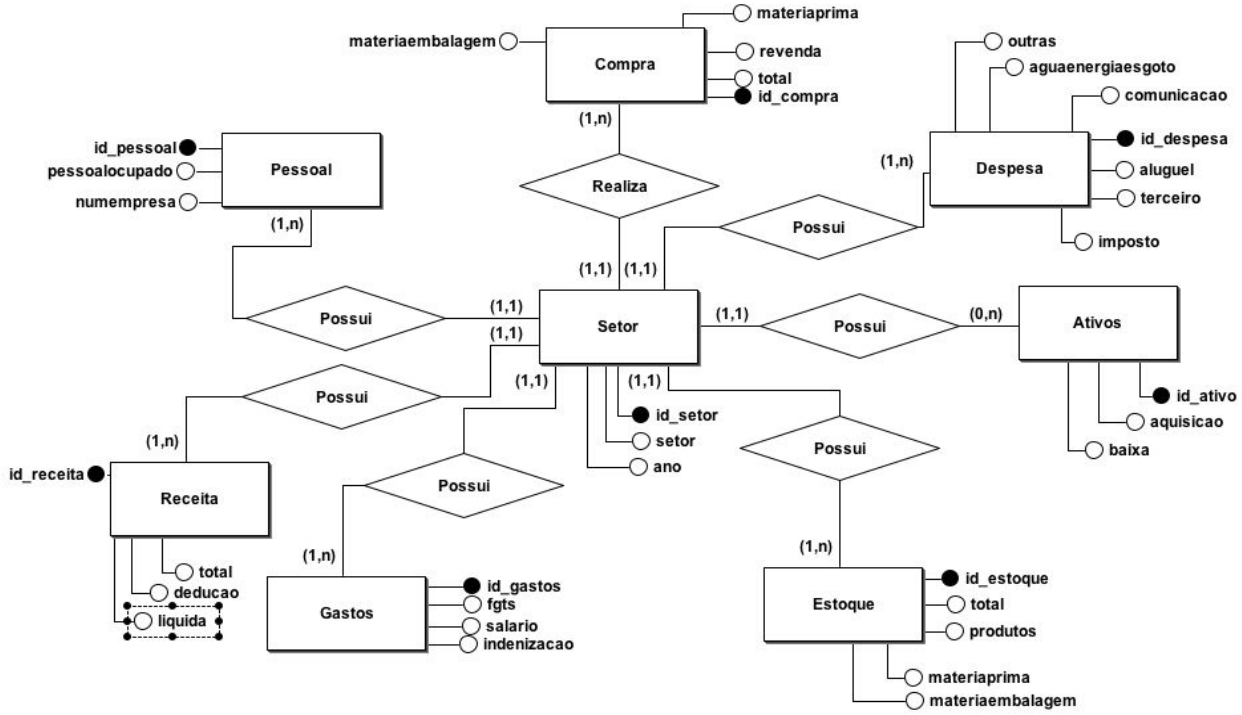
Relatório

Giovany da Silva Santos - 2018007758
João Marcos Calixto Moreira - 2018009636
Leonardo dos Santos De Souza - 2018012444

1.MER



O modelo de entidade e relacionamento prevê como vamos construir nosso banco.

Entidade	Atributos
SETOR: responsável por armazenar todos as categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. setor: responsável pelo armazenamento do nome. ano: responsável pelo armazenamento do ano da pesquisa.
ATIVOS:responsável por armazenar todos os ativos tangíveis das categorias dos setores do comércio.	id:responsável pela identificação de cada entidade. aquisicao: número de aquisições de ativos. baixa: número de baixas de ativos.
ESTOQUE:responsável por armazenar todos os estoques das categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. total: número total no estoque. produtos:número de produtos no estoque. materiaPrima:número de matéria prima para o produto no estoque.

