



PROJETO CUSTOMIZAÇÕES

Robô Genérico de Execução de Atividades

Amanda Alvim

1. Objetivo

O objetivo do robô genérico de executar atividades é realizar as execuções automáticas das atividades que os clientes não desejam aprovar manualmente e as atividades que nós precisamos que sejam aprovadas automaticamente para o funcionamento do processo desenhado.

O Robô genérico de execução de atividades se encontra no Projeto Base no nosso git (<http://git.lecom.com.br/PSP/Projeto-Base-BPM>) com o nome de **RoboExecutarAtividades**, que está no package `com.lecom.workflow.robo.satellite.generico`, para funcionamento dele existe dois arquivos que são como base que são os properties: **RoboExecutarAtividades.properties** e **automatico.properties**, eles ficam no caminho `(upload/cadastros/config)` do projeto.

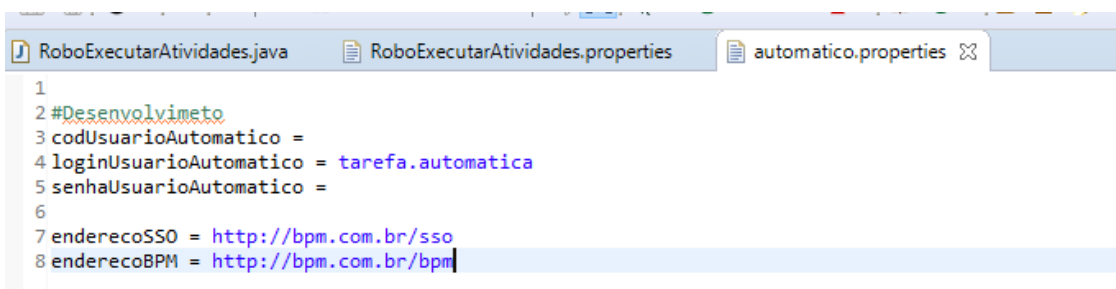
2. Como usar

No momento que você for colocar em um cliente esse robô você precisa ir na classe java e gerar um `.jar` pelo eclipse e subir no Lecom BPM (Menu Studio – Serviços – Aba Robos) e colocar o `.properties` no lugar correto do servidor, explicado abaixo melhor no menu de properties.

3. Properties

Os properties são a base para o funcionamento do robô conforme mencionado acima, nele nós iremos configurar as informações que o robô irá usar para realizar as execuções automáticas, os properties são usados em várias situações no decorrer dos nossos desenvolvimentos, eles ficam diretamente no servidor do cliente no caminho `/opt/lecom/app/tomcat/webapps/bpm/upload/cadastros/config` para ambientes anteriores a nova forma de atualização das customizações e na nova forma fica no caminho `/opt/lecom/custom/web-content/config`.

Vamos ver o que temos nos dois properties citados, para o arquivo `automatico.properties` ele contém o seguinte conteúdo:



```
1
2 #Desenvolvimeto
3 codUsuarioAutomatico =
4 loginUsuarioAutomatico = tarefa.automatica
5 senhaUsuarioAutomatico =
6
7 enderecoSSO = http://bpm.com.br/sso
8 enderecoBPM = http://bpm.com.br/bpm
```

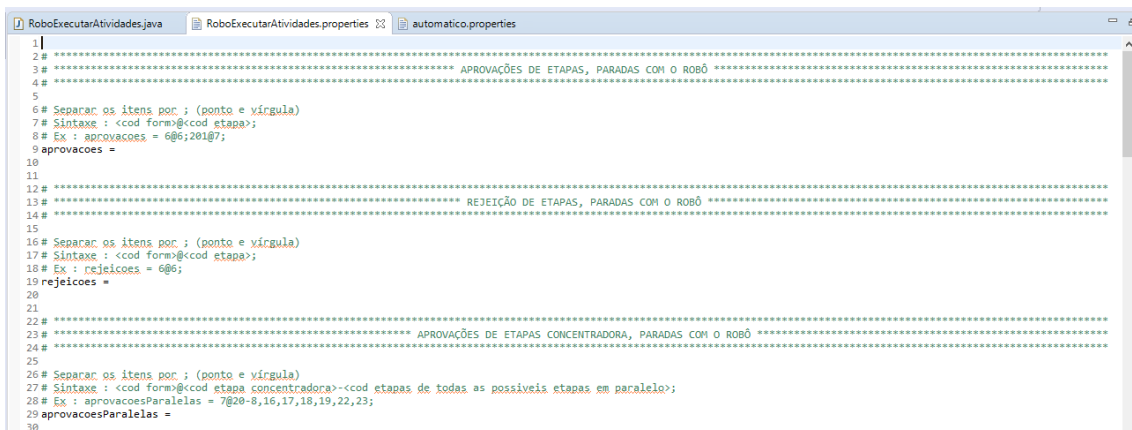
Na imagem acima temos os campos de configuração padrão para o nosso usuário automático, em todos os ambientes de clientes criamos um usuário específico denominando automático sem a opção de senha expira para que possamos conseguir deixar nossos properties com usuário e senha que somente nós consultores sabemos, sendo assim precisamos configurar:

- codUsuarioAutomatico: Nesse campo será preciso colocar o código do usuário automático criado, essa informação pegamos na tabela de usuário no banco do Lecom BPM,

precisamos dessa informação para poder fazer as buscas dos processos parados nesse usuário.

- **loginUsuarioAutomatico:** Nesse campo será preciso colocar o login do usuário automático criado, essa informação é necessária para que possamos realizar o login por fora da ferramenta e assim fazer aprovação automática das atividades.
- **senhaUsuarioAutomatico:** Nesse campo será preciso colocar a senha configurada no cadastro do usuário automático, essa informação é necessária para que possamos realizar o login por fora da ferramenta e assim fazer aprovação automática das atividades.
- **enderecoSSO:** Nesse campo será preciso colocar a url do cliente completa com o final SSO, conforme no exemplo (<https://bpm.lecom.com.br/sso>) , essa informação é necessária para a realização do login por fora.
- **enderecoBPM:** Nesse campo será preciso colocar a url do cliente completa com o final BPM, conforme no exemplo (<https://bpm.lecom.com.br/bpm>) , essa informação é necessária para a realização da aprovação das atividades.

O arquivo RoboExecutarAtividades.properties tem os seguintes conteúdos, como ele é muito grande iremos destacar e explicar abaixo por partes a configuração dele:



```

1 |
2 #
3 # ***** APROVAÇÕES DE ETAPAS, PARADAS COM O ROBÔ *****
4 #
5
6 # Separar os itens por ; (ponto e vírgula)
7 # Sintaxe : <cod form>@<cod etapa>;
8 # Ex : aprovacoes = 6@6;2@1@7;
9 aprovacoes =
10
11
12 #
13 # ***** REJEIÇÃO DE ETAPAS, PARADAS COM O ROBÔ *****
14 #
15
16 # Separar os itens por ; (ponto e vírgula)
17 # Sintaxe : <cod form>@<cod etapa>;
18 # Ex : rejeicoes = 6@6;
19 rejeicoes =
20
21
22 #
23 # ***** APROVAÇÕES DE ETAPAS CONCENTRADORA, PARADAS COM O ROBÔ *****
24 #
25
26 # Separar os itens por ; (ponto e vírgula)
27 # Sintaxe : <cod form>@<cod etapa concentradora>-<cod etapas de todas as possíveis etapas em paralelo>;
28 # Ex : aprovacoesParalelas = 7@20-8,16,17,18,19,22,23;
29 aprovacoesParalelas =
30

```

Na imagem acima temos o começo das configurações para os processos que forem do formulário antigo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **aprovacoes:** Nesse campo deve se configurar quais formulários e atividades você deseja que seja aprovada automaticamente, colocando o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).
- **rejeicoes:** Nesse campo deve se configurar quais formulários e atividades você deseja que seja rejeitada automaticamente, colocando o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).

- **aprovacoesParalelas**: Nesse campo deve se configurar as atividades concentradores que você deseja que seja aprovada automaticamente, lembrando que nesse caso ele só vai aprovar a atividade caso todos os seus paralelos já foram aprovados, então você precisa configurar aqui o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade concentradora, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, e os códigos de todas as atividades que são do paralelismo e todas as possibilidades que o roteamento chegue até a concentradora, conforme exemplo: 1@2-3,4; (formulário 1 concentradora 2 e paralelismo 3 e 4)

```

31
32#
33# ***** APROVAÇÕES DE ETAPAS EM UMA DATA, DETERMINADA POR UM CAMPO, DEFINIDO NO PROPERTIE, QUE ESTEJAM PARADAS COM O ROBÔ *****
34#
35
36# Separar os itens por ; (ponto e vírgula)
37# Sintaxe : <cod form>@<cod etapa>@<nome do campo>@<nome tabela>;
38# Ex : aprovacoesDataCampo = 33@1@DATA_INICIO_TRAB@f_contratacao_func;
39 aprovacoesDataCampo =
40
41
42#
43# ***** REPROVAÇÕES DE ETAPAS EM UMA DATA, DETERMINADA POR UM CAMPO, DEFINIDO NO PROPERTIE, QUE ESTEJAM PARADAS COM O ROBÔ *****
44#
45
46# Separar os itens por ; (ponto e vírgula)
47# Sintaxe : <cod form>@<cod etapa>@<nome do campo>@<nome tabela>;
48# Ex : rejeicoesDataCampo = 33@1@DATA_INICIO_TRAB@f_contratacao_func;
49 rejeicoesDataCampo =
50
51
52#
53# ***** APROVA ATIVIDADES QUE O "Prazo Máximo", DEFINIDO PARA ELA, TENHA EXCEDIDO E QUE ESTEJAM PARADAS COM O ROBÔ *****
54#
55
56# Separar os itens por ; (ponto e vírgula)
57# Sintaxe : <cod form>@<cod etapa>@<nome do campo observacao>@<mensagem para registrar aprovado automática>;
58# Ex : aprovacoesEtapasPrazoMaximoExcedido = 33@OBSERVACAO@Prazo para complementar informações excedido. Processo foi aprovado automaticamente;
59 aprovacoesEtapasPrazoMaximoExcedido =
60

```

Na imagem acima continuamos com as configurações para os processos que forem do formulário antigo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **aprovacoesDataCampo**: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a aprovação automática dessas etapas quando chegar na data do campo configurado, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).

- **rejeicoesDataCampo**: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a rejeição automática dessas etapas quando chegar na data do campo configurado, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).

