

Desafio Técnico CWI

Solução BACK-END para Gerenciar Sessões de Votação em Assembleias



Autor: Eliseu Silveira Brito

Data: 15/05/2020

GitHub: https://github.com/eliseusbrito/DesafioTecnicoCwiVotacaoAssembleia_SpringJPAHibernatePostgreSQL

App Heroku: <https://votacao-assembleia.herokuapp.com>

Sumário

Solução Back-end para Gerenciar Sessões de Votação em Assembleias	4
Utilização pelo Usuário.....	4
Operação	5
➤ Pauta	5
• Cadastro	5
• Listagem	7
• Edição.....	8
• Exclusão	8
➤ Sessão de Votação.....	9
• Abertura Sessão	9
• Tempo de Votação	9
• Comportamento	10
Localhost	10
Heroku	10
➤ Votos	12
• Cadastro	12
• Listagem	13
➤ Associados	14
• Cadastro	14
• Listagem	15
• Edição.....	16
• Exclusão	17
Resultado.....	17
➤ Mensageria e Filas	18
Programa Java	19
Tabela de Dados.....	19
Configuração.....	21
Classe de Configuração	21
Banco teste H2.....	22
Banco Postgre.....	22
➤ Banco Local e Heroku pelo Postgre:	22
➤ Dependências adicionadas:.....	22
➤ Obtendo script SQL a partir do PostgreSQL local	22
➤ 327. Executando script SQL no servidor remoto - HEROKU	24
➤ 329. Deploy do sistema no Heroku	26

Tratamento de Erros e exceções:	27
➤ Pesquisa por Id não existente da Classe Associado:	27
➤ Cadastrar associado sem informações no nome ou cpf:	27
➤ Adicionado mSG personalizada Ao cadastrar associado já existente	29
➤ Adicionado Mensagem QUANDO COLOCA LETRA ERRADA no Voto	29
➤ Adicionado Mensagem Quando Votação não ESTÁ aberta	29
➤ Mensagem quanto cadastra Pauta já existente	30
➤ Heroku	30
Instalações	30
RabbitMQ	30
Aprendizados	30
Tabela de Dados	30
Conceitos:	32
Pendências e Melhorias:	33
Conclusões	33
Anexos	34
Variação no api GET https://user-info.herokuapp.com/users/{cpf} :	34

Solução Back-end para Gerenciar Sessões de Votação em Assembleias

No cooperativismo, cada associado possui um voto e as decisões são tomadas em assembleias, por votação. Esta solução back-end foi criada para gerenciar essas sessões de votação.

Sendo executada na nuvem, promove as seguintes funcionalidades através de uma API REST:

- Cadastrar uma nova pauta;
- Abrir uma sessão de votação em uma pauta (a sessão de votação deve ficar aberta por um tempo determinado na chamada de abertura ou 1 minuto por default);
- Receber votos dos associados em pautas (os votos são apenas 'Sim'/'Não'. Cada associado é identificado por um id único e pode votar apenas uma vez por pauta);
- Contabilizar os votos e dar o resultado da votação na pauta.

As pautas e os votos são persistidos e não são perdidos com o restart da aplicação.

Para fins de exercício, a segurança das interfaces foi abstraída e qualquer chamada para as interfaces é considerada como autorizada. A linguagem escolhida foi Java, e utiliza ferramentas como o Spring, Rest, JPA, Hibernate, H2, PostgreSQL, Heroku, Docker, RabbitMQ, Json e Postman.

A solução também foi integrada com um sistema externo que verifica a partir do CPF do associado, se ele pode votar.

O resultado da votação pode ser informado para o restante da plataforma, isto é feito através de mensageria. A solução possui um Producer que pode ser acessado via end point através de comando Get. Quando a sessão de votação fechar o usuário pode postar uma mensagem com o resultado da votação, por exemplo, via Postman.

UTILIZAÇÃO PELO USUÁRIO

O responsável pela assembleia deve cadastrar os associados que ficarão aptos a votar.

Este também precisa cadastrar uma pauta de votação.

Após os dois pré-requisitos anteriores estarem atendidos, pode-se abrir uma votação com tempo de votação default de 1 minuto ou pode-se também definir este tempo.

Durante o período de votação, os associados poderão fazer seu voto com as opções de Sim ou Não.

Quando concluído o tempo de votação o sistema informará o resultado.

Caso seja aconteça um empate ou nenhum voto, uma nova pauta deverá ser cadastrada para esta nova votação.

OPERAÇÃO

Todos os comandos foram testados e funcionam utilizando o Postman, tanto no ambiente Heroku ou localhost.

Demais alternativas de envio de requisições, como por exemplo o Navegador Chrome Desktop ou Mobile, foram testadas, tendo comportamento semelhante ao Postman.

No GitHub do projeto tem duas Coleções, uma para testes no Heroku e outra para teste em localhost.

➤ PAUTA

- CADASTRO

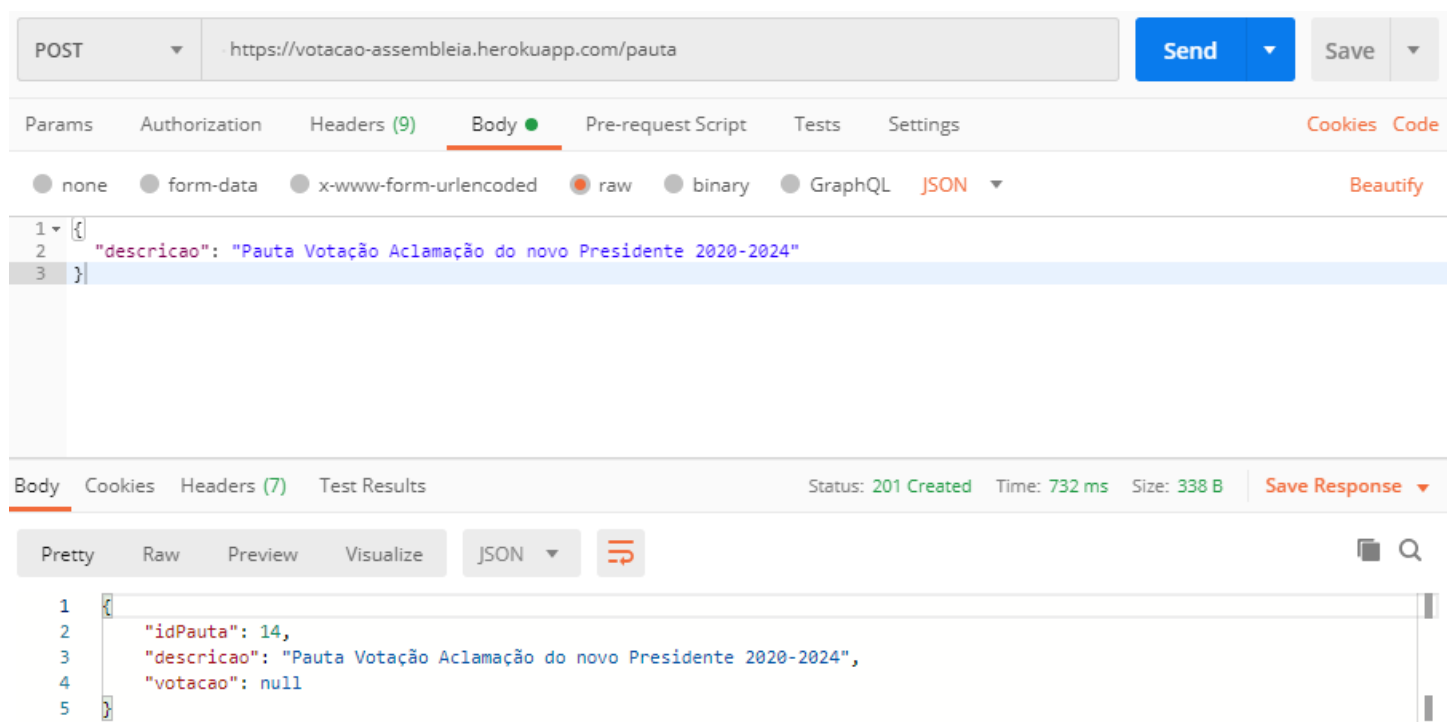
O utilizador pode cadastrar uma pauta utilizando o comando POST:

<https://votacao-assembleia.herokuapp.com/pauta> ou <http://localhost:8080/pauta>

No Body deve-se usar o padrão Json conforme a seguir:

```
{  
  "descricao": "Título da Pauta"  
}
```

Exemplo no Postman:



A solução foi integrada com um sistema externo que verifica, a partir do CPF do associado, se ele pode votar.

- GET `https://user-info.herokuapp.com/users/{cpf}`
- Caso o CPF seja inválido, a API retornará o HTTP Status 404 (Not found).
- Caso o CPF seja válido, a API retornará se o usuário pode (ABLE_TO_VOTE) ou não pode (UNABLE_TO_VOTE) executar a operação

Estas requisições foram integradas e caso o CPF seja válido a votação se dá normalmente.

Caso o CPF não seja válido o sistema retorna mensagem:

“CPF não é válido para votação: {"status":"UNABLE_TO_VOTE"}”

The screenshot shows a REST client interface with a POST request to `https://votacao-assembly.herokuapp.com/voto/pauta/1/associado/1/escolha/5`. The response status is 400 Bad Request. The response body is displayed in JSON format, showing the error message: `CPF não é válido para votação: {"status":"UNABLE_TO_VOTE"}`.

