Documentação Teste Sicredi

Ordem para testar endpoints:

1. Cadastrar Associado;
2. Cadastrar Pauta;
3. Abrir Sessão para a Pauta cadastrada;
4. Realizar o Voto
5. Quando a Sessão e/ou a votação estiver finalizada, poderá realizar a contagem de votos e mostrar o vencedor.

Endpoints para teste

Pauta

Cadastrar Pauta:

- POST: http://sicredi-production.up.railway.app/sicredi/pauta
    - Body:

```
{
    "titulo": "Titulo da Pauta",
    "limiteVotos": 10,
    "dataCriacao": "2023-04-08T13:00:00"
}
```

Voto

Contagem de votos:

- GET: http://sicredi-production.up.railway.app/sicredi/voto/1

Votar na Pauta:

- POST: http://sicredi-production.up.railway.app/sicredi/voto
    - Body:

```
{
    "idAssociado": 1,
    "valorVoto": "NAO",
    "idPauta": 1
}
```

Associado

Cadastrar Associado:

- POST: http://sicredi-production.up.railway.app/sicredi/associado
    - Body:

```
{
    "cpf": "12345678915"
}
```

Valida CPF:

- GET: http://sicredi-production.up.railway.app/sicredi/associado/12345678910

Sessão

Abrir Sessão:

- POST: http://sicredi-production.up.railway.app/sicredi/sessao
    - Body:

```
{
    "idPauta": 1,
    "dataLimite": "2023-04-08T20:00:00"
}
```

Verificador de CPF

Valida CPF:

- GET: http://valida-cpf-sicredi-production.up.railway.app/cpf/12345678910

Classes de domínio

Associado

Classe referente aos associados, contendo informações de ID e cpf, sendo utilizada para salvar os mesmos na tabela descrita na linha 19.

```java
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "db_associado")
@SequenceGenerator(name = "seq_db_associado", sequenceName = "seq_db_associado", allocationSize = 1)
public class Associado {

    @Id
    @GeneratedValue(generator = "seq_db_associado")
    @Column(name = "id", nullable = false)
    private Integer id;

    private String cpf;
}
```

Pauta

Esta é uma classe referente às pautas, contendo informações como ID, título, limite de votos para controle da finalização das sessões relacionadas a ela e data de criação. Ela é salva na tabela descrita na linha 21.

```java
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "db_pauta")
@SequenceGenerator(name = "seq_db_pauta", sequenceName = "seq_db_pauta", allocationSize = 1)
public class Pauta {

    @Id
    @GeneratedValue(generator = "seq_db_pauta")
    @Column(name = "id", nullable = false)
    private Integer id;

    private String titulo;

    private Integer limiteVotos;

    private LocalDateTime dataCriacao;

}
```

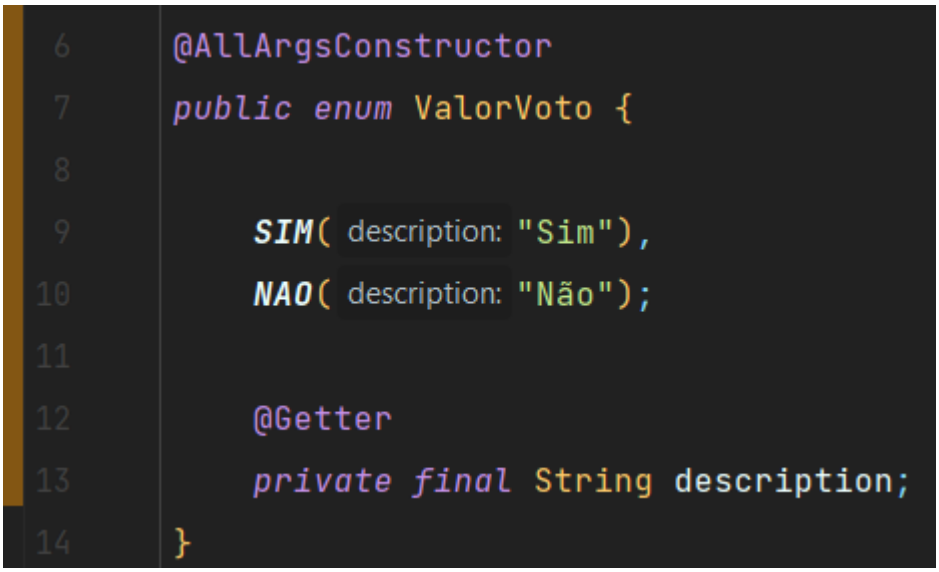
Sessão

Esta é a classe que representa as sessões, contendo informações como o ID, o ID da pauta que está sendo votada e a data limite para encerramento da sessão. Esses dados são salvos na tabela descrita na linha 21.

```java
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "db_sessao")
@SequenceGenerator(name = "seq_db_sessao", sequenceName = "seq_db_sessao", allocationSize = 1)
public class Sessao {

    @Id
    @GeneratedValue(generator = "seq_db_sessao")
    @Column(name = "id", nullable = false)
    private Integer id;

    private Integer idPauta;

    private LocalDateTime dataLimite;
}
```

Valor Voto

Enum para fixação e validação dos valores de votos a serem disponibilzados e/ou recebidos.



Voto

Esta é a classe responsável pelos votos a serem computados. Ela contém informações como o ID do voto, o ID do associado que realizou o voto, o ID da

pauta que está sendo votada e o valor do voto (Sim ou Não). Esses dados são salvos na tabela descrita na linha 19.

```java
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "db_voto")
@SequenceGenerator(name = "seq_db_voto", sequenceName = "seq_db_voto", allocationSize = 1)
public class Voto {

    @Id
    @GeneratedValue(generator = "seq_db_voto")
    @Column(name = "id", nullable = false)
    private Integer id;

    private Integer idAssociado;

    private Integer idPauta;

    private String valorVoto;
}
```

Repository

AssociadoRepository

Interface para utilização de métodos de interação com a tabela `"db_associado"` no banco de dados. Nenhum método declarado pois foram utilizados apenas os da interface que é extendida.

```java
@Repository
public interface AssociadoRepository extends CrudRepository<Associado, Integer> {
}
```

PautaRepository

Interface para utilização de métodos de interação com a tabela `"db_pauta"` no banco de dados. Nenhum método declarado pois foram utilizados apenas os da interface que é extendida.

```
7   @Repository
8   public interface PautaRepository extends CrudRepository<Pauta, Integer> {
9   }
10
```

SessaoRepository

Interface para utilização de métodos de interação com a tabela `"db_sessao"` no banco de dados.

