

iFlow

Criar um fluxo passo a passo

1. iFlowEditor – Bibliotecas e os seus principais Blocos

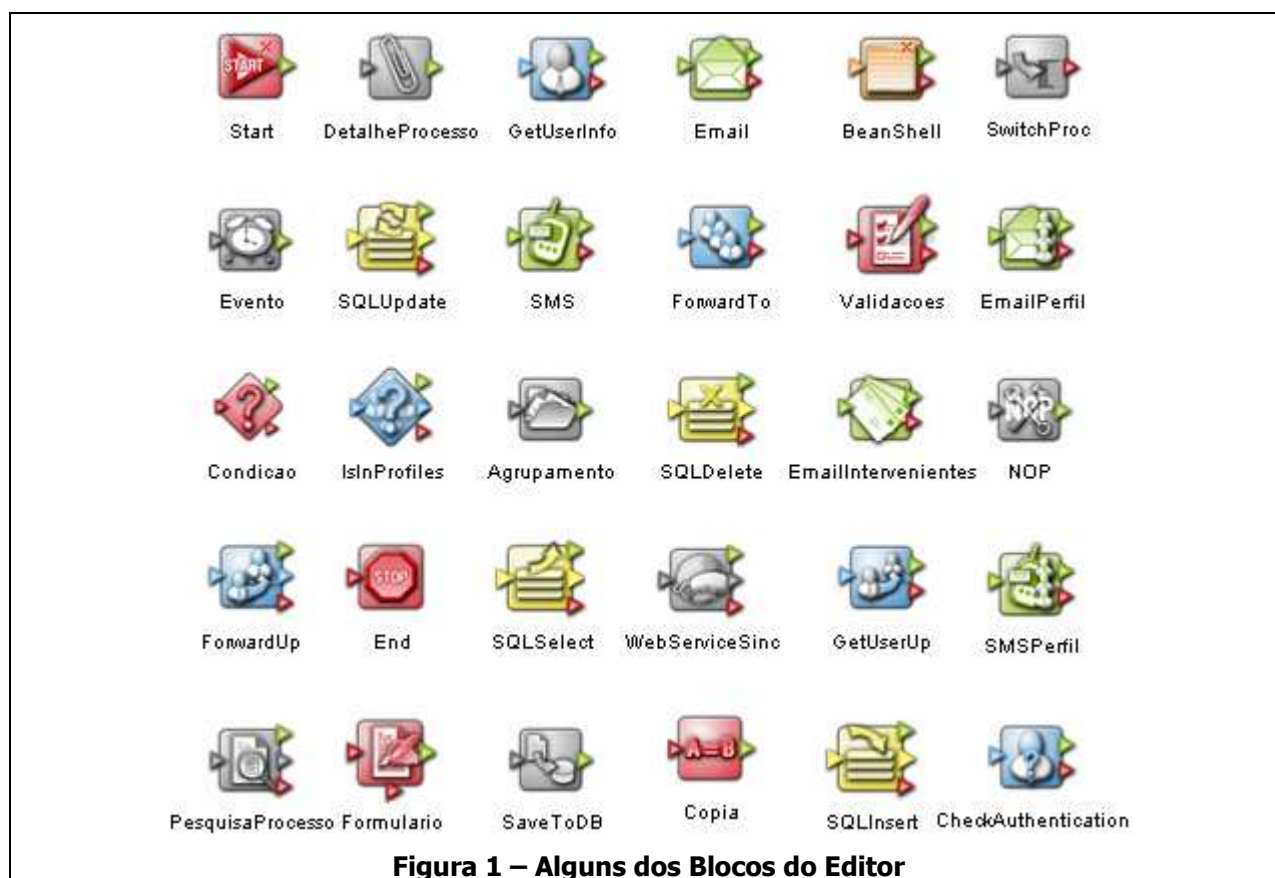


Figura 1 – Alguns dos Blocos do Editor

1.1. Biblioteca Básica

→ conjunto de blocos básicos, em média os mais usados num fluxo, desde os blocos de inicio e fim do fluxo, aos blocos para introdução de dados e validação dos mesmos, passando pelos blocos lógicos.



Start

O bloco **Start** é o bloco no qual todos os processos se iniciam. Nele é possível criar propriedades do fluxo (nome da propriedade e descrição), configuráveis à posteriori na

aplicação iFlow (área de Administração), que podem ser utilizadas nos diversos blocos. Essas propriedades podem ser simples (par nome/valor), funcionando na prática como uma variável do processo, ou listas. Neste caso, o nome da propriedade deve ter o prefixo “@”, como se ilustra na figura seguinte. De referir que, sendo este o ponto de partida para cada fluxo, só poderá existir um bloco de Start.

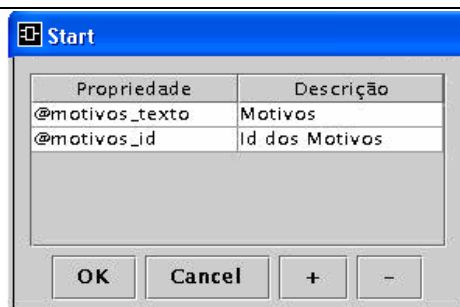


Figura 2 – Edição de Atributos do bloco Start



End

O bloco **End** serve para terminar o processo. Ao contrário do bloco de Start, pode existir mais que um bloco End. Este bloco não possui qualquer tipo de interacção.



Copia

O bloco de **Copia** serve para atribuir (ou criar caso ainda não existam) valores a variáveis do processo, definindo o nome da variável de destino e o seu valor, sendo este definido como string de Java (excepto quando se pretende atribuir um valor numérico, definido-o directamente). Na figura seguinte ilustram-se várias situações para a configuração deste bloco.



Figura 3 - Edição de Atributos do bloco Cópia

Assim sendo, é atribuído à variável “name” o texto “Name”, à variável “value” o valor e tipo (string ou double) que estiver na variável “startvalue”, os valores “1”, “2” e “3” nas posições

1, 2 e 3 respectivamente à variável (tipo lista) “array_test” e finalmente o número “13” à variável “numbervar”.



O bloco de **Validacoes** efectua as validações pretendidas, associando mensagens de erro a condições, tal como se pode ver na figura seguinte.