Caso o número do cpf cadastrado do Associado tenha alguma divergência com a base de dados pesquisada a mensagem é conforme a seguir:

“Erro ao obter dados do CPF na URL”

The screenshot shows a REST client interface with a POST request to `https://votacao-assembly.herokuapp.com/voto/pauta/1/associado/5/escolha/5`. The response status is 400 Bad Request. The response body is displayed in JSON format, showing the error message: `Erro ao obter dados do CPF na URL.`

- LISTAGEM

Para pesquisar todas as pautas cadastradas usa-se o comando GET

<https://votacao-assembly.herokuapp.com/pauta> ou <http://localhost:8080/pauta>

```
GET https://votacao-assembly.herokuapp.com/pauta Send

Pretty Raw Preview Visualize JSON

1 [
2   {
3     "idPauta": 1,
4     "descricao": "Votacao para aumentar numero de Sócios",
5     "votacao": {
6       "idVotacao": 1,
7       "sim": 2,
8       "nao": 1,
9       "total": 3,
10      "decisao": "Pauta id=1 Aprovada!"
11    }
12  },
13  {
14    "idPauta": 2,
15    "descricao": "Reduzir taxa de juros dos Financiamentos",
16    "votacao": {
17      "idVotacao": 2,
18      "sim": 1,
19      "nao": 1,
20      "total": 2,
21      "decisao": "Pauta id=2 precisa nova votação!"
22    }
23  },
24  {
```

Para listar uma pauta especifica usa-se o comando GET / número da pauta desejada

<https://votacao-assembly.herokuapp.com/pauta/{id}> ou <http://localhost:8080/pauta/{id}>

{id} = número da pauta que se quer listar

Ex.: <https://votacao-assembly.herokuapp.com/pauta/6>

```
GET https://votacao-assembly.herokuapp.com/pauta/6 Send

Body Cookies Headers (6) Test Results Status: 200 OK Time: 718 ms Size: 314 B Save

Pretty Raw Preview Visualize JSON

1 {
2   "idPauta": 6,
3   "descricao": "Aprovar verba para propaganda",
4   "votacao": {
5     "idVotacao": 6,
6     "sim": 2,
7     "nao": 0,
8     "total": 2,
9     "decisao": "Pauta id=6 Aprovada!"
10  }
11 }
```

- EDIÇÃO

Para editar a pauta desejada usa-se o comando PUT:

<https://votacao-assembly.herokuapp.com/pauta/{id}> ou <http://localhost:8080/pauta/{id}>

{id} = número da pauta que se quer editar

No Body deve-se usar o padrão Json conforme a seguir:

```
{  
  "descricao": "Alteração do Título da Pauta"  
}
```

Exemplo:

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** `https://votacao-assembly.herokuapp.com/pauta/3`
- Body Type:** JSON
- Body Content:**

```
1 {  
2   "id": 3,  
3   "descricao": "Aprovar verba para Propaganda"  
4 }
```

- EXCLUSÃO

Para excluir a pauta desejada usa-se o comando DELETE:

<https://votacao-assembly.herokuapp.com/pauta/{id}> ou <http://localhost:8080/pauta/{id}>

{id} = número da pauta que se quer excluir

Exemplo:

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `https://votacao-assembly.herokuapp.com/pauta/17`

<https://votacao-assembly.herokuapp.com/votacao/{id}/tempo/{t}> ou

<http://localhost:8080/votacao/{id}/tempo/{t}>

{id} = número da pauta que se quer abrir a sessão de votação.

{t} = tempo que a sessão de votação ficará aberta em minutos.

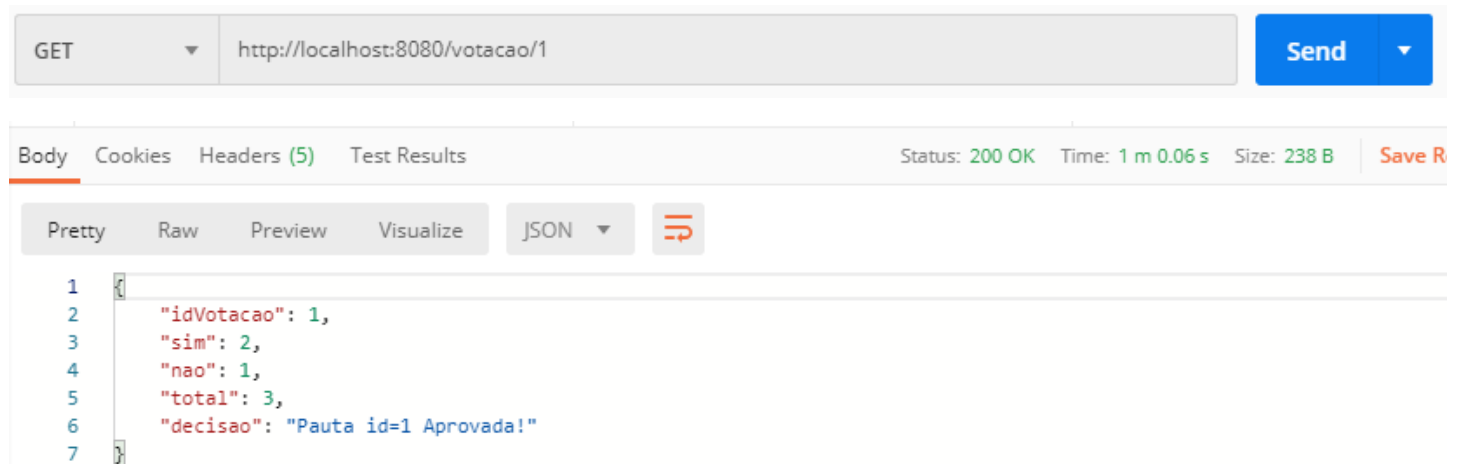
Exemplo: <https://votacao-assembly.herokuapp.com/votacao/1/tempo/2>

- COMPORTAMENTO

A resposta da requisição get, com tempo de votação dentro do Java, tem comportamentos diferentes dentro do localhost e do heroku. No local host responde normalmente. No Heroku responde com mensagens de erro mas os dados são gravados normalmente.

Localhost

Responde conforme imagem abaixo:



Testes mostram que tanto no ambiente de teste (Banco de dados H2) como no desenvolvimento (Banco de Dados Postdegre), tem o mesmo comportamento.

Heroku

Responde com erro conforme imagens abaixo: Could not get any response ou Status: 503 Service Unavailable