Métodos:

- Linha 10 (`findByIdPauta`): visa buscar uma ocorrência na tabela citada anteriormente, que contenha, na coluna idPauta, o valor informado no parâmetro.
- Linha 12 (`existsByIdPauta`): retorna `true` caso exista uma ocorrência na tabela que contenha, na coluna idPauta, o parâmetro informado. Caso contrário, retorna `false`.

```
7   @Repository
8   public interface SessaoRepository extends CrudRepository<Sessao, Integer> {
9
10      Sessao findByIdPauta(Integer idPauta);
11
12      boolean existsByIdPauta(Integer idPauta);
13  }
```

VotoRepository

Interface para utilização de métodos de interação com a tabela `"db_voto"` no banco de dados.

Métodos:

- Linha 12 (`existsByIdAssociadoAndIdPauta`): retorna `true` caso exista uma ocorrência na tabela que contenha, na coluna idAssociado e idPauta, os parâmetros informados. Caso contrário, retorna `false`
- Linha 14 (`countByIdPauta`): retorna a quantidade de ocorrências na tabela que contém, na coluna idPauta, o parâmetro informado.
- Linha 16 (`findAllByIdPauta`): retorna uma lista com todas as ocorrências na tabela que contém, na coluna idPauta, o parâmetro informado.

```
9   @Repository
10  public interface VotoRepository extends CrudRepository<Voto, Integer> {
11
12      boolean existsByIdAssociadoAndIdPauta(Integer idAssociado, Integer idPauta);
13
14      Integer countByIdPauta(Integer idPauta);
15
16      List<Voto> findAllByIdPauta(Integer idPauta);
17
18  }
```

CpfValidatorRepository

Interface para integração com serviço externo, o qual realiza a validação do cpf informado.

Métodos:

- Linha 11 (`validaCpf`): método que retorna um booleano de acordo com o retorno da chamada do serviço externo, passando o CPF como variável no path da URL.

```
7   @FeignClient(value = "cpfValidator", url = "https://valida-cpf-sicredi-production.up.railway.app")
8   public interface CpfValidatorRepository {
9
10      @GetMapping(value = "/cpf/{cpf}")
11      boolean validaCpf(@PathVariable String cpf);
12  }
```

Mapper

AssociadoMapper

Métodos:

- Linha 10 (`toDomain`): método utilizado para criar a classe domain de Associado, a partir da Request utilizada no cadastro de um novo Associado.

```
7     @Component
8     public class AssociadoMapper {
9
10 @      public Associado toDomain(CadastrarAssociadoRequest request) {
11            return Associado.builder()
12                    .cpf(request.getCpf())
13                    .build();
14        }
15    }
```

AssociadoMapper

Métodos:

- Linha 10 (`toDomain`): método utilizado para criar a classe domain de Pauta, a partir da Request utilizada no cadastro de uma nova Pauta.

```
7     @Component
8     public class PautaMapper {
9
10 @      public Pauta toDomain(final CadastrarPautaRequest request) {
11            return Pauta.builder()
12                    .titulo(request.getTitulo())
13                    .limiteVotos(request.getLimiteVotos())
14                    .dataCriacao(request.getDataCriacao())
15                    .build();
16        }
17    }
```

SessaoMapper

Métodos:

- Linha 12 (`toDomain`): método utilizado para criar a classe domain de Sessao, a partir da Request utilizada na criação de uma nova Sessao, bem como a data limite que é verificada na classe SessaoService.

VotoMapper

Métodos:

- Linha 13 (`toDomain`): método utilizado para criar a classe domain de Voto, a partir da Request utilizada na criação de um novo Voto.
- Linha 21 (`toVencedorResponse`): método para montar a Response indicando o vencedor (ou empate) da votação.
- Linha 27 (`montaFraseVencedor`): método para estruturar a frase que irá na Response, utilizando o título da pauta e o resultado da votação.

```java
@Component
public class VotoMapper {

    public Voto toDomain(final VotoRequest request) {
        return Voto.builder()
                .idAssociado(request.getIdAssociado())
                .valorVoto(ValorVoto.valueOf(request.getValorVoto()).getDescription())
                .idPauta(request.getIdPauta())
                .build();
    }

    public VencedorResponse toVencedorResponse(final Pauta pauta, String resultado) {
        return VencedorResponse.builder()
                .fraseVencedor(montaFraseVencedor(pauta.getTitulo(), resultado))
                .build();
    }

    private String montaFraseVencedor(String titulo, String resultado) {
        return "A pauta " + titulo + " teve o resultado: " + resultado;
    }
}
```

StatusMapper

Métodos:

- Linha 9 (`toResponse`): método utilizado para criar a classe de response do Status, a partir da mensagem passada como parâmetro.

```java
@Component
public class StatusMapper {

    public StatusResponse toResponse(String mensagem) {
        return StatusResponse.builder()
                .status(mensagem)
                .build();
    }
}
```

