas (representadas por documentos distintos referenciando o mesmo cliente).
- Uma encomenda pode conter múltiplas placas (armazenadas em arrays dentro do documento).
- Uma nota fiscal pode conter vários produtos e impostos (relacionados via referências ou subdocumentos).
- Cada fatura está associada a uma nota fiscal e a um cliente (por campos internos ou ID externo).
- Transportadoras e empresas podem ser referenciadas por seus identificadores nos documentos de nota fiscal.

A modelagem foi cuidadosamente planejada para preservar a consistência dos dados e facilitar as consultas mais frequentes, como relatórios de vendas, acompanhamento de entregas e emissão de documentos fiscais.

3. A conversão do modelo relacional para o MongoDB exigiu a reorganização dos dados, anteriormente distribuídos em tabelas normalizadas, para coleções compostas por documentos em formato JSON. Esse processo implicou não apenas uma mudança na estrutura física dos dados, mas também uma reinterpretação da modelagem conceitual, adaptando-a à abordagem orientada a documentos. As entidades principais do modelo relacional foram transformadas em coleções, enquanto os relacionamentos, principalmente os de um-para-muitos e muitos-para-muitos, foram representados por meio de subdocumentos e arrays, conforme a complexidade e a natureza dos dados envolvidos.

A nova modelagem foi cuidadosamente planejada para refletir fielmente as entidades e os relacionamentos originais, garantindo a preservação da integridade lógica e semântica do sistema. Foram adotadas boas práticas de modelagem em MongoDB, como a desnormalização controlada, visando reduzir a necessidade de junções complexas, que não são nativamente suportadas de forma eficiente em bancos NoSQL. O uso de subdocumentos permitiu agrupar dados relacionados de forma coesa, enquanto os arrays possibilitaram o armazenamento de múltiplos valores ou objetos dentro de um mesmo campo, facilitando o acesso e a manipulação de dados no contexto das operações de leitura.

Nesta seção, são apresentados exemplos representativos da estrutura de documentos adotada para as principais coleções do banco de dados, bem como trechos de código ilustrando a inserção de dados utilizando comandos do MongoDB. Além disso, são descritas consultas adaptadas ao novo modelo, utilizando a linguagem de agregação quando necessário, com o objetivo de demonstrar como as informações podem ser recuperadas de maneira eficiente e funcional, respeitando a lógica de negócios originalmente definida no sistema relacional.

Estrutura dos documentos (Modelagem física)

Coleção: empresa

```
{  
  "cnpj": "12.345.678/0001-99",  
  "nome": "Gráfica Express",  
  "logo": "logo1.png",  
  "chave_acesso": "CHAVE123456",  
  "natureza": "LTDA",  
  "inscricao_municipal": "IM12345",  
  "inscricao_estadual": "IE54321"  
}
```

Coleção: cliente

```
{  
  "cpf_cnpj": "123.456.789-00",  
  "nome": "Maria Silva",  
  "telefone": "11988887777",  
  "endereco": "Rua das Flores, 100",  
  "inscricao_social": "IS001"  
}
```

Coleção: encomenda

```
{
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "data_encomenda": "2025-05-01",
  "data_entrega": "2025-05-05",
  "valor_servico": 200.00,
  "valor_sinal": 50.00,
  "pagamento": "Dinheiro",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "tamanho": "30x20",
      "cor_frase": "Preto",
      "cor_placa": "Branco",
      "valor": 70.00,
      "utensilios": "Moldura, Suporte"
    }
  ]
}
```

Coleção: emite_nota_fiscal

```
{
  "numero": 1,
  "serie": "A1",
  "tipo": "Saída",
  "data_emissao": "2025-05-01",
  "codigo_fixo": "CF001",
  "desconto": 50.00,
  "valor_seguro": 10.00,
  "empresa_cnpj": "12.345.678/0001-99"
}
```

Coleção: fatura

```
{
  "numero": "FAT001",
  "valor_servico": 500.00,
  "data_vencimento": "2025-05-10",
  "nota_fiscal_numero": 1
}
```


Coleção: produto

```
{  
  "codigo": 101,  
  "preco": 50.00,  
  "est": "C01",  
  "un": "UN",  
  "cfop": "5102",  
  "descricao": "Cartão de visita 1000 unid.",  
  "nota_fiscal_numero": 1  
}
```

Coleção: imposto

```
{  
  "codigo": 1,  
  "nome": "ICMS",  
  "valor": 30.00,  
  "base_calculo": 500.00,  
  "nota_fiscal_numero": 1  
}
```

Coleção: transportador

```
{  
  "cnpj": "11.111.111/0001-11",  
  "nome": "TransLog",  
  "endereco": "Rua das Cargas, 10",  
  "quantidade": 2,  
  "marca": "Volks",  
  "peso_bruto": 1000.00,  
  "peso_liquido": 800.00,  
  "especie": "Caixa",  
  "placa": "ABC1234",  
  "numeracao": "N001",  
  "codigo_antt": "ANTT01",  
  "fonte_pagamento": "A prazo",  
  "inscricao_estadual": "IE99999",  
  "nota_fiscal_numero": 1  
}
```

Inserções de dados no MongoDB (equivalente ao SQL INSERT INTO)

Exemplo: Inserindo uma empresa

```
db.empresa.insertOne({  
  "cnpj": "12.345.678/0001-99",  
  "nome": "Grafica Express",  
  "logo": "logo1.png",  
  "chave_acesso": "CHAVE123456",  
  "natureza": "LTDA",  
})
```

```
"inscricao_municipal": "IM12345",  
"inscricao_estadual": "IE54321"  
});
```

Exemplo: Inserindo uma encomenda com subdocumentos

```
db.encomenda.insertOne({  
  "numero": "E001",  
  "nome_cliente": "Laura Mendes",  
  "data_encomenda": "2025-05-01",  
  "data_entrega": "2025-05-05",  
  "valor_servico": 200.00,  
  "valor_sinal": 50.00,  
  "pagamento": "Dinheiro",  
  "placas": [  
    {  
      "codigo": "P001",  
      "frase": "Feliz Aniversário",  
      "tamanho": "30x20",  
      "cor_frase": "Preto",  
      "cor_placa": "Branco",  
      "valor": 70.00,  
      "utensilios": "Moldura, Suporte"  
    }  
  ]  
});
```

Consultas em MongoDB

A seguir, são apresentadas 20 consultas desenvolvidas para o banco de dados implementado no MongoDB, com o objetivo de simular operações reais de gestão no contexto da gráfica especializada em placas personalizadas. Essas consultas foram adaptadas diretamente do projeto original em SQL, agora reestruturadas para o modelo orientado a documentos.

As operações demonstram a versatilidade do MongoDB em recuperar informações distribuídas em coleções, utilizando comandos como `find()` e operadores de comparação, além de agregações mais complexas com `aggregate()` e `$lookup`, que simulam junções entre coleções. Cada consulta é precedida por seu objetivo e seguida de uma breve descrição do resultado esperado, refletindo situações comuns no dia a dia de uma empresa gráfica, como o acompanhamento de encomendas, controle fiscal, consulta de produtos e filtragem por critérios comerciais.

A modelagem com documentos JSON permitiu reunir dados relacionados em estruturas mais compactas, eliminando a necessidade de múltiplas junções e tornando as consultas mais simples e diretas. Esta seção evidencia, na prática, como o MongoDB pode ser utilizado para sustentar operações administrativas, comerciais e fiscais com flexibilidade e eficiência.

Consulta 1 – Clientes com nota fiscal emitida

Objetivo: Listar todos os clientes que possuem nota fiscal emitida, com detalhes da nota.

Consulta:

```
db.cliente.aggregate([
  {
    $lookup: {
      from: "emite_nota_fiscal",
      localField: "cpf_cnpj",
      foreignField: "cliente_id",
      as: "notas_fiscais"
    }
  },
  { $match: { "notas_fiscais.0": { $exists: true } } }
]);
```

Resultado esperado: Maria Silva com nota A1 e Carlos Souza com nota B1.