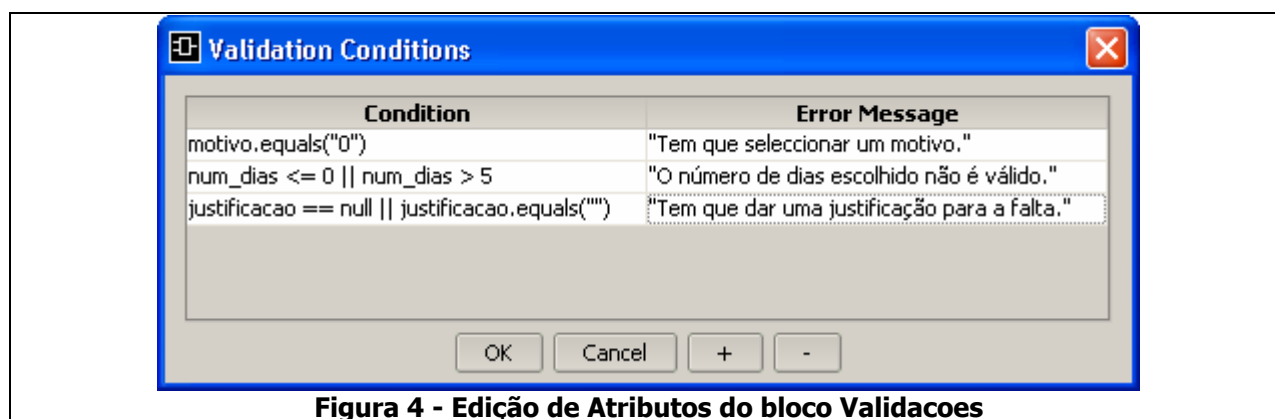


Figura 4 - Edição de Atributos do bloco Validacoes

As condições são condições de Java, sendo a mensagem de erro no formato de string de Java. No exemplo ilustrado, podemos concluir que se a variável motivo tiver o valor 0, o bloco atribui o erro “Tem de seleccionar um motivo.”. De notar que as condições são todas avaliadas, pelo que a mensagem de erro poderá ficar vazia (caso nenhuma se verifique) ou poderá ter todas as mensagens de erro (caso todas se verifiquem). O porto de saída do bloco depende disso mesmo, ou seja, se nenhuma das condições se verificar, o bloco sai pelo porto verde (superior); caso contrário, sai pelo porto vermelho (inferior), que geralmente irá ser ligado novamente ao bloco de entrada de dados (formulário).



O bloco de **Condicao** é similar ao de Validacoes, na medida em que também se configura a condição (como condições de Java). No entanto, só é permitido a configuração de uma condição e a saída do bloco faz-se pelo porto verde (superior) quando a condição é verificada e pelo porto vermelho (inferior) no caso contrário.



O bloco de **Formulario** serve, como o próprio nome indica, para criar formulários, sendo possível definir que campos o compõe, como se pode observar na figura seguinte.

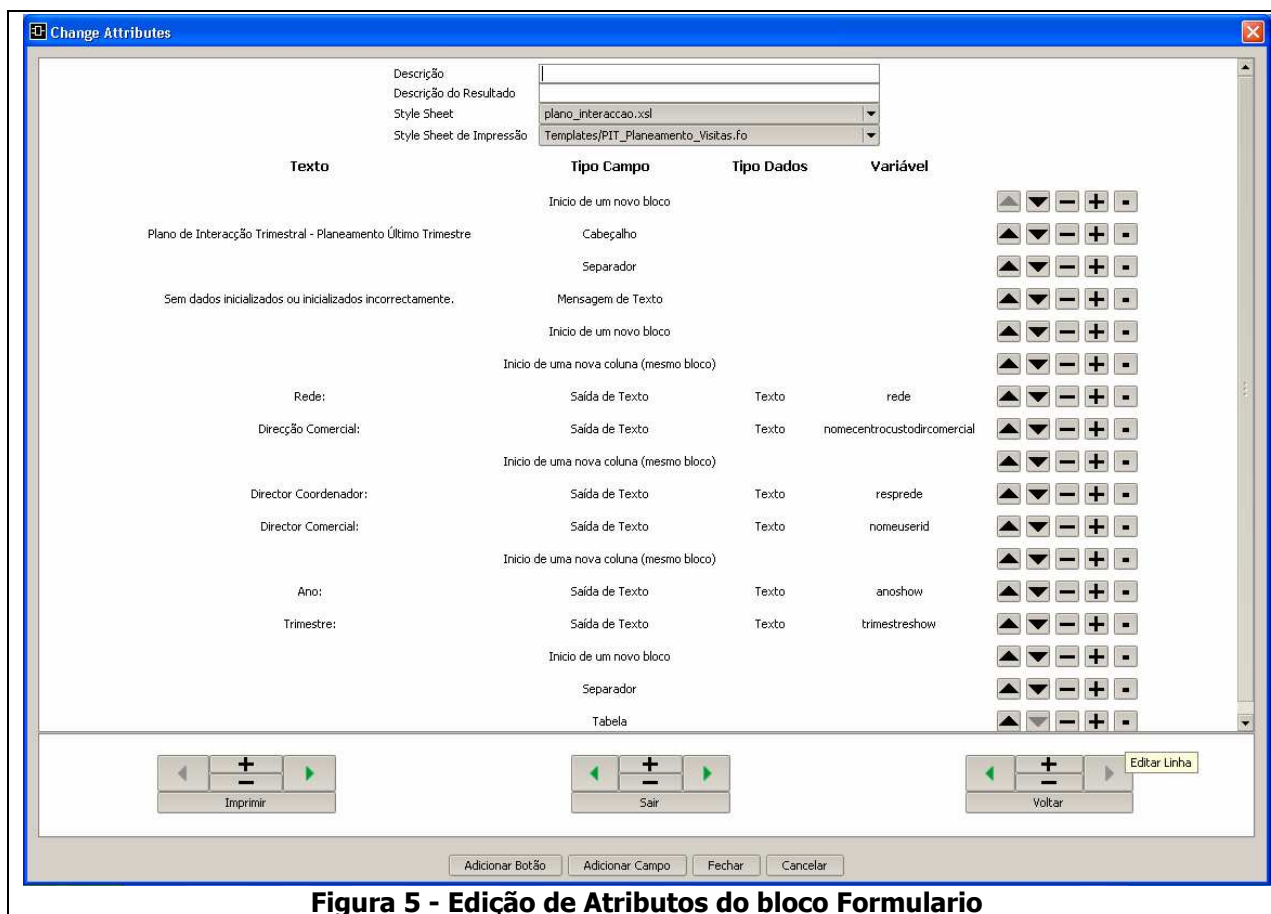


Figura 5 - Edição de Atributos do bloco Formulário

Como se pode observar, é possível definir os vários campos do formulário, a sua ordem no mesmo, o seu tipo de dados e a variável do processo associada, entre outros. Nas figuras seguintes encontra-se a janela de edição do campo “Lista de Selecção”, onde se pode observar a aplicação das propriedades do fluxo criadas no bloco Start, embora também seja possível definir a lista de selecção localmente embora de forma estática.