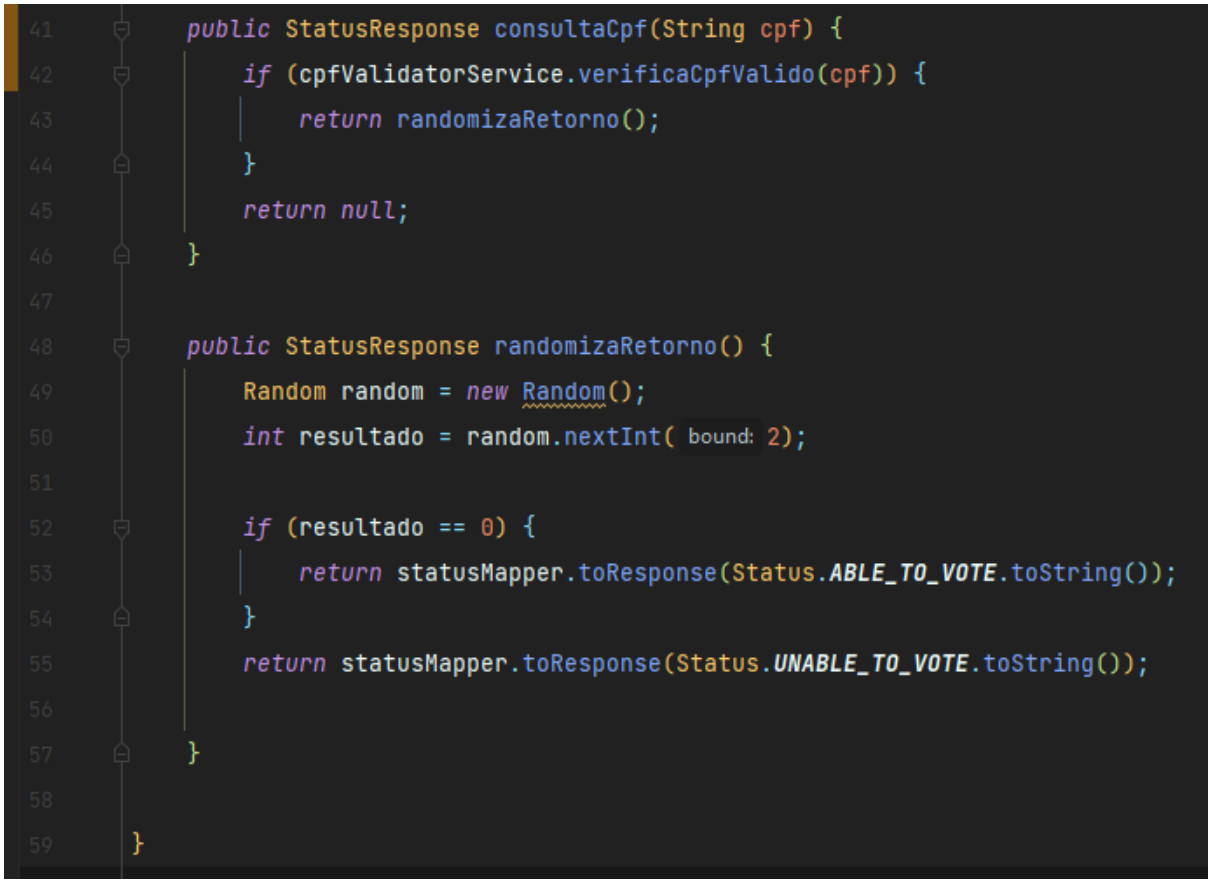
Service

AssociadoService

Métodos:

- Linha 23 (`cadastrarAssociado`): método para salvar o novo Associado cadastrado.
- Linha 43 (`consultaCpf`): método que utiliza o método `consultaCpf` da classe de serviço `cpfValidatorService`, que retorna um valor boolean determinando se o cpf é válido ou não, se for `true` um valor randômico será gerado e retornado, se for `false` será retornado `null`.
- Linha 48 (`randomizaRetorno`): método que gera um valor númerico randômico entre 0 e 1, posteriormente utilizando esse valor para decidir qual Status será retornado, caso seja 0 o Status será “ABLE_TO_VOTE”, caso seja 1 o Status será “UNABLE_TO_VOTE”.

```
13  @Slf4j
14  @Service
15  public class AssociadoService {
16
17      @Autowired
18      private AssociadoRepository repository;
19
20      @Autowired
21      private AssociadoMapper associadoMapper;
22
23      public void cadastrarAssociado(CadastrarAssociadoRequest request) {
24
25          Associado associado = associadoMapper.toDomain(request);
26
27          log.info("Salvando associado criado.");
28
29          repository.save(associado);
30      }
```

