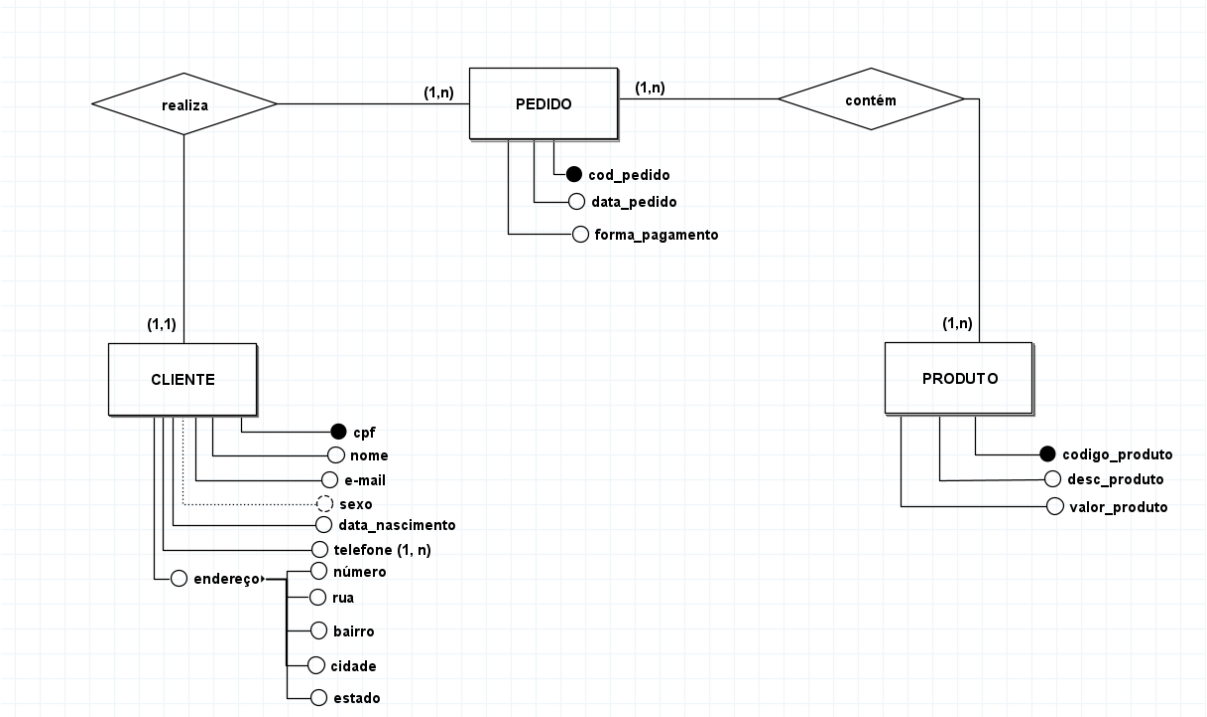

Modelo Conceitual	**Tipo no MC**	**MongoDB**	**Tipo no MongoDB**	**Observação**
Cliente	Entidade	Cliente	Coleção	Campo obrigatório
CPF	Atributo identificador	Campo de cliente	Campo simples	Campo obrigatório
Nome	Atributo simples	Campo de cliente	Campo simples	Campo obrigatório
E-mail	Atributo simples	Campo de cliente	Campo simples	Campo obrigatório
Sexo	Atributo opcional	Campo de cliente	Campo simples	Campo opcional
Data de nascimento	Atributo simples	Campo de cliente	Campo simples	Campo obrigatório
Telefone	Atributo multivalorado	Campo de cliente	Campo-Array	Campo obrigatório
Endereço	Atributo composto	Campo de cliente	Campo - estrutura	Campo obrigatório

