Você será capaz de:

- Incorporar dados aos documentos através de arrays ;
- Utilizar os operadores $pop , $pull e $push ;
- Utilizar o operador $addToSet ;
- Utilizar os operadores $each , $slice e $sort .

Por que isso é importante?

Os arrays são estruturas muito importante no contexto da modelagem de dados do MongoDB . Além de enriquecer os dados, essas estruturas também são responsáveis pelo "relacionamento" entre os dados. Aqui a palavra relacionamento aparece entre aspas porque, mesmo o MongoDB sendo um banco de dados não relacional , você sempre terá algum tipo de relação entre seus dados.
Com os arrays , você pode criar estruturas que simulam o relacionamento 1:N (um para muitos), e o MongoDB oferece um conjunto muito bom de operadores para que o trabalho com arrays fique mais simples e preciso!

Operador $push

O operador $push adiciona um valor a um *array* . Se o campo não existir no documento, um novo *array* com o valor em um elemento será adicionado.
Sintaxe:

Copiar

```
{ $push: { <campo1>: <valor1>, ... } }
```

Em conjunto com o $push , você pode utilizar o que chamamos de modificadores . Cada um desses modificadores tem funções específicas que você verá melhor com exemplos. São eles:

- $each : Adiciona múltiplos valores a um *array* ;
- $slice : Limita o número de elementos do *array* . Requer o uso do modificador $each ;
- $sort : Ordena os elementos do *array* . Requer o uso do modificador $each ;
- $position : Especifica a posição do elemento que está sendo inserido no *array* . Também requer o modificador $each . Sem o modificador $position , o operador $push adiciona o elemento no final do *array* .

Quando você utiliza um modificador, o processo de push ocorre na seguinte ordem, independentemente da ordem em que os modificadores aparecem:

1. Altera o *array* para adicionar os elementos na posição correta;
2. Aplica a ordenação ($sort), se especificada;

3. Limita o *array* ($slice), se especificado;
4. Armazena o *array* .

Veja alguns exemplos. Utilizaremos um banco de dados chamado sales .

Adicionando um valor a um array

Considere a coleção supplies , uma coleção vazia. A operação abaixo adiciona um objeto que tem as informações da compra de um produto ao array items do documento em que o _id seja igual a 1 na coleção supplies :
Para não precisarmos escrever uma *query* só para fazer o insert do documento, vamos usar a opção upsert: true para já adicionar o elemento ao mesmo tempo que usamos o operador $push . É importante ficar nítido que a condição upsert não influencia a forma como o $push funciona.

Copiar

```
use sales;
db.supplies.updateOne(
  { _id: 1 },
  {
    $push: {
      items: {
        "name": "notepad",
        "price":  35.29,
        "quantity": 2,
      },
    },
  },
  { upsert: true },
);
```

Veja, o método updateOne() é o mesmo que você já utilizou nos exemplos anteriores. A única diferença é a inclusão do operador $push . O resultado dessa operação é um documento com o seguinte formato:

Copiar

```
{
    _id : 1,
    items : [
        {
            "name" : "notepad",
            "price" : 35.29,
            "quantity" : 2,
        },
    ],
}
```

Adicionando múltiplos valores a um array

Se você quiser adicionar múltiplos valores a um *array* , isso também é possível utilizando o operador $push , mas dessa vez será necessário adicionar o modificador $each .
A operação abaixo adicionará mais dois produtos ao *array* items do primeiro documento na coleção supplies :

Copiar

```
db.supplies.updateOne(
  {},
  {
    $push: {
      items: {
        $each: [
          {
            "name": "pens",
            "price": 56.12,
            "quantity": 5,
          },
          {
            "name": "envelopes",
            "price": 19.95,
            "quantity": 8,
          },
        ],
      },
    },
  },
  { upsert: true },
);
```

O documento ficará assim:

Copiar

```
{
    _id : 1,
    items : [
        {
            "name" : "notepad",
            "price" : 35.29,
            "quantity" : 2,
        },
        {
            "name" : "pens",
            "price" : 56.12,
```

```
            "quantity" : 5,
        },
        {
            "name" : "envelopes",
            "price" : 19.95,
            "quantity" : 8,
        },
    ],
}
```

Múltiplos modificadores

O $push pode ser utilizado com múltiplos modificadores, fazendo várias operações ao mesmo tempo em um *array* .
Desconsidere as últimas alterações com $push (se quiser acompanhar, você pode utilizar db.dropDatabase() para remover as alterações anteriores) e veja a realização dele abaixo, com ainda mais opções!