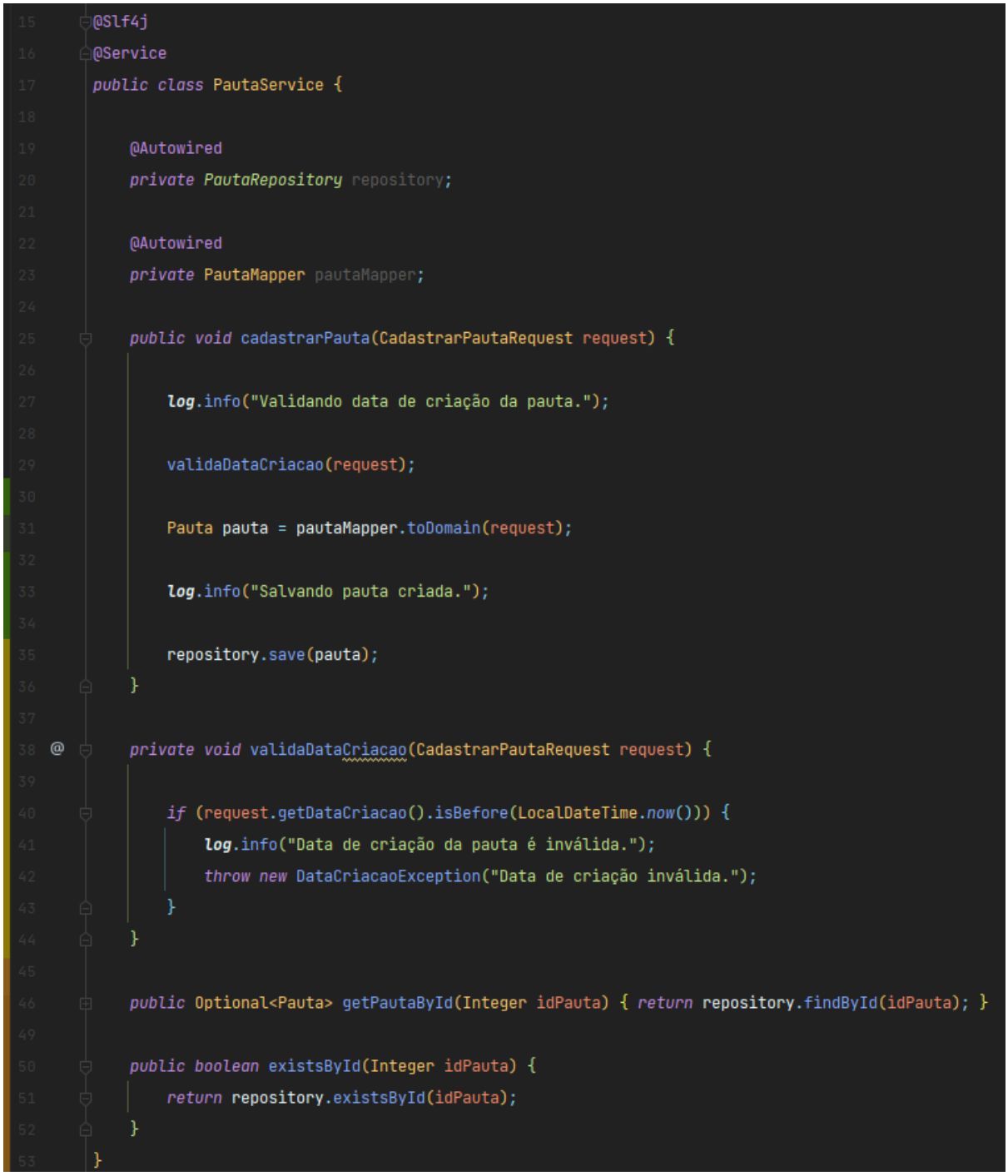


```java
    public StatusResponse consultaCpf(String cpf) {
        if (cpfValidatorService.verificaCpfValido(cpf)) {
            return randomizaRetorno();
        }
        return null;
    }

    public StatusResponse randomizaRetorno() {
        Random random = new Random();
        int resultado = random.nextInt( bound: 2);

        if (resultado == 0) {
            return statusMapper.toResponse(Status.ABLE_TO_VOTE.toString());
        }
        return statusMapper.toResponse(Status.UNABLE_TO_VOTE.toString());

    }

}
```

PautaService

Métodos:

- Linha 25 (`cadastrarPauta`): método para salvar a nova Pauta criada, utilizando um método para validar a data de criação da mesma.
- Linha 38 (`validaDataCriacao`): método para verificar se a data de criação da Pauta é válida, no caso verificando se a mesma não é anterior ao `LocalDateTime.now()`.
- Linha 46 (`getPautaById`): método que verifica a existência de uma Pauta que possua o id informado como parâmetro.
- Linha 50 (`existsById`): método para verificar se existe uma Pauta com o id passado como parâmetro.



```java
@Slf4j
@Service
public class PautaService {

    @Autowired
    private PautaRepository repository;

    @Autowired
    private PautaMapper pautaMapper;

    public void cadastrarPauta(CadastrarPautaRequest request) {

        log.info("Validando data de criação da pauta.");

        validaDataCriacao(request);

        Pauta pauta = pautaMapper.toDomain(request);

        log.info("Salvando pauta criada.");

        repository.save(pauta);
    }

    private void validaDataCriacao(CadastrarPautaRequest request) {

        if (request.getDataCriacao().isBefore(LocalDateTime.now())) {
            log.info("Data de criação da pauta é inválida.");
            throw new DataCriacaoException("Data de criação inválida.");
        }
    }

    public Optional<Pauta> getPautaById(Integer idPauta) { return repository.findById(idPauta); }

    public boolean existsById(Integer idPauta) {
        return repository.existsById(idPauta);
    }
}
```

SessaoService

Métodos:

- Linha 31 (`criarSessao`): método para salvar a nova Sessao criada, utilizando os seguintes métodos para validar as informações: `verificaSessaoExiste`, `verificaDataLimite` e `verificaSeAPautaExiste`.

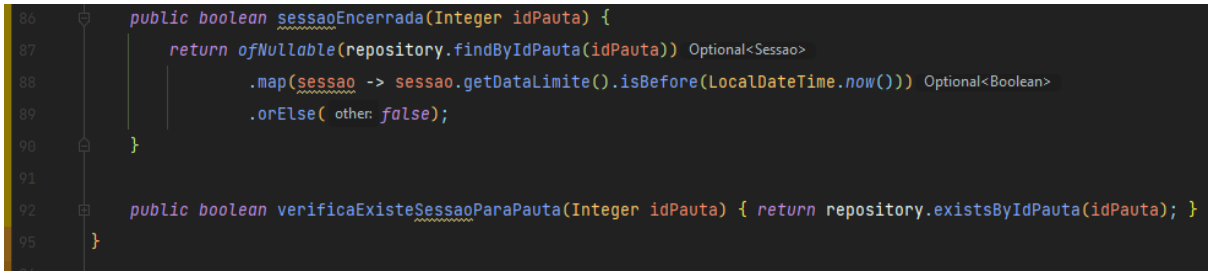
```java
@Slf4j
@Service
public class SessaoService {

    @Autowired
    private SessaoRepository repository;

    @Autowired
    private PautaService pautaService;

    @Autowired
    private SessaoMapper sessaoMapper;

    public void criarSessao(AbrirSessaoRequest request) {

        log.info("Verificando se já existe sessão para a pauta requisitada.");

        verificaSessaoExiste(request);

        log.info("Verificando se a data limite é válida.");

        LocalDateTime dataLimite = verificaDataLimite(request.getDataLimite());

        log.info("Verificando se a pauta existe.");

        verificaSeAPautaExiste(request);

        Sessao sessao = sessaoMapper.toDomain(request, dataLimite);

        log.info("Salvando sessão criada.");

        repository.save(sessao);
    }
```

- Linha 52 (`verificaSessaoExiste`): método que verifica se já não existe uma sessão para aquela pauta.
- Linha 60 (`verificaDataLimite`): método para verificar se foi informada uma data limite na requisição. Se a data for nula, o método definirá 1 minuto como padrão. Se a data for anterior à data atual, o método `validaDataLimite` lançará uma exceção.

