

tryber / **sd-010-a-mongodb-dataflights** Publicgenerated from [betrybe/sd-0x-mongodb-dataflights](#)

☆ 0 stars 🍴 13 forks

☆ Star

👁 Watch ▼

Code

Issues

Pull requests 124

Actions

Projects

Wiki

Security

Insights

🔗 master ▼

...



vinicius-vasconcelos ...

on 11 Aug

[View code](#)

☰ README.md



Termos e acordos

Ao iniciar este projeto, você concorda com as diretrizes do Código de Ética e Conduta e do Manual da Pessoa Estudante da Trybe.

Boas vindas ao repositório do projeto DataFlights!!

Você já usa o GitHub diariamente para desenvolver os exercícios, certo? Agora, para desenvolver os projetos, você deverá seguir as instruções a seguir. Fique atento a cada passo, e se tiver qualquer dúvida, nos envie por *Slack*! #vqv 🚀

Aqui você vai encontrar os detalhes de como estruturar o desenvolvimento do seu projeto a partir desse repositório, utilizando uma branch específica e um *Pull Request* para colocar seus códigos.

Sumário

- [Habilidades](#)
- [Entregáveis](#)

- O que deverá ser desenvolvido
- Desenvolvimento
- Data de entrega
- Instruções para entregar seu projeto
 - Antes de começar a desenvolver
 - Durante o desenvolvimento
- Como desenvolver
- Requisitos do projeto
 - Lista de requisitos
- Depois de terminar o desenvolvimento (Opcional)
- Avisos finais

Habilidades

- Buscar documentos no banco
- Usar filtros na busca
- Deletar documentos conforme filtro
- Contar documentos compreendidos nos filtros pedidos
- Inserir documentos no banco

Entregáveis

Temos, neste projeto, uma série de desafios com diferentes níveis de complexidade que devem ser resolvidos cada um em seu arquivo próprio.

1. Leia a pergunta e crie no diretório `challenges` um arquivo chamado `desafioN.js`, em que N é o número do desafio.
2. O arquivo deve conter apenas o código MQL (*Mongo Query Language*) do desafio resolvido. **Não se esqueça de incluir o ponto e vírgula (";")** no final de suas queries, como no exemplo a seguir:

```
db.voos.find();
```

⚠ Restrições ⚠:

- **Não se deve usar aspas simples** para especificar campos e/ou valores. Quando for necessário usar aspas, **use somente aspas duplas**;

- **Não se deve usar o comando `use dataFlights`** , haja visto que **os testes já se conectam automaticamente à base `dataFlights`** .
3. Faça isso até finalizar todos os desafios e depois siga as instruções de como entregar o projeto em [Instruções para entregar seu projeto](#).
 4. Para entregar o seu projeto você deverá criar um *Pull Request* neste repositório. Este *Pull Request* deverá conter no diretório `challenges` os arquivos `desafio1.js` , `desafio2.js` e assim por diante até o `desafio28.js` , que conterão seu código `MQL` de cada desafio, respectivamente.

 **É importante que seus arquivos tenham exatamente estes nomes!** 

Qualquer dúvida, procure a monitoria. Lembre-se que você pode consultar nosso conteúdo sobre [Git & GitHub](#) sempre que precisar!

O que deverá ser desenvolvido

Hoje você fará um projeto com o codinome *dataflights*. Neste projeto, você praticará todos os conceitos de **MongoDB** já ensinados até aqui.

Porém, você usará um banco de dados diferente dos utilizados nos exemplos e exercícios vistos até agora. Chamaremos esse banco de `dataFlights` . As instruções de como restaurar o banco podem ser lidas a seguir.

Desenvolvimento

Nesse projeto você vai elaborar *queries* em `mongo` para:

- Consultar a coleção do projeto, usando vários campos para filtrar essa busca,
- Deletar alguns voos conforme outros filtros.
- Contar voos compreendidos nos filtros.

Data de Entrega

- Projeto individual.
- Será um dia de projeto.
- Data de entrega para avaliação final do projeto: 18/08/2021 - 14:00h .