	materiaEmbalagem:número de matéria prima para o embalagem no estoque.
GASTOS:responsável por armazenar todos os gastos das categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. salario: número do valor em milhares dos custos com salário. fgts: número do valor em milhares dos custos com fgts. indenizacao:número do valor em milhares dos custos com indenizações.
RECEITA:responsável por armazenar todos as receitas das categorias dos setores do comércio.	id:responsável pela identificação de cada entidade. total: número do valor total em milhares das receitas. deducao: número do valor dedutivo em milhares das receitas. liquida:número do valor líquido em milhares das receitas.
COMPRAS:responsável por armazenar todos as compras das categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. total: número total de compras. materiaPrima:número de matéria prima para o produto compradas. materiaEmbalagem:número de matéria prima para o embalagem compradas.
PESSOAL: responsável por armazenar todos o número de pessoas e o número de empresas das categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. numEmpresas: número de empresas pessoalOcupado:número de pessoas da categoria registrado no dia 31 de dezembro.
DESPESA: responsável por armazenar todos as despesas das categorias dos setores do comércio.	id: responsável pela identificação de cada entidade. aluguel: despesas com aluguéis. terceiro: despesas com terceiros. comunicacao: despesas com comunicação. aguaEnergiaEsgoto: despesas com água, energia e esgoto. imposto: despesas com impostos outros: despesas com outras coisas que não encaixam nos itens anteriores.

2.Modelo Lógico (Modelo Relacional)

Setor(id_setor* , setor , ano)

Despesa(id_despesa* ,id_setor* , aluguel , terceiro , imposto , aguaenergiaesgoto , comunicacao , outras)
id_setor referencia Setor (id_setor)

Gastos(id_gastos* , id_setor* , fgts , salario , indenizacao)
id_setor referencia Setor(id_setor)

Receita(id_receita* , id_setor* , total , deducao , liquida)
id_setor referencia Setor (id_setor)

Pessoal (id_pessoal* , id_setor* , pessoalocupado , numempresa)
id_setor referencia Setor(id_setor)

Estoque(id_estoque* , id_setor* , total , produtos , materiaprima , materiaembalagem)
id_setor referencia Setor(id_setor)

Ativos(id_ativos* , id_setor* , aquisicao , baixa)
id_setor referencia Setor(id_setor)

Compras(id_compra* , id_setor* , total , revenda , materiaprima , materiaembalagem)
id_setor referencia Setor(id_setor)

3.ROLES

ROLE	PERMISSÕES
supervisor de encargos	tem permissão sobre operações CRUD nas tabelas gastos,despesa,e receita.Além disso também possui a permissão de executar a função consulta total que retorna o total de compras ,receita e estoque de um determinado setor em dado ano. Ex: select * from consultaTotal('1.Total',2009)

auxiliar de encargos	tem permissão sobre operações select nas tabelas gastos,despesa,e receita.
supervisor de insumos	tem permissão sobre operações CRUD nas tabelas ativos,compra e estoque
auxiliar de insumos	tem permissão sobre operações select nas tabelas ativos,compra e estoque
supervisor de pessoal	tem permissão sobre operações CRUD na tabela pessoal
auxiliar de pessoal	tem permissão sobre operações select na tabela pessoal

4.FUNÇÕES

Criamos uma função consulTotal que recebe dois parâmetros (tipo do setor e ano) e ela retorna uma consulta com o total de receita, compras e estoque desse determinado setor em um dado ano.Escopo da função:

```
create or replace function consultaTotal(tipo_setor varchar,ano_setor integer)
returns table(
        total_receita bigint,
        total_estoque int,
        total_compra bigint)
as $$
begin
return QUERY
        select r.total,e.total,c.total from setor s,
        receita r join estoque e on(r.id_setor=e.id_setor) join compra c
        on(r.id_setor=c.id_setor and e.id_setor=c.id_setor)
        where s.id_setor=r.id_setor and s.setor=tipo_setor and s.ano=ano_setor;
end;
$$ language plpgsql
```

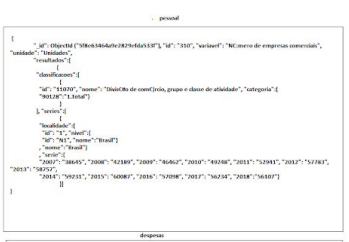
5.ÍNDICES

Nome atributo	Nome Tabela
Total	Receita
Total	Compra
Total	Estoque

Criamos os índices sobre os atributos “total” das tabelas: receita,compra e estoque, visto que a função consultaTotal utiliza esses três atributos, assim a execução desta função terá uma melhor performance,isto é útil, pois toda vez que o usuário do tipo supervisor encargos utilizá-la terá uma resposta mais rápida.

