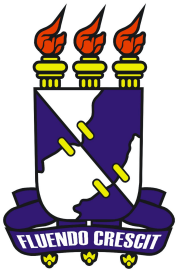


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

João Pedro Cardoso Arruda

Lucas Santana de Jesus

Patrick Augusto Oliveira Pinheiro

Paulo Henrique dos Santos Reis

BANCO DE DADOS 1

São Cristóvão - SE
29/07/2025

Trabalho prático - Projeto Lógico - Parte 2.2

Prof. André Britto de Carvalho

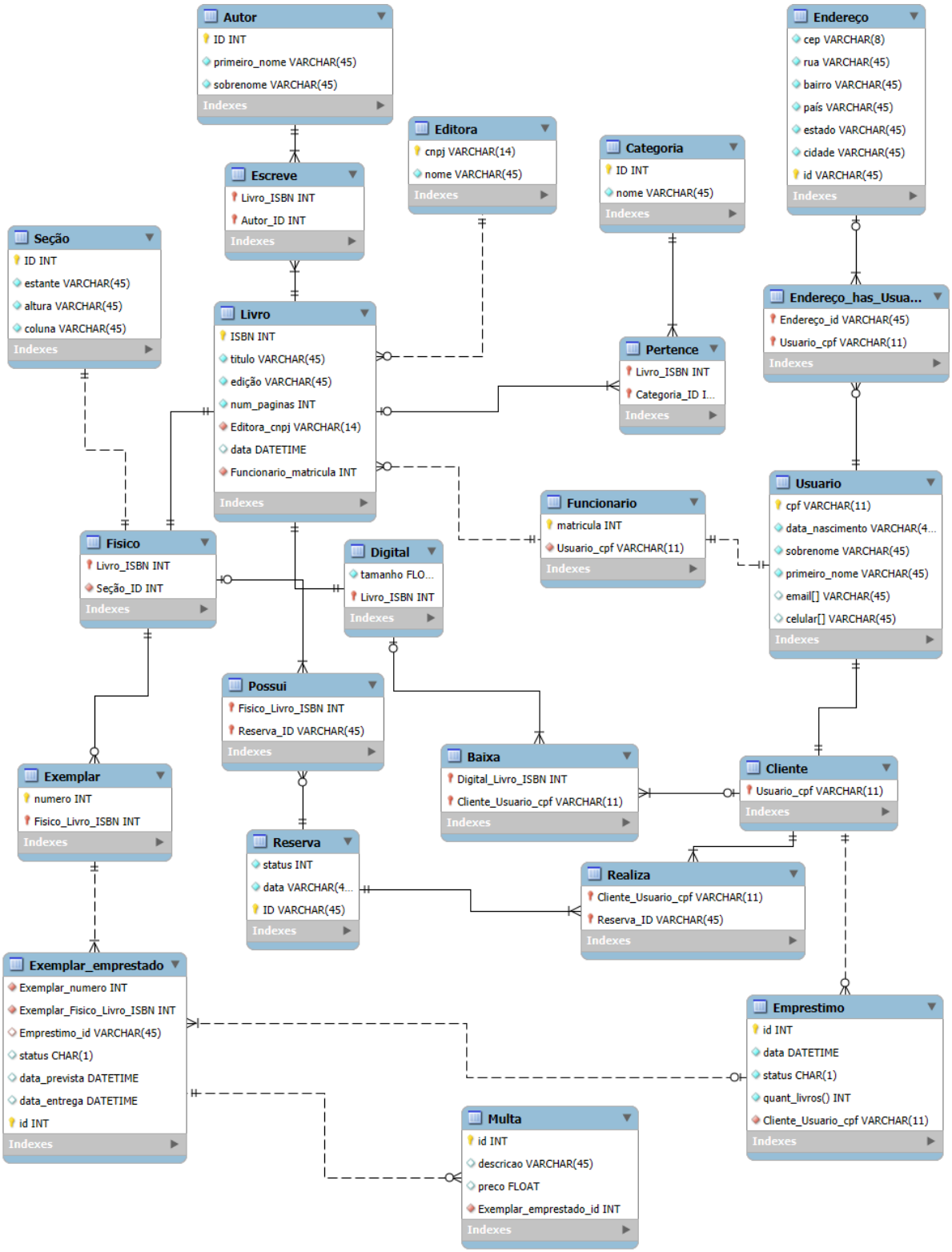
São Cristóvão - SE
29/07/2025

1. INTRODUÇÃO

Nessa parte do projeto foi feito o mapeamento do banco do modelo relacional para um modelo baseado em documentos utilizando do SGBD MongoDB. Para isso foram modeladas e codificadas as coleções do MongoDB para representar as entidades e seus relacionamentos e foi utilizado o recurso do “$jsonSchema” para que possa ser feita a validação das estruturas das coleções, o que garante que os documentos inseridos respeitem um formato e possuam todos os atributos correspondentes das entidades do modelo incluindo ainda restrições de domínio.