- **aprovacoesEtapasPrazoMaximoExcedido**: Nesse campo você deve configurar os formulários, atividades que você deseja que seja aprovada quando chegar na data limite configurada pelo Studio, sendo assim você precisa informar código do seu formulário,

informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)

```

RoboExecutarAtividades.java RoboExecutarAtividades.properties automatico.properties
61 #
62 # ***** REJEITA ATIVIDADES QUE O "Prazo Máximo", DEFINIDO PARA ELA, TENHA EXCEDIDO E QUE ESTEJAM PARADAS COM O ROBÔ *****
63 #
64 #
65
66 # Separar os itens por ; (ponto e vírgula)
67 # Sintaxe : <cod form>@<cod etapa>@<nome do campo observacao>@<mensagem para registrar execução automática>;
68 # Ex : rejeicoesEtapasPrazoMaximoExcedido = 3@3@OBSERVACAO@prazo para complementar informações excedido. Processo foi rejeitado automaticamente;
69 rejeicoesEtapasPrazoMaximoExcedido =
70
71

```

Na imagem acima continuamos com as configurações para os processos que forem do formulário antigo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **rejeicoesEtapasPrazoMaximoExcedido**: Nesse campo você deve configurar os formulários, atividades que você deseja que seja rejeitada quando chegar na data limite configurada pelo Studio, sendo assim você precisa informar código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)

```

RoboExecutarAtividades.java RoboExecutarAtividades.properties automatico.properties
72 #
73 # ***** PARAMETROS 5.30 FORM NOVO *****
74 #
75 #
76 #
77 # ***** APROVAÇÕES DE ETAPAS, PARADAS COM O ROBÔ OU NÃO *****
78 #
79 #
80
81 # Separar os itens por ; (ponto e vírgula)
82 # Sintaxe : <cod form>@<cod etapa>;
83 # Ex : aprovacoesFN = 6@6;
84 aprovacoesFN =
85 aprovacoesFNUC =
86
87
88 # ***** REJEIÇÃO DE ETAPAS, PARADAS COM O ROBÔ OU NÃO *****
89 #
90 #
91
92 # Separar os itens por ; (ponto e vírgula)
93 # Sintaxe : <cod form>@<cod etapa>;
94 # Ex : rejeicoesFN = 6@6;
95 rejeicoesFN =
96 rejeicoesFNUC =
97
98
99 # ***** APROVAÇÕES DE ETAPAS CONCENTRADORA, PARADAS COM O ROBÔ OU NÃO *****
100 #
101 #
102
103 # Separar os itens por ; (ponto e vírgula)
104 # Sintaxe : <cod form>@<cod etapa concentradora>-<cod etapas de todas as possíveis etapas em paralelo>;
105 # Ex : aprovacoesFNConcentradora = 7@20-8,16,17,18,19,22,23;
106 aprovacoesFNConcentradora =
107 aprovacoesFNConcentradoraUC =
108
109

```

Na imagem acima temos o começo das configurações para os processos que forem do formulário novo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **aprovacoesFN**: Nesse campo deve se configurar quais formulários e atividades você deseja que seja aprovada automaticamente, colocando o código do seu formulário,

informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).

- aprovacoesFNUC: Nesse campo deve se configurar quais formulários e atividades você deseja que seja aprovada automaticamente e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a aprovação, colocando o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).

- rejeicoesFN: Nesse campo deve se configurar quais formulários e atividades você deseja que seja rejeitada automaticamente, colocando o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).

- rejeicoesFNUC: Nesse campo deve se configurar quais formulários e atividades você deseja que seja rejeitada automaticamente e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a rejeição, colocando o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, e o código da sua atividade a ser aprovada, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, todos os códigos de formulário e etapa devem estar separados por ; conforme exemplo: 1@2;2@3 (formulário 1 etapa 2, formulário 2 etapa 3).

- aprovacoesFNConcentradora: Nesse campo deve se configurar as atividades concentradores que você deseja que seja aprovada automaticamente, lembrando que nesse caso ele só vai aprovar a atividade caso todos os seus paralelos já foram aprovados, então você precisa configurar aqui o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade concentradora, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, e os códigos de todas as atividades que são do paralelismo e todas as possibilidades que o roteamento chegue até a concentradora, conforme exemplo: 1@2-3,4; (formulário 1 concentradora 2 e paralelismo 3 e 4)

- aprovacoesFNConcentradoraUC: Nesse campo deve se configurar as atividades concentradores que você deseja que seja aprovada automaticamente e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a aprovação, lembrando que nesse caso ele só vai aprovar a atividade caso todos os seus paralelos já foram aprovados, então você precisa configurar aqui o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade concentradora, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, e os códigos de todas as atividades que são do paralelismo e todas as possibilidades que o roteamento chegue até a concentradora, conforme exemplo: 1@2-3,4; (formulário 1 concentradora 2 e paralelismo 3 e 4)

```

110 # ***** REJEIÇÃO DE ETAPAS CONCENTRADORA, PARADAS COM O ROBÔ OU NÃO *****
111 #
112 #
113 #
114 # Separar os itens por : (ponto e vírgula)
115 # Sintaxe : <cod form>@<cod etapa concentradora>-<cod etapas de todas as possíveis etapas em paralelo>;
116 # Ex : rejeicoesFNConcentradora = 7@20-8,16,17,18,19,22,23;
117 rejeicoesFNConcentradora =
118 rejeicoesFNConcentradoraUC =
119
120
121 # ***** APROVAÇÕES DE ETAPAS EM UMA DATA, DETERMINADA POR UM CAMPO, DEFINIDO NO PROPERTIE, QUE ESTEJAM PARADAS COM O ROBÔ OU NÃO *****
122 #
123 #
124 #
125 # Separar os itens por : (ponto e vírgula)
126 # Sintaxe : <cod form>@<cod etapa>@<nome do campo>@<nome tabela>;
127 # Ex : aprovacoesFNDataCampo = 33@18@DATA_INICIO_TRAB@f_contratacao_func;
128 aprovacoesFNDataCampo =
129 aprovacoesFNDataCampoUC =
130
131
132 # ***** REPROVAÇÕES DE ETAPAS EM UMA DATA, DETERMINADA POR UM CAMPO, DEFINIDO NO PROPERTIE, QUE ESTEJAM PARADAS COM O ROBÔ OU NÃO *****
133 #
134 #
135 #
136 # Separar os itens por : (ponto e vírgula)
137 # Sintaxe : <cod form>@<cod etapa>@<nome do campo>@<nome tabela>;
138 # Ex : rejeicoesFNDataCampo = 33@18@DATA_INICIO_TRAB@f_contratacao_func;
139 rejeicoesFNDataCampo =
140 rejeicoesFNDataCampoUC =
141

```

Na imagem acima temos o começo das configurações para os processos que forem do formulário novo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **rejeicoesFNConcentradora**: Nesse campo deve se configurar as atividades concentradores que você deseja que seja rejeitada automaticamente, lembrando que nesse caso ele só vai rejeitar a atividade caso todos os seus paralelos já foram aprovados, então você precisa configurar aqui o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade concentradora, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, e os códigos de todas as atividades que são do paralelismo e todas as possibilidades que o roteamento chegue até a concentradora, conforme exemplo: 1@2-3,4; (formulário 1 concentradora 2 e paralelismo 3 e 4)
- **rejeicoesFNConcentradoraUC**: Nesse campo deve se configurar as atividades concentradores que você deseja que seja rejeitada automaticamente e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a rejeição, lembrando que nesse caso ele só vai rejeitar a atividade caso todos os seus paralelos já foram aprovados, então você precisa configurar aqui o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade concentradora, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, e os códigos de todas as atividades que são do paralelismo e todas as possibilidades que o roteamento chegue até a concentradora, conforme exemplo: 1@2-3,4; (formulário 1 concentradora 2 e paralelismo 3 e 4)
- **aprovacoesFNDataCampo**: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a aprovação automática dessas etapas quando chegar na data do campo configurado, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).

- aprovacoesFNDataCampoUC: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a aprovação automática dessas etapas quando chegar na data do campo configurado e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a aprovação, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).

- rejeicoesFNDataCampo: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a rejeição automática dessas etapas quando chegar na data do campo configurado, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).

- rejeicoesFNDataCampoUC: Nesse campo você irá configurar qual formulário, atividade e campo deve ser baseado para que ocorra a rejeição automática dessas etapas quando chegar na data do campo configurado e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a rejeição, para isso você precisa informar o código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, o nome do campo, informação que você pode pegar na lista de campos do seu formulário, e o nome da tabela do formulário, informação que você pode pegar na aba de Propriedades do seu formulário, conforme exemplo: 1@2@DATA_ESPERA@f_form1 (formulário 1 etapa 2 campo DATA_ESPERA e tabela f_form1).


```

137 # Sintaxe : <cod form><cod etapa><nome do campo><nome tabela>;
138 # Ex : rejeicoesFNDataCampo = 3@18@DATA_INICIO_TRAB@f_contratacao_func;
139 rejeicoesFNDataCampo =
140 rejeicoesFNDataCampoUC =
141
142
143 # ***** APROVA ATIVIDADES QUE O "Prazo Máximo", DEFINIDO PARA ELA, TENHA EXCEDIDO E QUE ESTEJAM PARADAS COM O ROBÔ OU NÃO *****
144 # *****
145 #
146
147 # Separar os itens por ; (ponto e vírgula)
148 # Sintaxe : <cod form><cod etapa><nome do campo observacao><mensagem para registrar: aprovado automaticamente>;
149 # Ex : aprovacoesFNEtapasPrazoMaximoExcedido = 3@3@OBSERVACAO@Prazo para complementar informações excedido. Processo foi aprovado automaticamente;
150 aprovacoesFNEtapasPrazoMaximoExcedido =
151 aprovacoesFNEtapasPrazoMaximoExcedidoUC =
152
153
154 # ***** REJEITA ATIVIDADES QUE O "Prazo Máximo", DEFINIDO PARA ELA, TENHA EXCEDIDO E QUE ESTEJAM PARADAS COM O ROBÔ OU NÃO *****
155 # *****
156 #
157
158 # Separar os itens por ; (ponto e vírgula)
159 # Sintaxe : <cod form><cod etapa><nome do campo observacao><mensagem para registrar execução automática>;
160 # Ex : rejeicoesFNEtapasPrazoMaximoExcedido = 31@11@MANIF_ADICIONAIS@Prazo para complementar informações foi excedido. Processo prosseguiu automaticamente!;
161 rejeicoesFNEtapasPrazoMaximoExcedido =
162 rejeicoesFNEtapasPrazoMaximoExcedidoUC =
163
164
165 # ***** CANCELA PROCESSOS QUE ESTÃO NA ATIVIDADE INICIAL, CICLO 1, PARADOS A MAIS DE X DIAS *****
166 # *****
167 #
168 # OBS : na opção "cancelaFNProcessoNaoEnviado", o Usuário do Robô precisa ser gestor do modelo, o cancelamento é feito como se fosse o gestor do modelo, cancelando o processo
169 # OBS2 : na opção "cancelaFNProcessoNaoEnviadoUC", o Usuário do Robô é incluído na processo_etapa_usu o cancelamento é feito como se o usuário clicasse no botão cancelar
170
171 # Separar os itens por ; (ponto e vírgula)
172 # Sintaxe : <cod form><gttd dias>;
173 # Ex : cancelaFNProcessoNaoEnviado = 31@19;
174 cancelaFNProcessoNaoEnviado =
175 cancelaFNProcessoNaoEnviadoUC =