Consulta 2 – Faturas com valor de serviço superior a R\$300,00

Objetivo: Exibir faturas cujo valor do serviço ultrapassa R\$300,00.

Consulta:

```
db.fatura.find({ valor_servico: { $gt: 300.00 } });
```

Resultado esperado: FAT001 no valor de R\$500,00.

Consulta 3 – Produtos com CST “C01”

Objetivo: Listar todos os produtos cujo código CST seja igual a “C01”.

Consulta:

```
db.produto.find({ cst: "C01" });
```

Resultado esperado: Cartão de visita e Imã de geladeira.

Consulta 4 – Impostos da nota fiscal nº 2

Objetivo: Exibir os impostos vinculados à nota fiscal de número 2.

Consulta:

```
db.imposto.find({ nota_fiscal_numero: 2 });
```

Resultado esperado: IPI com valor de R\$20,00.

Consulta 5 – Placas da encomenda “E001”

Objetivo: Listar todas as placas associadas à encomenda de código “E001”.

Consulta:

```
db.encomenda.find(  
  { numero: "E001" },  
  { "placas": 1, _id: 0 }  
);
```

Resultado esperado: Placa P001 com frase “Feliz Aniversário”.

Consulta 6 – Notas emitidas em maio de 2025

Objetivo: Buscar todas as notas fiscais emitidas no mês de maio de 2025.

Consulta:

```
db.emite_nota_fiscal.find(  
  data_emissao: {  
    $gte: new Date("2025-05-01"),  
    $lte: new Date("2025-05-31")  
  }  
));
```

Resultado esperado: Notas A1 e B1.

Consulta 7 – Empresas fora da cidade de São Paulo

Objetivo: Listar empresas que não estão localizadas na cidade de São Paulo.

Consulta:

```
db.empresa.find({ cidade: { $ne: "São Paulo" } });
```

Resultado esperado: Visual Design localizada em Campinas.

Consulta 8 – Produtos com preço superior a R\$100,00

Objetivo: Exibir todos os produtos cujo preço seja superior a R\$100,00.

Consulta:

```
db.produto.find({ preco: { $gt: 100.00 } });
```

Resultado esperado: Banner 2x1m no valor de R\$120,00.

Consulta 9 – Transportadores com marca “Volks”

Objetivo: Listar todos os transportadores cujos veículos são da marca “Volks”.

Consulta:

```
db.transportador.find({ marca: "Volks" });
```

Resultado esperado: TransLog com veículo de marca Volks.

Consulta 10 – Encomendas com sinal superior a R\$100,00

Objetivo: Exibir todas as encomendas cujo valor de sinal é maior que R\$100,00.

Consulta:

`db.encomenda.find({ valor_sinal: { $gt: 100.00 } });`

Resultado esperado: Nenhuma encomenda (no exemplo, sinal é R\$50,00).

Consulta 11 – Clientes físicos com telefone cadastrado

Objetivo: Exibir os clientes que possuem telefone registrado.

Consulta:

`db.cliente.find({ telefone: { $exists: true, $ne: "" } });`

Resultado esperado: Laura Mendes com telefone 11955554444.

Consulta 12 – Faturas com vencimento futuro

Objetivo: Listar todas as faturas com vencimento posterior à data atual.

Consulta:

`db.fatura.find({ data_vencimento: { $gt: new Date() } });`

Resultado esperado: Fatura FAT001 (vencimento em 10/06/2025, por exemplo).

Consulta 13 – Produtos da nota fiscal nº 1

Objetivo: Listar os produtos que estão vinculados à nota fiscal número 1.

Consulta:

`db.produto.find({ nota_fiscal_numero: 1 });`

Resultado esperado: Cartão de visita e Banner.

Consulta 14 – Impostos com valor acima de R\$10,00

Objetivo: Listar todos os impostos com valor superior a R\$10,00.

Consulta:

`db.imposto.find({ valor: { $gt: 10.00 } });`

Resultado esperado: ICMS e IPI.

Consulta 15 – Notas com desconto superior a R\$30,00

Objetivo: Listar notas fiscais cujo valor de desconto seja superior a R\$30,00.

Consulta:

`db.emite_nota_fiscal.find({ desconto: { $gt: 30.00 } });`

Resultado esperado: Nota A1 (desconto R\$50,00).

Consulta 16 – Produtos com quantidade superior a 1

Objetivo: Listar produtos que tenham quantidade maior que 1 unidade. (Assumindo que o campo "quantidade" exista na coleção produto.)

Consulta:

```
db.produto.find({ quantidade: { $gt: 1 } });
```

Resultado esperado: Banner com quantidade 3.

Consulta 17 – Clientes com endereço na Av. Paulista

Objetivo: Listar clientes cujo endereço contenha “Paulista”.

Consulta:

```
db.cliente.find({ endereco: /Paulista/i });
```

Resultado esperado: Carlos Souza com endereço na Av. Paulista.

Consulta 18 – Encomendas com frases personalizadas

Objetivo: Exibir todas as encomendas que possuem frases personalizadas nas placas.

Consulta:

```
db.encomenda.find({ "placas.frase": { $exists: true, $ne: "" } });
```

Resultado esperado: Encomenda E001 com frase "Feliz Aniversário”.

Consulta 19 – Empresas com nome iniciado por “Graf”

Objetivo: Listar empresas cujo nome comece com “Graf”.

Consulta:

```
db.empresa.find({ nome: { $regex: /^Graf/, $options: "i" } });
```

Resultado esperado: Grafica Express.

Consulta 20 – Notas fiscais com número maior que 1

Objetivo: Listar todas as notas fiscais cujo número seja superior a 1.

Consulta:

```
db.emite_nota_fiscal.find({ numero: { $gt: 1 } });
```

Resultado esperado: Nota fiscal número 2 (B1).

Exemplo de projeto desenvolvido utilizando MongoDB

Este exemplo tem como objetivo demonstrar, de forma prática e aplicada, como foi realizada a implementação de um sistema de gestão voltado para uma gráfica especializada na produção de placas personalizadas, utilizando o banco de dados NoSQL MongoDB. Durante o desenvolvimento, foram explorados os principais conceitos de modelagem orientada a documentos, característica marcante dessa tecnologia. O foco está na organização eficiente das informações e na flexibilidade da estrutura de dados, adequando-se às necessidades específicas do negócio. Na sequência, são apresentadas as principais coleções que compõem o banco de dados, exemplos de documentos reais armazenados nessas coleções, bem como as operações básicas de manipulação de dados (CRUD). Também são demonstradas consultas utilizando filtros e operações de agregação, evidenciando o potencial do MongoDB para atender demandas complexas de análise e recuperação de informações.

Coleções utilizadas

O sistema foi estruturado com as seguintes coleções:

- empresa
- cliente
- encomenda
- placa (adaptada dentro da encomenda)
- emite_nota_fiscal
- fatura
- produto
- imposto
- transportador

Exemplo de documento: Encomenda

```
{
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "data_encomenda": ISODate("2025-05-01T00:00:00Z"),
  "data_entrega": ISODate("2025-05-05T00:00:00Z"),
  "valor_servico": 200.00,
  "valor_sinal": 50.00,
  "pagamento": "Dinheiro",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "tamanho": "30x20",
      "cor_frase": "Preto",
      "cor_placa": "Branco",
    }
  ]
}
```

```

    "valor": 70.00,
    "utensilios": "Moldura, Suporte"
  }
]
}

```

Este modelo de dados apresentado acima, possibilita que uma única encomenda abrigue múltiplas placas personalizadas, cada uma com seus respectivos atributos, como tamanho, material, cor, tipo de acabamento, entre outros. Isso elimina a necessidade de criação e manutenção de tabelas auxiliares para representar o relacionamento entre encomendas e placas, simplificando a estrutura do banco de dados e otimizando o armazenamento e a consulta das informações.