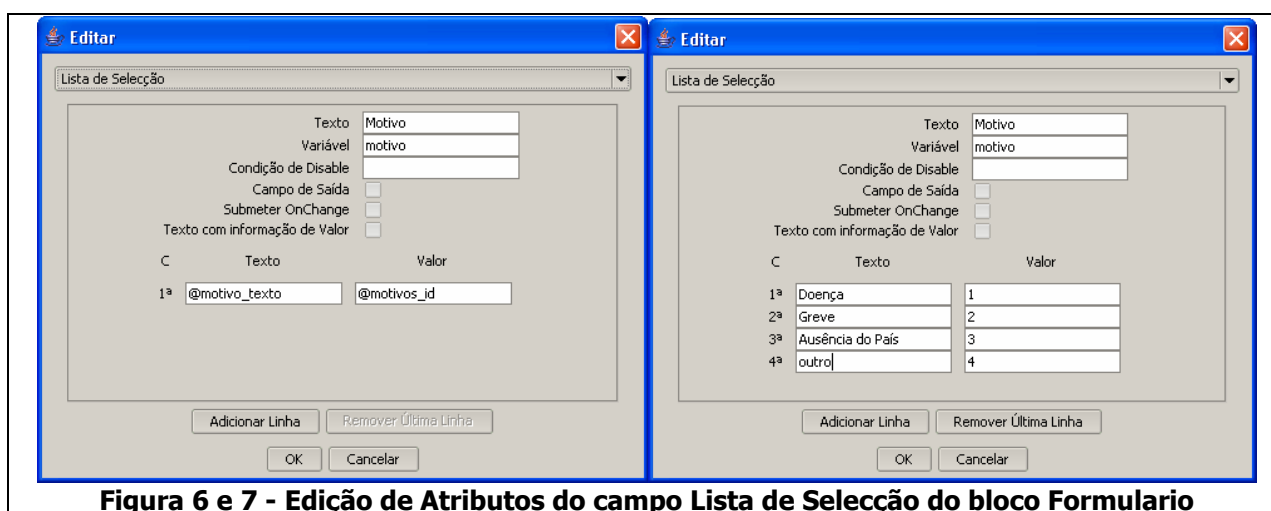


Figura 6 e 7 - Edição de Atributos do campo Lista de Selecção do bloco Formulário

Resumindo, podem-se definir os seguintes campos:

- Cabeçalho: permite definir um texto para um cabeçalho para o formulário;
- Caixa de Texto: permite criar uma caixa de texto no formulário, definindo o texto, a variável, o tipo de dados associado à variável e os comprimentos da caixa e do campo;
- Caixa de Password: similar à Caixa de Texto, não sendo possível definir o tipo de dados associado à variável, que por ser uma password, é texto;
- Tabela: permite criar uma tabela, definindo para cada coluna, o nome associado a ela, a variável, o tipo de dados associado à variável, o alinhamento na célula, algumas propriedades extra para alguns tipos de dados e ainda se é um campo de saída (disabled/readonly). A variável terá sempre de corresponder a uma lista, quer do processo, quer de propriedades do fluxo, sendo que neste último caso a variável terá de ter o prefixo “@”. Relativamente às propriedades extra, elas são importantes para os tipos de dados “Check Box”, “Radio Button” e “Link“, sendo as propriedades separadas por “,”. Para o primeiro, é possível definir as seguintes propriedades:

- “textvar” com o nome da variável com o texto a atribuir (ou acrescentar) à variável com o texto associada a esta coluna. Essa variável tem o mesmo nome da variável associada à coluna acrescentada do sufixo “_text”, ou seja, “<variável>_text” (em que <variável> é o nome atribuído à variável associada à coluna). Esta configuração é útil para poder mostrar os valores seleccionados num formato mais amigável num campo doutro tipo (como por exemplo, uma “Caixa de Texto”);
- “textsep” com o separador a aplicar entre o texto de cada checkbox seleccionada na variável “<variável>_text” (em que <variável> é o nome atribuído à variável associada à coluna);
- “checkedvalue” com o texto a atribuir à variável “<variável>_checkedtext” (em que <variável> é o nome atribuído à variável associada à coluna). Esta configuração é útil para poder mostrar os valores seleccionados num formato mais amigável quando se está a exportar a tabela (para o formato Microsoft Excel, por exemplo) ou noutra tabela com um formato mais simples.

Para o segundo tipo de dados, “Radio Button”, só é possível definir as mesmas propriedades definidas no tipo “Check Box” com a excepção de “textsep”, já que este tipo é exclusivo (só permite um valor seleccionado). Para o terceiro e último tipo, “Link”, é possível definir as seguintes propriedades:

- “text” com o texto do link;

- “vardep<n>” (<n> indica a dependência e terá de ser não negativo) com as dependências do link. Por exemplo, se a variável associada a esta coluna for “link”, existirem as dependências “vardep0=flowid,vardep1=pid” e o link for seleccionado, passam a existir a variável “link_flowid” com o valor da dependência “flowid” e a variável “link_pid” com o valor da dependência “pid”.

É ainda possível definir, para este campo, se se pretende disponibilizar o serviço de exportação, permitindo exportar os seus dados, e/ou o serviço de impressão, permitindo imprimir a tabela com a formatação com que é apresentada no formulário;

- Espaço: permite definir um espaço entre campos;
- Lista de Selecção: permite definir uma lista de selecção, sendo possível definir o texto, a variável, se está disabled (campo de saída) e as várias opções. Aqui é possível definir opção a opção, ou então definir uma variável do tipo lista, como foi exemplificado na **Erro! A origem da referência não foi encontrada.** De notar que o valor a atribuir à variável definida é o valor seleccionado pelo utilizador da coluna “Valor”. No entanto, é criada uma outra variável, de seu nome “<variável>_text” (em que <variável> é o nome da variável associada à Lista de Selecção), com o valor seleccionado pelo utilizador da coluna “Texto”, de modo a possuir um formato mais amigável;
- Saída de Texto: permite mostrar o conteúdo de variáveis, sendo possível definir o texto, a variável, o tipo de dados associado à variável e se se pretende ocultar o campo no caso da variável não ter valor;
- Área de Texto: permite definir uma área de texto, sendo possível definir o texto, a variável, o número de colunas e de linhas e se é um campo de saída (disabled). Sendo uma área de texto, o tipo de dados associado é texto;
- Link: permite definir um link. Este poderá ser um link do processo onde se insere ou um link para fora dele. Neste campo é possível definir: o texto, a classe da CSS, o script a executar quando o link é clicado, a mensagem de Status do browser quando se passa com o rato sobre o link, o alinhamento, a condição de enable (condição Java), se o link é de processo, sendo que nesse caso é possível definir a variável e o valor para a variável (como string de Java) se o link for pressionado, ao passo que se o link não for de processo é possível definir qual o URL e se se pretende abrir numa nova janela, sendo possível ainda definir neste caso, qual o nome a atribuir à janela;
- Mensagem de Texto: permite definir uma mensagem de texto. É similar ao cabeçalho, diferindo no facto deste campo ter outra formatação;