```

Na imagem acima continuamos com as configurações para os processos que forem do formulário novo, esse arquivo de properties ele já vem comentado com o que deve ser preenchido, mas vamos aos detalhes aqui:

- **aprovacoesFNEtapasPrazoMaximoExcedido**: Nesse campo você deve configurar os formulários, atividades que você deseja que seja aprovada quando chegar na data limite configurada pelo Studio, sendo assim você precisa informar código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)
- **aprovacoesFNEtapasPrazoMaximoExcedidoUC**: Nesse campo você deve configurar os formulários, atividades que você deseja que seja aprovada quando chegar na data limite configurada pelo Studio e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a aprovação, sendo assim você precisa informar código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)
- **rejeicoesFNEtapasPrazoMaximoExcedido**: Nesse campo você deve configurar os formulários, atividades que você deseja que seja rejeitada quando chegar na data limite configurada pelo Studio, sendo assim você precisa informar código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a

atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)

- rejeicoesFNEtapasPrazoMaximoExcedidoUC: Nesse campo você deve configurar os formulários, atividades que você deseja que seja rejeitada quando chegar na data limite configurada pelo Studio e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará a rejeição, sendo assim você precisa informar código do seu formulário, informação que consegue pegar na listagem de modelos do Studio, o código da sua atividade, informação que você pode pegar passando o mouse em cima do nome da atividade na tela do Studio na aba de etapas/atividades, nome de um campo de observação para que a atividade seja aprovada com uma mensagem diferente configurada por você, e a mensagem que deseja que apareça na sequência do processo informando que ele foi aprovado automaticamente, conforme exemplo: 1@2@OBS@Processo aprovado automaticamente após prazo máximo; (formulário 1 etapa 2 campo OBS e mensagem que será preenchida no campo OBS)

- cancelaFNProcessoNaoEnviado: Nesse campo você deve configurar os formulários que deseja que seja cancelados na primeira atividade ciclo 1 depois de x dias, lembrando que essa situação ocorre caso a pessoa ou um outro desenvolvimento abriu na primeira atividade e não deu sequência com o processo, para isso deve informar o código do formulário, informação que consegue pegar na listagem de modelos do Studio, e a quantidade de dias que deseja aguardar para que seja feito o cancelamento automático, conforme exemplo: 1@15;(formulário 1 e 15 dias)

- cancelaFNProcessoNaoEnviadoUC: Nesse campo você deve configurar os formulários que deseja que seja cancelados na primeira atividade ciclo 1 depois de x dias e que esteja parada com um usuário comum dessa forma o robô irá incluir o automático como um possível aprovador e realizará o cancelamento, lembrando que essa situação ocorre caso a pessoa ou um outro desenvolvimento abriu na primeira atividade e não deu sequência com o processo, para isso deve informar o código do formulário, informação que consegue pegar na listagem de modelos do Studio, e a quantidade de dias que deseja aguardar para que seja feito o cancelamento automático, conforme exemplo: 1@15;(formulário 1 e 15 dias)

4. Fonte Java

No tópico acima foi explicado como configurar os properties que serão o que de fato vocês irão configurar, mas para conhecimento e entendimento do fonte vamos passar por ele e os seus métodos para saberem o que está sendo feito.

```

1 package com.lecom.workflow.roboto.satelite.geratico;
2
3 import java.sql.Connection;
4
5 @RobotModule(value = "RoboExecutarAtividades")
6 @Version({1,0,12})
7 public class RoboExecutarAtividades implements WebServices {
8
9     private static final Logger logger = Logger.getLogger(RoboExecutarAtividades.class);
10    private String configPath = Funcoes.getRotaRootDir() + "/upload/cadastros/config/";
11    private String accessToken538550;
12
13    @Execution
14    public void AprovarEtapas() throws Exception {
15        logger.info(">>> Inicio RoboExecutarAtividades Genérico <<<");
16
17        try ( Connection cnLecom = DBUtils.getConnection("workflow") ) {
18
19            Map<String, String> parametros = Funcoes.getParametrosIntegracao(configPath + getClass().getSimpleName());
20
21            Map<String, String> automatico = Funcoes.getParametrosIntegracao(configPath + "automatico");
22            Integer codUsuarioAutomatico = new Integer ( automatico.get("codUsuarioAutomatico") );
23            String loginUsuarioAutomatico = automatico.get("loginUsuarioAutomatico");
24            String senhaUsuarioAutomatico = automatico.get("senhaUsuarioAutomatico");
25            String urlSSO = automatico.get("enderecoSSO");
26            String urlBPM = automatico.get("enderecoBPM");
27
28            // Data Atual
29            Calendar dataAtual = Calendar.getInstance();
30
31            // Cria um Objeto LocalDate com a data atual ( Java 8 ).
32            LocalDate dataAtual = LocalDate.now();
33
34            // Aprova etapas, em paralelo, definidas no properties, paradas com o Robô
35            executarAtividadeConcentradora(cnLecom, false, parametros.get("aprovacoesParalelas"), codUsuarioAutomatico, loginUsuarioAutomatico, senhaUsuarioAutomatico, "P", n
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70

```

Na imagem acima temos o começo da nossa classe JAVA:

- Na **linha 37** é onde informamos que essa classe é do tipo robô, colocando essa annotation `@RobotModule(value="Nome Robô")`, dentro dela apenas colocamos um value, um nome para classe, esse nome não interfere em nada no procedimento de execução do robô.
- Na **linha 38** informamos a annotation `@Version({1,0,0})`, nela você irá informar a versão da sua classe geralmente iniciamos em 1,0,0 e a cada subida com alguma alteração na classe altera-se o número assim você irá identificar que está subindo a versão correta da sua classe.
- Na **linha 41** definimos a variável de log, onde utilizamos a biblioteca do log4j que o Lecom BPM já utiliza, nessa linha somente estamos definindo o nome da variável e iremos utilizar mais para baixo e sempre usamos as opções debug, info e error.
- Na **linha 42** estamos definindo o caminho a qual irá estar o arquivo de properties (esse caminho completo está mencionado no começo do documento)
- Na **linha 46** é onde usamos a anotação de `@Execution`, isso informa que esse método é o que o Lecom BPM irá chamar primeiro, se não tiver a anotação ele não deixará subir o seu jar.
- Na **linha 51** estamos iniciando uma conexão com o banco de dados do Lecom BPM, utilizando uma classe do produto (DBUtils) que nos retorna a conexão com o bpm.
- Nas **linhas 53 e 55** estamos transformando esses properties que foram explicados acima em um `Map<String, String>` para que seja fácil de pegar os parâmetros que configuramos.
- Nas **linhas 56 a 60** estão pegando os parâmetros do properties automatico para utilizar as informações que foram preenchidas.

```

2639 //
2640 // Aprova etapas, em paralelo, definidas no properties, paradas com o Robô
2641 //
2642 private void executarAtividadeConcentradora(Connection cnLecom, boolean formNovo, String atividadesExecutar, Integer codUsuarioAutomatico, String loginUsuarioAutomatico,
2643 String senhaUsuarioAutomatico, String paramacaoAprovaRejeita, String urlBPM, boolean transferirRobo ) throws Exception{
2644
2645     logger.info("INICIO executarEtapasConcentradora : " + atividadesExecutar);
2646
2647     if( !"".equals(atividadesExecutar) ){
2648         for (String atividadeExecut : atividadesExecutar.split(";")) {
2649             String[] aprovacaoFormEtapas = atividadeExecut.split("@");
2650             String[] aprovacaoEtapasConcentradora = aprovacaoFormEtapas[1].split("-");
2651
2652             String codFormConcentradora = aprovacaoFormEtapas[0];
2653             String codEtapasConcentradora = aprovacaoEtapasConcentradora[0];
2654             String codEtapasParalelismo = aprovacaoEtapasConcentradora[1];
2655
2656             logger.info("PARALELO - Em análise - Form : " + codFormConcentradora + " / Etapa Concentradora : " + codEtapasConcentradora + " / Etapa Paralelismo : " + codEtapasParalelismo);
2657
2658             // consulta as etapas Concentradoras que ainda estão em aberto
2659             StringBuilder sqlConsultaAux1 = new StringBuilder();
2660             sqlConsultaAux1.append(" SELECT P.COD_FORM, PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO, P.IDE_BETA_TESTE ");
2661             sqlConsultaAux1.append(" FROM PROCESSO ETAPA PE ");
2662             sqlConsultaAux1.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
2663             sqlConsultaAux1.append(" WHERE PE.IDE_STATUS IN ('A') ");
2664             sqlConsultaAux1.append(" AND P.COD_FORM = ");
2665             sqlConsultaAux1.append(codFormConcentradora);
2666             sqlConsultaAux1.append(" AND PE.COD_ETAPA IN ( ");
2667             sqlConsultaAux1.append(codEtapasConcentradora);
2668             sqlConsultaAux1.append(" ) ");
2669
2670             if ( !transferirRobo ) {
2671                 sqlConsultaAux1.append(" AND PE.COD_USUARIO_ETAPA IN ( ");
2672                 sqlConsultaAux1.append(codUsuarioAutomatico);
2673                 sqlConsultaAux1.append(" ) ");
2674             }
2675
2676             sqlConsultaAux1.append(" ORDER BY PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO ");
2677
2678             try (PreparedStatement pstConsultaAux1 = cnLecom.prepareStatement(sqlConsultaAux1.toString());
2679                 ResultSet rsConsultaAux1 = pstConsultaAux1.executeQuery()); {
2680
2681                 while (rsConsultaAux1.next()) {
2682                     // Consulta as etapas do Paralelismo que ainda estão em aberto
2683                     StringBuilder sqlConsultaAux2 = new StringBuilder();
2684                     sqlConsultaAux2.append(" SELECT COUNT(P.COD_PROCESSO) QTD_ETAPAS_ABERTO ");
2685                     sqlConsultaAux2.append(" FROM PROCESSO_ETAPA PE ");
2686                     sqlConsultaAux2.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
2687                     sqlConsultaAux2.append(" WHERE PE.IDE_STATUS IN ('A') ");
2688                     sqlConsultaAux2.append(" AND P.COD_PROCESSO = ");
2689                     sqlConsultaAux2.append(rsConsultaAux1.getString("COD_PROCESSO"));
2690                     sqlConsultaAux2.append(" AND PE.COD_ETAPA IN ( ");
2691                     sqlConsultaAux2.append(codEtapasParalelismo);
2692                     sqlConsultaAux2.append(" ) ");
2693                     sqlConsultaAux2.append(" ORDER BY PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO ");
2694
2695                     try (PreparedStatement pstConsultaAux2 = cnLecom.prepareStatement(sqlConsultaAux2.toString());
2696                         ResultSet rsConsultaAux2 = pstConsultaAux2.executeQuery()); {
2697
2698                         while (rsConsultaAux2.next()) {
2699                             if (rsConsultaAux2.getInt("QTD_ETAPAS_ABERTO") == 0) {
2700                                 Integer codProcesso = rsConsultaAux1.getInt("COD_PROCESSO");
2701                                 Integer codEtapas = rsConsultaAux1.getInt("COD_ETAPA");
2702                                 Integer codCiclo = rsConsultaAux1.getInt("COD_CICLO");
2703                                 String modoTeste = rsConsultaAux1.getString("IDE_BETA_TESTE");
2704
2705                                 if ( !transferirRobo ) {
2706                                     if ( !verificarUsuarioProcessoEtapasUsu( cnLecom, codProcesso, codEtapas, codCiclo, codUsuarioAutomatico ) ) {
2707                                         inserirUsuarioEtapas ( cnLecom, codProcesso, codEtapas, codCiclo, codUsuarioAutomatico );
2708                                     }
2709                                 }
2710
2711                                 if ( formNovo ) {
2712                                     logger.info("APROVACOES - RETORNO FN : Proc / Etapa - ( " + codProcesso + " / " + codEtapas + " ) - " + executarAtividadeProcessoFormNovo(codProcesso.toString(), codEtapas));
2713                                 } else {
2714                                     String valores = "OBSERVACAO)Processo aprovado automaticamente.";
2715                                     String valores = "";
2716                                     logger.info("PARALELO - RETORNO : Proc / Etapa - ( " + codProcesso + " / " + codEtapas + " ) - " + executarEtapasProcesso(rsConsultaAux1.getInt("COD_FORM"), codProcesso));
2717                                 }
2718                             }
2719                         }
2720                     }
2721                 }
2722             }
2723         }
2724     }
2725 }

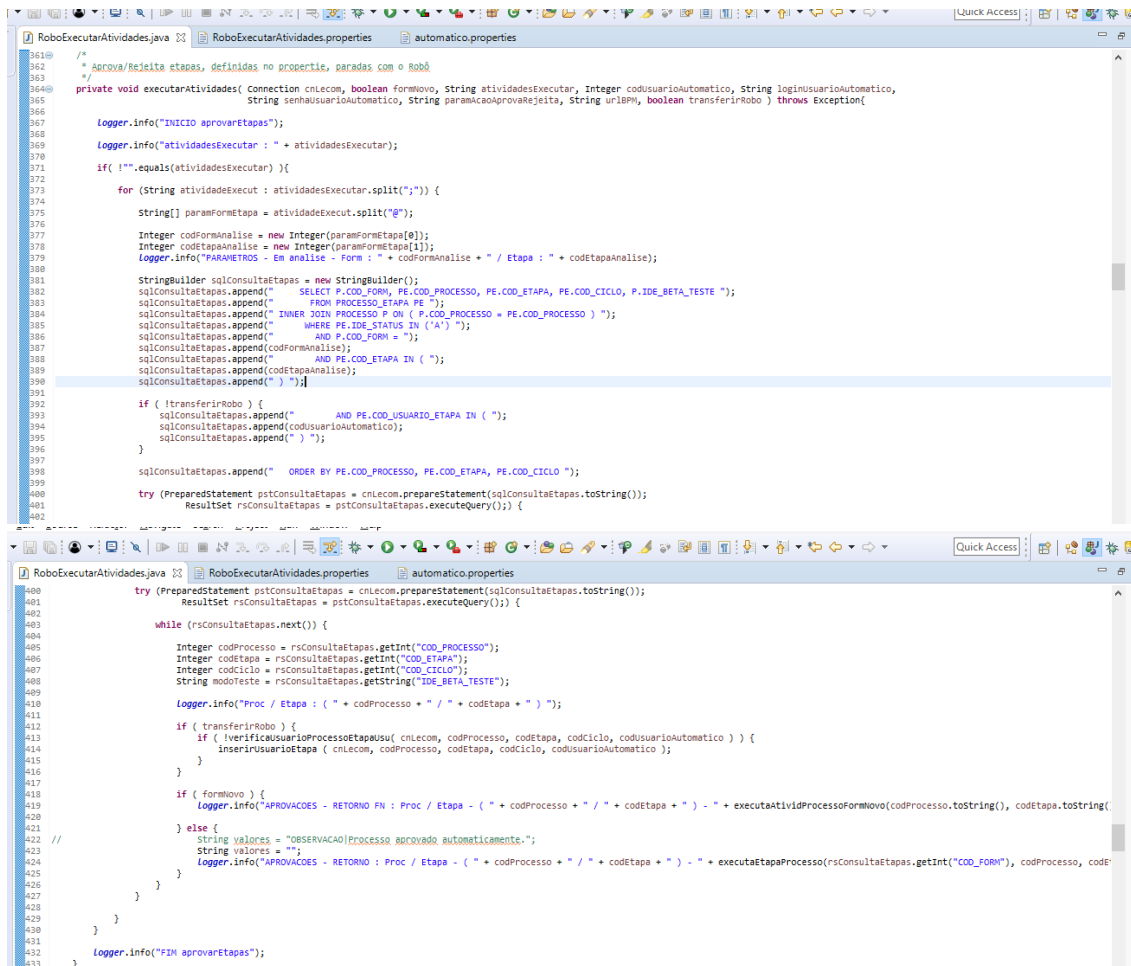
```

Na imagem acima temos o print do método **executarAtividadeConcentradora** que irá utilizar os parâmetros explicado nas páginas 3,6 e7, os parâmetros passados para esses métodos são:

- Conexão (a conexão que foi instanciada na linha 51)
- se é formulário novo (é passado true ou false para verificar se está realizando as aprovações dos formulários antigos ou novos)
- o valor dos parâmetros (aprovacoesParalelas, aprovacoesFNConcentradora, aprovacoesFNConcentradoraUC, rejeicoesFNConcentradora, rejeicoesFNConcentradoraUC)
- código de usuário automatico (variável instanciada na linha 56)
- login do usuário automatico (variável instanciada na linha 57)
- senha do usuário automático (variável instanciada na linha 58)
- parâmetro da ação (P se for aprovação, R se for rejeição)
- url do BPM (variável instanciada na linha 60)
- se irá realizar transferência para o usuário robô (é passado true se estiver utilizando os parâmetros aprovacoesFNConcentradoraUC e rejeicoesFNConcentradoraUC se não vem false)

No restante do método é feito um select para identificar todos os processos do formulário configurado que estão parados na atividade configurada e com ou não o usuário automatico, a partir desses processos retornados é avaliado se as etapas paralelas já foram aprovadas,