Instruções para entregar seu projeto

Antes de começar a desenvolver

1. Clone o repositório

- `git clone https://github.com/tryber/sd-010-a-mongodb-dataflights.git` .
- Entre na pasta do repositório que você acabou de clonar:
 - `cd sd-010-a-mongodb-dataflights`

2. Crie uma branch a partir da branch `master`

- Verifique que você está na branch `master`
 - Exemplo: `git branch`
- Se não estiver, mude para a branch `master`
 - Exemplo: `git checkout master`
- Agora crie uma branch à qual você vai submeter os `commits` do seu projeto
 - Você deve criar uma branch no seguinte formato: `nome-de-usuario-nome-do-projeto`
 - Exemplo: `git checkout -b seunome-mongodb-dataflights`

3. Para cada exercício você deve criar um novo arquivo JS **dentro de uma pasta na raiz do seu projeto chamada `challenges`** seguindo a seguinte estrutura:

- `desafio1.js`, `desafio2.js`, ..., `desafioN.js`

4. Adicione as mudanças ao `stage` do Git e faça um `commit`

- Verifique que as mudanças ainda não estão no `stage`
 - Exemplo: `git status` (deve aparecer o arquivo que você alterou como `desafio1.js`)
- Adicione o novo arquivo ao `stage` do Git
 - Exemplo:
 - `git add .` (adicionando arquivo de solução `challenges/desafio1.js` para desafio 1)
 - `git status` (deve aparecer listado o arquivo `challenges/desafio1.js` em verde)
- Faça o `commit` inicial
 - Exemplo:
 - `git commit -m 'iniciando o projeto MongoDB dataflights'` (fazendo o primeiro commit)
 - `git status` (deve aparecer uma mensagem tipo *nothing to commit*)



5. Adicione a sua branch com o novo `commit` ao repositório remoto

- Usando o exemplo anterior: `git push -u origin seunome-mongodb-dataflights`

6. Crie um novo `Pull Request` (PR)

- Vá até a página de *Pull Requests* do [repositório no GitHub](#)
- Clique no botão verde *"New pull request"*
- Clique na caixa de seleção *"Compare"* e escolha a sua branch **com atenção**
- Clique no botão verde *"Create pull request"*
- Adicione uma descrição para o *Pull Request* e clique no botão verde *"Create pull request"*
- **Não se preocupe em preencher mais nada por enquanto!**
- Volte até a [página de Pull Requests do repositório](#) e confira que o seu *Pull Request* está criado

Durante o desenvolvimento

-  **LEMBRE-SE DE CRIAR TODOS OS ARQUIVOS DENTRO DA PASTA `challenges`** 
- Faça `commits` das alterações que você fizer no código regularmente
- Lembre-se de sempre após um (ou alguns) `commits` atualizar o repositório remoto
- Os comandos que você utilizará com mais frequência são:
 - i. `git status` (para verificar o que está em vermelho - fora do stage - e o que está em verde - no stage)
 - ii. `git add` (para adicionar arquivos ao stage do Git)
 - iii. `git commit` (para criar um `commit` com os arquivos que estão no stage do Git)
 - iv. `git push -u origin nome-da-branch` (para enviar o `commit` para o repositório remoto na primeira vez que fizer o `push` de uma nova branch)
 - v. `git push` (para enviar o `commit` para o repositório remoto após o passo anterior)

Como desenvolver

Execute o seguinte comando para instalar as dependências de desenvolvimento do projeto:

```
npm install
```

Linter

Para garantir a qualidade do código, vamos utilizar neste projeto o linter ESLint. Assim o código estará alinhado com as boas práticas de desenvolvimento, sendo mais legível e de fácil manutenção! Para rodar o *linter* localmente no projeto, execute o comando abaixo:

```
npm run lint
```

⚠️ PULL REQUESTS COM ISSUES DE LINTER NÃO SERÃO AVALIADAS. ATENTE-SE PARA RESOLVÊ-LAS ANTES DE FINALIZAR O DESENVOLVIMENTO! ⚠️

Aqui encontram-se os requisitos do projeto. Em cada requisito você encontrara uma imagem de um protótipo de como sua aplicação deve ficar. Estilo da página não será avaliado.