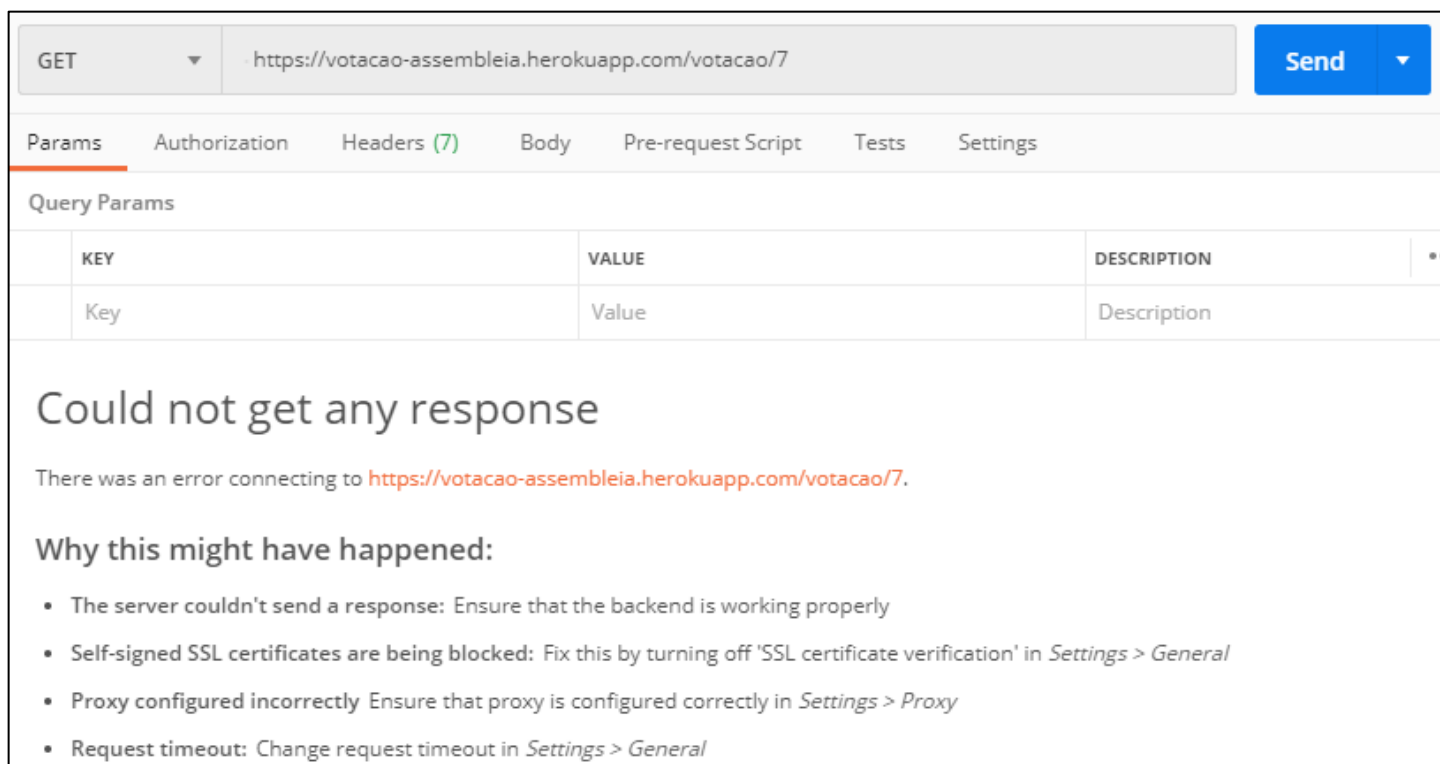


Imagem 1 - Could not get any response

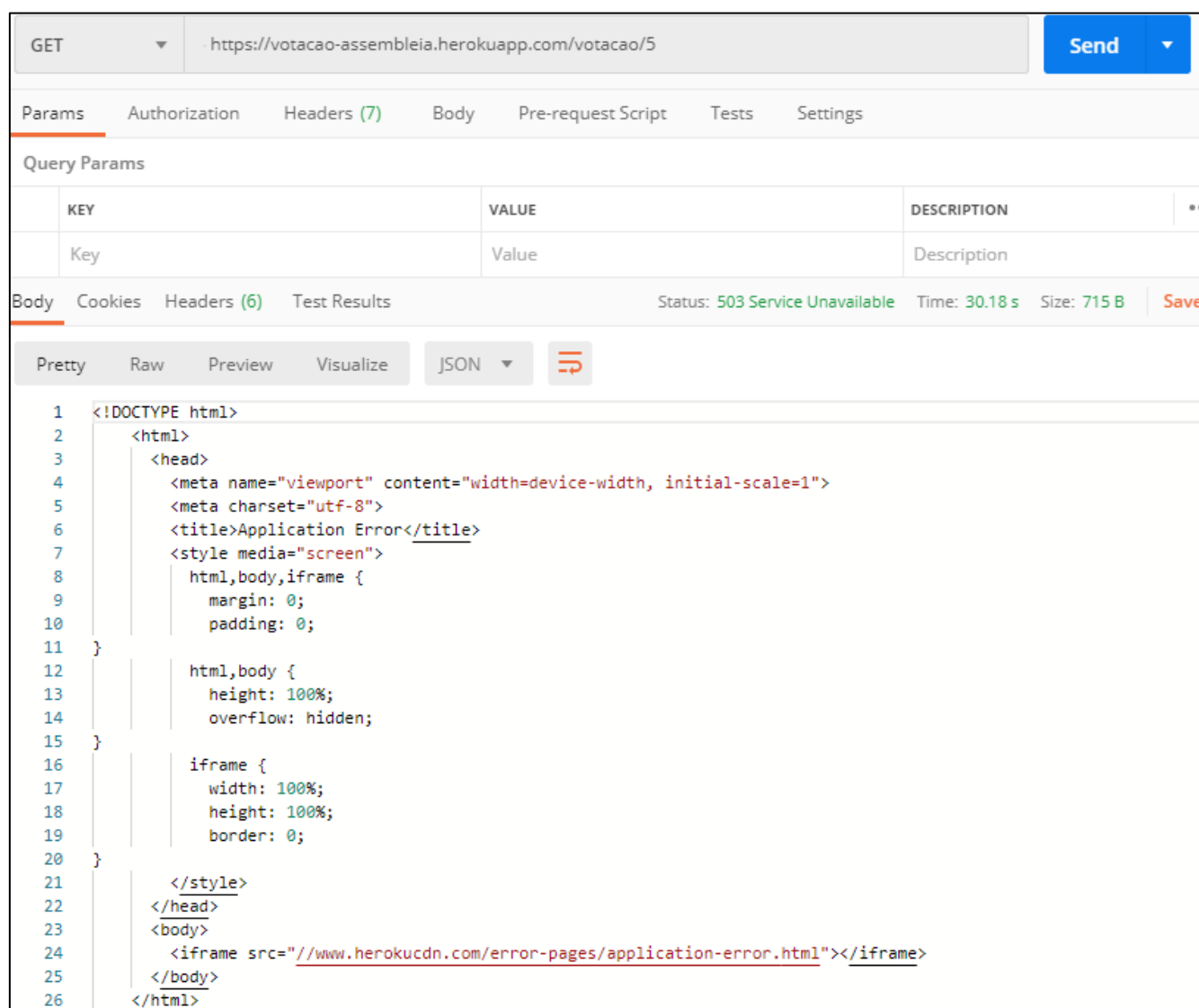
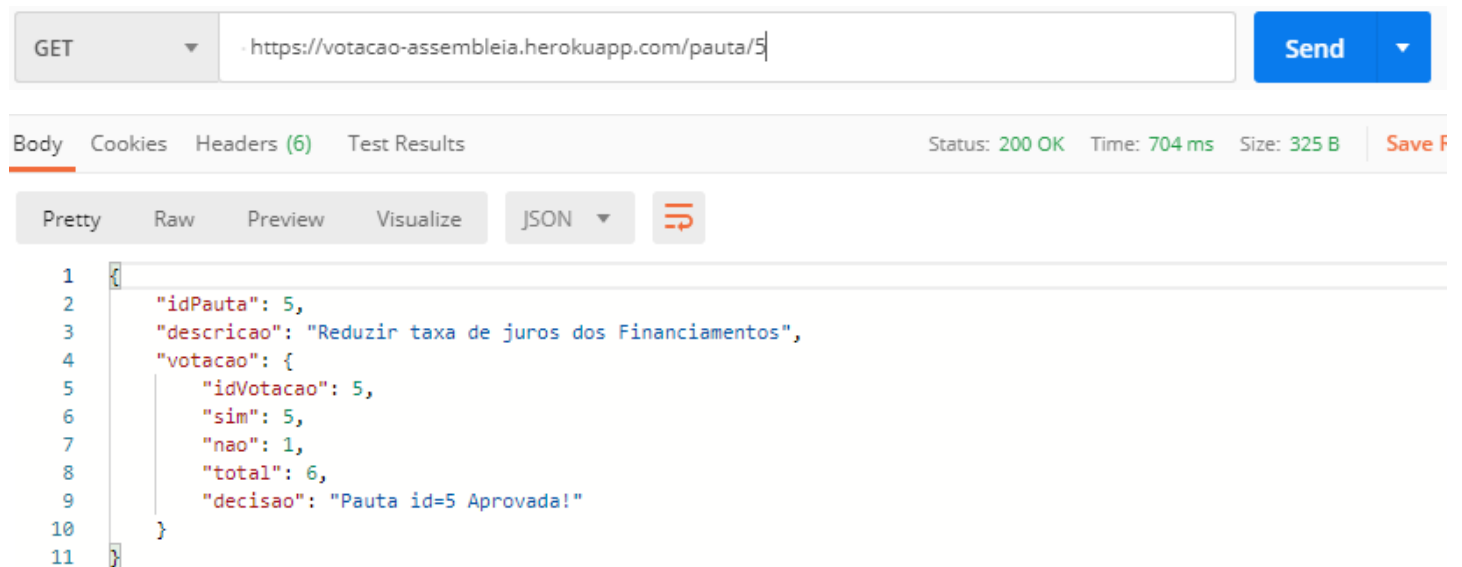


Imagem 2 - Status: 503 Service Unavailable

No entanto, os dados são gravados normalmente e pode-se confirmar listando a pauta que teve a votação.

Exemplo:



The screenshot shows a web browser interface for testing HTTP requests. The method is GET and the URL is `https://votacao-assembleia.herokuapp.com/pauta/5`. The response status is 200 OK, with a time of 704 ms and a size of 325 B. The response body is displayed in JSON format:

```
{
  "idPauta": 5,
  "descricao": "Reduzir taxa de juros dos Financiamentos",
  "votacao": {
    "idVotacao": 5,
    "sim": 5,
    "nao": 1,
    "total": 6,
    "decisao": "Pauta id=5 Aprovada!"
  }
}
```

Solução para este problema, que acontece somente no ambiente de produção, virá nas próximas atualizações em novos deploy para o heroku.

➤ VOTOS

- CADASTRO

Para votar o utilizador usa o comando POST:

<https://votacao-assembleia.herokuapp.com/voto/pauta/{idPauta}/associado/{idAssociado}/escolha/{escolha}>

ou <http://localhost:8080/voto/pauta/{idPauta}/associado/{idAssociado}/escolha/{escolha}>

{idPauta} = Número da Pauta que se quer votar;

{idAssociado} = identificação do associado que esta votando.

{escolha} = para votar sim usa-se o S e para votar não se usa o N.

Exemplo:

<https://votacao-assembleia.herokuapp.com/voto/pauta/6/associado/1/escolha/S>

POST ▼ <https://votacao-assembly.herokuapp.com/voto/pauta/7/associado/3/escolha/5> Send ▼

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	

Body Cookies Headers (6) Test Results Status: 200 OK Time: 577 ms Size: 379 B Save

Pretty Raw Preview Visualize JSON ▼ ≡

```

1 {
2   "idVoto": 21,
3   "pauta": {
4     "idPauta": 7,
5     "descricao": "Votacao para aumentar numero de Sócios",
6     "votacao": null,
7     "hibernateLazyInitializer": {}
8   },
9   "associado": {
10    "id": 3,
11    "nome": "Pedrinho",
12    "cpf": "22345678900"
13  },
14   "escolha": "S"
15 }

```

- LISTAGEM

Para pesquisar deve-se usar o comando GET:

Busca todos os votos

<https://votacao-assembly.herokuapp.com/voto> ou <http://localhost:8080/voto>

Para buscar um voto específico:

<https://votacao-assembly.herokuapp.com/voto/{id}> ou <http://localhost:8080/voto/{id}>

{id} = Número do voto que se quer pesquisar

Exemplo:

<https://votacao-assembly.herokuapp.com/voto/1>

GET <https://votacao-assembleia.herokuapp.com/voto/1> Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (6) Test Results Status: 200 OK Time: 723 ms Size: 421 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "idVoto": 1,
3   "pauta": {
4     "idPauta": 1,
5     "descricao": "Votacao para aumentar numero de Sócios",
6     "votacao": {
7       "idVotacao": 1,
8       "sim": 2,
9       "nao": 1,
10      "total": 3,
11      "decisao": "Pauta id=1 Aprovada!"
12    }
13  },
14  "associado": {
15    "id": 1,
16    "nome": "Maria Brown",
17    "cpf": "666666666666"
18  },
19  "escolha": "S"
20 }

```

➤ ASSOCIADOS

• CADASTRO

Para cadastrar usuários usa-se o POST

<https://votacao-assembleia.herokuapp.com/associado> ou <http://localhost:8080/associado>

com o Body Json:

```

{
  "nome": "Nome do Associado",
  "cpf": "xxxxxxxxxx"
}

```

```

1 {
2   "nome": "Maria",
3   "cpf": "57658785040"
4 }

```

POST <https://votacao-assembleia.herokuapp.com/associado> Send

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 201 Created Time: 824 ms Size: 286 B Save

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 21,
3   "nome": "Maria",
4   "cpf": "57658785040"
5 }

```

- LISTAGEM

Para pesquisar todos os usuários cadastrados usa-se o comando GET

<https://votacao-assembleia.herokuapp.com/associado> ou <http://localhost:8080/associado>

GET <https://votacao-assembleia.herokuapp.com/associado> Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies C

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk E
Key	Value	Description		

Body Cookies Headers (6) Test Results Status: 200 OK Time: 763 ms Size: 1.15 KB Save Response

Pretty Raw Preview Visualize JSON ↕

```

1 [
2   {
3     "id": 1,
4     "nome": "Maria Brown",
5     "cpf": "66666666666"
6   },
7   {
8     "id": 2,
9     "nome": "Alex Green",
10    "cpf": "12345678900"
11  },
12  {
13    "id": 3,
14    "nome": "Pedrinho",
15    "cpf": "22345678900"
  }
]
```

Para listar associado específico:

<https://votacao-assembleia.herokuapp.com/associado/{id}>

ou <http://localhost:8080/associado/{id}>

{id} = número da identificação do associado

Exemplo: <http://localhost:8080/associado/1>

GET <https://votacao-assembleia.herokuapp.com/associado/1> Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	

Body Cookies Headers (6) Test Results Status: 200 OK Time: 721 ms Size: 221 B Save

Pretty Raw Preview Visualize JSON ↕

```

1 {
2   "id": 1,
3   "nome": "Maria Brown",
4   "cpf": "66666666666"
5 }
```

- EDIÇÃO

Para editar o associado desejada usa-se o comando PUT:

<https://votacao-assemblya.herokuapp.com/associado/{id}>

ou <http://localhost:8080/associado/{id}>

{id} = número da pauta que se quer editar

No Body deve-se usar o padrão Json conforme a seguir:

```
{  
  "id":1,  
  "nome":"Nome do Associado",  
  "cpf":"xxxxxxxxxx"  
}
```

Exemplo:

The screenshot shows a REST client interface with a PUT request to `https://votacao-assemblya.herokuapp.com/associado/1`. The request body is a JSON object: `{ "id": 1, "nome": "João da Silva Antunes", "cpf": "66948824000" }`. The response status is 200 OK, with a time of 186 ms and a size of 223 B. The response body is also shown as a JSON object: `{ "id": 1, "nome": "João Antunes", "cpf": "66948824000" }`.

```
PUT https://votacao-assemblya.herokuapp.com/associado/1
```

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "id": 1,  
3   "nome": "João da Silva Antunes",  
4   "cpf": "66948824000"  
5 }
```

Body Cookies Headers (6) Test Results Status: 200 OK Time: 186 ms Size: 223 B Save

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 1,  
3   "nome": "João Antunes",  
4   "cpf": "66948824000"  
5 }
```


- EXCLUSÃO

Para excluir o associado desejado usa-se o comando DELETE:

<https://votacao-assembleia.herokuapp.com/associado/{id}>

ou <http://localhost:8080/associado/{id}>

{id} = número da pauta que se quer excluir

Exemplo:

DELETE	https://votacao-assembleia.herokuapp.com/associado/19	Send
--------	---------------------------------------------------------------------------------------------------------------------------	------

RESULTADO

O resultado da votação é informado ao terminar o tempo de votação como resposta desta solicitação.

GET	http://localhost:8080/votacao/1	Send
-----	-------------------------------------------------------------------------------	------

Body	Cookies	Headers (5)	Test Results	Status: 200 OK	Time: 1 m 0.06 s	Size: 238 B	Save R
<div>Pretty Raw Preview Visualize JSON</div> <pre>1 { 2 "idVotacao": 1, 3 "sim": 2, 4 "nao": 1, 5 "total": 3, 6 "decisao": "Pauta id=1 Aprovada!" 7 }</pre>							

Também é possível verificar o resultado consultando a pauta desejada.

GET	https://votacao-assembleia.herokuapp.com/pauta/1	Send
-----	-----------------------------------------------------------------------------------------------------------------	------

Pretty	Raw	Preview	Visualize	JSON
<pre>1 { 2 "idPauta": 1, 3 "descricao": "Votacao para aumentar numero de Sócios", 4 "votacao": { 5 "idVotacao": 1, 6 "sim": 2, 7 "nao": 1, 8 "total": 3, 9 "decisao": "Pauta id=1 Aprovada!" 10 } 11 }</pre>				

➤ MENSAGERIA E FILAS

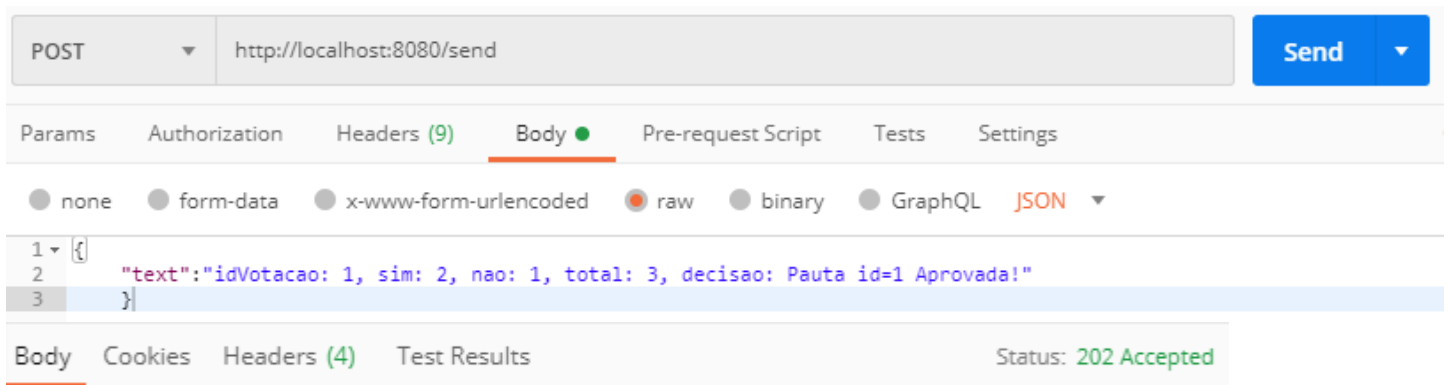
Nesta solução o resultado da votação pode ser enviado para um serviço de mensageria. Foi escrito um pacote no Java chamado Producer que produz esta mensagem e envia para o RabbitMQ, ficando na fila para ser consumida conforme disponibilidade de um Consumer. Esta disparada é via um endpoint via Postman utilizando o comando POST.

<http://localhost:8080/send>

No Body deve-se usar o padrão Json conforme a seguir:

```
{
  "text": "Resultado da Votação"
}
```

Exemplo:



Esta mensagem fica armazenada na fila do RabbitMQ, conforme assinalada na imagem abaixo:

The screenshot shows the RabbitMQ Admin interface. The 'Queues' tab is selected, showing a list of queues. The queue `rk.producer.votacaoAssembleia` is highlighted, and its 'Ready' message count of 3 is circled in red.

Overview				Messages			Message rates			
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	+/-
dl.producer.votacaoAssembleia	classic	D DLX DLK	idle	0	0	0				
pl.producer.votacaoAssembleia	classic	D	idle	0	0	0				
rk.producer.votacaoAssembleia	classic	D DLX DLK	idle	3	0	3	0.00/s	0.00/s	0.00/s	

Pode-se verificar o conteúdo da mensagem conforme abaixo:

The screenshot shows the RabbitMQ web interface at localhost:15672. The 'Queues' tab is selected. The message details for 'Message 3' are displayed. The message content is a JSON object: {"text": "idVotacao: 1, sim: 2, nao: 1, total: 3, decisao: Pauta id=1 Aprovada!"}. The message is 80 bytes long and is encoded as a string.

Exchange	Routing Key	Redelivered	Properties	Payload
ex.producer.votacaoAssembleia	rk.producer.votacaoAssembleia	0	priority: 0 delivery_mode: 2 headers: __TypeId__: VotacaoAssembleia.producer.dto.MessageQueue content_encoding: UTF-8 content_type: application/json	{"text": "idVotacao: 1, sim: 2, nao: 1, total: 3, decisao: Pauta id=1 Aprovada!"}

Uma opção de ativar o endpoint automaticamente ao termino de cada votação é possível e deverá estar disponível em versão futura desta solução.

Referencia:

<https://www.udemy.com/course/rabbitmq-com-springboot-e-docker/learn/lecture/17405674#overview>

PROGRAMA JAVA

A seguir, descreve-se pontos importantes durante o desenvolvimento

TABELA DE DADOS

A solução tem a estrutura de tabelas conforme abaixo. A imagem mostra a relação de tabelas juntamente com a programação JPA-Hibernate e resultado no banco de dados do H2-console.