caso o retorno do count seja 0 ele realiza a aprovação ou rejeição da etapa, indo para o método do formulário antigo ou do formulário novo dependendo do parâmetro que veio, também verifica se tem o true para a transferência do usuário automatico e assim realiza a chamada do método que faz essa parte de inserção da permissão para que o usuário possa aprovar uma atividade que não esteja com ele.



```
3610
362  /* Aprova/rejeita etapas, definidas no properties, paradas com o Robô
363  */
3640 private void executarAtividades( Connection cnLecom, boolean formNovo, String atividadesExecutar, Integer codusuarioAutomatico, String loginusuarioAutomatico,
365  String senhausuarioAutomatico, String parametroAprovaRejeita, String urlBPM, boolean transferirRobo ) throws Exception{
366
367  Logger.info("INICIO aprovarEtapas");
368  Logger.info("atividadesExecutar : " + atividadesExecutar);
369  if ( !"".equals(atividadesExecutar) ){
370  for (String atividadeExecut : atividadesExecutar.split(";")) {
371  String[] paramFormEtapas = atividadeExecut.split("@");
372  Integer codFormAnalise = new Integer(paramFormEtapas[0]);
373  Integer codEtapasAnalise = new Integer(paramFormEtapas[1]);
374  Logger.info("PARAMETROS - Em analise - Form : " + codFormAnalise + " / Etapa : " + codEtapasAnalise);
375  StringBuilder sqlConsultaEtapas = new StringBuilder();
376  sqlConsultaEtapas.append(" SELECT P.COD_FORMA, PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO, P.IDE_BETA_TESTE ");
377  sqlConsultaEtapas.append(" FROM PROCESSO_ETAPA PE ");
378  sqlConsultaEtapas.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
379  sqlConsultaEtapas.append(" WHERE PE.IDE_STATUS IN ('A') ");
380  sqlConsultaEtapas.append(" AND P.COD_FORM = ");
381  sqlConsultaEtapas.append(codFormAnalise);
382  sqlConsultaEtapas.append(" AND PE.COD_ETAPA IN ( ");
383  sqlConsultaEtapas.append(codEtapasAnalise);
384  sqlConsultaEtapas.append(" )");
385  if ( !transferirRobo ) {
386  sqlConsultaEtapas.append(" AND PE.COD_USUARIO_ETAPA IN ( ");
387  sqlConsultaEtapas.append(codusuarioAutomatico);
388  sqlConsultaEtapas.append(" )");
389  }
390  sqlConsultaEtapas.append(" ORDER BY PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO ");
391  try (PreparedStatement pstConsultaEtapas = cnLecom.prepareStatement(sqlConsultaEtapas.toString());
392  ResultSet rsConsultaEtapas = pstConsultaEtapas.executeQuery()); {
393  while ( rsConsultaEtapas.next()) {
394  Integer codProcesso = rsConsultaEtapas.getInt("COD_PROCESSO");
395  Integer codEtapas = rsConsultaEtapas.getInt("COD_ETAPA");
396  Integer codCiclo = rsConsultaEtapas.getInt("COD_CICLO");
397  String modoTeste = rsConsultaEtapas.getString("IDE_BETA_TESTE");
398  Logger.info("Proc / Etapa : ( " + codProcesso + " / " + codEtapas + " )");
399  if ( transferirRobo ) {
400  if ( !verificaUsuarioProcessoEtapasUsou( cnLecom, codProcesso, codEtapas, codCiclo, codusuarioAutomatico ) ) {
401  inserirUsuarioEtapas( cnLecom, codProcesso, codEtapas, codCiclo, codusuarioAutomatico );
402  }
403  }
404  if ( formNovo ) {
405  Logger.info("APROVACOES - RETORNO FN : Proc / Etapa - ( " + codProcesso + " / " + codEtapas + " ) - " + executaAtividProcessoFormNovo(codProcesso.toString(), codEtapas.toString());
406  }
407  else {
408  String valores = "OBSERVACAO|Processo aprovado automaticamente.";
409  String valores = "";
410  Logger.info("APROVACOES - RETORNO : Proc / Etapa - ( " + codProcesso + " / " + codEtapas + " ) - " + executaEtapasProcesso(rsConsultaEtapas.getInt("COD_FORM"), codProcesso, codEtapas, codCiclo, modoTeste);
411  }
412  }
413  }
414  }
415  }
416  }
417  }
418  }
419  }
420  }
421  }
422  }
423  }
424  }
425  }
426  }
427  }
428  }
429  }
430  }
431  }
432  }
433  }
434  }
435  }
436  }
437  }
438  }
439  }
440  }
441  }
442  }
443  }
444  }
445  }
446  }
447  }
448  }
449  }
450  }
451  }
452  }
453  }
454  }
455  }
456  }
457  }
458  }
459  }
460  }
461  }
462  }
463  }
464  }
465  }
466  }
467  }
468  }
469  }
470  }
471  }
472  }
473  }
474  }
475  }
476  }
477  }
478  }
479  }
480  }
481  }
482  }
483  }
484  }
485  }
486  }
487  }
488  }
489  }
490  }
491  }
492  }
493  }
494  }
495  }
496  }
497  }
498  }
499  }
500  }
501  }
502  }
503  }
504  }
505  }
506  }
507  }
508  }
509  }
510  }
511  }
512  }
513  }
514  }
515  }
516  }
517  }
518  }
519  }
520  }
521  }
522  }
523  }
524  }
525  }
526  }
527  }
528  }
529  }
530  }
531  }
532  }
533  }
534  }
535  }
536  }
537  }
538  }
539  }
540  }
541  }
542  }
543  }
544  }
545  }
546  }
547  }
548  }
549  }
550  }
551  }
552  }
553  }
554  }
555  }
556  }
557  }
558  }
559  }
560  }
561  }
562  }
563  }
564  }
565  }
566  }
567  }
568  }
569  }
570  }
571  }
572  }
573  }
574  }
575  }
576  }
577  }
578  }
579  }
580  }
581  }
582  }
583  }
584  }
585  }
586  }
587  }
588  }
589  }
590  }
591  }
592  }
593  }
594  }
595  }
596  }
597  }
598  }
599  }
600  }
601  }
602  }
603  }
604  }
605  }
606  }
607  }
608  }
609  }
610  }
611  }
612  }
613  }
614  }
615  }
616  }
617  }
618  }
619  }
620  }
621  }
622  }
623  }
624  }
625  }
626  }
627  }
628  }
629  }
630  }
631  }
632  }
633  }
634  }
635  }
636  }
637  }
638  }
639  }
640  }
641  }
642  }
643  }
644  }
645  }
646  }
647  }
648  }
649  }
650  }
651  }
652  }
653  }
654  }
655  }
656  }
657  }
658  }
659  }
660  }
661  }
662  }
663  }
664  }
665  }
666  }
667  }
668  }
669  }
670  }
671  }
672  }
673  }
674  }
675  }
676  }
677  }
678  }
679  }
680  }
681  }
682  }
683  }
684  }
685  }
686  }
687  }
688  }
689  }
690  }
691  }
692  }
693  }
694  }
695  }
696  }
697  }
698  }
699  }
700  }
701  }
702  }
703  }
704  }
705  }
706  }
707  }
708  }
709  }
710  }
711  }
712  }
713  }
714  }
715  }
716  }
717  }
718  }
719  }
720  }
721  }
722  }
723  }
724  }
725  }
726  }
727  }
728  }
729  }
730  }
731  }
732  }
733  }
734  }
735  }
736  }
737  }
738  }
739  }
740  }
741  }
742  }
743  }
744  }
745  }
746  }
747  }
748  }
749  }
750  }
751  }
752  }
753  }
754  }
755  }
756  }
757  }
758  }
759  }
760  }
761  }
762  }
763  }
764  }
765  }
766  }
767  }
768  }
769  }
770  }
771  }
772  }
773  }
774  }
775  }
776  }
777  }
778  }
779  }
780  }
781  }
782  }
783  }
784  }
785  }
786  }
787  }
788  }
789  }
790  }
791  }
792  }
793  }
794  }
795  }
796  }
797  }
798  }
799  }
800  }
801  }
802  }
803  }
804  }
805  }
806  }
807  }
808  }
809  }
810  }
811  }
812  }
813  }
814  }
815  }
816  }
817  }
818  }
819  }
820  }
821  }
822  }
823  }
824  }
825  }
826  }
827  }
828  }
829  }
830  }
831  }
832  }
833  }
834  }
835  }
836  }
837  }
838  }
839  }
840  }
841  }
842  }
843  }
844  }
845  }
846  }
847  }
848  }
849  }
850  }
851  }
852  }
853  }
854  }
855  }
856  }
857  }
858  }
859  }
860  }
861  }
862  }
863  }
864  }
865  }
866  }
867  }
868  }
869  }
870  }
871  }
872  }
873  }
874  }
875  }
876  }
877  }
878  }
879  }
880  }
881  }
882  }
883  }
884  }
885  }
886  }
887  }
888  }
889  }
890  }
891  }
892  }
893  }
894  }
895  }
896  }
897  }
898  }
899  }
900  }
901  }
902  }
903  }
904  }
905  }
906  }
907  }
908  }
909  }
910  }
911  }
912  }
913  }
914  }
915  }
916  }
917  }
918  }
919  }
920  }
921  }
922  }
923  }
924  }
925  }
926  }
927  }
928  }
929  }
930  }
931  }
932  }
933  }
934  }
935  }
936  }
937  }
938  }
939  }
940  }
941  }
942  }
943  }
944  }
945  }
946  }
947  }
948  }
949  }
950  }
951  }
952  }
953  }
954  }
955  }
956  }
957  }
958  }
959  }
960  }
961  }
962  }
963  }
964  }
965  }
966  }
967  }
968  }
969  }
970  }
971  }
972  }
973  }
974  }
975  }
976  }
977  }
978  }
979  }
980  }
981  }
982  }
983  }
984  }
985  }
986  }
987  }
988  }
989  }
990  }
991  }
992  }
993  }
994  }
995  }
996  }
997  }
998  }
999  }
1000 }
```

Na imagem acima temos o print do método **executarAtividades** que irá utilizar os parâmetros explicado nas páginas 2 e 5, os parâmetros passados para esses métodos são:

- Conexão (a conexão que foi instanciada na linha 51)
- se é formulário novo (é passado true ou false para verificar se está realizando as aprovações dos formulários antigos ou novos)
- o valor dos parâmetros (aprovacoes, rejeicoes, aprovacoesFN, aprovacoesFNUC, rejeicoesFN, rejeicoesFNUC)
- código de usuário automatico (variável instanciada na linha 56)
- login do usuário automatico (variável instanciada na linha 57)
- senha do usuário automático (variável instanciada na linha 58)
- parâmetro da ação (P se for aprovação, R se for rejeição)
- url do BPM (variável instanciada na linha 60)
- se irá realizar transferência para o usuário robô (é passado true se estiver utilizando os parâmetros aprovacoesFNConcentradoraUC e rejeicoesFNConcentradoraUC se não vem false)

No restante do método é feito um select para identificar todos os processos do formulário configurado que estão parados na atividade configurada e com ou não o usuário automatico, a partir desses processos retornados ele realiza a aprovação ou rejeição da etapa, indo para o método do formulário antigo ou do formulário novo dependendo do parâmetro que veio, também verifica se tem o true para a transferência do usuário automatico e assim realiza a chamada do método que faz essa parte de inserção da permissão para que o usuário possa aprovar uma atividade que não esteja com ele.

```

438
439 private void executarAtividadesDataCampo( Connection cnLecom, boolean formNovo, Calendar dataAtual, String execucaoDataCampo, Integer codusuarioAutomatico, String loginusuarioAutomatico,
440 String senhausuarioAutomatico, String paramacaoAprovaRejeita, String urlBPM, boolean transferirRobo ) throws Exception{
441
442     Logger.info("INICIO executarAtividadesDataCampo");
443     Logger.info("execucaoDataCampo : " + execucaoDataCampo);
444
445     if ( !"".equals(execucaoDataCampo) ){
446
447         for (String paramExec : execucaoDataCampo.split(";")) {
448
449             String[] paramFormEtapa = paramExec.split("@");
450
451             Integer codFormAnalise = new Integer(paramFormEtapa[0]);
452             Integer codEtapaAnalise = new Integer(paramFormEtapa[1]);
453             String nomeCampoAnalise = paramFormEtapa[2];
454             String nomeTabelaModelo = paramFormEtapa[3];
455             Logger.info("APROVACOES DATA CAMPO - Em analise - Form : " + codFormAnalise + " / Etapa : " + codEtapaAnalise + " / Campos : " + nomeCampoAnalise);
456
457             Map<String, String> paramFormModelo = Funcoes.getParametrosIntegracao(configPath + String.format("modelo_%1$s.properties", codFormAnalise));
458             String tableName = paramFormModelo.get("table_name");
459
460             // Pega todos os processos que estão na etapa "Aguarda_data_modific"
461             StringBuilder sqlConsultaEtapas = new StringBuilder();
462             sqlConsultaEtapas.append(" SELECT P.COD_FORM, PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO, P.IDE_BETA_TESTE, F.");
463             sqlConsultaEtapas.append(" FROM PROCESSO_ETAPA PE ");
464             sqlConsultaEtapas.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
465             sqlConsultaEtapas.append(" INNER JOIN ");
466             sqlConsultaEtapas.append(" ON ( PE.COD_PROCESSO = F.COD_PROCESSO_F AND PE.COD_ETAPA = F.COD_ETAPA_F AND PE.COD_CICLO = F.COD_CICLO_F ) ");
467             sqlConsultaEtapas.append(" WHERE PE.IDE_STATUS = 'A' ");
468             sqlConsultaEtapas.append(" AND P.COD_FORM = ");
469             sqlConsultaEtapas.append(codFormAnalise);
470             sqlConsultaEtapas.append(" AND PE.COD_ETAPA = ");
471             sqlConsultaEtapas.append(codEtapaAnalise);
472
473             if ( !transferirRobo ) {
474                 sqlConsultaEtapas.append(" AND PE.COD_USUARIO_ETAPA IN ( ");
475                 sqlConsultaEtapas.append(codusuarioAutomatico);
476                 sqlConsultaEtapas.append(" ) ");
477             }
478
479             sqlConsultaEtapas.append(" ");
480
481             sqlConsultaEtapas.append(" AND F.");
482             sqlConsultaEtapas.append(nomeCampoAnalise);
483             sqlConsultaEtapas.append(" IS NOT NULL ");
484
485             // logger.info("sqlConsultaEtapas : " + sqlConsultaEtapas);
486
487             try (PreparedStatement pstConsultaEtapas = cnLecom.prepareStatement(sqlConsultaEtapas.toString());
488                 ResultSet rsConsultaEtapas = pstConsultaEtapas.executeQuery()) {
489
490                 while (rsConsultaEtapas.next()) {
491
492                     Integer codProcesso = rsConsultaEtapas.getInt("COD_PROCESSO");
493                     Integer codEtapa = rsConsultaEtapas.getInt("COD_ETAPA");
494                     Integer codCiclo = rsConsultaEtapas.getInt("COD_CICLO");
495                     String modoTeste = rsConsultaEtapas.getString("IDE_BETA_TESTE");
496                     Calendar dataReferencia = DateToCalendar(rsConsultaEtapas.getDate(nomeCampoAnalise));
497
498                     // logger.info("codProcesso : " + codProcesso);
499                     // logger.info("codEtapa : " + codEtapa);
500                     // logger.info("datReferencia : " + dataReferencia);
501
502                     // Se a data Referencia for igual ou inferior a data atual, então executa
503                     if (dataReferencia.compareTo(dataAtual) <= 0) {
504
505                         if ( transferirRobo ) {
506                             if ( !verificarUsuarioProcessoEtapaUsu( cnLecom, codProcesso, codEtapa, codCiclo, codusuarioAutomatico ) ) {
507                                 inserirUsuarioEtapa( cnLecom, codProcesso, codEtapa, codCiclo, codusuarioAutomatico );
508                             }
509                         }
510
511                         if ( formNovo ) {
512                             Logger.info("APROVACOES - RETORNO FN : Proc / Etapa - ( " + codProcesso + " / " + codEtapa + " ) - " + execucaoDataCampoFormNovo(codProcesso.toString(), codEtapa.toString(),
513                             String valores = "";
514                             Logger.info("APROVACOES EXECUTA ETAPAS DATA CAMPO - RETORNO : Proc / Etapa - ( " + codProcesso + " / " + codEtapa + " ) - " + execucaoDataCampoFormNovo(rsConsultaEtapas.getInt(
515                             }
516                         }
517                     }
518                 }
519             }
520         }
521     }