- Separador: permite definir um separador entre campos. É similar ao espaço, diferindo no facto deste ter outra formatação.
- Caixa de Data: permite criar uma caixa de data no formulário, definindo o texto, a variável, o tipo de data associado à variável e a definição da data actual por default;
- Lista de Selecção SQL: (...)
- Início de um novo bloco: permite definir grupos de campos dentro de um formulário para aplicação de regras/comportamentos (exemplo, aplicar um condição de activação dos campos do grupo definido);
- Início de uma nova coluna (mesmo bloco): permite definir grupos de campos dentro de um bloco para aplicação de regras/comportamentos;
- Imagem: (...)
- Ficheiro: (...)
- Gráfico: (...)
- Sub Cabeçalho: permite definir um texto para um sub cabeçalho para o formulário. Quando se pretende introduzir apenas texto, o mesmo terá de estar entre aspas, quando isso não acontecer, irá interpretar o texto como uma variável do fluxo e apresentará o seu valor;

Após definir os campos do formulário, é também necessário definir qual a stylesheet XSL que irá ser usada para a apresentação dos dados, a stylesheet usada para a impressão, caso essa funcionalidade seja pretendida para o formulário em questão e configurar os botões usados (o seu tipo, a sua posição e o texto associado).

1.2. Biblioteca Perfis

→ conjunto de blocos de acesso a informação de estruturas organizacionais do género LDAP, desde o acesso à informação genérica de um dado utilizador, como de unidade orgânica, passando também pelo encaminhamento de processos para um ou vários utilizadores.



O bloco **GetUserInfo** obtém dados do utilizador no LDAP, sendo possível configurar a variável com o utilizador para o qual se vai obter dados e as variáveis do processo que irão guardar esses mesmos dados, como se ilustra na figura seguinte (não definindo variável para dados que não sejam necessários ou que não se pretendam obter). A coluna de Input define qual a variável usada pelo bloco para servir de chave de busca no LDAP.

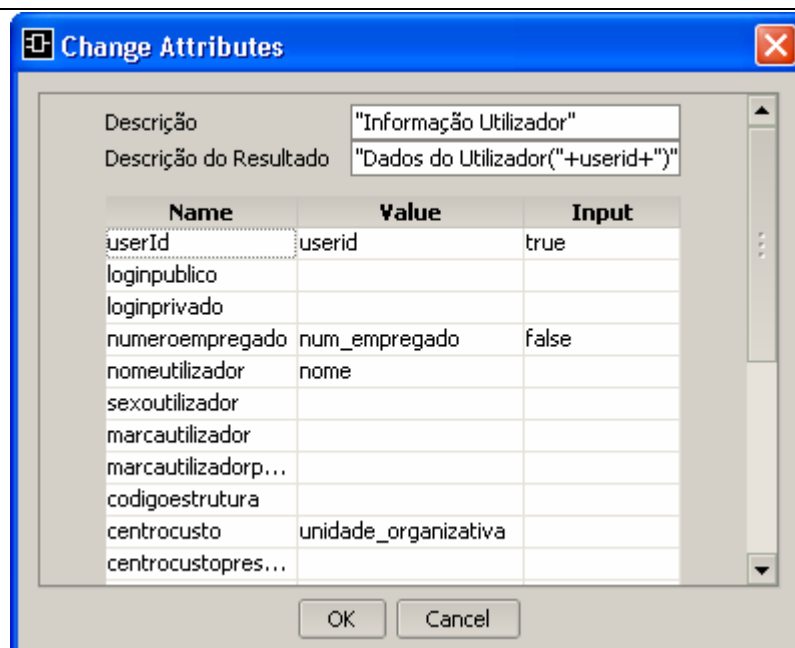


Figura 8 - Edição de Atributos do bloco GetUserInfo

Como se pode observar, existem atributos para Descrição e Descrição do Resultado. O primeiro guarda a descrição associada ao bloco, que irá ser visualizado na listagem das tarefas pendentes bem como para posterior consulta do histórico do processo. Quanto à Descrição do Resultado, só é visível no histórico do processo e serve para descrever o estado ou o resultado do bloco depois de ser executado. De notar que os valores destes campos estão formatados como strings de Java, sendo assim possível obter descrições que dependam de variáveis do processo (neste caso o utilizador, userid).



Os blocos **ForwardUp** e **ForwardTo** servem para fazer o reencaminhamento do processo para outros utilizadores. No primeiro, o processo é reencaminhado para o(s) superior(es) hierárquico(s) do utilizador detentor do processo (não é configurável). O segundo serve para reencaminhar o processo para um utilizador específico (neste caso o valor é no formato de string de Java) (figura 11) ou então para um perfil (grupo de utilizadores com uma determinada característica) (figura 9) ou ainda para um perfil definido em texto (neste caso o valor do perfil é no formato de string de Java) e não por selecção (figura 10). As configurações encontram-se ilustradas nas figuras seguintes.