Copiar

```
db.supplies.updateOne(
  { _id: 1 },
  {
    $push: {
      items: {
        $each: [
          {
            "name" : "notepad",
            "price" : 35.29,
            "quantity" : 2,
          },
          {
            "name": "envelopes",
            "price": 19.95,
            "quantity": 8,
          },
          {
            "name": "pens",
            "price": 56.12,
            "quantity": 5,
          },
        ],
        $sort: { quantity: -1 },
        $slice: 2,
      },
    },
```

```
  },
  { upsert: true },
);
```

Essa operação utiliza os seguintes modificadores:

- O modificador `$each` para adicionar múltiplos documentos ao *array* `items` ;
- O modificador `$sort` para ordenar todos os elementos alterados no *array* `items` pelo campo `quantity` em ordem decrescente;
- E o modificador `$slice` para manter apenas os dois primeiros elementos ordenados no *array* `items` .

Em resumo, essa operação mantém no *array* `items` apenas os dois documentos com a *quantidade* (campo `quantity`) mais alto. Veja o resultado logo abaixo:

Copiar

```
{
  _id : 1,
  items : [
    {
      "name" : "envelopes",
      "price" : 19.95,
      "quantity" : 8,
    },
    {
      "name" : "pens",
      "price" : 56.12,
      "quantity" : 5,
    },
  ],
}
```

Operador $pop

Uma maneira simples de remover o primeiro ou o último elemento de um *array* é utilizar o operador `$pop` . Passando o valor -1 ao operador `$pop` você removerá o primeiro elemento. Já ao passar o valor 1 , você removerá o último elemento do *array* . Simples, não é?!
Dado o seguinte documento na coleção `supplies` :

Copiar

```
{
  _id: 1,
  items: [
    {
      "name" : "notepad",
```

```
    "price" : 35.29,
    "quantity" : 2,
  },
  {
    "name": "envelopes",
    "price": 19.95,
    "quantity": 8,
  },
  {
    "name": "pens",
    "price": 56.12,
    "quantity": 5,
  },
],
}
```

Removendo o primeiro item de um array

Para remover o primeiro elemento do *array* items , utilize a seguinte operação:

Copiar

```
db.supplies.updateOne({ _id: 1 }, { $pop: { items: -1 } });
```

O documento será alterado, e o primeiro elemento será removido do *array* items :

```
{
  _id: 1,
  items: [
  {
    "name": "envelopes",
    "price": 19.95,
    "quantity": 8,
  },
  {
    "name": "pens",
    "price": 56.12,
    "quantity": 5,
  },
],
}
```

Removendo o último item de um array

Para remover o último elemento do *array* items , utilize a seguinte operação:

Copiar

```
db.supplies.updateOne({ _id: 1 }, { $pop: { items: 1 } });
```

Executando essa operação, é removido o último elemento do *array* `items` , e o documento ficará assim:

```
{
  _id: 1,
  items: [
    {
      "name" : "notepad",
      "price" : 35.29,
      "quantity" : 2,
    },
    {
      "name": "envelopes",
      "price": 19.95,
      "quantity": 8,
    },
  ],
}
```

Operador $pull

O operador `$pull` remove de um *array* existente todos os elementos com um ou mais valores que atendam à condição especificada.

Removendo todos os itens iguais a um valor especificado

Vamos considerar os seguintes documentos na coleção `supplies` :

```
{
  _id: 1,
  items: [
    {
      "name" : "notepad",
      "price" : 35.29,
      "quantity" : 2,
    },
    {
      "name": "envelopes",
      "price": 19.95,
      "quantity": 8,
    },
    {
      "name": "pens",
      "price": 56.12,
```

```
      "quantity": 5,
    },
  ],
},
{
  _id: 2,
  items: [
    {
      "name" : "pencil",
      "price" : 5.29,
      "quantity" : 2,
    },
    {
      "name": "envelopes",
      "price": 19.95,
      "quantity": 8,
    },
    {
      "name": "backpack",
      "price": 80.12,
      "quantity": 1,
    },
    {
      "name": "pens",
      "price": 56.12,
      "quantity": 5,
    },
  ],
}
```