Operações CRUD

As operações CRUD (Create, Read, Update, Delete) são fundamentais em qualquer sistema que manipule dados. No MongoDB, essas operações são realizadas diretamente sobre coleções, utilizando documentos no formato JSON (ou BSON internamente). A seguir, explicam-se cada uma dessas operações no contexto da aplicação da gráfica de placas personalizadas.

Create (Criar)

A operação Create é utilizada para inserir novos dados no banco. No MongoDB, isso é feito por meio dos métodos `insertOne()` ou `insertMany()`, que adicionam um ou mais documentos a uma coleção. No sistema da gráfica, a operação de criação é usada, por exemplo, ao cadastrar uma nova encomenda de placa personalizada, contendo informações como cliente, frase, tamanho da placa, forma de pagamento e outros detalhes. Essa estrutura permite registrar todos os dados relevantes de forma consolidada em um único documento.

Read (Ler/Consultar)

A operação Read permite buscar e visualizar os dados armazenados. É realizada com o método `find()` ou `findOne()`, podendo incluir filtros para localizar registros específicos. No caso do projeto, essa operação é utilizada para localizar as encomendas de um cliente, consultar produtos cadastrados, ou verificar notas fiscais emitidas. O MongoDB oferece operadores de comparação e expressões regulares para tornar as consultas mais flexíveis e eficientes.

Update (Atualizar)

A operação Update é usada para modificar dados existentes. No MongoDB, é executada com métodos como `updateOne()` ou `updateMany()`, combinados com operadores como `$set` (atualiza campos), `$inc` (incrementa valores), entre outros. Por exemplo, na gráfica, um cliente pode alterar a data de entrega de uma encomenda ou modificar a frase gravada em uma placa. Essa operação é útil para manter os dados sempre atualizados conforme mudanças de última hora ou correções.

Delete (Excluir)

A operação Delete remove dados do banco. Em MongoDB, isso é feito com `deleteOne()` ou `deleteMany()`. Em um sistema real, essa ação é utilizada com cautela, e geralmente acompanhada de validações. No projeto da gráfica, um exemplo de uso seria a exclusão de uma encomenda cancelada ou registrada por

engano. O MongoDB executa a exclusão diretamente sobre os documentos que correspondem aos critérios definidos.

- Essas quatro operações formam a base da manipulação de dados em MongoDB, permitindo que o sistema da gráfica opere de forma completa: do cadastro de informações até sua consulta, atualização ou remoção.

Create (Inserção de uma nova encomenda)

```
db.encomenda.insertOne({
  "numero": "E002",
  "nome_cliente": "Carlos Souza",
  "data_encomenda": new Date("2025-06-01"),
  "data_entrega": new Date("2025-06-04"),
  "valor_servico": 250.00,
  "valor_sinal": 100.00,
  "pagamento": "PIX",
  "placas": [
    {
      "codigo": "P002",
      "frase": "Bem-vindo",
      "tamanho": "40x30",
      "cor_frase": "Azul",
      "cor_placa": "Amarela",
      "valor": 80.00,
      "utensilios": "Base magnética"
    }
  ]
});
```

Read (Buscar encomendas feitas por "Laura Mendes")

```
db.encomenda.find({ nome_cliente: "Laura Mendes" });
```

Update (Alterar o valor do sinal de uma encomenda)

```
db.encomenda.updateOne(
  { numero: "E001" },
  { $set: { valor_sinal: 60.00 } }
);
```

Delete (Remover uma encomenda específica)

```
db.encomenda.deleteOne({ numero: "E002" });
```

Consulta com agregação: Valor total arrecadado por encomenda

```
db.encomenda.aggregate([
{
  $project: {
    numero: 1,
    cliente: "$nome_cliente",
    valor_total: { $sum: ["$valor_servico"] }
  }
}
]);
```

Resultado esperado: Lista com o número da encomenda, nome do cliente e o valor total de serviço prestado.

Resumo da modelagem utilizada

- Coleções como unidades lógicas: cada entidade do negócio é representada por uma coleção (empresa, cliente, etc.).
- Subdocumentos e arrays: usados para representar dependências internas como placas dentro de uma encomenda.
- Documentos autônomos: cada documento contém todas as informações necessárias para representar uma instância completa da entidade.
- Consultas simplificadas: o acesso aos dados se torna mais direto sem a necessidade de múltiplos relacionamentos.

Este exemplo representa um sistema funcional desenvolvido com MongoDB, demonstrando a aplicação prática das boas práticas do modelo orientado a documentos. A estrutura do banco foi cuidadosamente projetada para refletir de maneira fiel os requisitos do sistema original, garantindo a integridade e a consistência dos dados, mesmo em um ambiente NoSQL. Além disso, o projeto preserva toda a lógica de negócio anteriormente implementada em um banco de dados relacional, ao mesmo tempo em que se beneficia das vantagens oferecidas pela tecnologia NoSQL, como maior flexibilidade na modelagem de dados, escalabilidade horizontal e desempenho otimizado em operações de leitura e escrita. Dessa forma, a migração para o MongoDB não compromete a robustez da aplicação, mas sim a complementa, tornando o sistema mais adaptável a diferentes cenários de uso e volumes de dados.

Análise comparativa entre banco relacional e NoSQL

A implementação do projeto inicialmente em um banco de dados relacional (MySQL) e, posteriormente, sua adaptação para um banco de dados NoSQL (MongoDB) permitiu uma análise comparativa entre os dois paradigmas. Ambos possuem vantagens e limitações, sendo indicados para diferentes tipos de aplicações, conforme as características dos dados e os requisitos do sistema.

No modelo relacional, os dados são organizados em tabelas com esquemas rígidos, fortemente estruturados por meio de chaves primárias e estrangeiras. Essa abordagem favorece a integridade referencial e é ideal para sistemas com regras de negócio bem definidas e pouca variação estrutural. No entanto, em contextos como o da gráfica de placas personalizadas, onde há grande variação nos detalhes das

encomendas, personalizações específicas por cliente, e dados compostos e aninhados, o modelo relacional exige múltiplas tabelas auxiliares e operações de junção (joins) para representar as informações de forma completa.

Já o MongoDB, por ser um banco de dados orientado a documentos, permite armazenar dados com estruturas flexíveis, sem a necessidade de seguir um esquema fixo. Informações como frases personalizadas, dimensões da placa, cores e utensílios podem ser representadas dentro de um único documento, mantendo a coesão dos dados e eliminando a necessidade de operações complexas de junção. Isso se traduz em consultas mais simples, maior performance em leitura, e agilidade na evolução da estrutura de dados.

Adiante, apresenta - se uma tabela comparativa entre as duas abordagens utilizadas no projeto:

Característica	Banco Relacional (MySQL)	Banco NoSQL (MongoDB)
Modelo de Dados	Tabelas com esquema fixo	Documentos JSON com esquema flexível
Normalização	Elevada (3FN ou superior)	Desnormalização por embedding
Relacionamentos	Por chaves primária/ estrangeira	Por subdocumentos ou referências
Performance em Leitura	Boa, mas depende de joins	Excelente em leitura direta de documentos
Flexibilidade para dados variáveis	Limitada	Alta
Evolução de Estrutura	Requer alterações no esquema	Permite modificações sem recriar estrutura
Operações CRUD	SQL padrão (INSERT, SELECT, etc.)	Operações com insert, find, etc.

Estratégias de modelagem com MongoDB

Ao adaptar o projeto da gráfica de placas personalizadas para o modelo NoSQL utilizando o MongoDB, foi necessário reestruturar completamente a modelagem dos dados. A abordagem orientada a documentos exige decisões diferentes daquelas tomadas em bancos relacionais, especialmente no que diz respeito ao modo como as informações são agrupadas e acessadas.

Duas estratégias principais de modelagem são aplicáveis no MongoDB:

- Embedding (documentos aninhados)
- Referencing (documentos separados com referências)

Cada uma dessas estratégias apresenta vantagens e implicações práticas que devem ser analisadas com cautela, considerando o contexto específico de uso do sistema.