			embutida	
Pedido	Entidade	Pedido	Coleção	Principal entidade do projeto
Código do pedido	Atributo identificador	Campo de pedido	Campo simples	Campo obrigatório
Data do pedido	Atributo simples	Campo de pedido	Campo simples	Campo obrigatório
Forma de pagamento	Atributo simples	Campo de pedido	Campo simples	Campo obrigatório
Realizado[Cliente]	relacionamento	Campo de pedido	Referência única	Campo obrigatório/ Referência através do _id o cliente que fez o pedido
Contém[Produto]	relacionamento	Campo de pedido	Campo - Array de referência a itens	Campo obrigatório/ Referência através dos _id os produtos que integram o pedi
Produto	Entidade	Produto	Coleção	
Código do produto	Atributo identificador	Campo de produto	Campo simples	Campo obrigatório
Descrição do produto	Atributo simples	Campo de produto	Campo simples	Campo obrigatório
Valor do produto	Atributo simples	Campo de produto	Campo simples	Campo obrigatório

```
// Criação das coleções

db.createCollection("Cliente", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "CPF", "Nome", "Email", "dataNascimento", "Endereco"],
      properties: {
        _id: {
          bsonType: "objectId",
          description: "Deve ser um ObjectId único para o cliente."
```

```
},
CPF: {
  bsonType: "string",
  description: "Deve ser uma string contendo o CPF do cliente."
},
Nome: {
  bsonType: "string",
  description: "Deve ser uma string contendo o nome do cliente."
},
Email: {
  bsonType: "string",
  pattern: "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
  description: "Deve ser uma string contendo um endereço de e-mail válido."
},
Sexo: {
  bsonType: "string",
  enum: ["Masculino", "Feminino", "Outro"],
  description: "Deve ser uma string com valor 'Masculino', 'Feminino' ou 'Outro'."
},
dataNascimento: {
  bsonType: "date",
  description: "Deve ser uma data válida de nascimento."
},
telefone: {
  bsonType: "array",
  description: "Pode ser uma array contendo números de telefone.",
  items: {
    bsonType: "string",
    description: "Cada elemento deve ser uma string contendo um número de telefone."
  }
},
Endereco: {
  bsonType: "object",
  required: ["numero", "rua", "bairro", "cidade", "estado"],
  properties: {
    numero: {
      bsonType: "int",
      description: "Deve ser um número inteiro representando o número do endereço."
    },
    rua: {
      bsonType: "string",
      description: "Deve ser uma string contendo o nome da rua."
    },
    bairro: {
      bsonType: "string",
      description: "Deve ser uma string contendo o nome do bairro."
    },
    cidade: {
```

```
            bsonType: "string",
            description: "Deve ser uma string contendo o nome da cidade."
          },
          estado: {
            bsonType: "string",
            description: "Deve ser uma string contendo a sigla do estado."
          }
        }
      }
    }
  }
});


// Criação da coleção de Produto com validação
db.createCollection("Produto", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "codigoProduto", "descricaoProduto", "valorProduto"],
      properties: {
        _id: {
          bsonType: "objectId",
          description: "Deve ser um ObjectId único para o produto."
        },
        codigoProduto: {
          bsonType: "string",
          description: "Deve ser uma string contendo o código do produto."
        },
        descricaoProduto: {
          bsonType: "string",
          description: "Deve ser uma string contendo a descrição do produto."
        },
        valorProduto: {
          bsonType: "double",
          description: "Deve ser um número contendo o valor do produto."
        }
      }
    }
  }
});


db.createCollection("Pedido", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
```

```
      required: ["codigoPedido", "dataPedido", "formaPagamento", "clienteId", "produtos"],
      properties: {
        codigoPedido: {
          bsonType: "string",
          description: "Deve ser uma string contendo o código do pedido."
        },
        dataPedido: {
          bsonType: "string",
          description: "Deve ser uma string contendo a data do pedido."
        },
        formaPagamento: {
          bsonType: "string",
          description: "Deve ser uma string contendo a forma de pagamento."
        },
        clienteId: {
          bsonType: "objectId",
          description: "Deve ser um ObjectId único referenciando o cliente."
        },
        produtos: {
          bsonType: "array",
          minItems: 1,  // Garante que haja no mínimo um elemento no array
          description: "Deve ser um array contendo ObjectId(s) referenciando o(s) produto(s).",
          items: {
            bsonType: "objectId",
            description: "Cada elemento deve ser um ObjectId referenciando um produto."
          }
        }
      }
    }
  }
});
```

```
// ------------------------------------------------------------------------------------------------------
```

// Criação dos índices

// Criando um índice para a data do pedido em ordem descendente, aprimorando o desempenho das consultas de ordenação por data e agilizando a recuperação dos primeiros pedidos. O índice visa otimizar a eficiência na busca e classificação de registros com base na cronologia dos pedidos.

```
db.Pedido.createIndex({dataPedido: -1 })
```

```
// ------------------------------------------------------------------------------------------------------
```

// Criando um índice para a data de nascimento do cliente, buscando melhorar o desempenho em consultas e operações relacionadas a esse campo. Essa medida acelera a recuperação de dados e otimiza a identificação rápida dos usuários mais antigos.

```
db.Cliente.createIndex({ dataNascimento: -1 })
```