- Linha 69 (`validaDataLimite`): método que realiza a validação se a a data não é anterior a data atual. Caso positivo, ele lança uma exceção.
- Linha 79 (`verificaSeAPautaExiste`): método para verifica se existe uma pauta com o idPauta informado na Request, caso negativo uma exceção será lançada.

```
52 @    private void verificaSessaoExiste(AbrirSessaoRequest request) {
53
54          if (repository.existsByIdPauta(request.getIdPauta())) {
55              log.info("Já existe sessão para a pauta requisitada.");
56              throw new SessaoException("A sessão dessa pauta já existe.");
57          }
58      }
59
60      private LocalDateTime verificaDataLimite(LocalDateTime dataLimite) {
61
62          log.info("Verificando se a data limite é nula (para criação de uma default) ou se ela é inválida.");
63
64          return ofNullable(dataLimite)
65                  .map(this::validaDataLimite)
66                  .orElse(LocalDateTime.now().plusMinutes(1));
67      }
68
69 @    private LocalDateTime validaDataLimite(LocalDateTime request) {
70
71          if (request.isBefore(LocalDateTime.now())) {
72              log.info("Data limite é inválida.");
73              throw new DataLimiteException("Data limite não pode ser antes da data atual.");
74          }
75
76          return request;
77      }
78
79 @    private void verificaSeAPautaExiste(AbrirSessaoRequest request) {
80
81          if (!pautaService.existsById(request.getIdPauta())) {
82              throw new PautaNaoExisteException("Pauta não existe.");
83          }
84      }
```

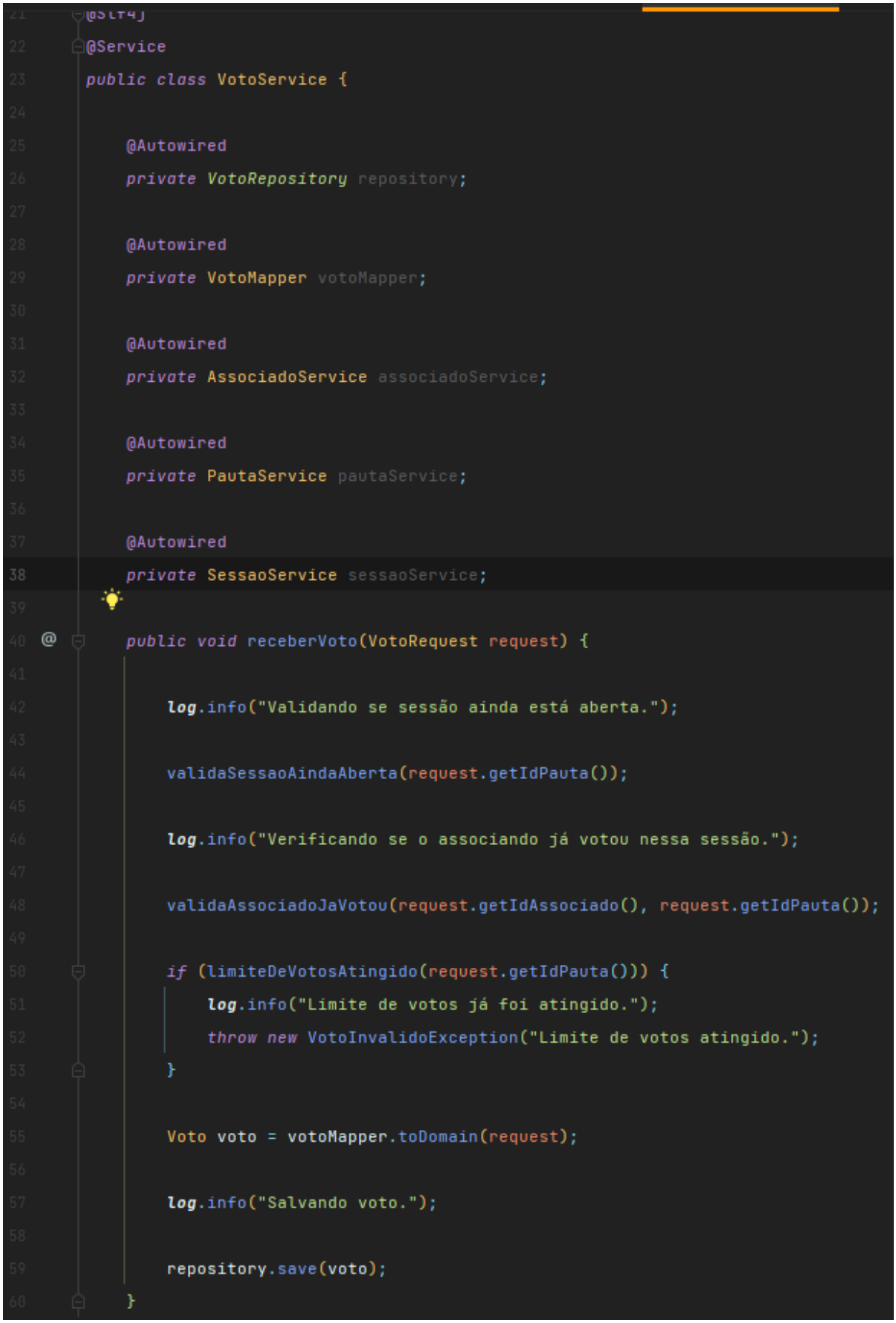
- Linha 86 (`sessaoEncerrada`): método para verificar se a Sessao referente a Pauta a qual o id é passado como parâmetro, está encerrada ou não.
- Linha 92 (`verificaExisteSessaoParaPauta`): método para verificar se existe uma Sessao para o idPauta informado no parâmetro.