6.API e o modelo Orientado a documentos

Os bancos orientados a documentos são projetados para fazer o armazenamento ,recuperação e gerência de informações orientadas a documentos.Para o nosso trabalho utilizamos a API do IBGE , que retorna uma url do objeto do tipo JSON de acordo com os dados que selecionamos e a partir dessa, fazemos a inserção de um objeto do tipo json em uma base de dados no SGBD Mongo.Abaixo temos o modelo orientado a documentos:



7.COUNT TABELAS

Foi feito uma seleção da quantidade de registros e cadas uma das tabelas possui a quantidade de registros igual a 600, já que são 12 anos diferentes com cada registro indo de 0 até 49 no json:

```
select count(*) from ativos
```

	count bigint
1	600

```
select count(*) from empresa
```

	count bigint
1	600

```
select count(*) from compra
```

	count bigint
1	600

```
select count(*) from estoque
```

	count bigint
1	600

```
select count(*) from despesa
```

	count bigint
1	600

```
select count(*) from gastos
```

	count bigint
1	600

```
select count(*) from receita
```

	count bigint
1	600

```
select count(*) from pessoal
```

	count bigint
1	600

8.COUNT COLLECTIONS

```
> use trabalho
switched to db trabalho
> db.ativos.count()
2
> db.compras.count()
4
> db.depesas.count()
6
> db.estoques.count()
4
> db.gastos.count()
3
> db.pessoal.count()
2
```

```
> db.receitas.count()
3
```

Link Video:
https://drive.google.com/file/d/1r2u8KAT_pDvB0KU5SimivuTb0yYdGZQU/view?usp=sharing

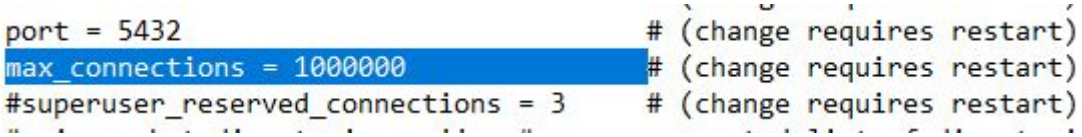
9.Testes JMeter
9.1Teste Postgres
9.1.1 Configurações

Os testes com o JMeter foram realizados de modo local e em uma máquina com as seguintes configurações:

- Processador Intel Core i5-7200 CPU@2.50GHz 2.71GHz.
- Memória 4.00 GB.
- Sistema Operacional Windows 10 Home Single Language de 64 bits.

A versão do JMeter utilizada foi a 2.1.0, a versão do JDBC para a conexão com o Postgres foi a 42.2.18 e a versão do SGBD Postgres, a 13.1.A versão do Mongo é a 4.4.1 e a do driver JMeter para teste do Mongo 3.12.6.

Antes de iniciar os testes configuramos o número de conexões simultâneas máximas para 1.000.000 de conexões como pode ser visto na figura abaixo.



Após isso, seguimos os testes da seguinte maneira:
Iniciamos os testes com 10 usuários seguindo em uma escala multiplicativa de fator 10.Foi encontrado erro de requisições para 1000 usuários.Então, utilizamos a ideia de sempre retroceder ao último teste que deu certo e “subir” o número de usuários em uma escala menor.Após retroceder para o 100, aumentamos os testes de 100 em 100,houve erro na casa dos 900.Então, a nova mudança de escala foi “subir” de 10 em 10, na qual obtivemos erros na casa do 860 usuários.A partir daí, percebemos que o limite para a consulta estava nesse intervalo 850 -860,de modo mais preciso o limite encontrado foi de 855 usuários simultâneos.

A consulta escolhida foi uma consulta com a seguinte descrição “retornar todas as informações referentes a estoques,compras,ativos,receitas,gastos,despesas,pessoal de um determinado setor em um dado ano”.E foi fixada uma requisição por usuário.Abaixo temos o comando SQL executado no JMeter:

Consulta:

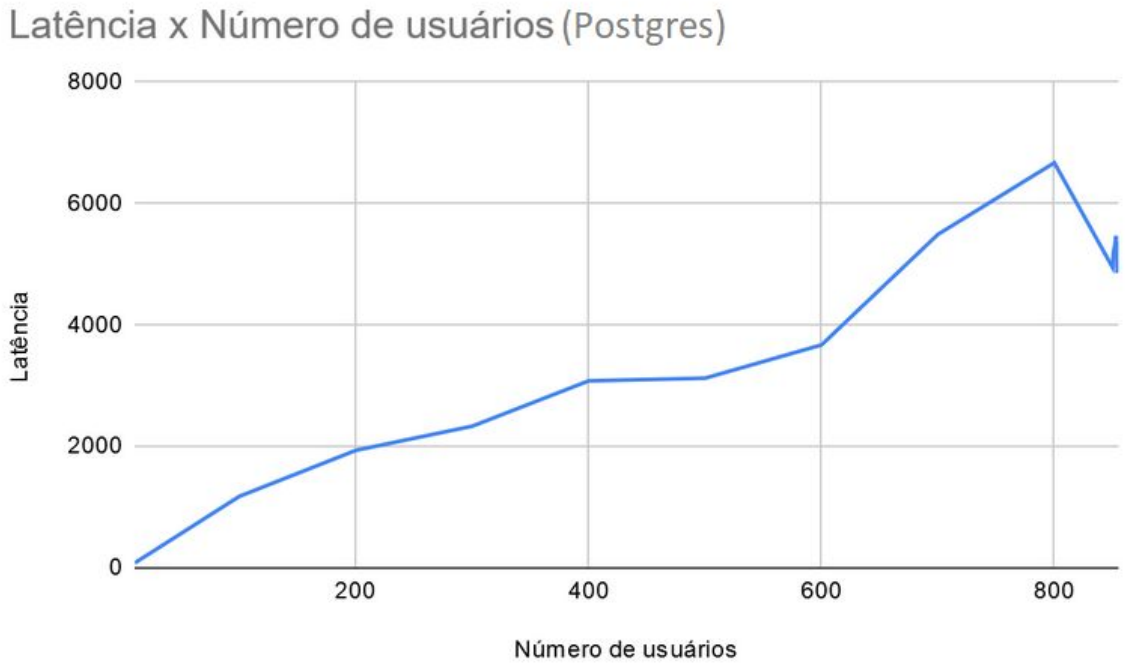
```
1 select * from setor s join  estoque e on (s.id_setor=e.id_setor) join compra c on (s.id_setor=c.
  id_setor) join ativos a on (s.id_setor=a.id_setor) join
2 receita r on (s.id_setor=r.id_setor) join gastos g on (s.id_setor=g.id_setor) join despesa d on (s.
  id_setor=d.id_setor) join pessoal p on (s.id_setor=p.id_setor)
3 where s.setor='2.Comercio de veiculos, pecas e motocicletas' and s.ano='2010'
```

9.1.2 Latência x Número de usuários

Para a análise do tempo entre uma requisição e a sua resposta (latência) montamos um gráfico que foi plotado a partir dos números de usuários, no eixo x, e a média de latência para o número de usuários, no eixo y.

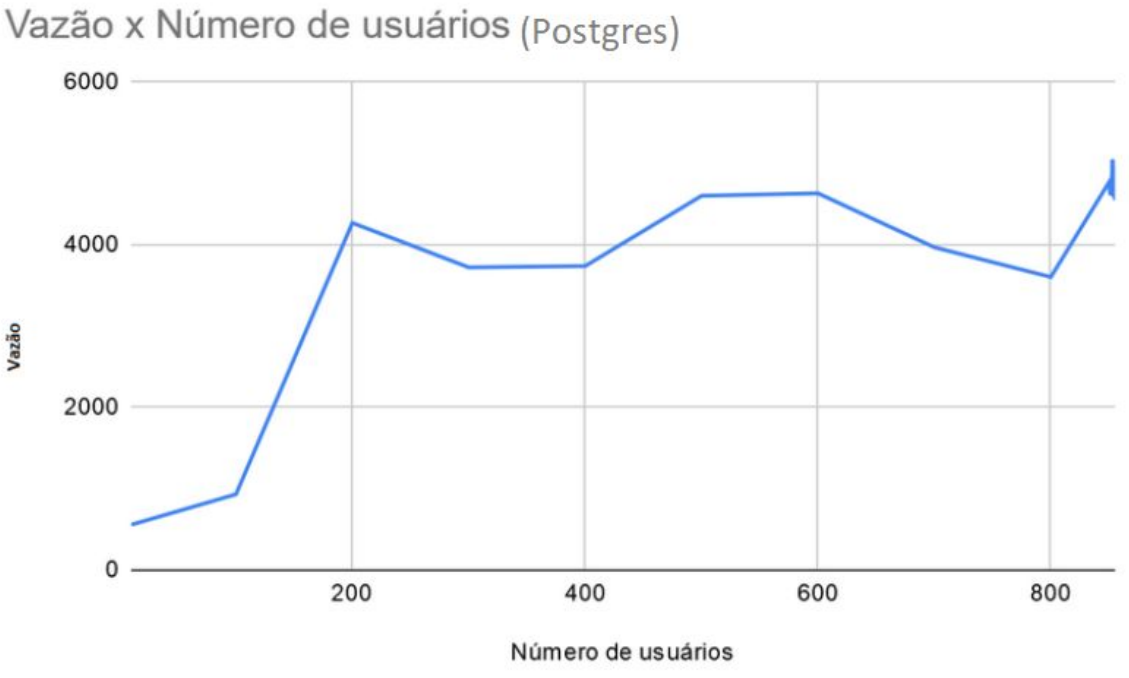
Como podemos observar, a variação de latência média de 10 para 100 usuários foi muito expressiva, indicando assim que o tempo de resposta da requisição cresce à medida que o número de usuários também cresce.Ao se aproximar do limite de usuários encontrados, percebeu se uma queda da latência indicando que o tempo de resposta não seguiria mais a tendência do número de usuários.A partir do limite 855 houve falha nas