Instruções para restaurar o banco de dados dataFlights

1. Abra o terminal e conecte-se à sua instância local do **MongoDB**. Se você receber uma mensagem de erro com uma mensagem como **Connection refused**, tente reiniciar sua instância clicando ([nesse link do course](#)) e através do menu lateral, no item Conectando .
2. Agora que você tem certeza de que a sua instância está no ar e que você está conectado a ela, digite `exit` . Você voltará ao terminal para iniciar a importação dos dados.
3. Na raiz do diretório do projeto, execute o seguinte comando que fará a restauração da base de dados `dataFlights` :

```
DBNAME=dataFlights ./scripts/resetdb.sh assets
```

- A execução desse script criará um banco de dados chamado `dataFlights` e importará os dados para a coleção `voos` .

⚠️ Como tanto esse script quanto o script de execução local dos testes (mostrado na [seção seguinte](#)), **restauram a base de dados `dataFlights`** , se atente a salvar seu progresso nos arquivos de desafio! ⚠️

Implementações técnicas

Para executar localmente os testes, é preciso escrever o seguinte no seu terminal, estando na raiz do diretório do projeto:

```
./scripts/evaluate.sh
```

Esse script passará por **todos os desafios** e imprimirá um relatório indicando se passou ou não para cada desafio. Como a execução do script **envolve restauração da base de dados dataFlights** de um teste para outro, recomenda-se esperar pela sua execução completa.

Para executar somente o teste de um desafio, execute o comando abaixo, substituindo N pelo número do desafio

```
./scripts/evaluate.sh desafioN
```

Requisitos do projeto

Durante a execução do projeto, utilize *queries* do mongo para retornar os valores pedidos nos requisitos.

Você deve criar uma pasta chamada `challenges` na raiz do projeto, contendo dentro dela arquivos no formato `desafioX.js` onde `X` é o número do requisito.

Dentro dos arquivos `desafioX.js`, **crie uma query** ou mais (se necessário), para retornar o que o requisito pede.

1 - Retorne a quantidade de documentos inseridos na coleção `voos`.

2 - Retorne os 10 primeiros documentos com voos da empresa `AZUL`.

3 - Retorne a quantidade de voos da empresa `AZUL`.

4 - Retorne a quantidade de voos da empresa `GOL`.

5 - Retorne o `vooId` do décimo ao décimo segundo documento da coleção `voos`.

6 - Retorne apenas os campos `empresa.sigla`, `empresa.nome` e `passageiros` do voo com o campo `vooId` igual a `756807`.

7 - Retorne a quantidade de voos em que o ano seja menor do que `2017`.

8 - Retorne a quantidade de voos em que o ano seja maior do que 2016 .

9 - Retorne a quantidade de voos entre os anos de 2017 e 2018 .

10 - Retorne apenas os 10 primeiros documentos com voos da empresa GOL do ano de 2017 . Exiba apenas os campos `vooId` , `empresa.nome` , `aeroportoOrigem.nome` , `aeroportoDestino.nome` , `mes` e `ano` .

11 - Retorne a quantidade de documentos em que o campo `aeroportoDestino.pais` não seja igual a ESTADOS UNIDOS .

12 - Retorne a quantidade de documentos em que o campo `aeroportoDestino.pais` seja igual a BRASIL , ARGENTINA ou CHILE .

13 - Retorne a quantidade de documentos em que o campo `aeroportoDestino.continente` não seja igual a EUROPA , ÁSIA e OCEANIA .

14 - Retorne o total de voos em que o país de origem não seja BRASIL .

15 - Retorne o total de voos com mais de 20 decolagens .

16 - Retorne o total de voos em que o campo `natureza` possui o valor Internacional .

17 - Retorne o total de voos em que o campo `natureza` possui o valor Doméstica .

18 - Retorne o `vooId` , `mes` e `ano` do primeiro voo com mais de 7000 passageiros pagos.

19 - Retorne o `vooId` do primeiro voo em que o campo `litrosCombustivel` exista.

20 - Retorne o `vooId` do primeiro voo em que o campo `rtk` não exista.

21 - Retorne o `vooId` do primeiro voo em que o campo `litrosCombustivel` seja maior ou igual a 1000 .