```

Na imagem acima temos o print do método **executarAtividadesDataCampo** que irá utilizar os parâmetros explicado nas páginas 3,7 e 8, os parâmetros passados para esses métodos são:

- Conexão (a conexão que foi instanciada na linha 51)
- se é formulário novo (é passado true ou false para verificar se está realizando as aprovações dos formulários antigos ou novos)
- o valor dos parâmetros (aprovacoesDataCampo, rejeicoesDataCampo, aprovacoesFNDDataCampo, aprovacoesFNDDataCampoUC, rejeicoesFNDDataCampo, rejeicoesFNDDataCampoUC)
- código de usuário automatico (variável instanciada na linha 56)
- login do usuário automatico (variável instanciada na linha 57)

- senha do usuário automático (variável instanciada na linha 58)
- parâmetro da ação (P se for aprovação, R se for rejeição)
- url do BPM (variável instanciada na linha 60)
- se irá realizar transferência para o usuário robô (é passado true se estiver utilizando os parâmetros aprovacoesFNConcentradoraUC e rejeicoesFNConcentradoraUC se não vem false)

No restante do método é feito um select para identificar todos os processos do formulário configurado que estão parados na atividade configurada e com ou não o usuário automatico pegando também da tabela do modelo o campo configurado e verificando se o campo tem valor, a partir desses processos retornados ele verifica que se o valor do campo configurado é igual a data de hoje ai ele realiza a aprovação ou rejeição da etapa, indo para o método do formulário antigo ou do formulário novo dependendo do parâmetro que veio, também verifica se tem o true para a transferência do usuário automatico e assim realiza a chamada do método que faz essa parte de inserção da permissão para que o usuário possa aprovar uma atividade que não esteja com ele.

```

540 private void executarAtividadesPrazoMaximoExcedido( Connection cnLecom, boolean formNovo, Calendar dataAtual, String paramProcExecutar, Integer codUsuarioAutomatico,
541 String loginUsuarioAutomatico, String senhaUsuarioAutomatico, String paramacaoaprovaRejeita, String urlBPM, boolean transferirRobo ) throws Excepti
542
543 logger.info("INICIO aprovarEtapasPrazoMaximoExcedido");
544 logger.info("aprovacoesEtapasPrazoMaximoExcedido : " + paramProcExecutar);
545 if( !"".equals(paramProcExecutar) ){
546 for (String paramProc : paramProcExecutar.split(";")) {
547 String[] paramFormEtapa = paramProc.split("@");
548 // <cod form>@<cod etapa>@<nome do campo observacao>@<mensagem para registrar execução automática>;
549 Integer codFormAnalise = new Integer(paramFormEtapa[0]);
550 Integer codFormAnalise = new Integer(paramFormEtapa[1]);
551 String nomeCampoObservacao = "";
552 String mensagemExecAutomatica = "";
553 if( paramFormEtapa.length > 2 ){
554 nomeCampoObservacao = paramFormEtapa[2];
555 mensagemExecAutomatica = paramFormEtapa[3];
556 }
557 logger.info("APROVA ETAPAS PRAZO MAXIMO EXCEDIDO - Em analise - Form : " + codFormAnalise + " / Etapa : " + codFormAnalise + " / Campo Observação : " + nomeCampoObservacao + " / Mensagem
558 StringBulder sqlConsultaAtividades = new StringBulder();
559 sqlConsultaAtividades.append(" SELECT P.COD_FORM, PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO, E.COD_TIPO_ETAPA, PE.DAT_LIMITE, P.IDE_BETA_TESTE ");
560 sqlConsultaAtividades.append(" FROM PROCESSO_ETAPA PE ");
561 sqlConsultaAtividades.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
562 sqlConsultaAtividades.append(" WHERE P.IDE_STATUS IN ('A') ");
563 sqlConsultaAtividades.append(" AND PE.DAT_LIMITE IS NOT NULL ");
564 sqlConsultaAtividades.append(" AND P.COD_FORM = ");
565 sqlConsultaAtividades.append(codFormAnalise);
566 sqlConsultaAtividades.append(" AND PE.COD_ETAPA IN ( ");
567 sqlConsultaAtividades.append(codFormAnalise);
568 sqlConsultaAtividades.append(" ) ");
569 if ( !transferirRobo ) {
570 sqlConsultaAtividades.append(" AND PE.COD_USUARIO_ETAPA IN ( ");
571 sqlConsultaAtividades.append(codUsuarioAutomatico);
572 sqlConsultaAtividades.append(" ) ");
573
574 sqlConsultaAtividades.append(" ORDER BY PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO ");
575 logger.info("sqlConsultaAtividades : " + sqlConsultaAtividades);
576 try ( PreparedStatement pstConsultaEtapas = cnLecom.prepareStatement(sqlConsultaAtividades.toString());
577 ResultSet rsConsultaEtapas = pstConsultaEtapas.executeQuery(); ) {
578 while ( rsConsultaEtapas.next() ) {
579 Integer codProcesso = rsConsultaEtapas.getInt("COD_PROCESSO");
580 Integer codEtapa = rsConsultaEtapas.getInt("COD_ETAPA");
581 Integer codCiclo = rsConsultaEtapas.getInt("COD_CICLO");
582 String modoteste = rsConsultaEtapas.getString("IDE_BETA_TESTE");
583 Calendar dataReferencia = DateToCalendar(rsConsultaEtapas.getDate("DAT_LIMITE"));
584 // Se o tipo de etapa for 1 = Inicial, a ação precisa ser de cancelamento
585 Integer codTipoEtapa = rsConsultaEtapas.getInt("COD_TIPO_ETAPA");
586 if ( !"R".equalsIgnoreCase(paramacaoaprovaRejeita) && codTipoEtapa.compareTo(new Integer(1)) == 0 ){
587 paramacaoaprovaRejeita = "C";
588 logger.info("nova ação : " + paramacaoaprovaRejeita);
589 }
590 logger.info("codProcesso : " + codProcesso);
591 logger.info("codEtapa : " + codEtapa);
592 logger.info("dataReferencia : " + dataReferencia);
593 // Se a data Referencia for igual ou inferior a data atual, então execute
594 if (dataReferencia.compareTo(dataAtual) <= 0) {
595 if ( transferirRobo ) {
596 if ( !verificaUsuarioProcessoEtapaUsu( cnLecom, codProcesso, codEtapa, codCiclo, codUsuarioAutomatico ) ) {
597 inserirUsuarioEtapa( cnLecom, codProcesso, codEtapa, codCiclo, codUsuarioAutomatico );
598 }
599 }
600 if( formNovo ) {
601 Map<String,String> camposValores = new HashMap<String, String>();
602 if ( !"".equals(nomeCampoObservacao) ) {
603 camposValores.put(nomeCampoObservacao, mensagemExecAutomatica);
604 }
605 logger.info("APROVACOES - RETORNO FN : Proc / Etapa - ( " + codProcesso + " / " + codEtapa + " ) - " + executarAtividadeFormNovo(codProcesso.toString(), codEtapa.toStr

```


Na imagem acima temos o print do método **executarAtividadesPrazoMaximoExcedido** que irá utilizar os parâmetros explicado nas páginas 4, 9 e 10, os parâmetros passados para esses métodos são:

- Conexão (a conexão que foi instanciada na linha 51)
- se é formulário novo (é passado true ou false para verificar se está realizando as aprovações dos formulários antigos ou novos)
- o valor dos parâmetros (aprovacoesEtapasPrazoMaximoExcedido, rejeicoesEtapasPrazoMaximoExcedido, aprovacoesFNEtapasPrazoMaximoExcedido, aprovacoesFNEtapasPrazoMaximoExcedidoUC, rejeicoesFNEtapasPrazoMaximoExcedido, rejeicoesFNEtapasPrazoMaximoExcedidoUC)
- código de usuário automatico (variável instanciada na linha 56)
- login do usuário automatico (variável instanciada na linha 57)
- senha do usuário automático (variável instanciada na linha 58)
- parâmetro da ação (P se for aprovação, R se for rejeição)
- url do BPM (variável instanciada na linha 60)
- se irá realizar transferência para o usuário robô (é passado true se estiver utilizando os parâmetros aprovacoesFNConcentradoraUC e rejeicoesFNConcentradoraUC se não vem false)