respostas das requisições, e com isso, obtivemos a limitação para essa consulta no banco(sua escalabilidade).



9.1.3 Vazão X Número de usuários

Temos também o gráfico relacionado a quantidade de operações realizadas em um determinado espaço de tempo(vazão), que no nosso caso é medida em minutos para todos os testes.Abaixo temos o gráfico Vazão x Número de usuários:

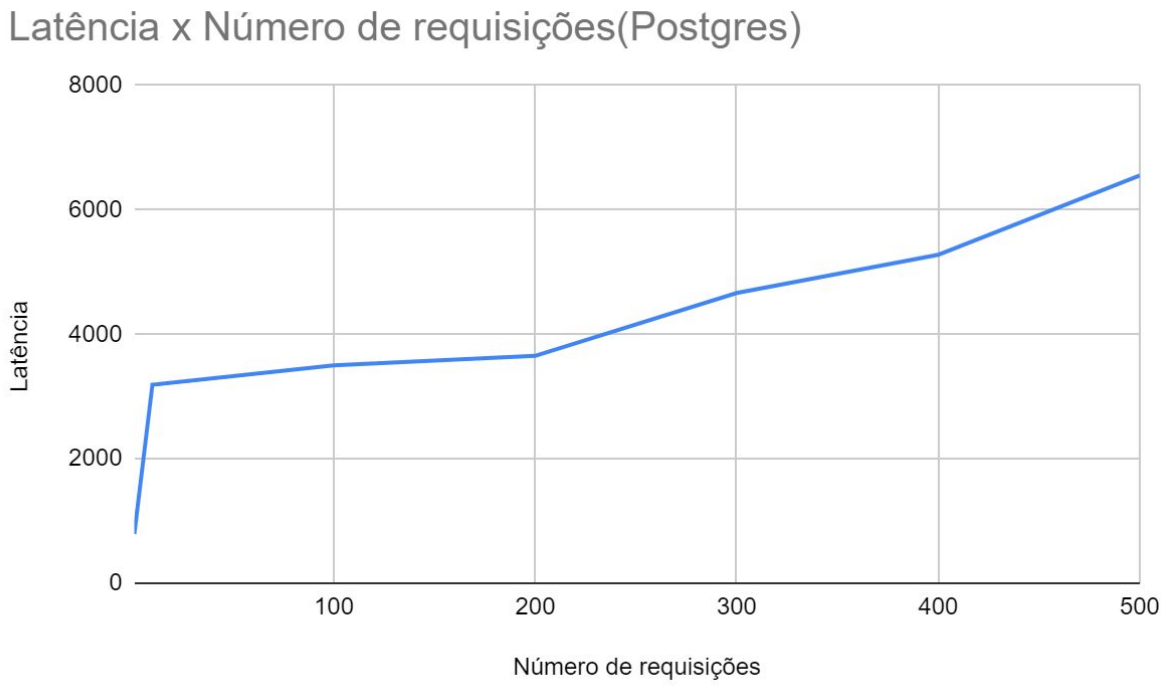


Como podemos observar: a vazão sofreu um grande acréscimo quando alteramos o número de usuários de 100 para 200.Uma outra observação é que, após a vazão atingir a faixa dos 3K-5K, a vazão ficou nessa região até o limite encontrado de usuários para consulta.Desse modo foi possível observar que essa faixa é o limite do número de operações por minuto para a consulta no banco.

9.1.4 Latência x Número de requisições

Para os testes relacionados a variação do número de requisições, foram feitas uma limitação de 400 usuários simultâneos.