Embedding: Documentos aninhados

O embedding foi utilizado em casos onde os dados fazem parte de uma mesma entidade lógica e são frequentemente acessados em conjunto. Um exemplo claro é a estrutura das placas dentro de uma encomenda. Como cada encomenda contém uma ou mais placas personalizadas com atributos únicos (frase, tamanho, cores, valor, utensílios), optou-se por armazenar essas placas como subdocumentos em um array dentro do próprio documento de encomenda.

Exemplo:

```
{
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "cor_frase": "Preto"
    }
  ]
}
```

Vantagens dessa abordagem:

- Reduz a necessidade de múltiplas consultas ou junções.
- Facilita a leitura completa de uma encomenda em uma única operação.
- Mantém integridade lógica entre entidades fortemente acopladas.

Referencing: Documentos relacionados por ID

A estratégia de referencing foi aplicada em entidades que possuem uma vida útil independente dentro do sistema ou que são compartilhadas por diversas outras entidades. Exemplos típicos dessa abordagem incluem clientes, empresas, transportadoras, notas fiscais e produtos. Essas entidades apresentam características próprias, sendo utilizadas em diferentes contextos e operações, o que justifica o seu armazenamento em coleções separadas.

Por meio dessa estratégia, o documento principal não armazena todas as informações detalhadas da entidade relacionada, mas sim um campo de referência que aponta para o documento correspondente em outra coleção. Essa referência geralmente é feita utilizando um identificador único, como o `_id` gerado pelo MongoDB ou, quando aplicável, um campo natural como o CNPJ no caso de empresas.

Essa técnica traz benefícios importantes para a integridade e a consistência dos dados, facilitando a manutenção e evitando a redundância de informações. Além disso, ela permite que atualizações em dados centrais, como o cadastro de um cliente ou o preço de um produto, reflitam automaticamente em todas as operações relacionadas, sem a necessidade de atualização manual em documentos dependentes.

A adoção do referencing também favorece a escalabilidade do sistema, especialmente em cenários onde o volume de dados das entidades referenciadas pode crescer de forma independente, sem impactar diretamente o tamanho ou a estrutura dos documentos principais.

Exemplo:

```
{
  "numero": 1,
  "data_emissao": "2025-05-01",
  "empresa_cnpj": "12.345.678/0001-99",
  "cliente_id": "123.456.789-00"
}
```

Vantagens dessa abordagem:

- Permite reutilização de dados (ex: um cliente com várias notas).
- Evita redundância.
- Facilita atualizações parciais sem afetar múltiplos documentos.

Critérios para escolha entre Embedding e Referencing

No desenvolvimento do projeto, a decisão entre utilizar embedding (aninhamento de documentos) ou referencing (referência entre documentos) foi cuidadosamente orientada por três critérios fundamentais, com base nas boas práticas de modelagem de dados para o MongoDB:

Frequência de acesso conjunto:

Quando os dados de duas ou mais entidades são frequentemente acessados ou consultados de forma conjunta, a estratégia de embedding foi a preferida. Isso permite reduzir a quantidade de operações de leitura e melhorar o desempenho, uma vez que todas as informações necessárias ficam armazenadas dentro de um único documento. Essa abordagem é ideal para cenários onde a recuperação de dados precisa ser rápida e otimizada, evitando a necessidade de múltiplas consultas ou operações de junção (joins) entre coleções.

Independência e reutilização da entidade:

Para entidades com vida útil independente, que podem ser modificadas ou reutilizadas por diferentes documentos ou processos no sistema, foi adotado o referencing. Neste caso, os documentos armazenam apenas o identificador (por exemplo, o ObjectId, CPF ou CNPJ) da entidade relacionada. Isso garante maior flexibilidade na atualização desses dados, evitando redundâncias e inconsistências. Essa abordagem foi aplicada principalmente em entidades como clientes, empresas, transportadoras, notas fiscais e produtos, que podem ser referenciados por diversas encomendas ou transações dentro do sistema.

Tamanho dos documentos:

Outro fator determinante foi o cuidado com o tamanho final dos documentos gerados. O MongoDB possui um limite máximo de 16 MB por documento. Para evitar o risco de atingir esse limite, optou-se por evitar o embedding excessivo em casos de entidades com potencial de crescimento descontrolado, como listas de transações, logs ou históricos extensos. Assim, o modelo buscou um equilíbrio entre desempenho e escalabilidade, garantindo que os documentos permanecessem dentro de limites aceitáveis e de fácil manutenção.

- Essa modelagem permitiu que o sistema fosse implementado de forma coerente com o comportamento real da empresa gráfica, equilibrando desempenho, integridade e escalabilidade.

Integração com aplicações

A utilização do MongoDB no projeto da gráfica de placas personalizadas não se limita apenas ao armazenamento e consulta de dados. Uma das grandes vantagens desse SGBD NoSQL é sua facilidade de integração com diferentes linguagens de programação e frameworks, o que amplia significativamente suas possibilidades de aplicação em diversos tipos de sistemas reais, como websites, aplicativos móveis ou plataformas de gestão interna. Essa flexibilidade permite que os desenvolvedores construam soluções mais dinâmicas e escaláveis, adaptando o banco de dados às necessidades específicas do negócio.

Além disso, o MongoDB oferece suporte nativo a bibliotecas populares em linguagens como JavaScript (Node.js), Python, Java, entre outras, o que facilita o desenvolvimento de APIs RESTful e aplicações com arquitetura orientada a serviços. No contexto da gráfica, isso significa a possibilidade de criar interfaces administrativas para controle de pedidos, dashboards para acompanhamento de produção e até mesmo lojas virtuais para que os clientes façam encomendas online de forma prática e rápida.

Outro diferencial importante é o suporte a consultas complexas com agregações, índices geoespaciais e operações de texto, o que viabiliza a implementação de recursos como busca inteligente por produtos, localização de clientes por região e relatórios personalizados. Dessa forma, o MongoDB não apenas atende aos requisitos básicos de armazenamento, mas também contribui para a inovação e modernização dos processos internos da gráfica.

Drivers oficiais e linguagens suportadas

O MongoDB oferece drivers oficiais para as principais linguagens de desenvolvimento, tendo como exemplo:

- Node.js (JavaScript)
- Python
- Java
- C# (.NET)
- PHP
- Go
- C++

Essa ampla variedade de drivers permite que o banco de dados seja integrado a diferentes tipos de aplicações, desde interfaces web até sistemas desktop e APIs RESTful.

Exemplo de integração prática: Node.js com MongoDB

Um exemplo de integração possível para o projeto seria o uso de Node.js, uma das plataformas mais utilizadas atualmente no desenvolvimento de aplicações web. Através do driver oficial do MongoDB para Node.js ou utilizando frameworks como o Mongoose, seria possível criar uma API que:

- Permite o cadastro de novas encomendas.
- Realiza a consulta de clientes e suas encomendas.
- Atualiza o status de produção e entrega das placas.
- Gera relatórios financeiros e fiscais.

Abaixo é apresentado um exemplo simples de conexão com MongoDB usando Node.js:

```
const { MongoClient } = require('mongodb');
async function main() {
  const uri = "mongodb://localhost:27017";
  const client = new MongoClient(uri);
  try {
    await client.connect();
    const database = client.db("grafica");
    const clientes = database.collection("cliente");
    const resultado = await clientes.findOne({ nome: "Maria Silva" });
    console.log(resultado);
  } finally {
    await client.close();
  }
}
main().catch(console.error);
```

Benefícios da integração

Ao integrar o MongoDB com aplicações reais, o sistema da gráfica passa a oferecer:

- Agilidade no desenvolvimento de novas funcionalidades.
- Escalabilidade, permitindo crescimento conforme a demanda aumenta.
- Facilidade de manutenção e atualização.
- Acesso remoto aos dados através de APIs.
- Possíveis chances de criação de interfaces gráficas amigáveis para os usuários finais.