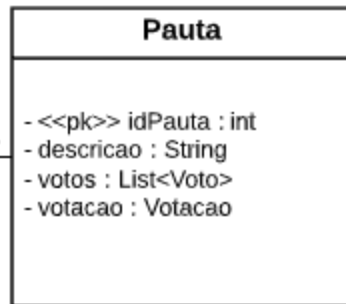
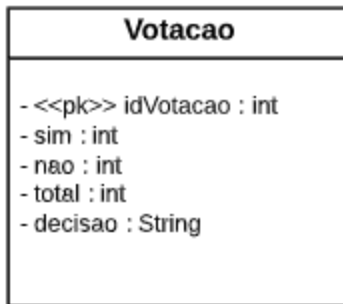
SELECT * FROM TB_VOTACAO;

DECISAO	NAO	SIM	TOTAL	PAUTA_ID_PAUTA
Pauta id=3 Aprovada!	0	3	3	3

(1 row, 1 ms)

```
@JsonIgnore
@OneToOne
@MapsId
private Pauta pauta;
```

```
@OneToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
private Votacao votacao;
```



SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO
1	Votacao para aumentar numero de Sócios
2	Reduzir taxa de juros dos Financiamentos
3	Aprovar verba para propaganda

(3 rows, 1 ms)

```
@JsonIgnore
@OneToMany(mappedBy = "voto")
private List<Voto> votos = new ArrayList<>();
```

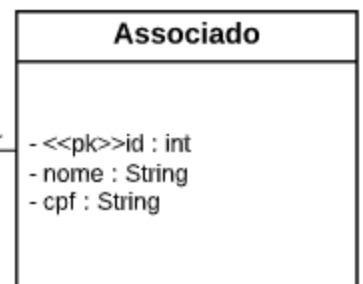
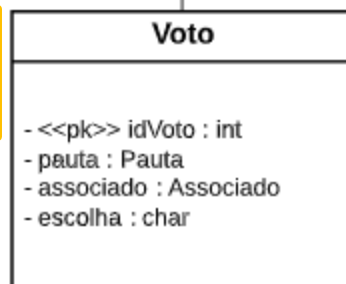
0...1

1

1

*

```
@ManyToOne
@JoinColumn(name = "pauta_id")
private Pauta pauta;
```



*

1

```
@ManyToOne
@JoinColumn(name = "associado_id")
private Associado associado;
```

```
@JsonIgnore
@OneToMany(mappedBy = "associado")
private List<Voto> votos = new ArrayList<>();
```

SELECT * FROM TB_VOTO;

ID_VOTO	ESCOLHA	ASSOCIADO_ID	PAUTA_ID
1	S	1	1
2	S	2	1
3	N	3	1
4	S	1	2
5	N	2	2
6	P	3	2

(6 rows, 3 ms)

SELECT * FROM TB_ASSOCIADO;

ID	CPF	NOME
1	66666666666	Maria Brown
2	12345678900	Alex Green
3	22345678900	Pedrinho

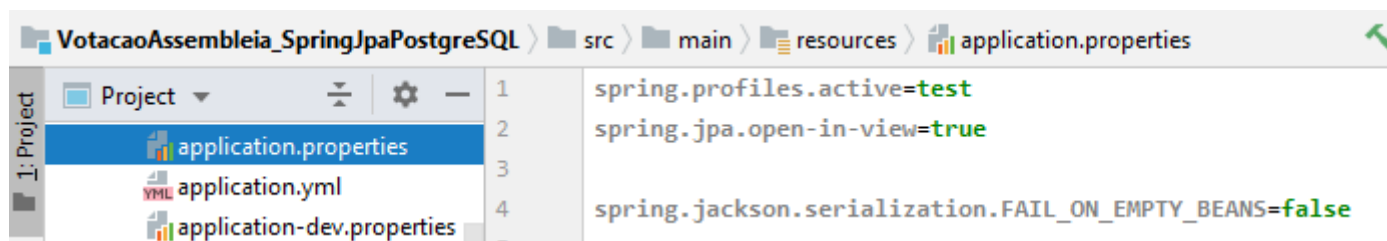
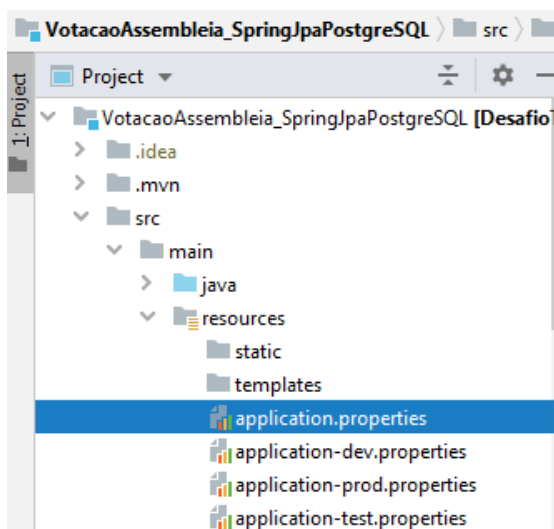
(3 rows, 2 ms)

Desenhado em Lucidchart: https://www.lucidchart.com/documents/edit/47b72a3f-6a36-4c11-bed0-0d635f7db075/0_0?beaconFlowId=48421A8D3D6A0B09#?folder_id=home&browser=icon

CONFIGURAÇÃO

O desenvolvedor tem três possibilidades de configuração de ambientes:

- application-test.properties: ambiente para testes utilizando o banco de dados H2;
- application-dev.properties: ambiente para desenvolvimento utilizando o bando de dados Postgres na máquina local;
- application-prod.properties: ambiente de produção usando bando de dados Postgres na nuvem utilizando o Heroku.



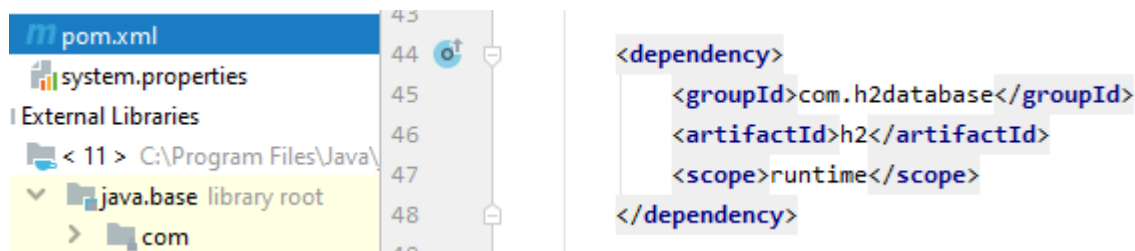
CLASSE DE CONFIGURAÇÃO

Classe auxiliar para salvar dados iniciais que carregam no inicio do programa no bando de teste.



BANCO TESTE H2

Um banco teste H2 console foi configurado para testes. A cada vez que o programa reinicia o banco zera todos os seus dados. É necessário acrescentar no arquivo pom.xml as dependências para este banco funcionar.



<http://localhost:8080/h2-console>

<http://localhost:8080/h2-console/login.do?jsessionid=2046bb4921a409a56a1848b8e008774c>

BANCO POSTGRE

O Postgre foi escolhido para armazenar os dados para o ambiente de desenvolvimento e para o de Produção.

- BANCO LOCAL E HEROKU PELO POSTGRE:



<http://127.0.0.1:65024/browser/>

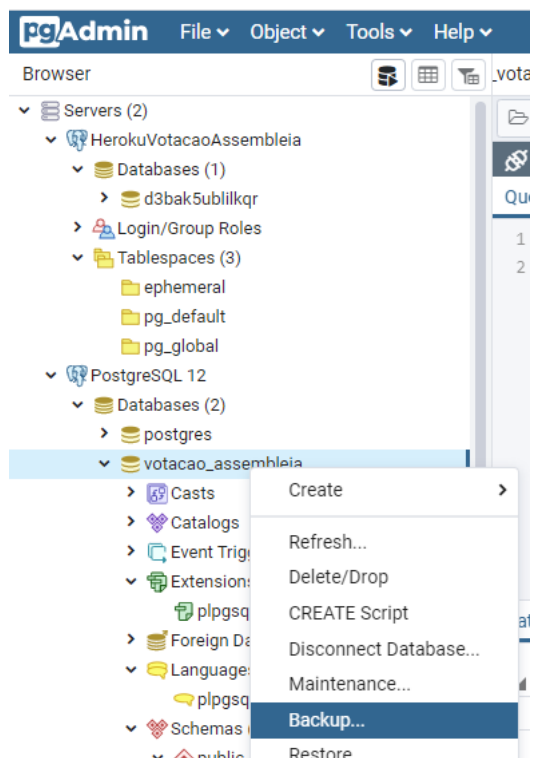
- DEPENDÊNCIAS ADICIONADAS:



- OBTENDO SCRIPT SQL A PARTIR DO POSTGRESQL LOCAL

Fonte: Udemey

→ 326. <https://www.udemy.com/course/java-curso-completo/learn/lecture/17057402#bookmarks>