// ---

// Inserções de Cliente

```
db.Cliente.insertOne({
  _id: ObjectId("657f7976fc6a7aacea011641"),
  CPF: "11122233344",
  Nome: "Ana Silva",
  Email: "ana@example.com",
  Sexo: "Feminino",
  dataNascimento: new Date("1984-04-12"),
  telefone: ["111111111", "222222222"],
  Endereco: {
    numero: 123,
    rua: "Rua das Flores",
    bairro: "Centro",
    cidade: "São Paulo",
    estado: "SP"
  }
})


db.Cliente.insertOne({
  _id: ObjectId("657f797cfc6a7aacea011642"),
  CPF: "22233344455",
  Nome: "Bruno Oliveira",
  Email: "bruno@example.com",
  Sexo: "Masculino",
  dataNascimento: new Date("1970-05-12"),
  telefone: ["333333333", "444444444"],
  Endereco: {
    numero: 456,
    rua: "Avenida Principal",
    bairro: "Copacabana",
    cidade: "Rio de Janeiro",
    estado: "RJ"
  }
})


db.Cliente.insertOne({
  _id: ObjectId("657f7983fc6a7aacea011643"),
```

```
  CPF: "33344455566",
  Nome: "Camila Santos",
  Email: "camila@example.com",
  Sexo: "Feminino",
  dataNascimento: new Date("1982-09-12"),
  telefone: ["555555555", "666666666"],
  Endereco: {
    numero: 789,
    rua: "Rua do Comércio",
    bairro: "Boa Viagem",
    cidade: "Recife",
    estado: "PE"
  }
})


db.Cliente.insertOne({
  _id: ObjectId("657f7988fc6a7aacea011644"),
  CPF: "44455566677",
  Nome: "Daniel Pereira",
  Email: "daniel@example.com",
  Sexo: "Masculino",
  dataNascimento:new Date("2000-01-11"),
  telefone: ["777777777", "888888888"],
  Endereco: {
    numero: 1011,
    rua: "Avenida dos Esportes",
    bairro: "Jardim Botânico",
    cidade: "Curitiba",
    estado: "PR"
  }
})


db.Cliente.insertOne({
  _id: ObjectId("657f798dfc6a7aacea011645"),
  CPF: "55566677788",
  Nome: "Elisa Souza",
  Email: "elisa@example.com",
  Sexo: "Feminino",
  dataNascimento: new Date("1960-03-12"),
  telefone: ["999999999", "101010101"],
  Endereco: {
    numero: 1213,
    rua: "Rua da Paz",
    bairro: "Itaim Bibi",
    cidade: "São Paulo",
    estado: "SP"
```

```
  }
})


db.Cliente.insertOne({
  _id: ObjectId("657f7993fc6a7aacea011646"),
  CPF: "66677788899",
  Nome: "Fernando Silva",
  Email: "fernando@example.com",
  Sexo: "Masculino",
  dataNascimento: new Date("1999-04-12"),
  telefone: ["121212121", "131313131"],
  Endereco: {
    numero: 1415,
    rua: "Alameda das Flores",
    bairro: "Vila Madalena",
    cidade: "São Paulo",
    estado: "SP"
  }
})


// Inserções de Produto

db.Produto.insertOne({
  _id: ObjectId("657f06a49ec1681c515b140b"),
  codigoProduto: "P001",
  descricaoProduto: "Strogonoff de Frango",
  valorProduto: 25.99
})


db.Produto.insertOne({
  _id: ObjectId("657f06b09ec1681c515b140c"),
  codigoProduto: "P002",
  descricaoProduto: "Pizza Margherita",
  valorProduto: 29.99
})


db.Produto.insertOne({
  _id: ObjectId("657f06b99ec1681c515b140d"),
  codigoProduto: "P003",
  descricaoProduto: "Hamburguer Gourmet",
  valorProduto: 15.99
})
```

```
db.Produto.insertOne({
  _id: ObjectId("657f06c49ec1681c515b140e"),
  codigoProduto: "P004",
  descricaoProduto: "Lasanha à Bolonhesa",
  valorProduto: 22.50
})


db.Produto.insertOne({
  _id: ObjectId("657f06cd9ec1681c515b140f"),
  codigoProduto: "P005",
  descricaoProduto: "Sushi Combo",
  valorProduto: 34.99
})


db.Produto.insertOne({
  _id: ObjectId("657f06d79ec1681c515b1410"),
  codigoProduto: "P006",
  descricaoProduto: "Salada Caesar",
  valorProduto: 12.99
})


// Inserções de Pedido

db.Pedido.insertOne({
  codigoPedido: "PED001",
  dataPedido: "15012023",
  formaPagamento: "Cartão de Crédito",
  clienteId: ObjectId("657f7976fc6a7aacea011641"),
  produtos: [
    ObjectId("657f06b09ec1681c515b140c"),
    ObjectId("657f06a49ec1681c515b140b")
  ]
})

db.Pedido.insertOne({
  codigoPedido: "PED002",
  dataPedido: "20022023",
  formaPagamento: "Boleto Bancário",
  clienteId: ObjectId("657f797cfc6a7aacea011642"),
  produtos: [
    ObjectId("657f06d79ec1681c515b1410"),
    ObjectId("657f06b99ec1681c515b140d")
  ]
})
```

```
db.Pedido.insertOne({
  codigoPedido: "PED003",
  dataPedido: "25032023",
  formaPagamento: "Transferência Bancária",
  clienteId: ObjectId("657f7983fc6a7aacea011643"),
  produtos: [
    ObjectId("657f06a49ec1681c515b140b"),
    ObjectId("657f06cd9ec1681c515b140f")
  ]
})

db.Pedido.insertOne({
  codigoPedido: "PED004",
  dataPedido: "10042023",
  formaPagamento: "Dinheiro",
  clienteId: ObjectId("657f7988fc6a7aacea011644"),
  produtos: [
    ObjectId("657f06cd9ec1681c515b140f"),
    ObjectId("657f06c49ec1681c515b140e")
  ]
})


db.Pedido.insertOne({
  codigoPedido: "PED005",
  dataPedido: "05052023",
  formaPagamento: "Pix",
  clienteId: ObjectId("657f798dfc6a7aacea011645"),
  produtos: [
    ObjectId("657f06cd9ec1681c515b140f"),
    ObjectId("657f06b09ec1681c515b140c")
  ]
})

// Consultas


// 2 consultas com pelo menos filtros diversos (IN, GT, etc), sem projeção;

// 1 - Com o intuito de personalizar o conteúdo do aplicativo, estamos resgatando todas as informações de clientes que nasceram a partir dos anos de 1990.

db.Cliente.find({
  dataNascimento: {
    $gte: new Date("1990-01-01")
  }
})
```