Essa capacidade de integração reforça a escolha do MongoDB como uma solução moderna e eficiente para o gerenciamento de dados da gráfica. A facilidade com que o MongoDB se conecta a diversas linguagens de programação, frameworks e ferramentas de análise de dados permite a criação de aplicações mais ágeis, escaláveis e adaptáveis às necessidades do negócio. Além disso, o suporte a consultas complexas, agregações e operações em tempo real garante que o sistema possa evoluir para atender demandas futuras, como integração com e-commerce, aplicativos móveis ou painéis de BI (Business Intelligence). Dessa forma, o MongoDB não apenas atende os requisitos atuais da gráfica, mas também oferece uma base sólida para futuras inovações tecnológicas no processo de gestão e atendimento ao cliente.

Ferramentas utilizadas no projeto

O desenvolvimento do projeto de banco de dados NoSQL para a gráfica de placas personalizadas contou com o apoio de diversas ferramentas, tanto para a fase de modelagem quanto para a implementação, testes e visualização dos dados. A escolha dessas ferramentas considerou critérios como acessibilidade, compatibilidade com o MongoDB, facilidade de uso e aderência às práticas acadêmicas, visando garantir um processo de desenvolvimento eficiente e alinhado com os objetivos do projeto.

Durante a modelagem, foram utilizadas ferramentas de diagramação que possibilitaram representar de forma clara a estrutura das coleções, os relacionamentos entre documentos e as regras de negócio envolvidas. Na etapa de implementação, o uso do MongoDB Compass e de ambientes de linha de comando, como o MongoDB Shell, foi fundamental para a criação das coleções, inserção de documentos e execução de consultas de maneira prática e intuitiva.

Além disso, para os testes e validação das operações CRUD e consultas complexas, foram utilizadas ferramentas de simulação de dados e extensões para integração com linguagens como JavaScript (Node.js), permitindo verificar o funcionamento do sistema em situações reais.

Por fim, a fase de visualização dos dados contou com recursos gráficos oferecidos pelo MongoDB Compass, que facilitaram a análise dos documentos e o entendimento da estrutura interna das coleções. Essa combinação de ferramentas tornou o processo de desenvolvimento mais ágil, didático e próximo das práticas aplicadas no mercado de tecnologia da informação.

BrModelo

Ainda que o foco final do projeto seja o MongoDB, a ferramenta BrModelo foi utilizada durante a etapa inicial de concepção, especialmente na criação do modelo conceitual baseado na notação Entidade-Relacionamento (ER). Essa escolha se justifica pela necessidade de manter a equivalência e o alinhamento com o projeto relacional anterior, desenvolvido em MySQL, garantindo uma transição mais segura e estruturada entre os dois paradigmas de banco de dados.

O uso do BrModelo permitiu uma representação gráfica clara das entidades, atributos e relacionamentos existentes no sistema original da gráfica de placas personalizadas. Essa etapa foi fundamental para uma análise mais detalhada das regras de negócio e para a identificação de quais estruturas seriam mantidas, quais seriam ajustadas e quais poderiam ser otimizadas no processo de migração para o modelo orientado a documentos.

Além disso, a criação do modelo ER auxiliou na discussão sobre estratégias de modelagem no MongoDB, como a escolha entre embedding e referencing, e a identificação de possíveis pontos de desnormalização. Dessa forma, o BrModelo funcionou como um elo importante entre a lógica relacional tradicional e a nova estrutura NoSQL, proporcionando maior clareza no mapeamento das entidades para as futuras coleções de documentos.

Essa abordagem também trouxe benefícios do ponto de vista acadêmico, pois seguiu uma sequência metodológica didática: partindo da análise conceitual para, posteriormente, realizar a adaptação ao novo formato tecnológico, respeitando os fundamentos da engenharia de dados e garantindo a consistência do projeto em todas as suas fases.

Abaixo serão apresentadas às respectivas imagens (1.0 e 2.0) de como foi elaborado o Modelo Conceitual para a Gráfica especializada na produção e entrega de placas personalizadas.

Imagem 1.0

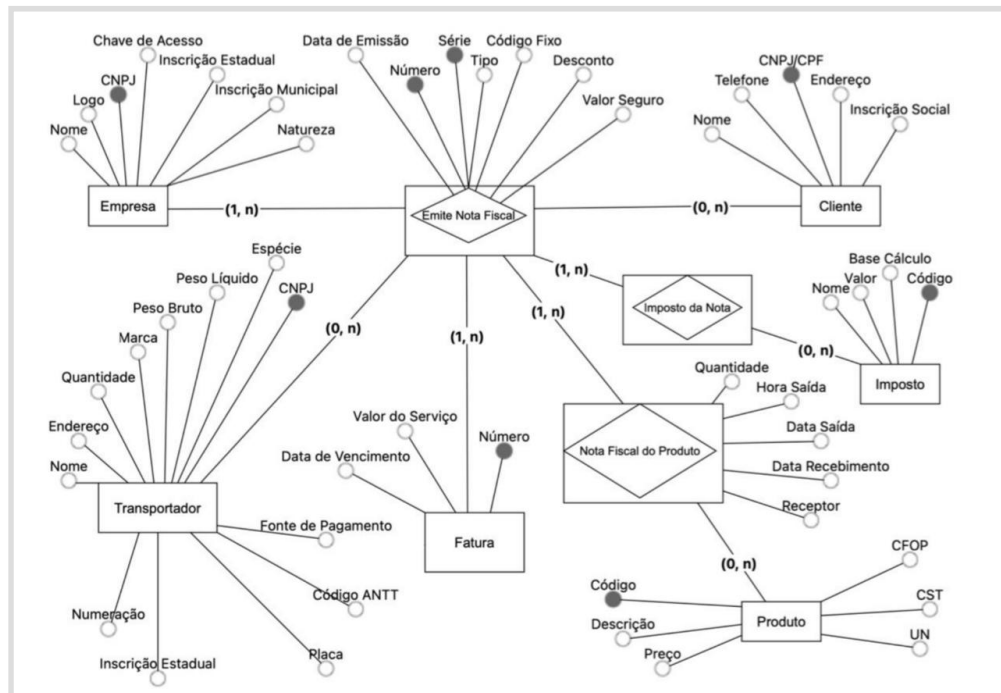
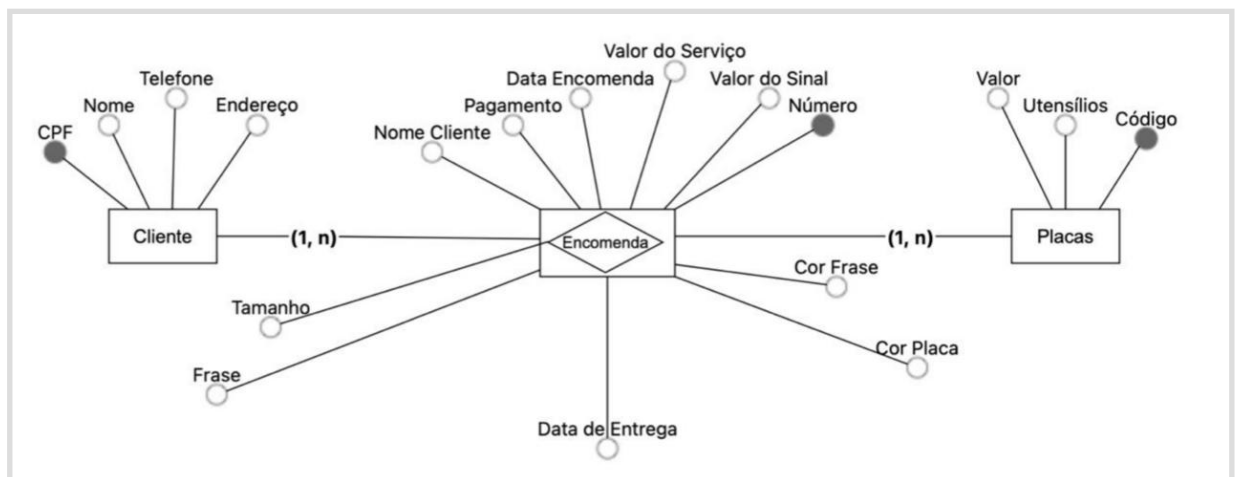


Imagem 2.0



MongoDB Community Server

O MongoDB Community Server disponibiliza uma infraestrutura completa para a criação de coleções, inserção de documentos e execução de consultas, tanto simples quanto avançadas, utilizando a linguagem de consultas baseada em JSON (MongoDB Query Language - MQL). Com essa versão, foi possível testar diferentes cenários de manipulação de dados, realizar operações CRUD (Create, Read, Update, Delete), aplicar filtros, ordenar resultados e até mesmo executar pipelines de agregação, recurso essencial para análises mais complexas.