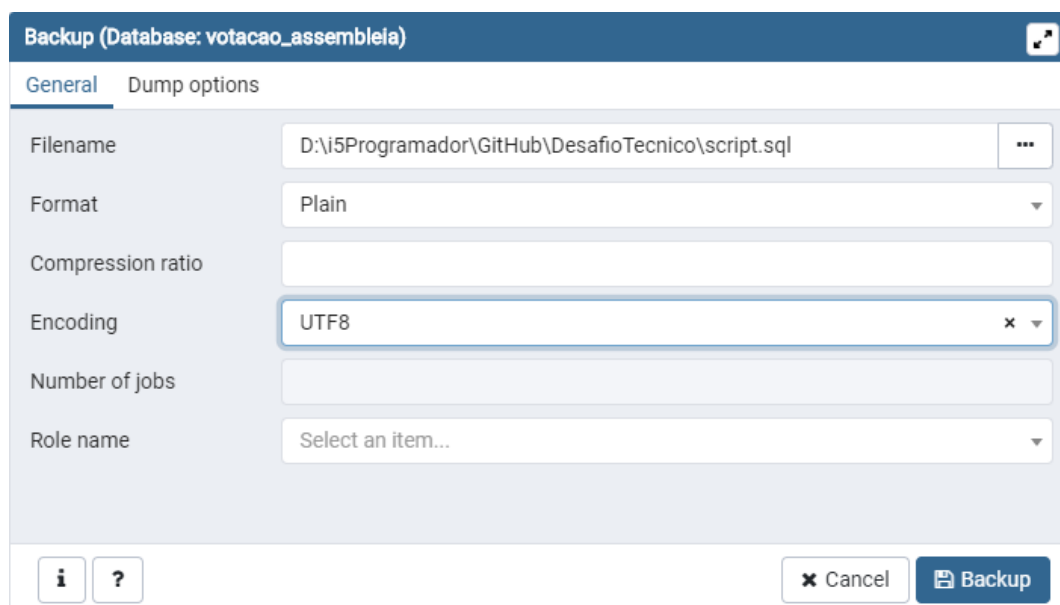


+

Tools -> Backup

- Format: Plain
- Encoding: UTF8
- Dump options:
 - Only schema: YES
 - Blobs: NO
 - Do not save: (ALL)
 - Verbose messages: NO

+



+

Backup (Database: votacao_assembleia)

General Dump options

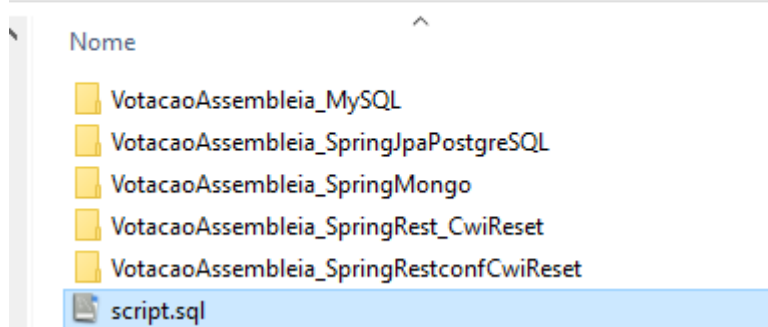
Sections	
Pre-data <input type="checkbox"/> No	Data <input type="checkbox"/> No
Post-data <input type="checkbox"/> No	
Type of objects	
Only data <input type="checkbox"/> No	Only schema <input checked="" type="checkbox"/> Yes
Blobs <input type="checkbox"/> No	
Do not save	
Owner <input checked="" type="checkbox"/> Yes	Privilege <input checked="" type="checkbox"/> Yes
Tablespace <input checked="" type="checkbox"/> Yes	Unlogged table data <input checked="" type="checkbox"/> Yes
Comments <input checked="" type="checkbox"/> Yes	
Queries	
Use Column Inserts <input type="checkbox"/> No	Use Insert Commands <input type="checkbox"/> No
Include CREATE DATABASE statement <input type="checkbox"/> No	Include DROP DATABASE statement <input type="checkbox"/> No
Load Via Partition Root <input type="checkbox"/> No	
Disable	
Trigger <input type="checkbox"/> No	\$ quoting <input type="checkbox"/> No
Disable	
Trigger <input type="checkbox"/> No	\$ quoting <input type="checkbox"/> No
Miscellaneous	
With OID(s) <input type="checkbox"/> No	Verbose messages <input type="checkbox"/> No
Force double quote on identifiers <input type="checkbox"/> No	Use SET SESSION AUTHORIZATION <input type="checkbox"/> No

+

 Backup

+

Desktopi5 (D:) > i5Programador > GitHub > DesafioTecnico >



+ Apagar as linhas do início do arquivo

➤ 327. EXECUTANDO SCRIPT SQL NO SERVIDOR REMOTO - HEROKU

HerokuVotacaoAssembleia

- Databases (1)
 - d3bak5ublilkqr
 - Cast
 - Catalog
 - Event Tr
 - Extensio
 - Foreign
 - Language
 - Schema
 - publ
 - Query Tool...

script.sql	3.4 kB	Thu May 14 20:57:05 2020
------------	--------	--------------------------

+ Ctrl A +

pgAdmin File Object Tools Help

Browser Servers (2) HerokuVotacaoAssembleia Databases (1) d3bak5ublilkqr

Query Editor Query History

```

1  --
2  -- Name: tb_associado; Type: TABLE; Schema: public; Owner: -
3  --
4
5  CREATE TABLE public.tb_associado (
6      id integer NOT NULL,
7      cpf character varying(255),
8      nome character varying(255)
9  );
10
11
12  --
13  -- Name: tb_associado_id_seq; Type: SEQUENCE; Schema: public; Owner: -
14  --
15

```

Data Output Explain Messages Notifications

ALTER TABLE

Query returned successfully in 270 msec.

Tables (6)

- associado
- pauta
- tb_associado
- tb_pauta
- tb_votacao
- tb_voto



➤ 329. DEPLOY DO SISTEMA NO HEROKU

1º) Faz este procedimento na primeira vez para sincronizar pasta local com app criado no heroku

```
heroku git:remote -a votacao-assembleia
D:\i5Programador\GitHub\DesafioTecnico\VotacaoAssembleia_SpringJpaPostgreSQL>heroku git:remote -a votacao-assembleia
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/38c19f9e-8689-437c-b10d-f6467b4b8924
Logging in... done
set git remote heroku to https://git.heroku.com/votacao-assembleia.git
```

Gerou o endereço no heroku

<https://git.heroku.com/votacao-assembleia.git>

Para verificar se ficou ok:

git remote -v

```
D:\i5Programador\GitHub\DesafioTecnico\VotacaoAssembleia_SpringJpaPostgreSQL>git remote -v
heroku https://git.heroku.com/votacao-assembleia.git (fetch)
heroku https://git.heroku.com/votacao-assembleia.git (push)
origin https://github.com/eliseusbrito/DesafioTecnicoCwiVotacaoAssembleia_SpringJpaHibernatePostgreSQL.git (fetch)
origin https://github.com/eliseusbrito/DesafioTecnicoCwiVotacaoAssembleia_SpringJpaHibernatePostgreSQL.git (push)
```

Enviar para o Heroku

- Send to Heroku:

```
git add .
git commit -m "Deploy app to Heroku"
git push heroku master
```

git add .

git commit -m "Deploy app to Heroku"

git push heroku master

```
remote: [INFO] -----
remote: [INFO] BUILD SUCCESS
remote: [INFO] -----
remote: [INFO] Total time: 20.290 s
remote: [INFO] Finished at: 2020-05-13T20:43:08Z
remote: [INFO] -----
remote: -----> Discovering process types
remote: Procfile declares types -> (none)
remote: Default types for buildpack -> web
remote: -----> Compressing...
remote: Done: 93.3M
remote: -----> Launching...
remote: Released v5
remote: https://votacao-assembleia.herokuapp.com/ deployed to Heroku
remote: Verifying deploy... done.
To https://git.heroku.com/votacao-assembleia.git
* [new branch] master -> master
```

<https://votacao-assembleia.herokuapp.com>

TRATAMENTO DE ERROS E EXCEÇÕES:

➤ PESQUISA POR ID NÃO EXISTENTE DA CLASSE ASSOCIADO:

Mensagens foram personalizadas para a aplicação nos casos onde o usuário pesquisa por um id não existente:

Erro era Status 500 e agora 404

Mensagem padrão:

```
Status: 500 Internal Server Error

1  {
2    "timestamp": "2020-05-13T19:54:53.062+0000",
3    "status": 500,
4    "error": "Internal Server Error",
5    "message": "No value present",
6    "path": "/associado/10"
7  }

//
public Associado findById(Integer id){
    Optional<Associado> obj=associadoRepository.findById(id);
    return obj.get();
    //      return obj.orElseThrow(() -> new ResourceNotFoundException(id));
}
```

Mensagem Customizada:

```
Status: 404 Not Found

1  {
2    "timestamp": "2020-05-13T19:57:00Z",
3    "status": 404,
4    "error": "Associado não encontrado.",
5    "message": "Associado id=10 não encontrado no cadastro.",
6    "path": "/associado/10"
7  }

//
public Associado findById(Integer id){
    Optional<Associado> obj=associadoRepository.findById(id);
    //      return obj.get();
    return obj.orElseThrow(() -> new ResourceNotFoundException(id));
}
```

➤ CADASTRAR ASSOCIADO SEM INFORMAÇÕES NO NOME OU CPF:

Situação Inicial: Erro e código

```
Status: 500 Internal Server Error
```

```

1 {
2     "timestamp": "2020-05-14T11:11:37.702+0000",
3     "status": 500,
4     "error": "Internal Server Error",
5     "message": "No message available",
6     "path": "/associado"
7 }

```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
private String nome;
private String cpf;

```

Classe Entities(Camada de Domínio)

```

@PostMapping
public ResponseEntity<Associado> insert(@RequestBody Associado obj){

```

Classe Rest (Camada

de Controller):

Corrigido mensagem

Status: 400 Bad Request

Código adicionado Annotation JPA @NotBlank e @Valid

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
@NotBlank
private String nome;
@NotBlank
private String cpf;

```

Código adicionado @Valid

```

@PostMapping
public ResponseEntity<Associado> insert(@RequestBody @Valid Associado obj){