22 - Retorne o `vooId` do primeiro voo em que a empresa seja DELTA AIRLINES ou AMERICAN AIRLINES , a sigla do aeroporto de origem seja SBGR e a sigla do aeroporto de destino seja KJFK .

23 - Retorne o `vooId` e `litrosCombustivel` do primeiro voo em que o campo `litrosCombustivel` não seja maior do que 1000 e o campo `litrosCombustivel` exista.

24 - Retorne o `vooId` , `empresa.nome` e `litrosCombustivel` do primeiro voo em que `litrosCombustivel` não seja maior do que 600 e a empresa não seja GOL ou AZUL , e o campo `litrosCombustivel` exista.

25 - Remova todos os voos da empresa AZUL em que a quantidade de combustível seja menor do que 400 . Informe a quantidade de documentos removidos.

26 - Remova todos os voos da empresa GOL em que a quantidade de passageiros pagos esteja entre 5 e 10 , incluindo os casos em que a quantidade é 5 e 10 . Informe a quantidade de documentos removidos.

27 - Retorne a quantidade total de voos de natureza Doméstica que a empresa PASSAREDO possui, via uso de uma nova coleção chamada resumoVoos .

Ou seja, a coleção `resumoVoos` conterá documentos onde cada um indica para cada empresa a quantidade total de voos que ela possui de natureza Doméstica .

Para isso, escreva no arquivo `desafio27.js` duas queries, **nesta ordem**:

1. Conte quantos voos da empresa `PASSAREDO` cujo campo `natureza` possua valor igual a `Doméstica` e crie uma query que insira na coleção `resumoVoos` um documento com os campos: `empresa` (nome da empresa) e `totalVoosDomesticos` (o total retornado anteriormente).
2. Em uma segunda query, retorne a `empresa` e o `totalVoosDomesticos` do primeiro documento presente na coleção `resumoVoos` em que a empresa seja `PASSAREDO` .

28 - Retorne a quantidade total de voos de natureza Doméstica que a empresa LATAM AIRLINES BRASIL possui, via uso de uma nova coleção chamada resumoVoos .

Para isso, escreva no arquivo `desafio28.js` duas queries, **nesta ordem**:

1. Conte quantos voos da empresa `LATAM AIRLINES BRASIL` cujo campo `natureza` possua valor igual a `Doméstica` e crie uma query que insira na coleção `resumoVoos` um documento com os campos: `empresa` (nome da empresa) e `totalVoosDomesticos` (o total retornado anteriormente).
2. Em uma segunda query, retorne a `empresa` e o `totalVoosDomesticos` do primeiro documento presente na coleção `resumoVoos` em que a empresa seja `LATAM AIRLINES BRASIL` .

Revisando um pull request

Use o conteúdo sobre [Code Review](#) para te ajudar a revisar os *Pull Requests*.

#VQV 🚀

Depois de terminar o desenvolvimento (OPCIONAL)

Para sinalizar que o seu projeto está pronto para o "Code Review" dos seus colegas, faça o seguinte:

- Vá até a página **DO SEU Pull Request**, adicione a label de "code-review" e marque seus colegas:
 - No menu à direita, clique no link "**Labels**" e escolha a label **code-review**;
 - No menu à direita, clique no link "**Assignees**" e escolha **o seu usuário**;
 - No menu à direita, clique no link "**Reviewers**" e digite `students` , selecione o time `tryber/students-sd-00` .

Caso tenha alguma dúvida, [aqui tem um video explicativo](#).

Avisos finais

Ao finalizar e submeter o projeto, não se esqueça de avaliar sua experiência preenchendo o formulário. Leva menos de 3 minutos!

Link: [FORMULÁRIO DE AVALIAÇÃO DE PROJETO](#)

O avaliador automático não necessariamente avalia seu projeto na ordem em que os requisitos aparecem no readme. Isso acontece para deixar o processo de avaliação mais rápido. Então, não se assuste se isso acontecer, ok?

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 3





jeanpsv Jean Paulo Silva Vasconcelos



vinicius-vasconcelos Vinicius-Vasconcelos



GabrielCoruja Gabriel Dalseco

Languages

● **Shell** 100.0%