Outro benefício importante dessa versão é a possibilidade de monitoramento de desempenho, gestão de índices e uso de ferramentas de diagnóstico, o que proporciona ao desenvolvedor um controle mais detalhado sobre o comportamento do banco de dados.

Além disso, por ser uma solução de código aberto, o MongoDB Community Server permite a exploração de configurações avançadas, aprendizado prático e acesso a uma comunidade ativa de desenvolvedores e estudantes que compartilham experiências, boas práticas e soluções para os desafios mais comuns em projetos NoSQL.

Essa combinação de recursos fez com que a escolha dessa versão atendesse plenamente os objetivos acadêmicos e práticos do projeto, permitindo a criação de um sistema funcional, escalável e alinhado com as tecnologias mais utilizadas no mercado.

Exemplo de uso no projeto: Consulta de uma encomenda pelo número.

Código:

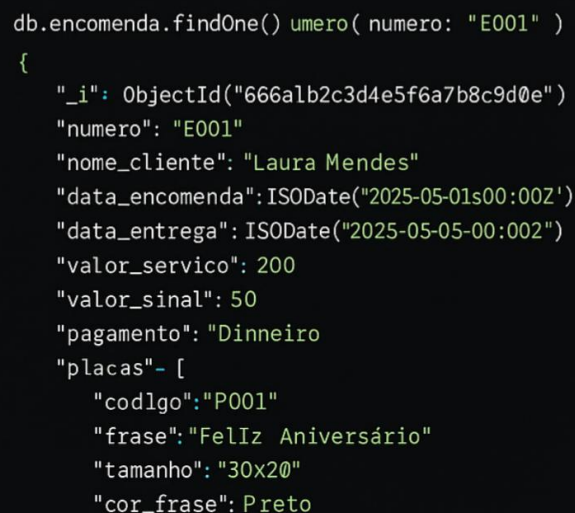
```
db.encomenda.findOne({ numero: "E001" });
```

Saída esperada:

```
{
  "_id": ObjectId("666a1b2c3d4e5f6a7b8c9d0e"),
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "data_encomenda": ISODate("2025-05-01T00:00:00Z"),
  "data_entrega": ISODate("2025-05-05T00:00:00Z"),
  "valor_servico": 200,
  "valor_sinal": 50,
  "pagamento": "Dinheiro",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "tamanho": "30x20",
      "cor_frase": "Preto"
    }
  ]
}
```

Segue a imagem abaixo da consulta de uma encomenda realizada pelo número na prática:

Imagem 3.0



```
db.encomenda.findOne( numero( numero: "E001" )
{
  "_i": ObjectId("666a1b2c3d4e5f6a7b8c9d0e")
  "numero": "E001"
  "nome_cliente": "Laura Mendes"
  "data_encomenda": ISODate("2025-05-01s00:00Z")
  "data_entrega": ISODate("2025-05-05-00:002")
  "valor_servico": 200
  "valor_sinal": 50
  "pagamento": "Dinneiro"
  "placas"- [
    "codlgo": "P001"
    "frase": "FelIz Aniversário"
    "tamanho": "30x20"
    "cor_frase": Preto
```

MongoDB Compass

O MongoDB Compass foi a ferramenta gráfica utilizada para a visualização e manipulação dos documentos dentro das coleções durante o desenvolvimento do projeto. Sua interface amigável e intuitiva desempenhou um papel fundamental na interação com o banco de dados, especialmente durante as fases de testes e validação das operações.

Com o MongoDB Compass, foi possível realizar inserções, atualizações e consultas de forma visual e interativa, sem a necessidade exclusiva de utilização de comandos pela linha de comando. A ferramenta oferece campos pré-configurados para edição de documentos em formato JSON, possibilitando a adição e alteração de dados de maneira rápida e prática.

Além das operações CRUD básicas, o Compass também proporcionou recursos avançados, como a construção de filtros por meio de assistentes visuais, visualização de índices, análise de performance das consultas e a execução de pipelines de agregação com interface gráfica, o que facilitou a criação de relatórios e agrupamentos de dados diretamente pela ferramenta.

Outro aspecto importante foi a capacidade do MongoDB Compass de exibir estatísticas detalhadas das coleções, como quantidade de documentos, tamanho total armazenado e distribuição de campos, o que auxiliou na análise e validação da modelagem orientada a documentos.

Essa combinação de funcionalidades fez com que o MongoDB Compass se tornasse uma ferramenta essencial para o acompanhamento da evolução do banco de dados, tornando as tarefas de manipulação e análise mais acessíveis e compreensíveis, principalmente em um contexto acadêmico, onde a visualização clara dos dados facilita o entendimento do funcionamento interno do MongoDB.

Exemplo gráfico: Visualização de uma encomenda no MongoDB Compass.

Coleção: encomenda

- Visualização no MongoDB Compass (modo Tree View)

```
{
  "_id": ObjectId("666a1b2c3d4e5f6a7b8c9d0e"),
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "data_encomenda": ISODate("2025-05-01T00:00:00Z"),
  "data_entrega": ISODate("2025-05-05T00:00:00Z"),
  "valor_servico": 200,
  "valor_sinal": 50,
  "pagamento": "Dinheiro",
  "placas": [
    {
      "codigo": "P001",
      "frase": "Feliz Aniversário",
      "tamanho": "30x20",
      "cor_frase": "Preto",
      "cor_placa": "Branco",
      "valor": 70,
      "utensilios": "Moldura, Suporte"
    }
  ]
}
```

```
]
}
```

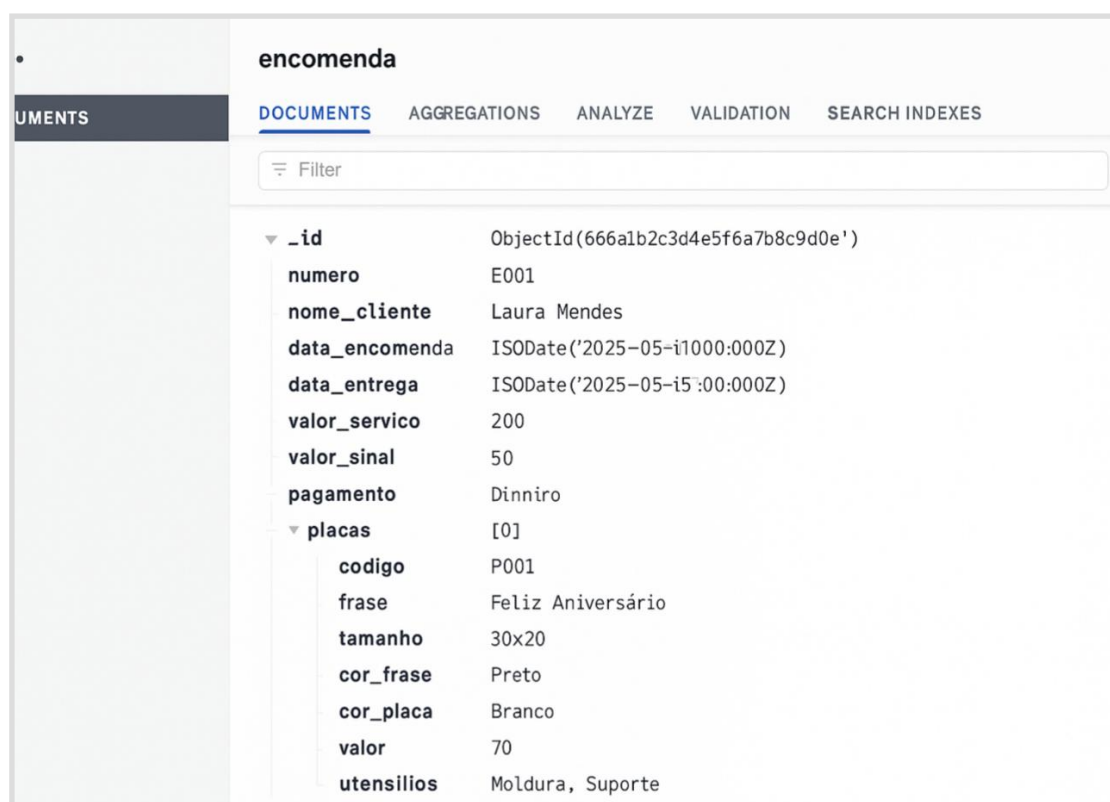
No MongoDB Compass, a visualização dos documentos dentro das coleções é apresentada em um formato gráfico de árvore, o que facilita a navegação e compreensão da estrutura de dados. Cada documento é exibido com seus campos no formato chave/valor, sendo que cada campo pode ser expandido individualmente para detalhar seu conteúdo. Quando o documento contém arrays, como no caso do campo "placas", esses elementos aparecem como listas organizadas, que podem ser abertas e expandidas para visualizar cada item individualmente.