```

```

1 {
2     "timestamp": "2020-05-14T11:17:47.537+0000",
3     "status": 400,
4     "error": "Bad Request",
5     "errors": [
6         {
7             "codes": [
8                 "NotBlank.associado.cpf",
9                 "NotBlank.cpf",
10                "NotBlank.java.lang.String",
11                "NotBlank"
12            ],
13            "arguments": [
14                {
15                    "codes": [
16                        "associado.cpf",
17                        "cpf"
18                    ],
19                    "arguments": null,
20                    "defaultMessage": "cpf",
21                    "code": "cpf"
22                }
23            ],
24            "defaultMessage": "não pode estar em branco",
25            "objectName": "associado",
26            "field": "cpf",
27            "rejectedValue": "",
28            "bindingFailure": false,
29            "code": "NotBlank"
30        }
31    ],
32    "message": "Validation failed for object='associado'. Error count: 1",
33    "path": "/associado"
34 }

```

```

"defaultMessage": "não pode estar em branco",
"objectName": "associado",
"field": "cpf"

```

Referencia: <https://www.udemy.com/course/fundamentos-de-programacao-com-java/learn/lecture/18619834#overview>

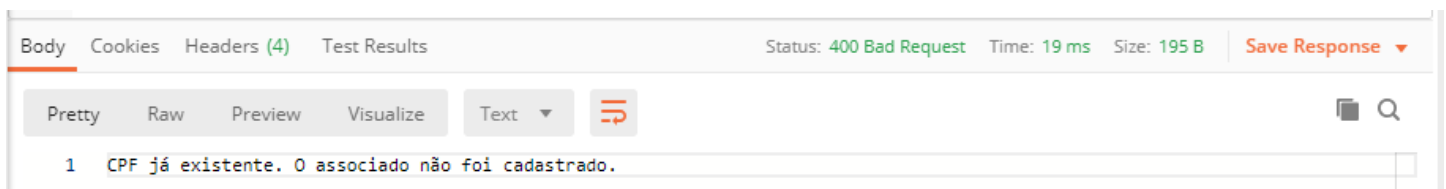
➤ ADICIONADO MSG PERSONALIZADA AO CADASTRAR ASSOCIADO JÁ EXISTENTE

Mensagem original : Retornava o usuário já cadastrado

Status: 200 OK

Mensagem Atualizada: Mensagem Status 400 Bad Request personalizada

```
@PostMapping
public ResponseEntity<> insert(@RequestBody @Valid Associado obj){
    obj = associadoGerenciador.insert(obj);
    if(cadastroExistente==1){
        cadastroExistente=0;
        return ResponseEntity.badRequest().body( t: "CPF já existente. O associado não foi cadastrado.");
    };
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")
        .buildAndExpand(obj.getId()).toUri();
    return ResponseEntity.created(uri).body(obj);
}
```

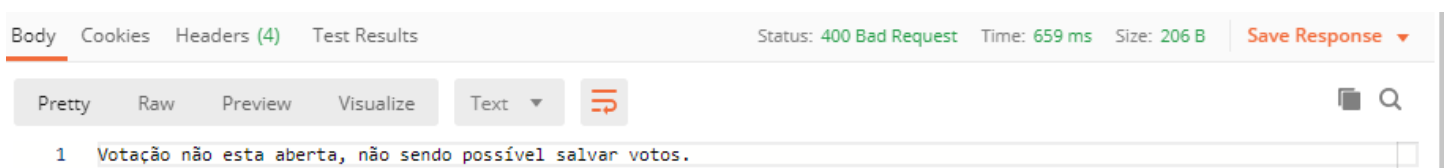


➤ ADICIONADO MENSAGEM QUANDO COLOCA LETRA ERRADA NO VOTO

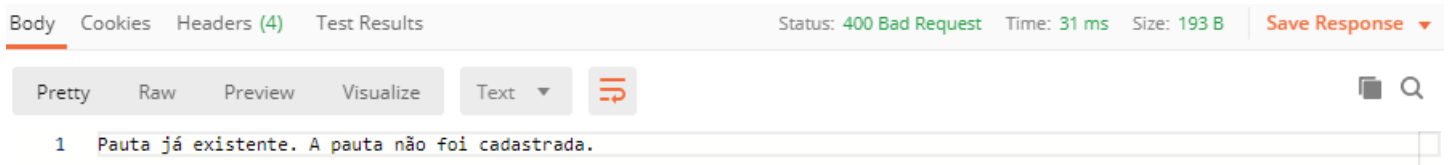
1 Você deve votar [S] para sim ou [N] para não. Este voto não foi considerado.

➤ ADICIONADO MENSAGEM QUANDO VOTAÇÃO NÃO ESTÁ ABERTA

Se uma votação não esta aberta (ainda não foi realizada ou já foi realizada) o programa não deve permitir votos. Também deve dar mensagem que não é possível votar antes de contar o tempo default ou definido pelo usuário para votação.

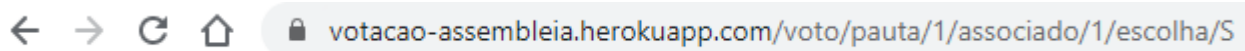


➤ MENSAGEM QUANTO CADASTRA PAUTA JÁ EXISTENTE



➤ HEROKU

Exceções vinda do Heroku ainda não foram tratadas nesta solução.



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sun May 17 14:53:00 UTC 2020

There was an unexpected error (type=Method Not Allowed, status=405).

Request method 'GET' not supported

INSTALAÇÕES

RABBITMQ

Referência:

<https://dev.senior.com.br/documentacao/instalacao-simplificada/2-instalando-o-rabbitmq/>

<https://www.youtube.com/watch?v=3sEPqKrFQf8>

<https://www.erlang.org/downloads/23.0>

<https://www.rabbitmq.com/install-windows.html>

APRENDIZADOS

TABELA DE DADOS

Testes para verifica comportamento dos comandos JPA para definição de relacionamento entre as Classes Pauta e Votação. Para cada Estudo tem uma configuração e um resultado de comportamento das tabelas de dados.

Classe Pauta:

Classe Votação

Estudo 1:

```
// @ManyToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
// @ManyToOne(mappedBy = "pauta" )
private Votacao votacao;
```

SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO	TESTE	VOTACAO
1	Votacao para aumentar numero de Sócios	0	null
2	Reduzir taxa de juros dos Financiamentos	0	null
3	Aprovar verba para propaganda	20	null

(3 rows, 1 ms)

```
// @ManyToOne
// @MapsId
private Pauta pauta;
```

SELECT * FROM TB_VOTACAO;

ID_VOTACAO	DECISAO	NAO	PAUTA	SIM	TOTAL
------------	---------	-----	-------	-----	-------

(no rows, 1 ms)

Estudo 2:

```
// @ManyToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
// @ManyToOne(mappedBy = "pauta" )
private Votacao votacao;
```

SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO	TESTE	VOTACAO
1	Votacao para aumentar numero de Sócios	0	null
2	Reduzir taxa de juros dos Financiamentos	0	null
3	Aprovar verba para propaganda	20	null

(3 rows, 1 ms)

```
@ManyToOne
@MapsId
private Pauta pauta;
```

SELECT * FROM TB_VOTACAO;

DECISAO	NAO	SIM	TOTAL	PAUTA_ID_PAUTA
---------	-----	-----	-------	----------------

(no rows, 3 ms)

Estudo 3:

```
// @ManyToOne(mappedBy = "pauta" )
@ManyToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
private Votacao votacao;
```

```
// @ManyToOne
// @MapsId
private Pauta pauta;
```

: Application run failed

Estudo 4:

```
// @ManyToOne(mappedBy = "pauta" )
@ManyToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
private Votacao votacao;
```

```
@ManyToOne
@MapsId
private Pauta pauta;
```

SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO	TESTE
1	Votacao para aumentar numero de Sócios	0
2	Reduzir taxa de juros dos Financiamentos	0
3	Aprovar verba para propaganda	20

(3 rows, 3 ms)

SELECT * FROM TB_VOTACAO;

DECISAO	NAO	SIM	TOTAL	PAUTA_ID_PAUTA
---------	-----	-----	-------	----------------

(no rows, 1 ms)

Estudo 5:

```
// @ManyToOne(mappedBy = "pauta", cascade = CascadeType.ALL ) @ManyToOne
@ManyToOne(mappedBy = "pauta" )
private Votacao votacao;
```

SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO	TESTE
1	Votacao para aumentar numero de Sócios	0
2	Reduzir taxa de juros dos Financiamentos	0
3	Aprovar verba para propaganda	20

(3 rows, 3 ms)

```
@ManyToOne
@MapsId
private Pauta pauta;
```

SELECT * FROM TB_VOTACAO;

DECISAO	NAO	SIM	TOTAL	PAUTA_ID_PAUTA
---------	-----	-----	-------	----------------

(no rows, 2 ms)

Estudo 6:

```
// @OneToOne(mappedBy = "pauta", cascade = CascadeType.ALL) // @OneToOne
// @OneToOne(mappedBy = "pauta" ) // @MapsId
private Votacao votacao;
private Pauta pauta;

: Application run failed
```

Estudo 7:

```
// @OneToOne(mappedBy = "pauta", cascade = CascadeType.ALL )
// @OneToOne(mappedBy = "pauta" )
private Votacao votacao;
```

SELECT * FROM TB_PAUTA;

ID_PAUTA	DESCRICAO	TESTE	VOTACAO
1	Votacao para aumentar numero de Sócios	0	null
2	Reduzir taxa de juros dos Financiamentos	0	null
3	Aprovar verba para propaganda	20	null

(3 rows, 3 ms)