VotoService

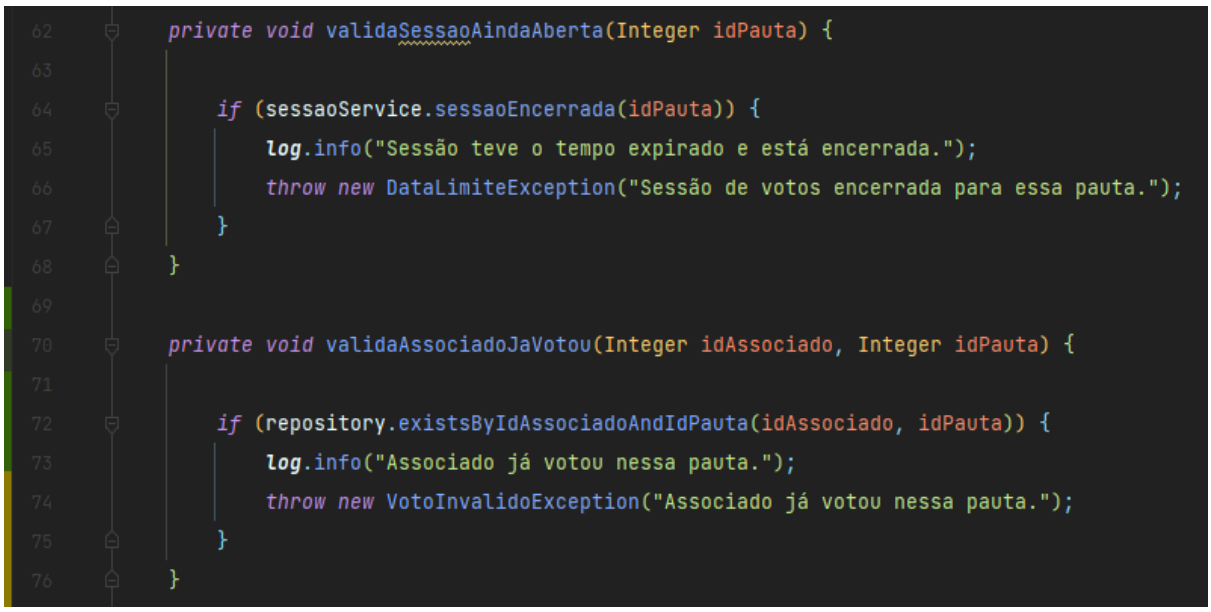
Métodos:

- Linha 40 (`receberVoto`): método para salvar o Voto criado, utilizando os seguintes métodos para validar as informações: `validaSessaoAindaAberta`, `validaAssociadoJaVotou` e `limiteDeVotosAtingido` para verificar se já foi atingido o limite de votos permitidos naquela Pauta.



```java
@Slf4j
@Service
public class VotoService {

    @Autowired
    private VotoRepository repository;

    @Autowired
    private VotoMapper votoMapper;

    @Autowired
    private AssociadoService associadoService;

    @Autowired
    private PautaService pautaService;

    @Autowired
    private SessaoService sessaoService;

    public void receberVoto(VotoRequest request) {

        log.info("Validando se sessão ainda está aberta.");

        validaSessaoAindaAberta(request.getIdPauta());

        log.info("Verificando se o associando já votou nessa sessão.");

        validaAssociadoJaVotou(request.getIdAssociado(), request.getIdPauta());

        if (limiteDeVotosAtingido(request.getIdPauta())) {
            log.info("Limite de votos já foi atingido.");
            throw new VotoInvalidoException("Limite de votos atingido.");
        }

        Voto voto = votoMapper.toDomain(request);

        log.info("Salvando voto.");

        repository.save(voto);
    }
```

- Linha 62 (`validaSessaoAindaAberta`): método que valida se o tempo da sessão já expirou. Caso positivo, uma exceção é lançada.
- Linha 70 (`validaAssociadoJaVotou`): método para validar se o associado já votou na sessão referente àquela pauta. Caso positivo, uma exceção é lançada.



```
62    private void validaSessaoAindaAberta(Integer idPauta) {
63
64        if (sessaoService.sessaoEncerrada(idPauta)) {
65            log.info("Sessão teve o tempo expirado e está encerrada.");
66            throw new DataLimiteException("Sessão de votos encerrada para essa pauta.");
67        }
68    }
69
70    private void validaAssociadoJaVotou(Integer idAssociado, Integer idPauta) {
71
72        if (repository.existsByIdAssociadoAndIdPauta(idAssociado, idPauta)) {
73            log.info("Associado já votou nessa pauta.");
74            throw new VotoInvalidoException("Associado já votou nessa pauta.");
75        }
76    }
```