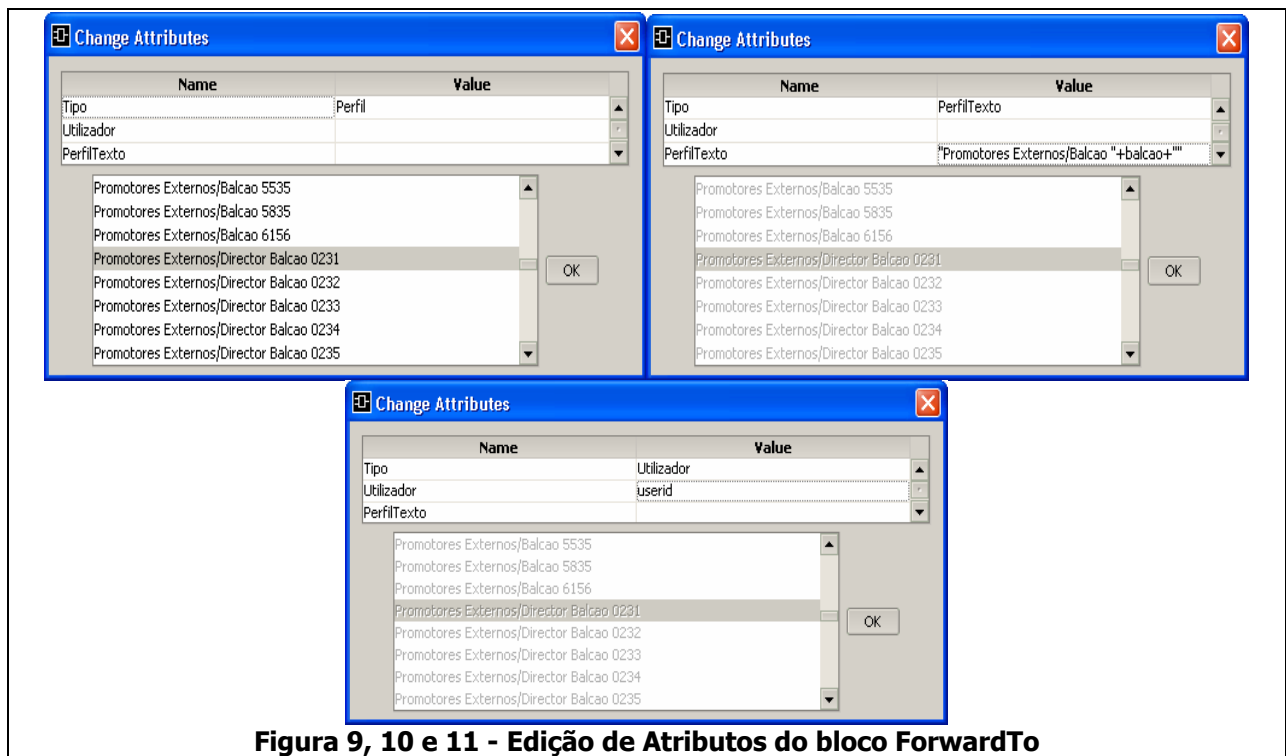


Figura 9, 10 e 11 - Edição de Atributos do bloco ForwardTo



GetUserUp

O bloco **GetUserUp** permite obter o(s) superior(es) hierárquico(s) para o utilizador detentor do processo para a variável configurada no bloco. Caso sejam precisos mais dados sobre o(s) superior(es) hierárquico(s), deverá conjugar-se este bloco com o bloco GetUserInfo já descrito anteriormente.

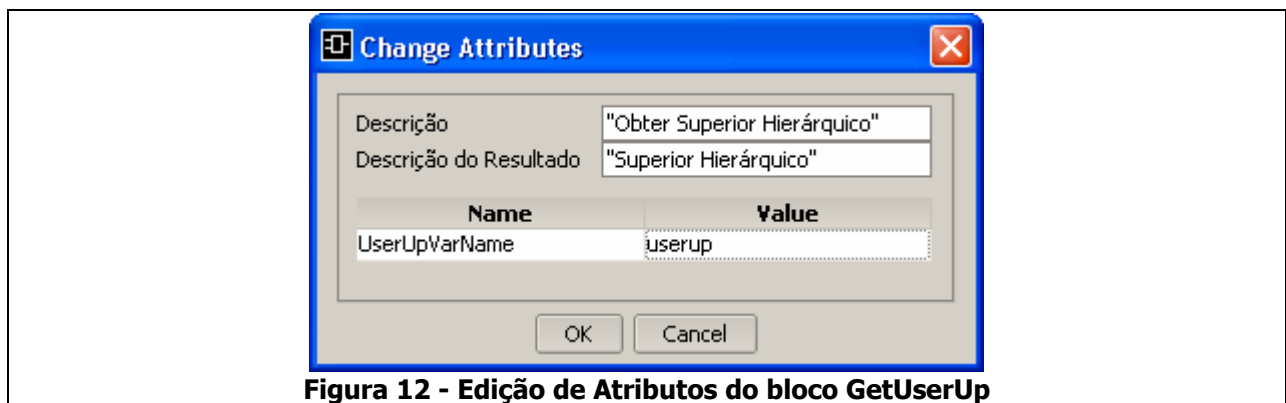


Figura 12 - Edição de Atributos do bloco GetUserUp



IsInProfiles

O bloco **IsInProfiles** permite verificar se o utilizador detentor do processo tem um determinado perfil. Para tal, é necessário configurar “regras”, definindo a condição para a qual se vai efectuar a verificação (Java), o perfil que se vai verificar caso a condição se verifique e a mensagem de erro caso o utilizador não tenha o perfil configurado (como string de Java). Desta maneira é possível definir regras de verificação baseadas em

condições. Caso se pretenda verificar somente o perfil, a condição a utilizar deverá ser “true”. A janela de edição para uma regra de exemplo encontra-se ilustrada na figura 13.



O bloco **IsInProfilesText** permite verificar se o utilizador detentor do processo tem um determinado perfil à semelhança do IsInProfiles. A única diferença consiste no campo Profile que é de preenchimento manual (avaliado como uma String Java) e não em lista de selecção. A janela de edição para uma regra de exemplo encontra-se ilustrada na figura 14.

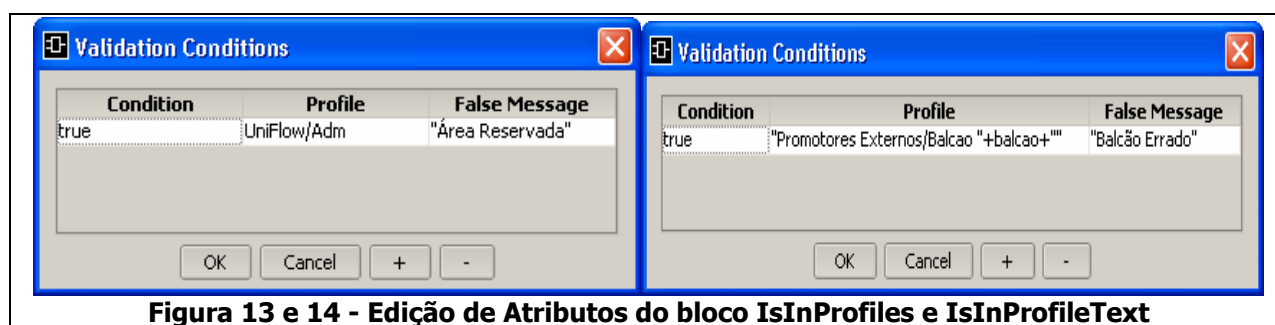


Figura 13 e 14 - Edição de Atributos do bloco IsInProfiles e IsInProfileText



GetOrganicalUnitInfo



GetOrganicalUnitParent

O bloco **GetOrganicalUnitInfo** obtém dados de uma dada Unidade Organizacional no LDAP. O seu funcionamento é similar ao bloco GetUserInfo descrito anteriormente, variando apenas o tipo de dados disponível já que se trata de unidades organizacionais e não users. O bloco **GetOrganicalUnitParent** obtém o mesmo tipo de dados que o bloco anterior, mas para a unidade organizacional superior (em termos hierárquicos) à qual o user actual do processo se insere.



CheckAuthentication

O bloco **CheckAuthentication** permite efectuar efectuar uma autenticação a um par variáveis user/password a ser preenchidas, esta autenticação é efectuada no sistema LDAP. O bloco tem duas saídas possíveis, uma de sucesso e outra de insucesso da autenticação. Na figura seguinte encontra se ilustrada o preenchimento do bloco.