No restante do método é feito um select para identificar todos os processos do formulário configurado que estão parados na atividade configurada e com ou não o usuário automatico, a partir desses processos retornados ele verifica se a data limite da atividade é igual a data de hoje ai ele realiza a aprovação ou rejeição da atividade, indo para o método do formulário antigo ou do formulário novo dependendo do parâmetro que veio, também verifica se tem o true para a transferência do usuário automatico e assim realiza a chamada do método que faz essa parte de inserção da permissão para que o usuário possa aprovar uma atividade que não esteja com ele.

```
RoboExecutarAtividades.java RoboExecutarAtividades.properties automatico.properties
727 private void cancelarProcessosNaoEnviado( Connection cnLecom, LocalDate dataAtual, String atividadesExecutar, Integer codusuarioAutomatico, String urlBPM, boolean transferirRobo ) throws Exception {
728
729     logger.info("INICIO cancelarProcessosNaoEnviado");
730
731     logger.info("atividadesExecutar : " + atividadesExecutar);
732
733     if ( !"".equals(atividadesExecutar) ){
734
735         for ( String atividadeExecut : atividadesExecutar.split(";") ) {
736
737             String[] paramFormEtapa = atividadeExecut.split("&");
738
739             Integer codFormAnalise = new Integer(paramFormEtapa[0]);
740             Integer auxQtddDiasAguardar = new Integer(paramFormEtapa[1]);
741             int qtddDiasAguardar = -auxQtddDiasAguardar; // Deixa o número negativo
742
743             logger.info("PARAMETROS - Em análise - Form : " + codFormAnalise + " / qtddDiasAguardar : " + qtddDiasAguardar);
744
745             StringBuilder sqlConsultaEtapas = new StringBuilder();
746             sqlConsultaEtapas.append(" SELECT P.COD_FORM, PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO, P.IDE_BETA_TESTE, PE.DAT_GRAVACAO ");
747             sqlConsultaEtapas.append(" FROM PROCESSO.ETAPA PE ");
748             sqlConsultaEtapas.append(" INNER JOIN PROCESSO P ON ( P.COD_PROCESSO = PE.COD_PROCESSO ) ");
749             sqlConsultaEtapas.append(" INNER JOIN ETAPA E ON ( E.COD_FORM = P.COD_FORM AND E.COD_VERSAO = P.COD_VERSAO AND E.COD_ETAPA = PE.COD_ETAPA ) ");
750             sqlConsultaEtapas.append(" WHERE PE.IDE_STATUS = 'A' ");
751             sqlConsultaEtapas.append(" AND PE.IDE_TEMPORARIO = 'N' ");
752             sqlConsultaEtapas.append(" AND E.COD_TIPO_ETAPA = 1 ");
753             sqlConsultaEtapas.append(" AND PE.COD_CICLO = 1 ");
754             sqlConsultaEtapas.append(" AND P.COD_FORM = ");
755             sqlConsultaEtapas.append(codFormAnalise);
756             sqlConsultaEtapas.append(" ORDER BY PE.COD_PROCESSO, PE.COD_ETAPA, PE.COD_CICLO ");
757
758             try { PreparedStatement pstConsultaEtapas = cnLecom.prepareStatement(sqlConsultaEtapas.toString());
759                 ResultSet rsConsultaEtapas = pstConsultaEtapas.executeQuery(); }
760
761             while ( rsConsultaEtapas.next() ) {
762
763                 Integer codProcesso = rsConsultaEtapas.getInt("COD_PROCESSO");
764                 Integer codEtapa = rsConsultaEtapas.getInt("COD_ETAPA");
765                 Integer codCiclo = rsConsultaEtapas.getInt("COD_CICLO");
766                 String modoTeste = rsConsultaEtapas.getString("IDE_BETA_TESTE");
767                 Calendar dataAbertura = DateToCalendar(rsConsultaEtapas.getDate("DAT_GRAVACAO"));
768                 // LocalDateTime dataAbertura = LocalDateTime.parse( rsConsultaEtapas.getString("DAT_GRAVACAO") , DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.S"));
769                 LocalDateTime dataAbertura = LocalDateTime.parse( rsConsultaEtapas.getString("DAT_GRAVACAO") , DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.S"));
770                 logger.info("dataAbertura : " + dataAbertura);
771
772                 // calcula a diferença de dias entre as duas datas / Se a qtdd vier negativa, quer dizer q a data analisada é inferior a atual, se for positiva, a data analisada é superior a atual
773                 long difDiasDataAbertura = ChronoUnit.DAYS.between(dataAtual, dataAbertura);
774                 logger.info("codProcesso / difDiasDataAbertura : " + codProcesso + " / " + difDiasDataAbertura);
775
776                 // Se a data Referência for igual ou inferior a data atual, então execute
777                 if( qtddDiasAguardar > difDiasDataAbertura ) {
778
779                     if ( transferirRobo ) {
780                         if ( !verificarUsuarioProcessoEtapaUsu( cnLecom, codProcesso, codEtapa, codCiclo, codusuarioAutomatico ) ) {
781                             inserirUsuarioEtapa ( cnLecom, codProcesso, codEtapa, codCiclo, codusuarioAutomatico );
782                         }
783                     }
784
785                     logger.info("CANCELAR PROCESSO - RETORNO FN : Proc / Etapa - ( " + codProcesso + " / " + codEtapa + " ) - " + cancelarProcessosFormMovto(codProcesso.toString(), codEtapa.toString());
786
787                     } else {
788                         logger.debug("Aguardar!");
789                     }
790
791                 }
792             }
793         }
794     }
795     logger.info("FIM cancelarProcessosNaoEnviado");
796 }
797
798 }
```

Na imagem acima temos o print do método **cancelarProcessosNaoEnviado** que irá utilizar os parâmetros explicado nas páginas 10, os parâmetros passados para esses métodos são:

- Conexão (a conexão que foi instanciada na linha 51)
- Data atual
- o valor dos parâmetros (cancelarFNProcessosNaoEnviado, cancelarFNProcessosNaoEnviadoUC)
- código de usuário automatico (variável instanciada na linha 56)
- url do BPM (variável instanciada na linha 60)
- se irá realizar transferência para o usuário robô (é passado true se estiver utilizando os parâmetros aprovacoesFNConcentradoraUC e rejeicoesFNConcentradoraUC se não vem false)

No restante do método é feito um select para identificar todos os processos do formulário configurado que estão parados na atividade inicial no ciclo 1, a partir desses processos retornados ele verifica se o valor do parâmetro de dias configurado for maior que a diferença entre a data de abertura do processo e a data atual ai ele realiza a aprovação ou rejeição da atividade, indo para o método do formulário antigo ou do formulário novo dependendo do parâmetro que veio, também verifica se tem o true para a transferência do usuário

automatico e assim realiza a chamada do método que faz essa parte de inserção da permissão para que o usuário possa aprovar uma atividade que não esteja com ele.

```

229     return retornoAprovacao;
230 }
231
232
233
234 private String cancelaProcessoFormNovo( String codProcessoExecutar, String codAtividadeExecutar, String codCicloExecutar, String modoTeste,
235 Integer codUsuarioAutomatico, String urlBPM ) throws Exception {
236
237     String retornoCancelamento = "falha";
238
239     try {
240         Logger.info("INICIO retornoCancelamento");
241
242         DadosProcessoAbertura procOrigemUtil = new DadosProcessoAbertura();
243         procOrigemUtil.setProcessInstanceId(codProcessoExecutar);
244         procOrigemUtil.setCurrentActivityInstanceId(codAtividadeExecutar);
245         procOrigemUtil.setCurrentCycle(codCicloExecutar);
246         procOrigemUtil.setModoTeste(modosTeste.equals("S") ? "true" : "false");
247
248         CancelarProcesso cancelar = new CancelarProcesso(urlBPM, accessToken38550, procOrigemUtil, procOrigemUtil.getModoTeste(), codUsuarioAutomatico.toString());
249         retornoCancelamento = cancelar.cancelarProcesso();
250         Logger.debug("retornoCancelamento = " + retornoCancelamento);
251     } catch ( CancelarProcessoException e1 ) {
252         Logger.error("ERRO CancelaProcessoException : ", e1);
253     }
254
255     Logger.info("FIM retornoCancelamento");
256     return retornoCancelamento;
257 }
258
259
260

```

Na imagem acima temos o print do método **cancelaProcessoFormNovo** que irá realizar a ação de cancelamento do processos, recebendo os seguintes parâmetros:

- código do processo
- código da atividade
- código do ciclo
- se está no modo teste ou não o processo (S ou N)
- código do usuário automatico
- url do BPM