- Linha 78 (`contagemVotosVencedor`): método que retorna a classe informando o vencedor da respectiva sessão. Antes disso, o método `verificaExisteSessaoParaPauta` é utilizado para verificar a existência de uma sessão para a pauta em questão, bem como se o limite de votos foi atingido (`limiteDeVotosAtingido`) ou se aquela sessão teve seu tempo esgotado (`sessaoEncerrada`). Caso alguma dessas verificações seja positiva, é realizada a contagem de votos (`contagemVotos`) e a verificação de quem foi o vencedor ou se houve empate (`verificaVencedor`). Caso contrário, é lançada uma exceção.
- Linha 105 (`contagemVotos`): método para realizar a contagem de votos que possuem o respectivo valor informado no parâmetro valorVoto.

```java
public VencedorResponse contagemVotosVencedor(Integer idPauta) {

    if (!sessaoService.verificaExisteSessaoParaPauta(idPauta)) {
        log.info("Sessão de votos para essa pauta não existe.");
        throw new SessaoException("Sessão de votos para essa pauta não existe.");
    }

    if (limiteDeVotosAtingido(idPauta) || sessaoService.sessaoEncerrada(idPauta)) {
        List<Voto> todosVotosDaPauta = repository.findAllByIdPauta(idPauta);
        Pauta pauta = pautaService.getPautaById(idPauta).orElseThrow();

        log.info("Contabilizando votos 'Sim'.");

        int contagemSim = contagemVotos(todosVotosDaPauta, valorVoto: "Sim");

        log.info("Contabilizando votos 'Não'.");

        int contagemNao = contagemVotos(todosVotosDaPauta, valorVoto: "Não");

        log.info("Verificando vencedor.");

        return verificaVencedor(pauta, contagemSim, contagemNao);
    }
    log.info("Sessão ainda em aberto, não atingiu limite de data e/ou votos.");
    throw new VotoInvalidoException("Sessão ainda em aberto, não atingiu limite de data e/ou votos.");
}

private int contagemVotos(List<Voto> votosTotais, String valorVoto) {
    return votosTotais
            .stream() Stream<Voto>
            .filter(voto -> voto.getValorVoto().equals(valorVoto))
            .toList() List<Voto>
            .size();
}
```

- Linha 113 (`verificaVencedor`): método para verificar qual valor de voto obteve maior contagem, caso sejam contagens iguais, o método passa o valor “Empate” para o mapper.
- Linha 124 (`limiteDeVotosAtingido`): método que verifica se aquela pauta ja atingiu o seu limite de votos totais.

```
    private VencedorResponse verificaVencedor(Pauta pauta, int contagemSim, int contagemNao) {

        if (contagemSim > contagemNao) {
            return votoMapper.toVencedorResponse(pauta, ValorVoto.SIM.getDescription());
        } else if (contagemNao > contagemSim) {
            return votoMapper.toVencedorResponse(pauta, ValorVoto.NAO.getDescription());
        }

        return votoMapper.toVencedorResponse(pauta, resultado: "Empate");
    }

    private boolean limiteDeVotosAtingido(Integer idPauta) {

        Optional<Pauta> pautaASerVotada = pautaService.getPautaById(idPauta);

        return pautaASerVotada.map(pauta ->
                        Objects.equals(pauta.getLimiteVotos(), repository.countByIdPauta(idPauta)))
                .orElse( other: false);
    }

}
```

CpfValidatorService

Métodos:

- Linha 13 (`verificaCpfValido`): método que utiliza a classe `CpfValidatorRepository` para validação do cpf informado como parâmetro.

```
@Service
public class CpfValidatorService {

    @Autowired
    private CpfValidatorRepository cpfValidatorRepository;

    public boolean verificaCpfValido(String cpf) {

        return cpfValidatorRepository.validaCpf(cpf);
    }
}
```

Controller

AssociadoController

Métodos:

- Linha 28 (`cadastrarAssociado`): método POST para realizar o cadastro do Associado.
- Linha 34 (`consultaCpf`): método GET para realizar a consulta do cpf informado no path da URL, retornando o status do cpf caso seja válido, caso negativo ele retorna “not found” (status code 404).

```
19  @RestController
20  @RequestMapping("associado")
21  public class AssociadoController {
22
23      @Autowired
24      AssociadoService associadoService;
25
26      @PostMapping
27      @ResponseStatus(HttpStatus.CREATED)
28      public void cadastrarAssociado(@RequestBody final CadastrarAssociadoRequest request) {
29          associadoService.cadastrarAssociado(request);
30      }
31
32      @GetMapping("{cpf}")
33      @ResponseStatus(HttpStatus.OK)
34      public ResponseEntity<StatusResponse> consultaCpf(@PathVariable String cpf) {
35          return Optional.ofNullable(associadoService.consultaCpf(cpf)) Optional<StatusResponse>
36                  .map(cpfResponse -> ResponseEntity.ok().body(cpfResponse)) Optional<ResponseEntity<StatusResponse>>
37                  .orElseGet(() -> ResponseEntity.notFound().build());
38      }
39  }
```

PautaController

Métodos:

- Linha 22 (`cadastrarPauta`): método POST para realizar o cadastro da Pauta.

```java
@RestController
@RequestMapping("pauta")
public class PautaController {

    @Autowired
    PautaService pautaService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void cadastrarPauta(@RequestBody final CadastrarPautaRequest request) {
        pautaService.cadastrarPauta(request);
    }
}
```

SessaoController

Métodos:

- Linha 22 (`abrirSessao`): método POST para realizar a abertura de uma Sessao.

```java
@RestController
@RequestMapping("sessao")
public class SessaoController {

    @Autowired
    SessaoService sessaoService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void abrirSessao(@RequestBody final AbrirSessaoRequest request) {
        sessaoService.criarSessao(request);
    }
```

VotoController

Métodos:

- Linha 25 (`votarPauta`): método POST votar em uma Pauta.
- Linha 31 (`resultadoPauta`): método para realizar a contagem de votos de uma pauta específica, informada através do idPauta no path da URL.

```java
@RestController
@RequestMapping("voto")
public class VotoController {

    @Autowired
    VotoService votoService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void votarPauta(@RequestBody final VotoRequest request) {
        votoService.receberVoto(request);
    }

    @GetMapping("{idPauta}")
    @ResponseStatus(HttpStatus.OK)
    public VencedorResponse resultadoPauta(@PathVariable Integer idPauta) {
        return votoService.contagemVotosVencedor(idPauta);
    }
}
```

Bônus 01: Integração com sistemas externos

Foi criada e disponibilizada na nuvem uma API que realiza a validação de CPF, retornando `false` caso o CPF seja inválido ou `true` caso seja válido. Essa API foi integrada na principal para que esse valor retornado seja utilizado para informar se o CPF do Associado é **ABLE_TO_VOTE** ou **UNABLE_TO_VOTE**, ou então retornando “not found” (status code 404) caso seja um CPF inválido.