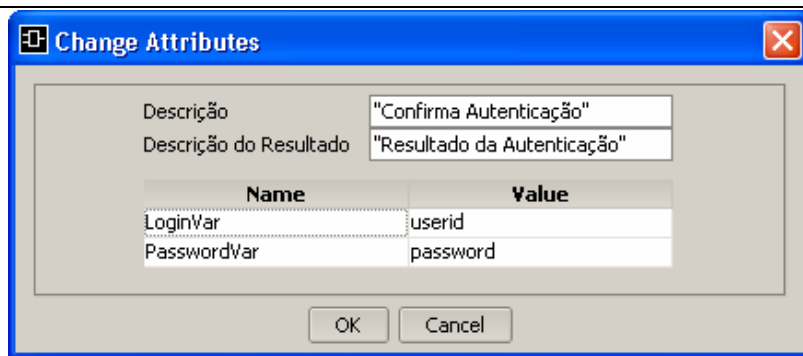
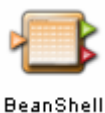


Figura 15 - Edição de Atributos do bloco CheckAuthentication

1.3. Biblioteca Utils

→ conjunto de blocos de índole genérica e tal como o nome indica de utilidade para todo o género de fluxo.



O bloco **BeanShell** permite efectuar processamento Java sobre o processo, usando uma biblioteca ligeiramente alterada da biblioteca BeanShell (<http://www.beanshell.org>). Neste bloco, a única configuração necessária corresponde ao código Java que se pretende executar. De notar que as variáveis do processo devem ter o prefixo “_”, ou seja, se se pretender usar a variável do processo “telefone” no código deste bloco, deve-se utilizar “_telefone”. Para variáveis definidas apenas no âmbito do bloco não é necessário o uso do mesmo prefixo.

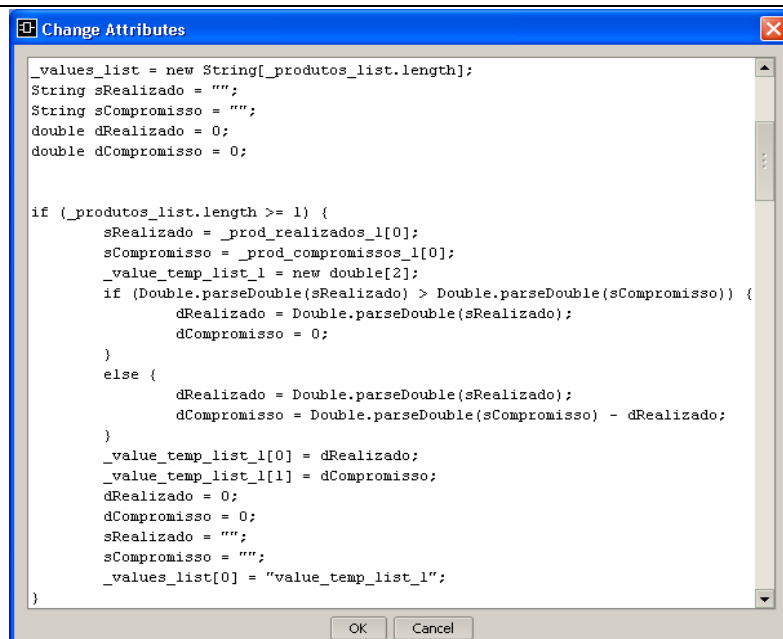


Figura 16 – Exemplo de uso do bloco BeanShell



Os blocos Bifurcacao, Sincronizacao e JuncaoExclusiva

(...)

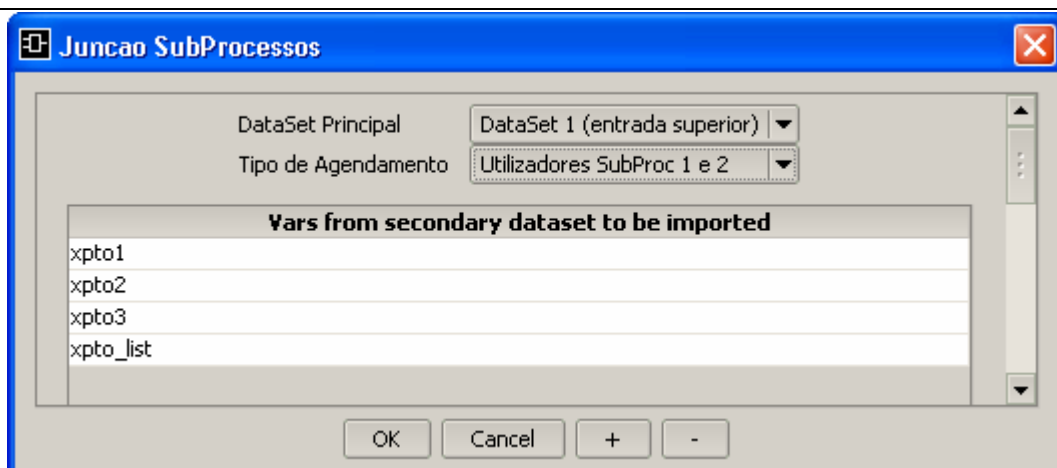


Figura 17 - Edição de Atributos do bloco Sincronizacao



O bloco SubFlow permite executar um novo e diferente fluxo dentro do fluxo actual, permitindo também mapear variáveis do fluxo principal para o novo fluxo (sub fluxo). No fim do sub fluxo, o focus retorna ao fluxo principal ou não, dependendo da configuração efectuada ao bloco. Existe também a possibilidade de no fim do sub fluxo as variáveis do mesmo serem mapeadas para variáveis do fluxo principal.

A configuração deste fluxo passa pela edição de 3 quadros distintos. No primeiro quadro (figura 18) é necessário definir qual o sub fluxo que se pretende instanciar, se se pretende manter no fim do mesmo as actividades do fluxo original e se pretende bifurcar o processo, passando assim a existir dois processos distintos, um que executa o fluxo principal, outro que executa o sub fluxo. No segundo quadro (figura 19) são definidas as variáveis que irão ser mapeadas do fluxo principal para o sub fluxo, a 1ª coluna com as variáveis do fluxo principal a mapear e na segunda coluna as variáveis do sub fluxo correspondentes. Finalmente, no terceiro quadro (figura 20) são definidas as variáveis que irão ser mapeadas do sub fluxo para o sub fluxo principal, a 1ª coluna com as variáveis do fluxo principal que irão receber os valores e na segunda coluna as variáveis do sub fluxo correspondentes.