//2 - Com o objetivo de mandar promoções direcionadas para as pessoas que moram em São Paulo e Rio de Janeiro, essa consulta é útil pra resgatar as informações das pessoas que moram nesses // estados.

```
db.Cliente.find({ "Endereco.estado": { $in: ["RJ", "SP"] } })
```

// 2 consultas com pelo menos filtros diversos e com projeção;

// 1 - O aplicativo que enviar avisos semanais apenas para pessoas do sexo feminino, e com essa consulta irá resgatar os e-mails delas.

```
db.Cliente.find({Sexo:"Feminino"},{Email:1,_id:0})
```

// 2 - Descobrir a descrição dos produtos que custam acima de 30 reais.

```
db.Produto.find({ valorProduto: { $gt: 30.00 } },{descricaoProduto:1,_id:0})
```

// 1 consulta com apenas projeção (sem filtro);

// 1 - Resgatar todas as formas de pagamento feitas nos pedidos

```
db.Pedido.find({},{formaPagamento:1,_id:0})
```

// 1 consulta com pelo menos acesso a elemento de array;

// Um erro ocorreu durante o cadastro de um cliente, tornando essencial notificar a pessoa sobre o problema por e-mail. Infelizmente, o aplicativo dispõe apenas do número de celular da pessoa. Neste contexto, torna-se necessário realizar uma consulta para recuperar o endereço de e-mail associado a esse número de celular.

```
db.Cliente.find(
  { "telefone": "666666666" },
  { "Email": 1, "_id": 0 }
)
```

// 1 consulta com pelo menos acesso a estrutura/objeto embutido;

// Resgatar a quantidade de pessoas que moram no Estado de São Paulo

```
db.Cliente.count({ "Endereco.estado": "SP" })
```

// 1 consulta com pelo menos sort e limit e filtros e projeções;

// Resgatar a descrição dos 2 produtos mais baratos acima de 10 reais.

```
db.Produto.find({ valorProduto: { $gt: 10.00 } }, { descricaoProduto: 1, _id: 0 }).sort({
valorProduto: 1 }).limit(2)

//1 consulta com pelo menos aggregate e lookup;

// Resgatar o nome, código do pedido, data do pedido e forma de pagamento dos clientes
que pediram Sushi Combo

db.Pedido.aggregate([
 {
  $match: {
   "produtos": ObjectId("657f06cd9ec1681c515b140f")
  }
 },
 {
  $lookup: {
   from: "Cliente",
   localField: "clienteId",
   foreignField: "_id",
   as: "cliente"
  }
 },
 {
  $unwind: "$cliente"
 },
 {
  $project: {
   _id: 0,
   "Nome do Cliente": "$cliente.Nome",
   "Código do Pedido": "$codigoPedido",
   "Data do Pedido": "$dataPedido",
   "Forma de Pagamento": "$formaPagamento"
  }
 }
])


// 1 outra consulta a seu critério, dentro do contexto da aplicação.

-- Resgatar os nomes das pessoas que possuem de 2 telefones pra cima, com o intuito de
realizar uma pesquisa interna

db.Cliente.find({
 $expr: { $gt: [{ $size: "$telefone" }, 1] }
}, {
 _id: 0,
 Nome: 1
})
```