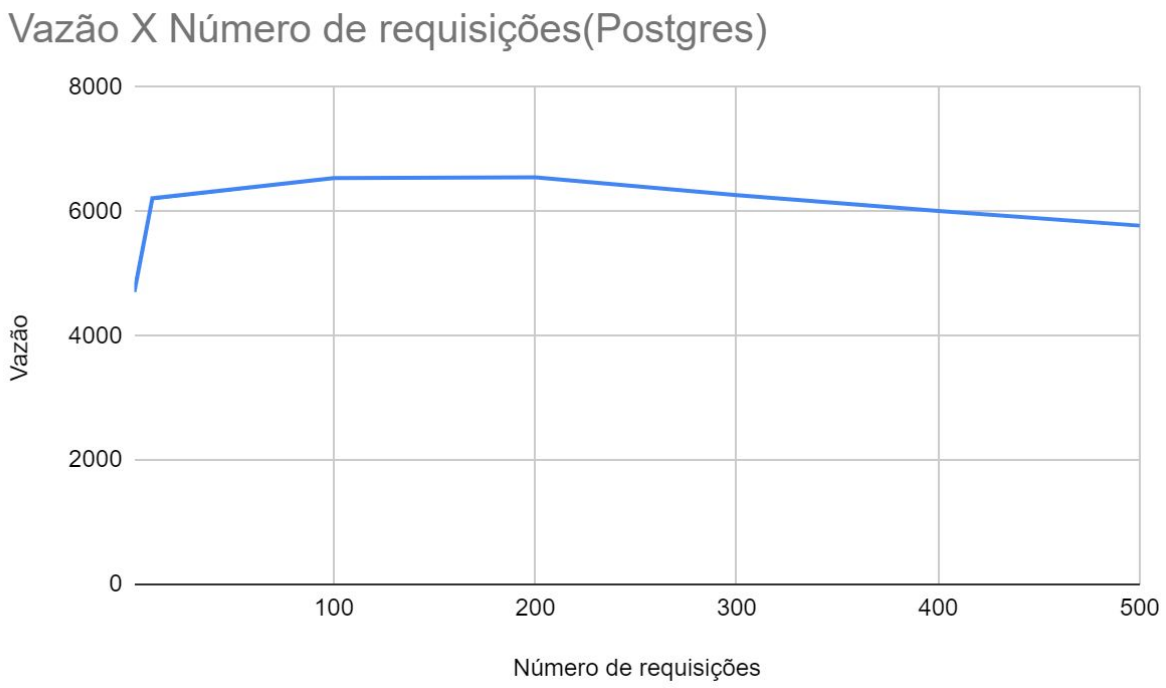
Avaliando o número de requisições, tivemos a seguinte adoção de teste: Aumentamos o número de requisições em um fator grande, porém quando obtivemos erro, retrocedemos ao último teste que possuiu êxito em todas as requisições.Mas, nesse teste fizemos o limite aproximado de requisições, uma vez que para o teste de requisições a duração foi bem longa.Abaixo temos o gráfico Latência x Número de requisições:



O testes foram iniciados para 1 requisição, podemos notar que houve uma grande variação (aproximadamente 3K) de latência quando aumentamos o número de requisições para 10.As variações de latência no intervalo 10-200 foram bem pequenas.A partir de 200 foi obtido uma variação quase que linear da latência.

A perda de resposta das requisições ocorreram quando colocamos 500 requisições para cada usuário.Assim, temos um limite aproximado de 500 requisições por usuário quando há 400 usuários simultâneos para a consulta especificada.