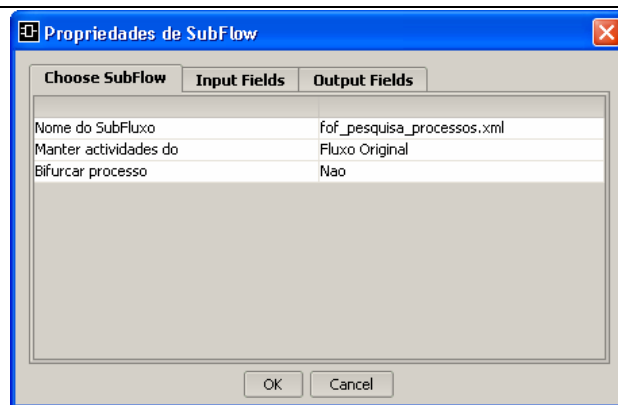


Figura 18 - Edição de Atributos do bloco Subflow (parte 1)

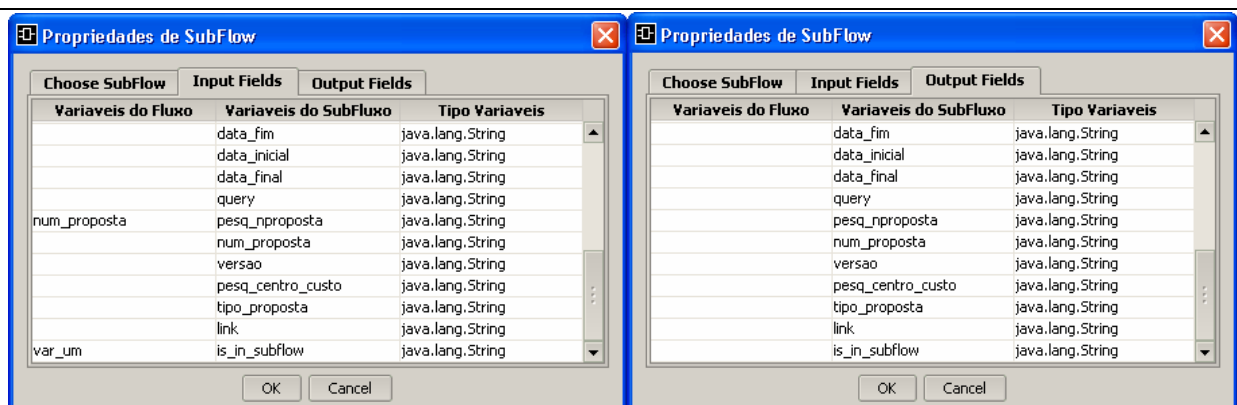


Figura 19 e 20 - Edição de Atributos do bloco Subflow (parte 2 e 3)



Date

O bloco **Date** permite obter para variáveis várias informações à cerca de uma data. Introduzindo uma data como input (por default, a data actual), é possível obter em variáveis definidas: dia (Dia), mês (Mes), mês por extenso (Nome Mes), ano (Ano), semana (Semana do Mes), inicio da semana (Primeiro Dia da Semana), fim da semana (Ultimo Dia da Semana).

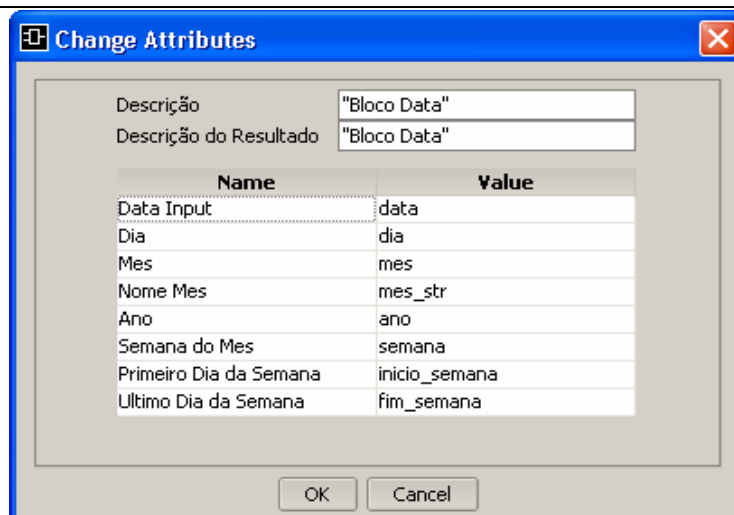


Figura 20 - Edição de Atributos do bloco Date

1.4. Biblioteca Processo

→ conjunto de blocos de índole genérica e tal como o nome indica de utilidade para todo o género de fluxo.



DetalheProcesso

O bloco de **Detalhe Processo** é similar ao bloco de Formulário, com a diferença que no Detalhe de Processo tem de se especificar as variáveis com o identificador do fluxo e o identificador do processo que se pretende consultar. A sua janela de edição de atributos encontra-se ilustrada na figura seguinte.