Digamos que você queira remover do *array* `items` os elementos `pens` e `envelopes` :

```
db.supplies.updateMany(
  {},
  {
    $pull: {
      items: {
        name: { $in: ["pens", "envelopes"] },
      },
    },
  },
);
```

Na atualização acima, foi utilizado o operador `$pull` combinado com o operador `$in` para alterar o *array* `items` :

```
{
  _id : 1,
  items : [
    {
      "name" : "notepad",
      "price" : 35.29,
      "quantity" : 2,
    },
  ],
},
{
  _id : 2,
  items : [
    {
      "name" : "pencil",
      "price" : 5.29,
      "quantity" : 2,
    },
    {
      "name" : "backpack",
      "price" : 80.12,
      "quantity" : 1,
    },
  ],
}
```

Removendo todos os itens que atendem a uma condição diretamente no $pull

Você pode especificar uma condição de remoção diretamente no `$pull` . Essa condição pode ser, por exemplo, um operador de comparação. Tendo o seguinte documento na coleção `profiles` :

Copiar

```
{ _id: 1, votes: [3, 5, 6, 7, 7, 8] }
```

Você pode remover todos os elementos do *array* `votes` que sejam maiores ou iguais a (`$gte`) 6 . Por exemplo:

Copiar

```
db.profiles.updateOne(
  { _id: 1 },
  {
    $pull: {
```

```
      votes: { $gte: 6 },
    },
  },
);
```

Depois dessa operação, o documento ficará apenas com valores menores do que 6 no *array* `votes` :

Copiar

```
{ _id: 1, votes: [3,  5] }
```

Removendo itens em um array de Documentos

Veja a coleção `survey` com os seguintes documentos:

Copiar

```
{
  _id: 1,
  results: [
    { item: "A", score: 5 },
    { item: "B", score: 8, comment: "Strongly agree" },
  ],
},
{
  _id: 2,
  results: [
    { item: "C", score: 8, comment: "Strongly agree" },
    { item: "B", score: 4 },
  ],
}
```

Os documentos têm um *array* chamado `results` que armazena documentos.
Com a operação abaixo, você consegue remover do *array* `results` todos os elementos que contenham o campo `score` igual a 8 e o campo `item` igual a "B" . Ou seja, o documento deve atender a ambas as condições.

Copiar

```
db.survey.updateMany(
  {},
  {
    $pull: {
      results: { score: 8 , item: "B" },
    },
  },
);
```

A expressão do operador `$pull` aplica as condições a cada elemento do *array* `results` como se estivesse no primeiro nível, isso porque os documentos dentro do *array* `results` não contêm outros campos com mais *arrays* . Se isso acontecer, você deve utilizar uma outra abordagem, que será detalhada mais à frente.
Após essa operação, os documentos ficarão assim:

Copiar

```
{
  _id: 1,
  results: [ { "item": "A", "score": 5 } ],
},
{
  _id: 2,
  results: [
    { "item": "C", "score": 8, "comment": "Strongly agree" },
    { "item": "B", "score": 4 },
  ],
}
```

Operador $addToSet

O operador `$addToSet` é utilizado quando você precisa garantir que os valores de um *array* não sejam duplicados. Ou seja, ele garante que apenas valores únicos estejam presentes no *array* , tratando o *array* como se fosse um conjunto (uma vez que conjuntos, na matemática, não podem conter elementos duplicados).
Você precisa ter em mente três aspectos sobre o `$addToSet` :

- Se você utilizá-lo em um campo que não existe no documento alterado, ele criará um campo do tipo *array* com o valor especificado na operação;
- Se você utilizá-lo em um campo já existente no documento, mas esse campo não for um *array* , a operação não funcionará;
- Se o valor passado for um documento, o MongoDB o considerará como duplicado se um documento existente no *array* for exatamente igual ao documento a ser adicionado, ou seja, possui os mesmos campos com os mesmos valores, e esses campos estão na mesma ordem.