Além disso, os campos de data são apresentados no formato padrão ISODate, o que garante precisão no armazenamento, mas também permite a visualização em um formato mais legível diretamente pela interface gráfica da ferramenta. Um outro aspecto importante é a presença do campo `_id`, que é automaticamente gerado pelo MongoDB. Esse campo recebe um valor único do tipo ObjectId, utilizado como identificador principal de cada documento dentro da coleção, garantindo a integridade e a unicidade dos registros.

Essa estrutura visual oferecida pelo Compass torna o processo de análise e manipulação de documentos mais intuitivo, especialmente para usuários que estão migrando de bancos relacionais e ainda não estão habituados aos conceitos de bancos NoSQL orientados a documentos.

Segue a imagem abaixo do exemplo gráfico, onde é apresentado a visualização de uma encomenda no MongoDB Compass.

Imagem 4.0



Mongo Shell

Durante os testes das operações CRUD e consultas mais complexas, foi utilizada a Mongo Shell, a interface de linha de comando oficial do MongoDB. Essa ferramenta é essencial para administradores de banco de dados e desenvolvedores que desejam interagir diretamente com o banco de dados de forma rápida e flexível, sem a necessidade de uma interface gráfica.

Por meio da Mongo Shell, foram executados comandos fundamentais como `insertOne()`, para a inserção de documentos individuais nas coleções; `find()`, para a realização de consultas simples ou com filtros específicos; e `updateOne()`, utilizado para modificar documentos existentes com base em critérios definidos. Além dessas operações básicas, também foram realizados testes com consultas mais avançadas utilizando o método `aggregate()`, que permite trabalhar com o framework de agregação do MongoDB, possibilitando o processamento de dados em múltiplos estágios, como filtros, projeções, agrupamentos e ordenações.

O uso da Mongo Shell durante o desenvolvimento foi importante para validar a integridade dos dados, testar a performance das consultas e garantir o correto funcionamento da modelagem proposta, servindo como uma ferramenta complementar ao MongoDB Compass e ao ambiente de desenvolvimento no Visual Studio Code.

Exemplo de uso no projeto: Consulta rápida de encomenda por número.

Código:

```
db.encomenda.findOne({ numero: "E001" });
```

Saída esperada:

```
{
  "_id": ObjectId("666a1b2c3d4e5f6a7b8c9d0e"),
  "numero": "E001",
  "nome_cliente": "Laura Mendes",
  "data_encomenda": ISODate("2025-05-01T00:00:00Z"),
  "data_entrega": ISODate("2025-05-05T00:00:00Z"),
  "valor_servico": 200,
  "valor_sinal": 50,
  "pagamento": "Dinheiro"
}
```

Segue a imagem abaixo da consulta rápida de uma encomenda realizada pelo número na prática:

Imagem 5.0

```
db.encomenda.fihdOne( { numero: "E001" } )
{
  "id" ObjectId("666a1b2c3d4e5f6a7b8c9d0e")
  "numero"."E001"
  "nome_cliente": Laura Mendes
  "data_encomenda": 2025-05-01T00:00:00Z
  "data_entrega": 2025-05-05T00:00:00Z"
  "valor_servico": 200
  "valor_sinal" 50
  "pagamento"."Dinheiro"
}
```

Visual Studio Code

O editor de código Visual Studio Code (VS Code) foi amplamente empregado durante o desenvolvimento do projeto para diversas finalidades relacionadas ao banco de dados MongoDB. Sua interface leve e altamente personalizável, combinada com a vasta gama de extensões disponíveis, tornou o processo de escrita, organização e execução dos scripts muito mais eficiente.

Foram desenvolvidos scripts de criação de documentos, comandos CRUD e exemplos de códigos de integração com o MongoDB, simulando cenários de uso real em aplicações web, principalmente com a utilização do Node.js e do driver oficial do MongoDB para JavaScript. Dessa forma, foi possível exemplificar como uma aplicação pode se conectar ao banco de dados, realizar operações como inserções, consultas, atualizações e até mesmo processos de agregação.

Além disso, o Visual Studio Code foi fundamental para a organização e o acompanhamento do código das consultas, permitindo uma melhor estruturação dos arquivos e facilitando tanto a manutenção quanto a realização de futuras modificações. Com o suporte a realce de sintaxe, o preenchimento automático inteligente e a integração com sistemas de controle de versão como o Git, o ambiente proporcionou maior produtividade e segurança durante o desenvolvimento e os testes de integração entre a aplicação e o banco de dados MongoDB.

Exemplo de código Node.js para consultar encomendas.

Objetivo desse código: Conectar no MongoDB, buscar a encomenda de número E001, e exibir o resultado no console.

Segue o código abaixo:

```
const { MongoClient } = require('mongodb');
async function main() {
  const uri = "mongodb://localhost:27017";
  const client = new MongoClient(uri);
  try {
    await client.connect();
    const database = client.db("grafica");
    const encomendas = database.collection("encomenda");
    const resultado = await encomendas.findOne({ numero: "E001" });
    console.log(resultado);
  } finally {
    await client.close();
  }
}
main().catch(console.error);
```

Segue a imagem abaixo do código Node.js para consultar encomendas:

Imagem 6.0

```
const async gction(mongodb' { 'mongodb')
use client = connectto(n(clest")
  filter=find0me(="F001)
  client.close()
} finally
```

Comparação com MySQL Workbench

Vale destacar que, no projeto relacional original, a ferramenta de apoio utilizada foi o MySQL Workbench, essencial para a modelagem lógica, criação do modelo físico e geração de scripts SQL utilizados na implementação do banco de dados relacional. O MySQL Workbench ofereceu uma interface gráfica poderosa que facilitou a definição de tabelas, relacionamentos, chaves primárias e estrangeiras, além da execução e depuração de consultas SQL durante as fases iniciais do projeto.

No entanto, com a decisão de migrar para o MongoDB e adotar uma abordagem NoSQL orientada a documentos, foi necessário readequar todo o fluxo de trabalho para o uso de ferramentas específicas desse novo ambiente tecnológico. Essa mudança envolveu uma transformação conceitual na forma de pensar a modelagem de dados, passando de um esquema rigidamente estruturado para um modelo mais flexível e dinâmico, baseado em documentos JSON.

Ferramentas como o MongoDB Compass e o Mongo Shell passaram a desempenhar um papel central nas etapas de desenvolvimento, testes e validação de dados. O Compass, com sua interface gráfica, possibilitou a visualização e manipulação das coleções de maneira mais acessível, enquanto o Mongo Shell permitiu a execução direta de comandos utilizando a linguagem própria do MongoDB (MQL), oferecendo maior controle para consultas avançadas e operações administrativas.