CpfService

Métodos:

- Linha 12 (`consultaCpf`): método que retorna um valor booleano de acordo com a verificação do CPF passado como parâmetro no método `verificaCpfValido`.

```java
@Service
public class CpfService {

    private static final Pattern CPF_PATTERN = Pattern.compile("\\d{3}\\.?\\d{3}\\.?\\d{3}-?\\d{2}");

    public boolean consultaCpf(String cpf) {

        return verificaCpfValido(cpf);
    }
```

- Linha 17 (`verificaCpfValido`): método que valida se o CPF informado como parâmetro é válido ou não, retornando respectivamente `true` ou `false`.

```java
public static boolean verificaCpfValido(String cpf) {
    if (cpf == null || !CPF_PATTERN.matcher(cpf).matches()) {
        return false;
    }

    cpf = cpf.replaceAll( regex: "[.-]",  replacement: "");

    int sum = 0;
    int weight = 10;
    for (int i = 0; i < 9; i++) {
        int digit = Character.getNumericValue(cpf.charAt(i));
        sum += digit * weight;
        weight--;
    }

    int firstVerifier = 11 - (sum % 11);
    if (firstVerifier > 9) {
        firstVerifier = 0;
    }

    sum = 0;
    weight = 11;
    for (int i = 0; i < 9; i++) {
        int digit = Character.getNumericValue(cpf.charAt(i));
        sum += digit * weight;
        weight--;
    }
    sum += firstVerifier * 2;

    int secondVerifier = 11 - (sum % 11);
    if (secondVerifier > 9) {
        secondVerifier = 0;
    }

    return cpf.charAt(9) - '0' == firstVerifier && cpf.charAt(10) - '0' == secondVerifier;
}
```

CpfController

Métodos:

- Linha 21 (`consultaCpf`): método GET que retorna um valor booleano de acordo com a verificação do CPF passado como parâmetro no método `consultaCpf`.

```java
@RestController
@RequestMapping()
public class CpfController {

    @Autowired
    CpfService cpfService;

    @GetMapping("{cpf}")
    @ResponseStatus(HttpStatus.OK)
    public boolean consultaCpf(@PathVariable String cpf) { return cpfService.consultaCpf(cpf); }
}
```

Bônus 02: Teste Performance

Foi executado um teste de carga com a finalidade de verificar a performance da aplicação. Os parâmetros utilizados estão indicados na figura 1, a requisição na figura 2, e os resultados obtidos estão ilustrados na figura 3.

- Para realização do teste de carga a aplicação foi inicializada no ambiente local, assim os limites de gastos na nuvem não são atingidos e as aplicações hospedadas nela não são bloqueadas/excluídas. Por questões de limitações de hardware, não foi possível executar o teste com “centenas de milhares de votos”.

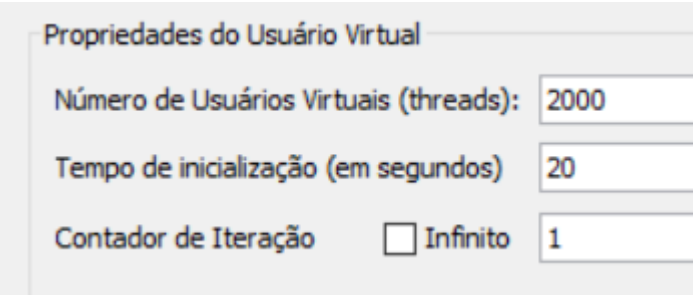


Figura 1.

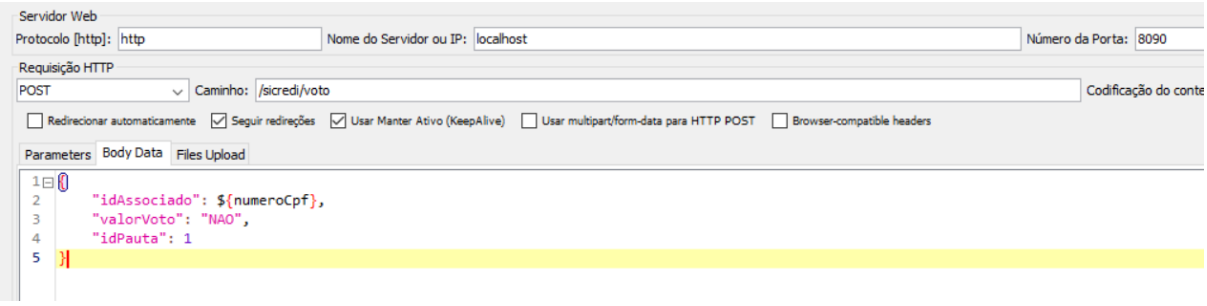


Figura 2.

Rótulo	# Amostras	Média	Mín.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
Requisição HTTP	2000	9	5	97	6,27	0,00%	99,1/sec	11,71	24,82	121,0
TOTAL	2000	9	5	97	6,27	0,00%	99,1/sec	11,71	24,82	121,0

Figura 3.

Bônus 03: Versionamento da API

Utilizaria a metodologia de versionamento que inclui a versão na URL. Podendo ter endpoints como **/api/v1/users** e **/api/v2/users**. Isso permite que as diferentes versões coexistam na mesma aplicação, oferecendo uma maneira fácil de distinguir entre elas e possibilitando que, caso seja necessário uma migração entre versões, ela ocorra de uma maneira mais gradual e sem prejudicar sistemas/serviços que consomem esses endpoints.

Também consideraria o versionamento via inclusão da versão da API no cabeçalho da solicitação. Isso permitiria manter a mesma URL para todas as versões, possivelmente exigindo menos retrabalho caso seja necessário uma migração entre elas.