São Cristóvão - SE
29/07/2025

Modelo lógico criado na etapa anterior:



São Cristóvão - SE
29/07/2025

2. MAPEAMENTO

O mapeamento foi feito seguindo as boas práticas recomendadas na documentação oficial do MongoDB, utilizando geralmente de coleções para representar as entidades e de referências ou aninhamento de documentos para representar as relações. As decisões de mapeamento foram feitas levando em consideração a fidelidade ao modelo original e suas restrições, o desempenho das consultas e a eficiência de armazenamento.

2.1. USUÁRIOS E ENDEREÇO

No modelo relacional em PostgreSQL, os dados dos usuários estão distribuídos entre três tabelas distintas: Usuário, que armazena os dados pessoais e de contato como CPF, nome, e-mail e celular; Endereço, responsável por manter os dados de localização como rua, bairro, cidade e CEP; e a tabela intermediária Endereco_Usuario, que realiza a associação entre usuários e seus respectivos endereços. Esse formato segue os princípios da normalização, promovendo a integridade referencial e evitando redundância de dados.

No MongoDB, as entidades Usuário e Endereço foram representadas em uma única coleção chamada usuários. Cada documento representa um usuário completo, contendo não apenas seus dados pessoais e contatos, mas também um subdocumento aninhado chamado endereço, com todas as informações de localização. Essa decisão permite que os dados do usuário sejam acessados de forma direta e eficiente, sem a necessidade de múltiplas consultas ou junções entre coleções. Além disso, para os atributos de e-mail, celular, data e cep é feita a validação por meio do “$jsonSchema” para verificar se as strings seguem um formato válido.

```
db.createCollection("usuarios", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "primeiro_nome", "sobrenome", "data_nascimento", "email", "celular", "endereco"],
      properties: {
        _id: { bsonType: "string" },
        primeiro_nome: { bsonType: "string" },
        sobrenome: { bsonType: "string" },
        data_nascimento: { bsonType: "string", pattern: "^\\d{4}-\\d{2}-\\d{2}$" },
        email: {
          bsonType: "array",
          items: { bsonType: "string", pattern: "^.+@.+\\..+$" }
        },
        celular: {
          bsonType: "array",
          items: { bsonType: "string", pattern: "^\\d{11}$" }
        },
        endereco: {
          bsonType: "object",
          required: ["pais", "estado", "cidade", "bairro", "rua", "cep"],
          properties: {
            pais: { bsonType: "string" },
            estado: { bsonType: "string" },
            cidade: { bsonType: "string" },
            bairro: { bsonType: "string" },
            rua: { bsonType: "string" },
            cep: { bsonType: "string", pattern: "^\\d{5}-\\d{3}$" }
          }
        }
      }
    }
  }});
);

```

São Cristóvão - SE
29/07/2025

2.2. FUNCIONÁRIO

Na estrutura relacional, os funcionários são definidos pela tabela Funcionario, que armazena sua matrícula e o CPF do usuário associado, com uma chave estrangeira para a tabela Usuario. Essa separação permite que informações pessoais e funcionais fiquem em tabelas distintas, mantendo a normalização.

No MongoDB, foi criada a coleção funcionarios, com documentos que contêm os campos matricula e usuario_cpf que referencia o cpf da coleção usuários. Esse modelo mantém a ligação com os dados do usuário por meio do CPF, mas não incorpora os dados pessoais diretamente no documento do funcionário. Essa escolha permite manter uma separação conceitual simples.