Nesse método é utilizada classes do jar RotasBPM, que é utilizado para ações do formulário novo, esse projeto foi desenvolvido pela nossa equipe de consultoria para utilizar as apis disponibilizadas pelo produto, sobre esse tema você pode ver mais afundo no documento RotasBPM, nesse método ele utiliza as classes de DadosProcessoAbertura para preenchimento das informações de código de processo, etapa, ciclo e modo teste, que são essenciais para que a classe CancelarProcesso funcione e realize o cancelamento automatico do processo.

```

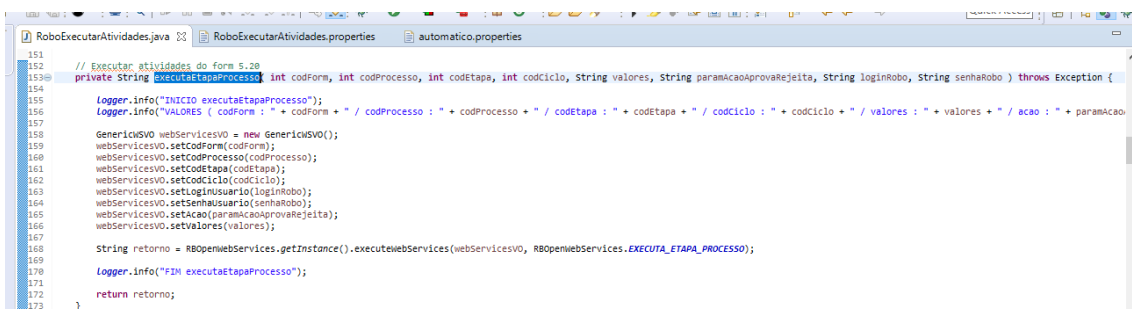
187     @return String retornoAprovacao
188     @throws Exception
189     /
190 private String executaAtivProcessoFormNovo( String codProcessoExecutar, String codAtividadeExecutar, String codCicloExecutar, String modoTeste,
191 Map<String,String> camposValores, Map<String,List<Map<String, Object>>> gridValores, Integer codUsuarioAutomatico,
192 String loginUsuarioAutomatico, String senhaUsuarioAutomatico, String paramacaoAprovarRejeita, String urlBPM ) throws Exception {
193
194     Logger.info("INICIO executaAtivProcessoFormNovo");
195     Logger.info("VALORES ( AccessTokens38550 : " + accessToken38550 + " / codProcessoExecutar : " + codProcessoExecutar + " / codAtividadeExecutar : " + codAtividadeExecutar + " / codCicloExecutar : " + codCicloExecutar + " / codUsuarioAutomatico : " + codUsuarioAutomatico + " / loginUsuarioAutomatico : " + loginUsuarioAutomatico + " / senhaUsuarioAutomatico : " + senhaUsuarioAutomatico );
196     Logger.info("VALORES ( codUsuarioAutomatico : " + codUsuarioAutomatico + " / loginUsuarioAutomatico : " + loginUsuarioAutomatico + " / senhaUsuarioAutomatico : " + senhaUsuarioAutomatico );
197
198     String retornoAprovacao = "";
199
200     try {
201         DadosProcesso dadosProcesso = new DadosProcesso(paramacaoAprovarRejeita);
202
203         if ( camposValores != null ) {
204             dadosProcesso.geraPadroes(camposValores);
205         }
206
207         if ( gridValores != null ) {
208             gridValores.forEach( (nomeGrid, valores) -> dadosProcesso.geraValoresGrid(nomeGrid, valores) );
209         }
210
211         DadosProcessoAbertura procOrigemUtil = new DadosProcessoAbertura();
212         procOrigemUtil.setProcessInstanceId(codProcessoExecutar);
213         procOrigemUtil.setCurrentActivityInstanceId(codAtividadeExecutar);
214         procOrigemUtil.setCurrentCycle(codCicloExecutar);
215         procOrigemUtil.setModoTeste(modosTeste.equals("S") ? "true" : "false");
216
217         Logger.debug("procOrigemUtil Info: " + procOrigemUtil.getCurrentActivityInstanceId() + " / " + procOrigemUtil.getCurrentCycle() + " / " + procOrigemUtil.getProcessInstanceId() + " / modoTeste: " + modoTeste );
218         AprovaProcesso aprovaProcesso = new AprovaProcesso( urlBPM, accessToken38550, procOrigemUtil, dadosProcesso, procOrigemUtil.getModoTeste(), codUsuarioAutomatico.toString() );
219         retornoAprovacao = aprovaProcesso.aprovaProcesso();
220         Logger.debug("retornoAprovacao = " + retornoAprovacao);
221
222     } catch ( Exception e ) {
223         Logger.error("executaAtivProcessoFormNovo : ", e);
224         retornoAprovacao = Funcoes.exceptionPrinter(e);
225     }
226
227     Logger.info("FIM executaAtivProcessoFormNovo");
228
229     return retornoAprovacao;
230

```

Na imagem acima temos o print do método **executaAtividProcessoFormNovo** que irá realizar a ação de aprovação/rejeição dos processos, recebendo os seguintes parâmetros:

- código do processo
- código da atividade
- código do ciclo
- se está no modo teste ou não o processo (S ou N)
- valores a serem preenchidos
- valores de grid
- código do usuário automatico
- login do usuário automatico
- senha do usuário automatico
- parâmetro de aprovação ou rejeição
- url do BPM

Nesse método é utilizada classes do jar RotasBPM, que é utilizado para ações do formulário novo, esse projeto foi desenvolvido pela nossa equipe de consultoria para utilizar as apis disponibilizadas pelo produto, sobre esse tema você pode ver mais afundo no documento RotasBPM, nesse método ele utiliza as classes de DadosProcessoAbertura para preenchimento das informações de código de processo, etapa, ciclo e modo teste, que são essenciais para que a classe AprovarProcesso funcione e realize a aprovação/rejeição automática do processo.



```

151 // Executar atividades do form 5.20
152 // private String executaAtividProcessoFormNovo(int codForm, int codProcesso, int codEtapa, int codCiclo, String valores, String paramAcaoAprovaRejeita, String loginRobo, String senhaRobo) throws Exception {
153 //
154 //
155 //
156 //
157 //
158 //
159 //
160 //
161 //
162 //
163 //
164 //
165 //
166 //
167 //
168 //
169 //
170 //
171 //
172 //
173 //

```

Na imagem acima temos o print do método **executaEtapaProcesso** que irá realizar a ação de aprovação/rejeição dos processos, recebendo os seguintes parâmetros:

- código do formulário
- código do processo
- código da atividade
- código do ciclo
- valores a serem preenchidos
- se está no modo teste ou não o processo (S ou N)
- login do usuário automático
- senha do usuário automático

Nesse método que é utilizado para realizar as ações de aprovação/rejeição dos processos que estão no formulário antigo, utilizando a biblioteca do produto chamada GenerecWSVO onde nela é setada as informações e passado esse objeto para o método de execução dos webservice que está dentro da instancia do RbOpenWebservice.

```
RoboExecutarAtividades.java RoboExecutarAtividades.properties automatico.properties
760 // Insere o usuário Robo na atividade para poder executar-la.
761 private void inserirUsuarioEtapa (Connection cnLecom, Integer codProcesso, Integer codEtapa, Integer codCiclo, Integer codUsuario ) throws Exception {
762     logger.info("=== INICIO inserirUsuarioEtapa ===");
763
764     StringBuilder processoEtapaUsu = new StringBuilder();
765     processoEtapaUsu.append(" INSERT INTO PROCESSO_ETAPA_USU ( COD_PROCESSO, COD_ETAPA, COD_CICLO, COD_USUARIO_ETAPA, IDE_VISUALIZADO ) ");
766     processoEtapaUsu.append(" VALUES ( ?, ?, ?, ?, 'N' ) ");
767
768     try(PreparedStatement pstInsert = cnLecom.prepareStatement(processoEtapaUsu.toString())) {
769
770         pstInsert.setInt(1, codProcesso);
771         pstInsert.setInt(2, codEtapa);
772         pstInsert.setInt(3, codCiclo);
773         pstInsert.setInt(4, codUsuario);
774
775         pstInsert.executeUpdate();
776         cnLecom.commit();
777
778     } catch (Exception e) {
779         logger.error("ERRO - inserirRoboEtapa", e);
780     }
781     logger.info("=== FIM inserirUsuarioEtapa ===");
782 }
```

Na imagem acima temos o print do método **inserirUsuarioEtapa** que irá inserir o usuário automatico na tabela responsável por gravar quem são os responsáveis da atividade, ele recebe os seguintes parâmetros:

- conexão do banco
- código do processo
- código da etapa
- código do ciclo
- código do usuário automatico

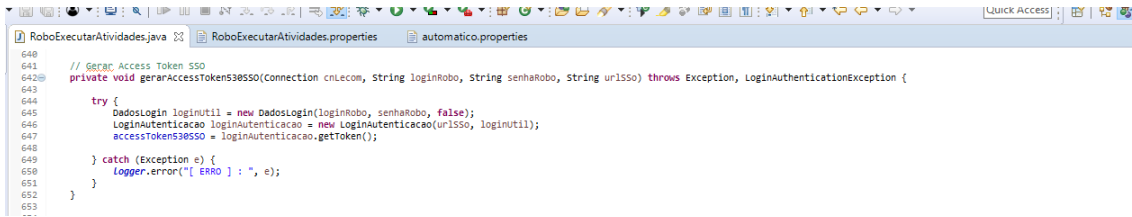
Nesse método é utilizado um insert direto no banco de dados até que tenhamos uma api do produto para isto.

```
RoboExecutarAtividades.java RoboExecutarAtividades.properties automatico.properties
664 }
665
666
667 private boolean verificaUsuarioProcessoEtapaUsu (Connection cnLecom, Integer codProcesso, Integer codEtapa, Integer codCiclo, Integer codUsuario ) throws Exception {
668
669     StringBuilder sql = new StringBuilder();
670     sql.append(" SELECT COUNT(*) AS total ");
671     sql.append(" FROM processo_etapa_usu ");
672     sql.append(" WHERE cod_processo = ? ");
673     sql.append(" AND cod_etapa = ? ");
674     sql.append(" AND cod_ciclo = ? ");
675     sql.append(" AND cod_usuario_etapa = ? ");
676
677     try (PreparedStatement pst = cnLecom.prepareStatement(sql.toString())) {
678         pst.setInt(1, codProcesso);
679         pst.setInt(2, codEtapa);
680         pst.setInt(3, codCiclo);
681         pst.setInt(4, codUsuario);
682
683         try (ResultSet rs = pst.executeQuery()) {
684             if (rs.next()) {
685                 int total = rs.getInt("total");
686
687                 if (total > 0) {
688                     return true;
689                 } else {
690                     return false;
691                 }
692             } else {
693                 return false;
694             }
695         }
696     }
697 }
698
699 }
```

Na imagem acima temos o print do método **verificaUsuarioProcessoEtapaUsu** que irá verificar se o usuário automatico está na tabela responsável por gravar quem são os responsáveis da atividade, ele recebe os seguintes parâmetros:

- conexão do banco
- código do processo
- código da etapa
- código do ciclo
- código do usuário automatico

Nesse método é utilizado um select que vai validar se o código do usuário existe para o processo, etapa, ciclo passado.



```
640
641 // gerar Access Token SSO
642 private void gerarAccessTokens530SSO(Connection cnLecom, String loginRobo, String senhaRobo, String urlSSO) throws Exception, LoginAuthenticationException {
643
644     try {
645         DadosLogin loginUtil = new DadosLogin(loginRobo, senhaRobo, false);
646         LoginAutenticacao loginAutenticacao = new LoginAutenticacao(urlSSO, loginUtil);
647         accessTokens530SSO = loginAutenticacao.getToken();
648     } catch (Exception e) {
649         Logger.error("[ ERRO ] : ", e);
650     }
651 }
652
653
```

Na imagem acima temos o print do método **gerarAccessTokens530SSO**, método utilizado para realizar o login por fora da ferramenta ele e assim ser utilizado o token retornado nos métodos de cancelamento e aprovação de atividades, ele recebe os seguintes parâmetros:

- conexão
- login do usuário automatico
- senha do usuário automatico
- url do sso

Nesse método é utilizada classes do jar RotasBPM, que é utilizado para ações do formulário novo, esse projeto foi desenvolvido pela nossa equipe de consultoria para utilizar as apis disponibilizadas pelo produto, sobre esse tema você pode ver mais a fundo no documento RotasBPM, ele utiliza o objeto DadosLogin para preencher as informações de login, senha do automatico e utiliza a classe LoginAutenticacao que fará o login por fora e retornará o token.