9.1.5 Vazão x Número de requisições

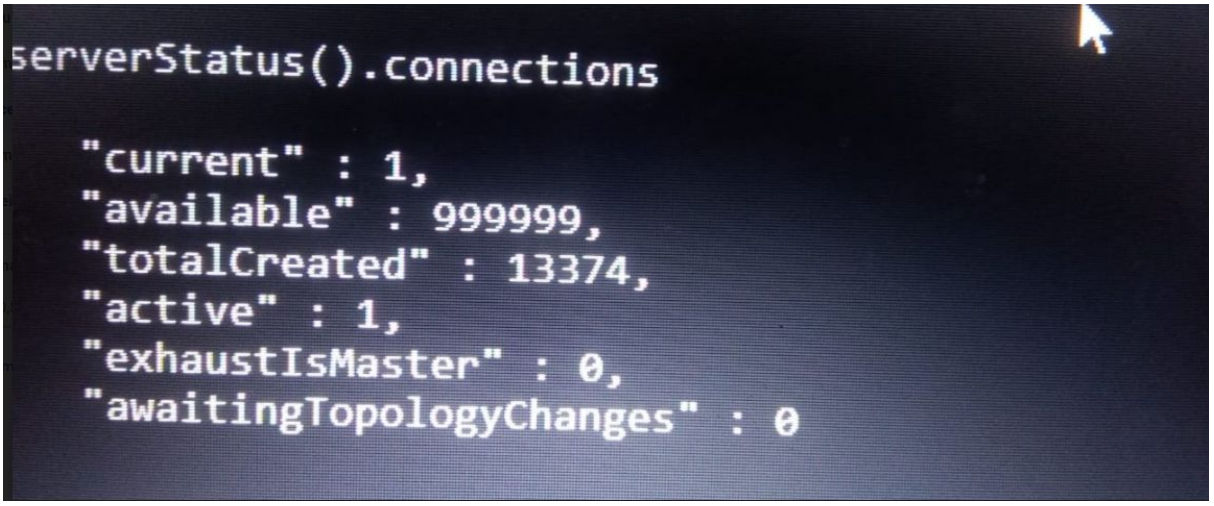


A vazão relacionada ao número de requisições apresentou um crescimento um pouco significativo(aproximadamente 2K) quando variamos o número de requisições de 1 para 10. Durante toda a execução dos testes a latência se manteve quase que constante, indicando um pequeno decrescimento no intervalo de 200-500.Uma observação é que as variações de latência entre os número de requisições indicadas foram muito similares neste intervalo.

9.2 Testes Mongo

9.2.1 Configurações

Abaixo, temos as configurações necessárias referentes ao Mongo.Notamos que a capacidade ,“available”, está setada para 999999 conexões máximas.Os nossos testes utilizaram um número bem menor de conexões, então esse limite não afetou nos resultados.

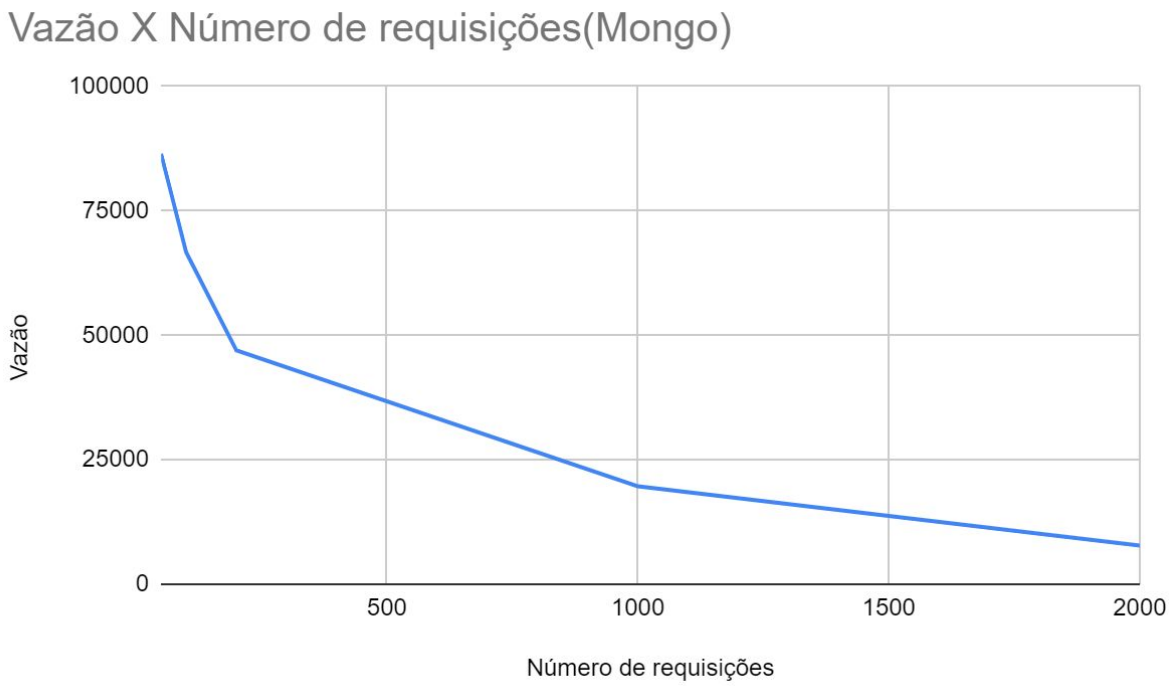


A consulta realizada foi a que tem o seguinte enunciado “Retornar todos os documentos da coleção receitas”.E que possui o seguinte comando : *db.receitas.find().*