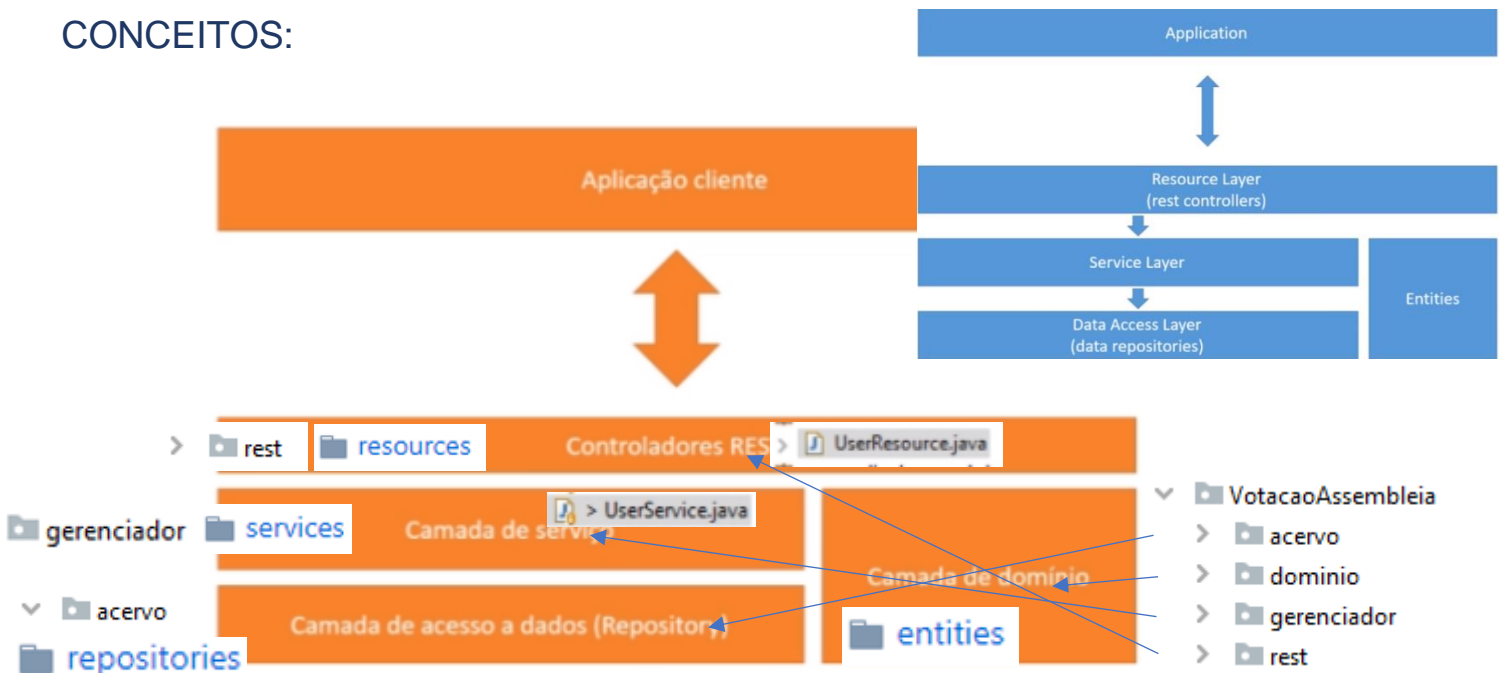
```
// @OneToOne
// @MapsId
private Pauta pauta;
```

SELECT * FROM TB_VOTACAO;

ID_VOTACAO	DECISAO	NAO	PAUTA	SIM	TOTAL
------------	---------	-----	-------	-----	-------

(no rows, 1 ms)

CONCEITOS:

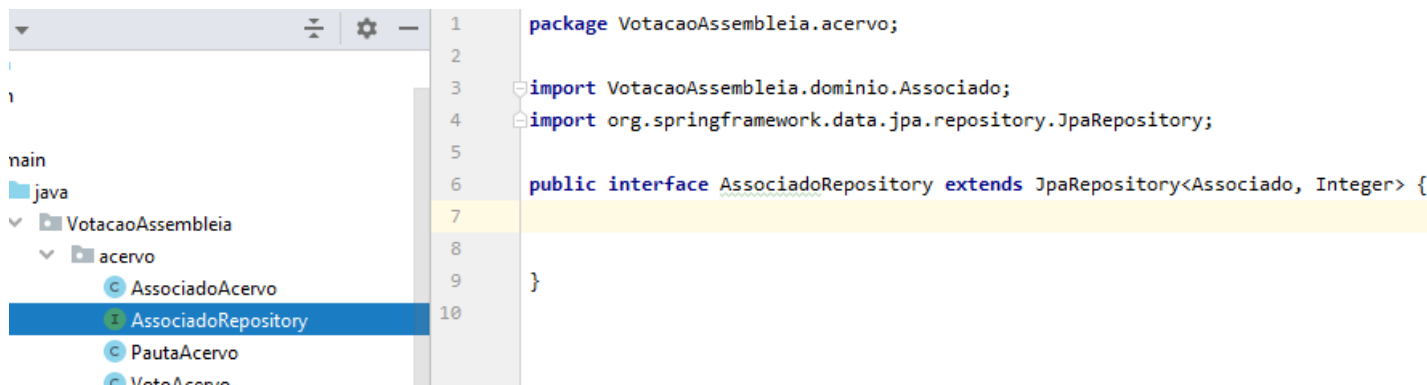


1) Classe domínio

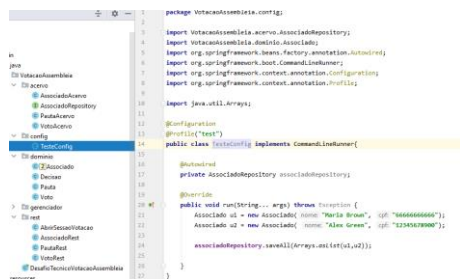
```
8 @Entity
9 public class Associado {
10
11 @Id
12 @GeneratedValue(strategy = GenerationType.IDENTITY)
13 private int id;
14 private String nome;
15 private String cpf;
16
```


2)Classe Repository(começa por baixo)

Criado uma interface



+ Teste Config



PENDÊNCIAS E MELHORIAS:

Criar explicit mapping for /error para exceções vinda do Heroku;

Talvez seja possível tirar equals e hashCode em alguns dos códigos;

Verificar necessidade de Colocar @NotBlank e @Valid nas outras classes;

Como usar o navegador para dar Comandos POST, PUT e DELETE;

Tornar automático o envio da mensagem para o RabbitMQ;

Alterar programação para não utilizar variáveis de controle para mensagens REST de erros.

CONCLUSÕES

Este desafio foi de extrema importância para colocar em prática os conhecimentos adquiridos no CWI Reset e pesquisar sobre novos conceitos e conhecimentos. Ter a capacidade de pesquisar e adequar ao seu projeto tecnologias ainda não conhecidas, acredito que pode diferenciar um futuro profissional de TI.

Neste desafio foi necessário pesquisar sobre Banco de Dados, converter dados entre bancos relacionais e linguagem orientada a objeto através de JPA(Java Persistence API) e Hibernate. Também foi necessário aprender sobre como colocar os dados e o aplicativo na nuvem. A solução encontrada para esta aplicação foi utilizar o Heroku e o PostgreSQL.

Já em relação a mensageria a melhor alternativa encontrada nas pesquisas foi utilizar o RabbitMQ, em função do material disponível e por ser um dos softwares mais utilizados para esta função.

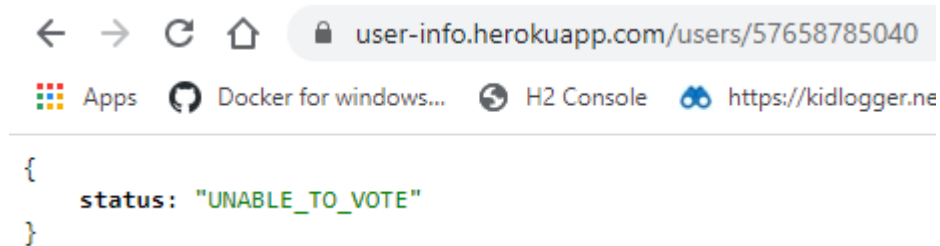
ANEXOS

VARIAÇÃO NO API GET [HTTPS://USER-INFO.HEROKUAPP.COM/USERS/{CPF}](https://user-info.herokuapp.com/users/{CPF}):

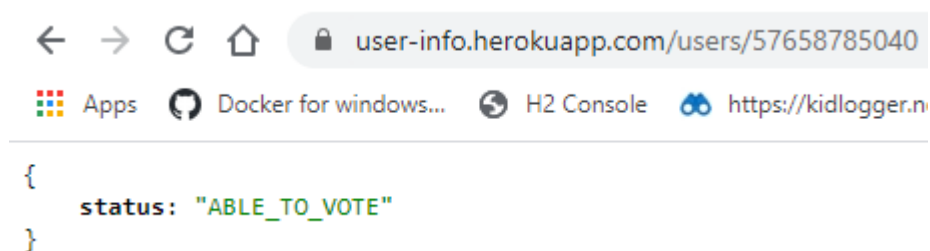
Conforme o momento da pesquisa um mesmo CPF pode ter seu Status alterado:

<https://user-info.herokuapp.com/users/57658785040>

CPF **57658785040** Inválido

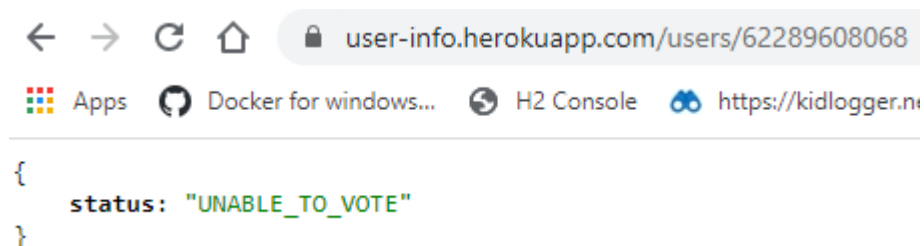


CPF **57658785040** Válido



<https://user-info.herokuapp.com/users/62289608068>

CPF **62289608068** Inválido



CPF **62289608068** Válido