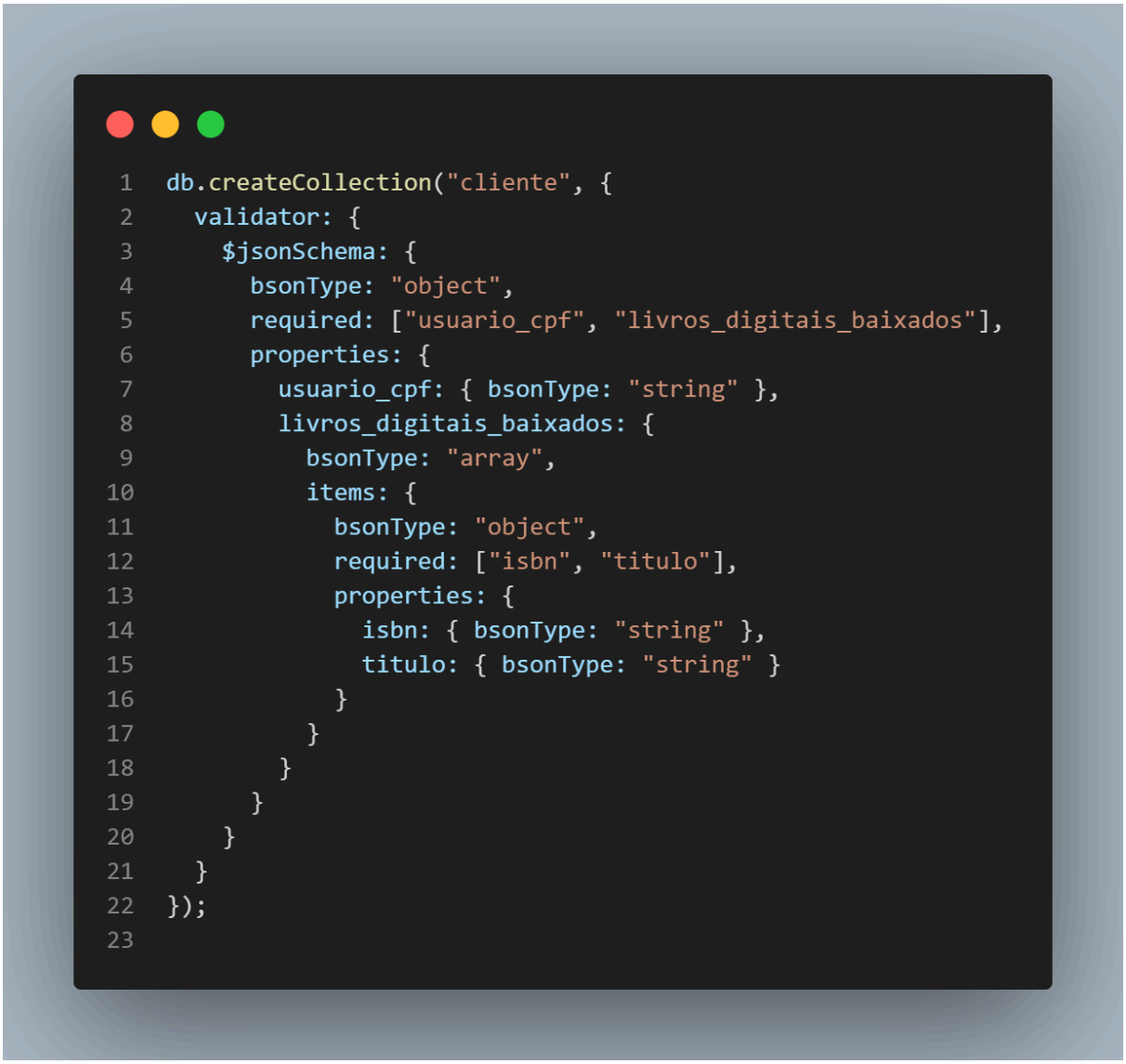
```
db.createCollection("funcionarios", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["matricula", "usuario_cpf"],
      properties: {
        matricula: { bsonType: "int" },
        usuario_cpf: { bsonType: "string" }
      }
    }
  }
});

```

2.3. CLIENTE

No modelo relacional, Cliente é uma entidade especializada de Usuario, definida por uma chave estrangeira (CPF) e que se relaciona com diversas outras tabelas, como Emprestimo, Reserva e Baixa (downloads de livros digitais).

No MongoDB, o cliente é representado por um documento na coleção cliente, contendo o usuario_cpf e um array chamado livros_digitais_baixados ambos são referências. Cada item do array representa um livro digital já baixado pelo cliente, com seu ISBN e título. Essa abordagem facilita a agregação de comportamento histórico dentro do próprio documento do cliente, facilitando a consulta direta às obras baixadas por cada usuário.



```
db.createCollection("cliente", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["usuario_cpf", "livros_digitais_baixados"],
      properties: {
        usuario_cpf: { bsonType: "string" },
        livros_digitais_baixados: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["isbn", "titulo"],
            properties: {
              isbn: { bsonType: "string" },
              titulo: { bsonType: "string" }
            }
          }
        }
      }
    }
  }
});
```

2.4. EDITORAS

A tabela Editora no PostgreSQL armazena o cnpj (como chave primária) e o nome da editora, garantindo unicidade.