Veja alguns exemplos considerando o documento abaixo na coleção `inventory` :

Copiar

```
{
  _id: 1,
  item: "polarizing_filter",
  tags: ["electronics", "camera"],
}
```

Adicionando ao array

A operação abaixo adiciona o elemento "accessories" ao *array* `tags` desde que "accessories" não exista no *array* :

Copiar

```
db.inventory.updateOne(
  { _id: 1 },
  { $addToSet: { tags: "accessories" } },
);
```

O *array* `tags` agora tem mais um elemento:

Copiar

```
{
  _id: 1,
  item: "polarizing_filter",
  tags: [
    "electronics",
    "camera",
    "accessories",
  ],
}
```

Se o valor existir

A operação abaixo tenta adicionar o elemento "camera" ao *array* `tags` . Porém, esse elemento já existe e a operação não surtirá efeito:

Copiar

```
db.inventory.updateOne(
  { _id: 1 },
  { $addToSet: { tags: "camera"  } },
);
```

Como resultado dessa operação, é retornada uma mensagem dizendo que o MongoDB encontrou um documento com o `_id` igual a 1 , mas não fez nenhuma alteração:

Copiar

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 0 }
```

Com o modificador $each

Você pode utilizar o operador `$addToSet` combinado com o modificador `$each` . Esse modificador permite que você adicione múltiplos valores a um *array* .
Veja o seguinte documento da coleção `inventory` :

Copiar
```
{ _id: 2, item: "cable", tags: ["electronics", "supplies"] }
```

A operação abaixo utiliza o operador `$addToSet` e o modificador `$each` para adicionar alguns elementos a mais no *array* `tags` :

Copiar
```
db.inventory.updateOne(
  { _id: 2 },
  {
    $addToSet: {
      tags: {
        $each: ["camera", "electronics", "accessories"],
      },
    },
  },
);
```

Como resultado, a operação adicionará ao *array* `tags` somente os elementos "camera" e "accessories" , uma vez que o elemento "electronics" já existia no *array* . Veja abaixo:

Copiar
```
{
  _id: 2,
  item: "cable",
  tags: ["electronics", "supplies", "camera", "accessories"],
}
```

Array Filters

Agora, imagine que você precisa de documentos que tenham apenas um valor que está dentro de um array de objetos, estranho, não é? Mas vamos ver um exemplo:

Copiar
```
db.recipes.insertMany([
  {
```

```
    title: "Panqueca Simples",
    ingredients: [
      { name: "Farinha", quantity: 2},
      { name: "Oleo", quantity: 4 },
      { name: "Leite", quantity: 1 },
    ],
  },
  {
    title: "Bolo de Cenoura",
    ingredients: [
      { name: "Farinha", quantity: 2},
      { name: "Oleo", quantity: 1, unit: "xícara" },
      { name: "Ovo", quantity: 3},
      { name: "Cenoura", quantity: 3},
      { name: "Fermento", quantity: 1},
    ],
  },
]);
```

Caso você saiba o `index` exato do elemento em que deseja-se alterar alguma propriedade, pode-se fazer algo como:

Copiar

```
db.recipes.updateOne( { title: "Panqueca Simples" }, { $set: {
"ingredients.1.unit": "xícara" } } );
```

Mas, e se você não soubesse qual posição do array que gostaria de alterar um objeto? Ou melhor, e se quisesse alterar dinamicamente todas as receitas que usam farinha, para usarem farinha integral e que a `unit` seja xícara? Vamos incrementando alguns exemplos até responder esta última suposição usando o `Array Filters` .

Copiar

```
db.recipes.updateOne(
  { title: "Panqueca Simples" },
  {
    $set : {
      "ingredients.$[elemento].name": "Farinha Integral",
    },
  },
  { arrayFilters: [ { "elemento.name": "Farinha" } ] },
);
```

Achamos um documento com `title` igual a `"Panqueca Simples"` e atualizamos o objeto com propriedade `name` igual a `"Farinha"` do array `ingredients` , alterando o valor do campo `name` para `"Farinha Integral"` . Agora, vamos adicionar `"xícara"` ao campo `unit` do objeto com `name` igual a `"Farinha Integral"` !

Copiar

```
db.recipes.updateOne(
  { title: "Panqueca Simples" },
  {
    $set : {
      "ingredients.$[elemento].unit": "xícara",
    },
  },
  { arrayFilters: [ { "elemento.name": "Farinha Integral" } ] },
);
```

Precisamos mudar o `arrayFilter` de `"Farinha"` para `"Farinha Integral"` , pois na `query` anterior alteramos o `name` desse ingrediente.
Se quiséssemos trocar todos os ingredientes da coleção que são `"Farinha"` por `"Farinha Integral"` e colocar `"xícara"` como valor de `unit` , poderíamos seguir o seguinte exemplo:

Copiar