Essa transição de ferramentas refletiu a própria mudança de paradigma entre os dois tipos de banco de dados: enquanto o MySQL Workbench representava a lógica de um modelo relacional com foco em integridade referencial e normalização, as ferramentas do MongoDB trouxeram uma abordagem mais flexível e orientada a desempenho, favorecendo a desnormalização e a rapidez no acesso aos dados.

Dessa forma, o projeto não apenas exigiu um aprendizado técnico sobre a tecnologia NoSQL, mas também uma adaptação metodológica, com a escolha de ferramentas adequadas para garantir a qualidade, a consistência e a eficiência no novo ambiente de desenvolvimento.

Descrição: No projeto original em MySQL Workbench, os dados de uma encomenda eram visualizados em formato de linhas e colunas dentro de uma tabela, exigindo uso de várias tabelas e joins para reunir informações de cliente, placas, pagamento e entrega.

Exemplo de saída (antes no SQL):

numero	nome_cliente	data_encomenda	valor_servico	...
E001	Laura Mendes	2025-05-01	200	...

Comparação: No MongoDB, todas essas informações agora aparecem dentro de um único documento JSON, com subdocumentos para placas, o que facilita a leitura e reduz a complexidade de consultas.

Resumo Comparativo Prático

Ferramenta	Função Principal
MongoDB Community Server	Execução local do banco de dados
Mongo Shell	Consultas e manipulação via linha de comando
Visual Studio Code	Desenvolvimento de código de integração
MySQL Workbench (comparativo)	Modelagem e visualização de dados SQL

Segue a imagem abaixo dos dados de uma encomenda visualizados em formato de linhas e colunas dentro de uma tabela.

Imagem 7.0

< →	Order E001	A ⋮ ⌕
	▼ _id	66631d*666a12c3de5f6a7b5c9dde)
	nunero	6001
	nome_cliente	Laura Mendes
	data_encomenda	2025-05-01 T00:00:00:00Z
	data_entrega	2025-05-05 T05:00:00:00Z
	valor_servico	200
	valor_sinal	50
	pagamento	Dinheiro

Considerações sobre a migração de SGBD

A migração do sistema de gestão da gráfica de placas personalizadas de um banco de dados relacional (MySQL) para um banco NoSQL (MongoDB) representou um desafio técnico e conceitual importante no desenvolvimento deste projeto. Esta mudança de paradigma exigiu não apenas a conversão de estruturas de dados, mas também uma adaptação na forma de pensar e manipular as informações.

Desafios encontrados durante o processo de migração:

Conversão de entidades para documentos

A transformação de tabelas relacionais para documentos JSON exigiu uma análise cuidadosa de quais entidades deveriam ser representadas como subdocumentos (embedding) e quais deveriam permanecer como coleções separadas com referências (referencing).

Adaptação de consultas

Consultas SQL envolvendo múltiplas junções precisaram ser reescritas utilizando os recursos de agregação (aggregate) e operações de lookup do MongoDB.

Mudança no controle de integridade referencial

Enquanto no MySQL a integridade dos relacionamentos era garantida por meio de chaves estrangeiras, no MongoDB essa responsabilidade passou a ser controlada pela aplicação ou através de práticas de modelagem cuidadosa.

Familiarização com os operadores MongoDB

Houve a necessidade de um período de adaptação para entender e aplicar corretamente os operadores específicos do MongoDB, como \$set, \$lookup, \$match, \$group, entre outros.

Benefícios observados

Apesar dos desafios iniciais, a migração proporcionou diversos benefícios práticos:

Maior flexibilidade na estruturação de dados:

Agora é possível armazenar diferentes formatos de placas, frases personalizadas e outras características únicas em um único documento.

Redução da complexidade das consultas:

A necessidade de múltiplas junções foi eliminada, permitindo consultas mais simples e diretas.

Melhor desempenho em leitura:

Recuperar todos os dados de uma encomenda ou de um cliente agora exige apenas uma única consulta.

Facilidade de manutenção e escalabilidade:

O sistema está mais preparado para futuras mudanças no modelo de dados, com possibilidade de incluir novos campos nos documentos sem a necessidade de alterar esquemas predefinidos.

Lições aprendidas

O processo de migração reforçou a importância de:

Analisar o perfil do sistema antes de escolher o SGBD:

Cada aplicação tem suas próprias necessidades em termos de estrutura de dados, volume, performance e flexibilidade.

Planejar bem a modelagem de documentos:

Decisões sobre embedding ou referencing impactam diretamente no desempenho e manutenção futura.

Conhecer bem as ferramentas de ambos os paradigmas:

Ter domínio tanto de SQL quanto de NoSQL amplia a capacidade de adaptação e escolha de soluções mais adequadas.

Quando optar por NoSQL no lugar de SQL

A experiência com este projeto indicou que o uso de MongoDB é especialmente vantajoso quando:

- Há grande variedade e personalização dos dados.
- Os requisitos do sistema estão em constante mudança.
- O volume de leitura é elevado e exige baixo tempo de resposta.
- Não há necessidade extrema de transações complexas entre múltiplas entidades.

Com essa experiência, conclui-se que o MongoDB atende com excelência as necessidades da gráfica de placas personalizadas, oferecendo uma solução moderna, eficiente e alinhada com as tendências de desenvolvimento de aplicações web e mobile.

4. O desenvolvimento deste projeto permitiu uma análise aprofundada da transição de um sistema de banco de dados relacional para um banco de dados não relacional, mantendo o tema e o contexto original da gráfica de placas personalizadas. A implementação com o MongoDB demonstrou que a adoção de um banco NoSQL é uma alternativa viável e eficaz para aplicações que exigem maior flexibilidade na modelagem de dados e escalabilidade.

Durante o processo de migração, foram enfrentados desafios relacionados à conversão da estrutura de tabelas para documentos, bem como à reformulação das consultas que antes dependiam fortemente de junções (joins) no MySQL. Entretanto, esses obstáculos foram superados com o uso das ferramentas apropriadas e com o entendimento aprofundado do modelo orientado a documentos adotado pelo MongoDB.

Entre os principais benefícios alcançados, destacam-se a simplificação na consulta e armazenamento de dados personalizados, a redução na complexidade de operações de leitura, e a possibilidade de estruturar os documentos de forma mais alinhada com a realidade do negócio da gráfica. A utilização de coleções, subdocumentos e arrays permitiu um gerenciamento de dados mais eficiente, representando com fidelidade as encomendas, clientes, produtos, transportadoras e demais entidades envolvidas no processo de produção e entrega de placas.

Além disso, a fácil integração do MongoDB com linguagens de programação modernas, como Node.js, abre oportunidades para futuras evoluções do sistema, incluindo a criação de APIs, painéis de gestão e aplicações web/mobile para atendimento ao cliente.

Por fim, este trabalho não apenas alcançou os objetivos propostos, como também proporcionou um importante aprendizado prático sobre o uso de bancos NoSQL, sua modelagem, ferramentas de apoio e técnicas de integração com aplicações modernas.

5. **Referências Bibliográficas**

Conceitos de modelagem de dados. Disponível em: <https://www.mongodb.com/pt-br/docs/manual/data-modeling/concepts/> .

Acesso em 13 de Junho de 2025.

O que é NoSQL? Disponível em: <https://www.oracle.com/br/database/nosql/what-is-nosql/>.

Acesso em 13 de Junho de 2025.

O que é MongoDB? Disponível em: <https://www.ibm.com/br-pt/topics/mongodb> .

Acesso em 13 de Junho de 2025.

MongoDB: o que é, quais suas características e benefícios e como trabalhar nessa ferramenta? Disponível em: <https://www.alura.com.br/artigos/mongodb?srsId=AfmBOorpUzFkIJuWVJ1UluphkT55NvSYLv2yxjoZXI7ZcGiRlswDqcOb>

Acesso em 13 de Junho de 2025.