Em MongoDB, essa estrutura é espelhada de forma simples na coleção editoras, em que o campo _id armazena o CNPJ e nome mantém o mesmo propósito. A validação garante que os tipos estejam corretos, mantendo a consistência mínima esperada dos dados.

```
db.createCollection("editoras", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "nome"],
      properties: {
        _id: { bsonType: "string" },
        nome: { bsonType: "string" }
      }
    }
  });
```

2.5. AUTORES

No modelo relacional, Autor possui um identificador, nome e sobrenome, e se relaciona com a tabela Livro por meio da tabela associativa Escreve.

No MongoDB, cada autor é representado por um documento na coleção autores, contendo seu _id e um objeto nome, com os campos primeiro_nome e sobrenome. Além disso, foi criada uma estrutura embutida para referenciar a coleção livros chamada livros_escritos, que armazena uma lista dos livros associados ao autor, com seus respectivos ISBNs e títulos. Isso evita a necessidade

de junções e fornece uma visualização direta do histórico do autor com suas publicações.

```
db.createCollection("autores", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "nome", "livros_escritos"],
      properties: {
        _id: { bsonType: "int" },
        nome: {
          bsonType: "object",
          required: ["primeiro_nome", "sobrenome"],
          properties: {
            primeiro_nome: { bsonType: "string" },
            sobrenome: { bsonType: "string" }
          }
        },
        livros_escritos: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["isbn", "titulo"],
            properties: {
              isbn: { bsonType: "string" },
              titulo: { bsonType: "string" }
            }
          }
        }
      }
    }
  }
});
```

2.6. LIVROS

A modelagem relacional do livro é complexa e envolve diversas tabelas: Livro, Digital, Fisico, Categoria, Pertence, Escreve e Exemplar. Cada aspecto do livro é armazenado separadamente, utilizando herança de tabelas para distinguir os formatos digital e físico.

No MongoDB, foi criada a coleção livros, que representa um documento unificado por livro. Esse documento inclui os dados básicos do livro, dados de cadastro e publicação, além de campos embutidos como autores, categorias e os campos digital (com tamanho_mb, se aplicável) e fisico (com secao_id e uma lista de exemplares) para armazenar as informações de cada versão caso exista. Essa unificação elimina a fragmentação dos dados, consolidando tudo em um único documento, o que reduz drasticamente a complexidade de consultas e melhora o desempenho em sistemas voltados à leitura intensiva.

```
db.createCollection("livros", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [
        "_id", "titulo", "edicao", "num_paginas", "data_cadastro",
        "editora_cnpj", "funcionario_matricula", "data_publicacao",
        "autores", "categorias", "digital", "fisico"
      ],
      properties: {
        _id: { bsonType: "string" },
        titulo: { bsonType: "string" },
        edicao: { bsonType: "string" },
        num_paginas: { bsonType: "int" },
        data_cadastro: { bsonType: "date" },
        editora_cnpj: { bsonType: "string" },
        funcionario_matricula: { bsonType: "int" },
        data_publicacao: { bsonType: "string", pattern: "^\\d{4}-\\d{2}-\\d{2}$" },
        autores: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["autor_id", "nome"],
            properties: {
              autor_id: { bsonType: "int" },
              nome: { bsonType: "string" }
            }
          }
        },
        categorias: {
          bsonType: "array",
          items: { bsonType: "string" }
        },
        digital: {
          bsonType: "object",
          properties: {
            tamanho_mb: { bsonType: "double" }
          }
        },
        fisico: {
          bsonType: "object",
          required: ["secao_id", "exemplares"],
          properties: {
            secao_id: { bsonType: "int" },
            exemplares: {
              bsonType: "array",
              items: {
                bsonType: "object",
                required: ["numero", "status"],
                properties: {
                  numero: { bsonType: "int" },
                  status: { bsonType: "string", enum: ["disponível", "emprestado"] }
                }
              }
            }
          }
        }
      }
    }
  }
});
```

2.7. RESERVAS

Na estrutura relacional, reservas são representadas pelas tabelas Reserva, Realiza e Possui, conectando clientes a livros físicos reservados.