```
db.recipes.updateMany( // Passamos de updateOne para updateMany.
  {}, // Retiramos a restrição do título.
  {
    $set : {
      "ingredients.$[elemento].unit": "xícara", // Setamos `unit` como "xícara",
      "ingredients.$[elemento].name": "Farinha Integral", // `name` como "Farinha Integral".
    },
  },
  { arrayFilters: [ { "elemento.name": "Farinha" } ] }, // Filtramos os arrays que o valor da propriedade `name` seja "Farinha".
);
```

Agora, a prática

Você continuará utilizando o mesmo dataset de filmes do dia anterior. Se você fez todos os exercícios corretamente, apenas siga para o primeiro

exercício de hoje. Caso contrário, conecte-se à sua instância e utilize o trecho de código abaixo para inserir os documentos e ficar na mesma página!

Copiar

```
db.movies.drop();
db.movies.insertMany([
  {
    title: "Batman",
    category: [
      "action",
      "adventure",
    ],
    imdbRating: 7.7,
    budget: 35,
  },
  {
    title: "Godzilla",
    category: [
      "action",
      "adventure",
      "sci-fi",
    ],
    imdbRating: 6.6,
    budget: 1,
  },
  {
    title: "Home Alone",
    category: [
      "family",
      "comedy",
    ],
    imdbRating: 7.4,
  },
]);
```

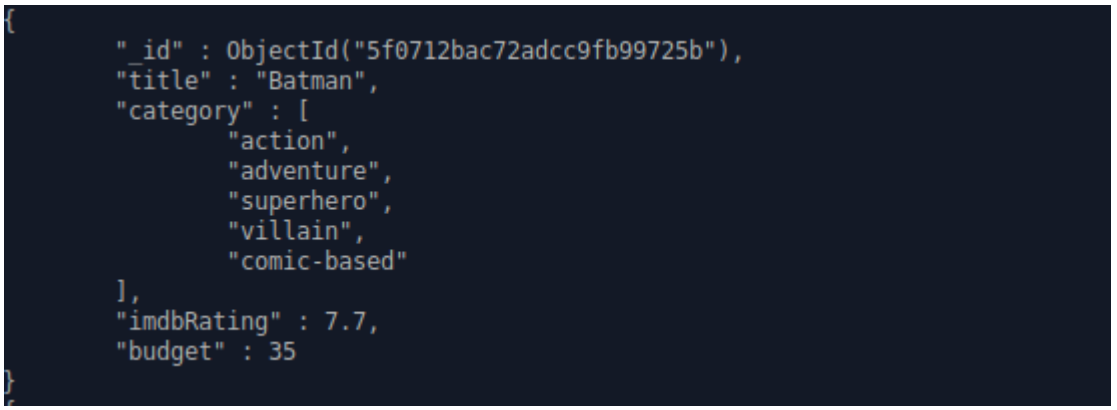
Para cada execução, utilize o método find() para conferir as alterações nos documentos

O `MongoDB` possui diversas ferramentas, como, por exemplo, `mongo` , `mongosh` , `Compass` e outras ferramentas de terceiros. Você pode utilizar o que achar melhor para executar as *queries* . O importante é realizá-las.

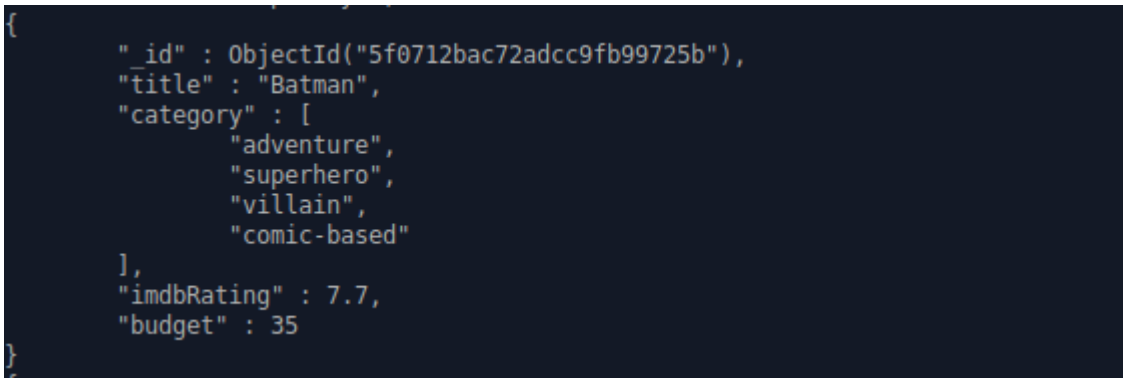
Exercício 1: Adicione a categoria `"superhero"` ao filme `Batman` .

Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:

Exercício 2: Utilizando o modificador `$each` , adicione as categorias `"villain"` e `"comic-based"` ao filme `Batman` .
Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:



Exercício 3: Remova a categoria `"action"` do filme `Batman` .
Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:



Exercício 4: Remova o primeiro elemento do *array* `category` do filme `Batman` .
Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:

```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725b"),
        "title" : "Batman",
        "category" : [
                "superhero",
                "villain",
                "comic-based"
        ],
        "imdbRating" : 7.7,
        "budget" : 35
}
```

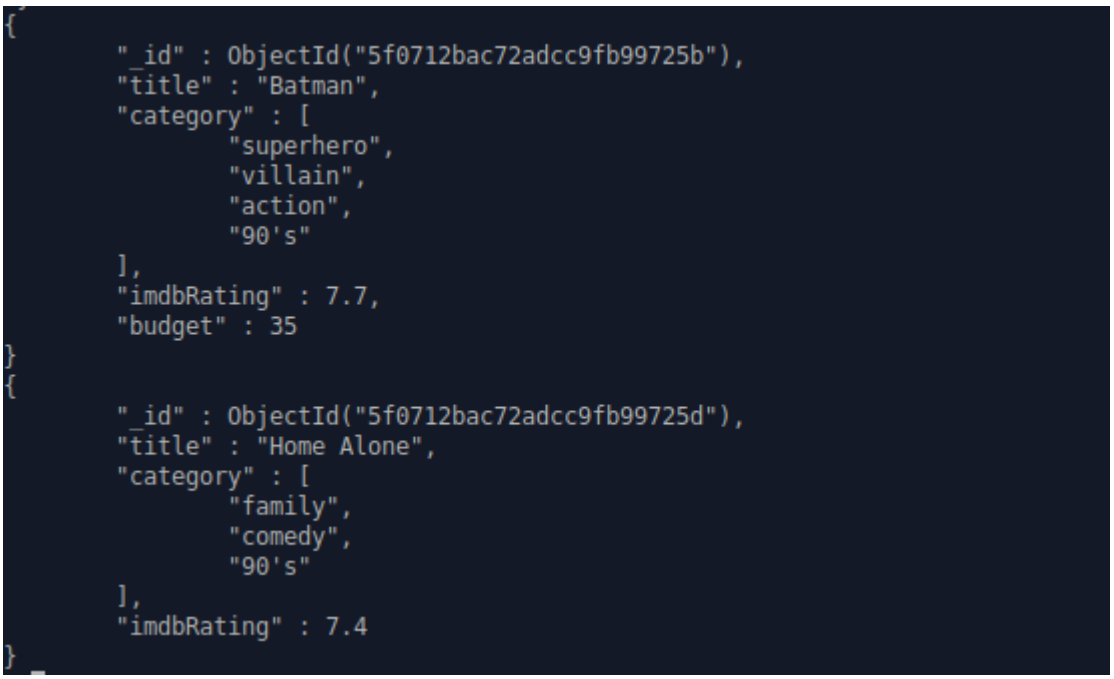

Exercício 5: Remova o último elemento do *array* `category` do filme `Batman` .
Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:


```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725b"),
        "title" : "Batman",
        "category" : [
                "superhero",
                "villain"
        ],
        "imdbRating" : 7.7,
        "budget" : 35
}
```


Exercício 6: Adicione o elemento `"action"` ao *array* `category` do filme `Batman` , garantindo que esse valor não se duplique.
Após a execução do método `.find().pretty()` o resultado do filme `Batman` será parecido com o dessa imagem:


```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725b"),
        "title" : "Batman",
        "category" : [
                "superhero",
                "villain",
                "action"
        ],
        "imdbRating" : 7.7,
        "budget" : 35
}
```


Exercício 7: Adicione a categoria `"90's"` aos filmes `Batman` e `Home Alone` .
Após a execução do método `.find().pretty()` , o resultado do filme `Batman` e do filme `Home Alone` será parecido com o dessa imagem:



Exercício 8: Crie um *array* de documentos chamado `cast` para o filme `Home Alone` com os seguintes dados:

Copiar

```
{
  "actor": "Macaulay Culkin",
  "character": "Kevin"
},
{
  "actor": "Joe Pesci",
  "character": "Harry"
},
{
  "actor": "Daniel Stern"
}
```

Após a execução do método `.find().pretty()` , o resultado do filme `Home Alone` será parecido com o dessa imagem:

```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725d"),
        "title" : "Home Alone",
        "category" : [
                "family",
                "comedy",
                "90's"
        ],
        "imdbRating" : 7.4,
        "cast" : [
                {
                        "actor" : "Macaulay Culkin",
                        "character" : "Kevin"
                },
                {
                        "actor" : "Joe Pesci",
                        "character" : "Harry"
                },
                {
                        "actor" : "Daniel Stern"
                }
        ]
}
```


Exercício 9: Adicione o campo `character` com o valor `Marv` ao *array* de `cast` em que o campo `actor` seja igual a `Daniel Stern` no filme `Home Alone` .
`Dica` : Para isso, leia aqui sobre o operador `$` .
Após a execução do método `.find().pretty()` , o resultado do filme `Home Alone` será parecido com o dessa imagem:


```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725d"),
        "title" : "Home Alone",
        "category" : [
                "family",
                "comedy",
                "90's"
        ],
        "imdbRating" : 7.4,
        "cast" : [
                {
                        "actor" : "Macaulay Culkin",
                        "character" : "Kevin"
                },
                {
                        "actor" : "Joe Pesci",
                        "character" : "Harry"
                },
                {
                        "actor" : "Daniel Stern",
                        "character" : "Marv"
                }
        ]
}
```


Exercício 10: Crie um *array* de documentos chamado `cast` para o filme `Batman` com os seguintes dados:

Copiar


```
{
```

```
  "character": "Batman"
},
{
  "character": "Alfred"
},
{
  "character": "Coringa"
}
```

Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:

Exercício 11: Produza três *querys* para o filme `Batman` :

- Adicione o campo `actor` , que deve ser um array com o valor `Christian Bale` , ao *array* de `cast` em que o campo `character` seja igual a `Batman` ;
- Adicione o campo `actor` , que deve ser um array com o valor `Michael Caine` , ao *array* de `cast` em que o campo `character` seja igual a `Alfred` ;
- Adicione o campo `actor` , que deve ser um array com o valor `Heath Ledger` , ao *array* de `cast` em que o campo `character` seja igual a `Coringa` .

`Dica` : Para isso, leia aqui sobre o operador `$` .
Após a execução do método `.find().pretty()` o resultado do filme `Batman` será parecido com o dessa imagem:

Exercício 12: Adicione aos atores de `cast` do `character Batman` do filme `Batman` os valores `"Michael Keaton"` , `"Val Kilmer"` e `"George Clooney"` , e deixe o array em ordem alfabética.

`Dica` : Para isso, leia aqui sobre o operador `$` .

Após a execução do método `.find().pretty()` , o resultado do filme `Batman` será parecido com o dessa imagem:

```
{
        "_id" : ObjectId("5f0712bac72adcc9fb99725b"),
        "title" : "Batman",
        "category" : [
                "superhero",
                "villain",
                "action",
                "90's"
        ],
        "imdbRating" : 7.7,
        "budget" : 35,
        "cast" : [
                {
                        "character" : "Batman",
                        "actor" : [
                                "Christian Bale",
                                "George Clooney",
                                "Michael Keaton",
                                "Val Kilmer"
                        ]
                },
                {
                        "character" : "Alfred",
                        "actor" : [
                                "Michael Caine"
                        ]
                },
                {
                        "character" : "Coringa",
                        "actor" : [
                                "Heath Ledger"
                        ]
                }
        ]
}
```