Para os testes com o banco Mongo também foi adotada a ideia de ter um número de variação maior entre a quantidade de requisições e também entre a quantidade de usuários.Outra consideração é que, os testes apresentaram latência zero para ambas os gráficos (usuário e requisição) em todos os testes.Isso pode ser explicado pois os bancos orientado a documentos têm uma alta performance no tempo de resposta de consultas.Então, os gráficos que apresentamos são apenas relacionados a vazão.Além disso, a limitação de usuários simultâneos foi bem menor para os testes com o Mongo, uma vez que quando iniciamos os testes com uma quantidade maior de usuários o computador travava constantemente, impactando assim, na necessidade de sua reinicialização.

9.2.2 Vazão x Número de requisições

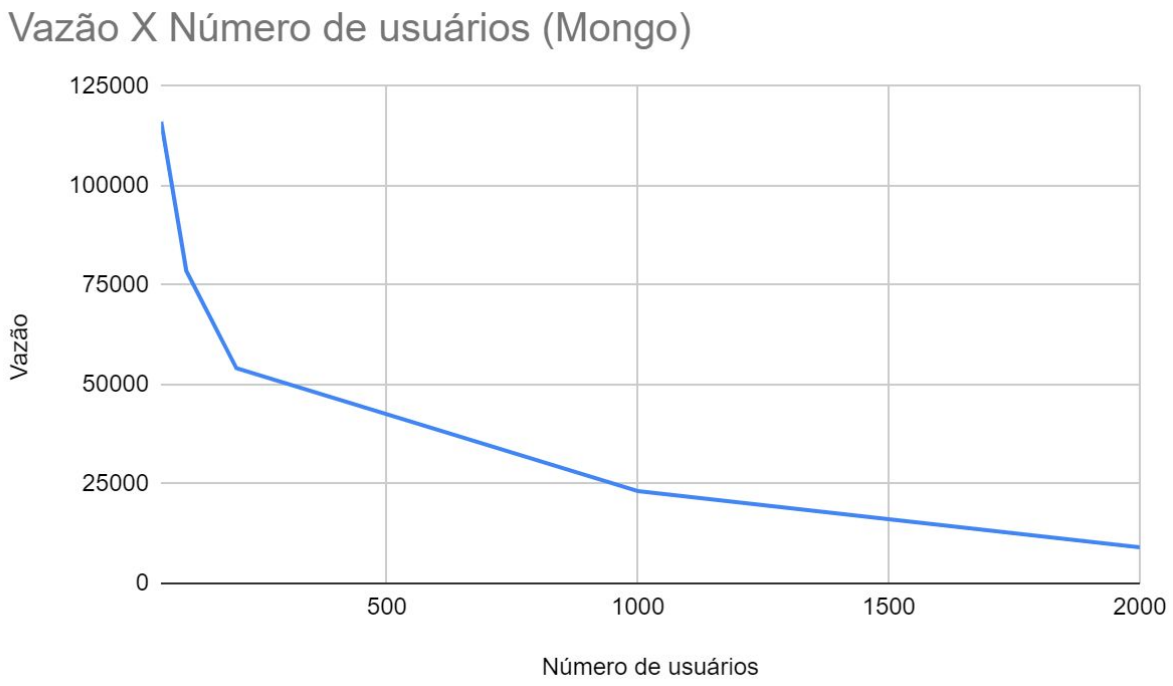
Para esse teste fixamos o número de usuários simultâneos em 25.



O teste apresentou um decrescimento da vazão à medida que o número de requisições foi crescendo.A variação mais alta de vazão foi no intervalo de 50-1000 requisições.

9.2.3 Vazão x Número de usuários

Para esse teste fixamos o número de requisições por usuário em 25.



Pelo gráfico é possível notar que, a vazão também apresenta uma tendência contrária a do parâmetro avaliado, que nesse caso é o número de usuários.Para esse teste, vemos uma outra similaridade que é a variação da vazão no intervalo 50-200, que foi bem acentuada igualmente ao ocorrido no teste quando fixamos o número de usuários(teste de requisições).

Para o nosso trabalho não será possível fazer a comparação dos testes realizados entre os diferentes bancos (Postgres e Mongo), uma vez que as consultas realizadas foram diferentes.