No MongoDB, a coleção reservas concentra todos esses dados em um único documento, que contém o identificador da reserva (_id), status, data da reserva, CPF do cliente e uma lista de livros físicos associados (com ISBN). Isso facilita o rastreamento completo da reserva e de seus itens sem depender de múltiplas coleções ou relacionamentos.

```
db.createCollection("reservas", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "status", "data_reserva", "cliente_usuario_cpf", "livros_fisicos"],
      properties: {
        _id: { bsonType: "int" },
        status: { bsonType: "string" },
        data_reserva: { bsonType: "string", pattern: "^\\d{4}-\\d{2}-\\d{2}$" },
        cliente_usuario_cpf: { bsonType: "string" },
        livros_fisicos: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["isbn"],
            properties: {
              isbn: { bsonType: "string" }
            }
          }
        }
      }
    }
  }
});

```

2.8. SEÇÃO

A tabela relacional Seção define a localização física dos livros nas estantes. No MongoDB, ela é representada pela coleção secao, com cada documento contendo o _id da seção e um subdocumento localizador que guarda os dados de estante, coluna e altura. Esse formato reflete diretamente a estrutura original e facilita a leitura dos dados sem a complexidade de junções.

```
db.createCollection("secao", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "localizador"],
      properties: {
        _id: { bsonType: "int" },
        localizador: {
          bsonType: "object",
          required: ["estante", "coluna", "altura"],
          properties: {
            estante: { bsonType: "string" },
            coluna: { bsonType: "int" },
            altura: { bsonType: "int" }
          }
        }
      }
    }
  }
});

```

2.9. EMPRÉSTIMOS

O empréstimo no modelo relacional é representado pelas tabelas Emprestimo, Exemplar_emprestado e seus vínculos com Cliente, Exemplar, Livro e Multa.

No MongoDB, foi criada a coleção emprestimos, na qual cada documento representa um empréstimo realizado por um cliente. Ele contém o identificador, CPF do cliente, data, status, quantidade de livros e um array de itens, que representam os exemplares emprestados. Cada item contém informações completas do exemplar e das datas de empréstimo, prevista e real. Isso elimina a necessidade de joins entre várias tabelas para reconstruir a estrutura completa de um empréstimo.

```
db.createCollection("emprestimos", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "cliente_usuario_cpf", "data_emprestimo", "status", "quant_livros", "itens"],
      properties: {
        _id: { bsonType: "int" },
        cliente_usuario_cpf: { bsonType: "string" },
        data_emprestimo: { bsonType: "date" },
        status: { bsonType: "string" },
        quant_livros: { bsonType: "int" },
        itens: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["_id", "livro_isbn", "exemplar_numero", "data_prevista"],
            properties: {
              _id: { bsonType: "int" },
              livro_isbn: { bsonType: "string" },
              exemplar_numero: { bsonType: "int" },
              data_prevista: { bsonType: "string", pattern: "^\\d{4}-\\d{2}-\\d{2}$" },
              data_entrega: {
                bsonType: ["null", "string"],
                pattern: "^\\d{4}-\\d{2}-\\d{2}$"
              }
            }
          }
        }
      }
    }
  }
});
```

2.10. MULTAS

No modelo relacional, a Multa está ligada a um Exemplar_emprestado específico e inclui uma descrição e valor.

Na coleção multas do MongoDB, cada documento armazena os mesmos dados: _id, descrição, preço, emprestimo_id e item_emprestimo_id (referindo-se ao item do array itens no documento de empréstimo). Apesar de manter referências, a estrutura é mais simples e direta, e a lógica pode ser gerenciada diretamente na aplicação.

```
db.createCollection("multas", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "descricao", "preco", "emprestimo_id", "item_emprestimo_id"],
      properties: {
        _id: { bsonType: "int" },
        descricao: { bsonType: "string" },
        preco: { bsonType: "double" },
        emprestimo_id: { bsonType: "int" },
        item_emprestimo_id: { bsonType: "int" }
      }
    }
  }
});
```

3. REFERÊNCIAS

MONGODB. *Mapeando relacionamentos entre documentos*. Disponível em: https://www.mongodb.com/pt-br/docs/manual/data-modeling/schema-design-process/map-relationships/#std-label-data-modeling-map-relationships. Acesso em: 29 jul. 2025.

MONGODB. *Modelagem de dados no MongoDB*. Disponível em: https://www.mongodb.com/pt-br/docs/manual/data-modeling/. Acesso em: 29 jul. 2025.

MONGODB. *Mapeamento de esquema no Relational Migrator*. Disponível em: https://www.mongodb.com/pt-br/docs/relational-migrator/mapping-rules/schema-mapping/. Acesso em: 29 jul. 2025.