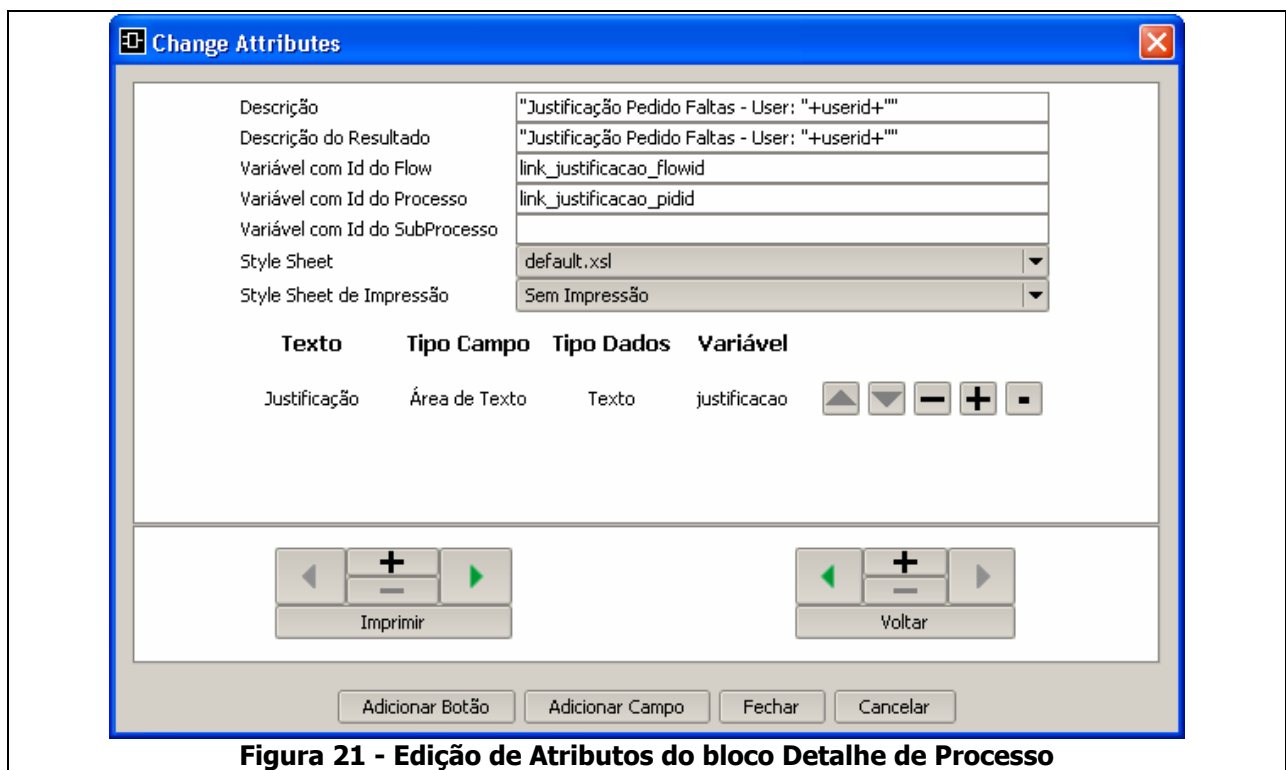


Figura 21 - Edição de Atributos do bloco Detalhe de Processo



PesquisaProcesso

O bloco de **Pesquisa Processo** permite pesquisar e/ou obter dados de outros processos, respeitando os critérios configurados. A figura seguinte ilustra a janela de edição de atributos deste bloco.

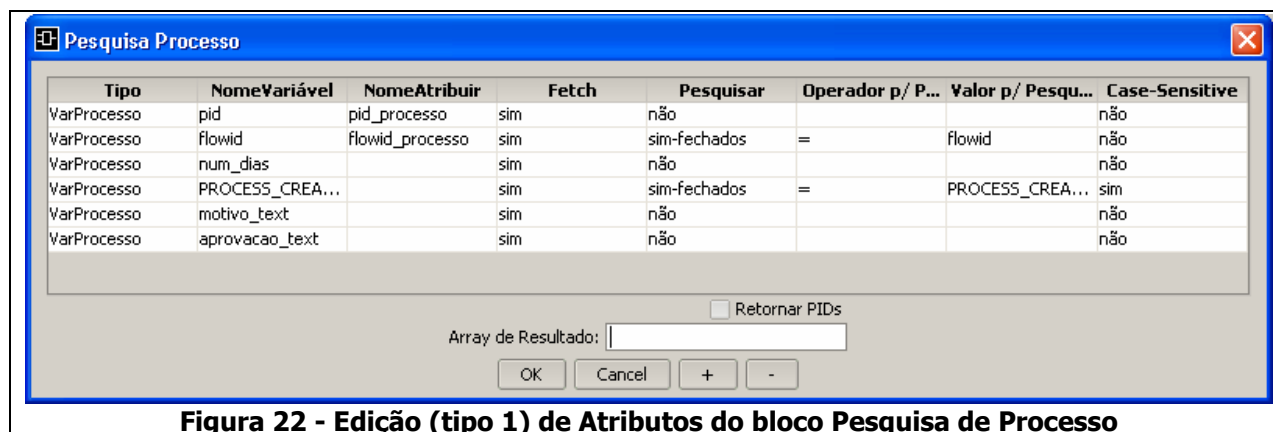


Figura 22 - Edição (tipo 1) de Atributos do bloco Pesquisa de Processo

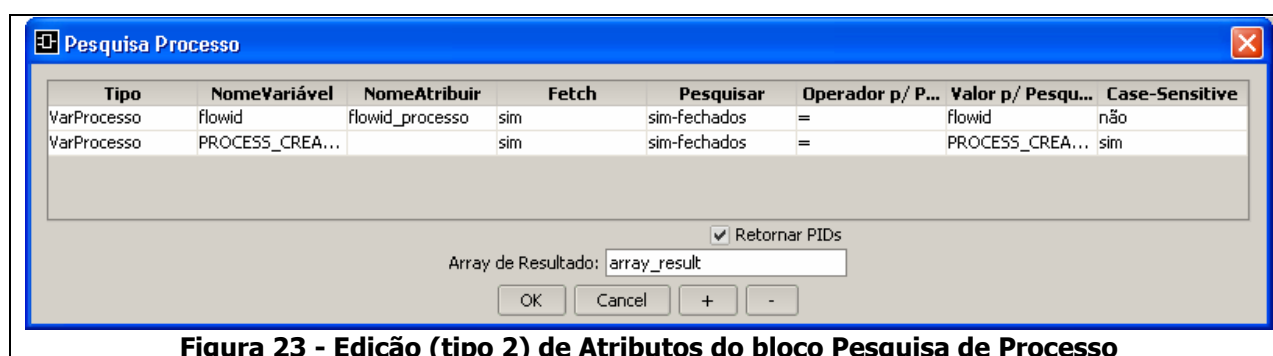


Figura 23 - Edição (tipo 2) de Atributos do bloco Pesquisa de Processo

Como se pode ver, neste bloco é necessário configurar as variáveis que se pretendem obter para cada processo e/ou sobre as quais se pretende pesquisar. Assim sendo, para cada variável é necessário definir:

- o seu tipo (se é variável do processo, ou se corresponde a uma data de estado ou ainda a uma descrição de estado);
- o seu nome;
- se se pretende obter o seu valor variável para o processo;
- se se pretende usá-la como critério de pesquisa e se sim, quais os processos que se pretende pesquisar (abertos, fechados ou todos);
- qual o operador SQL para a pesquisa (no formato string de Java, ex.: "=", ">", "is not null");
- o valor para usar na pesquisa (no formato de string de Java);
- se é sensível a maiúsculas/minúsculas.

Estas variáveis deverão ser visualizadas num campo do tipo tabela do bloco de Formulário, já que cada processo encontrado irá ser inserido numa lista (de processos) composta pelas variáveis que se pretendam obter (formando assim uma tabela), ou noutro bloco que permita visualizar tabelas de dados. (edição tipo 1)

Existe também a possibilidade de apenas introduzir linhas para critério de pesquisa e o bloco devolver um array (configurável no campo “Array de Resultado”) o pid, subpid e flowid dos processos englobados nos critérios de pesquisa. Para tal é necessário activar a checkbox “Retornar PIDs”. De salientar também que a opção descrita anteriormente é exclusiva em relação ao fetch normal da variáveis que se pretendem extrair dos processos. Este array apenas poderá ser usado em conjunto com o tipo de campo do bloco formulário “Tabela de Processo”.(edição tipo 2)



O bloco **Agrupamento**, tal como o nome indica, serve para agrupar processos segundo um determinado critério de agrupamento. A sua janela de configuração encontra-se ilustrada na figura seguinte:

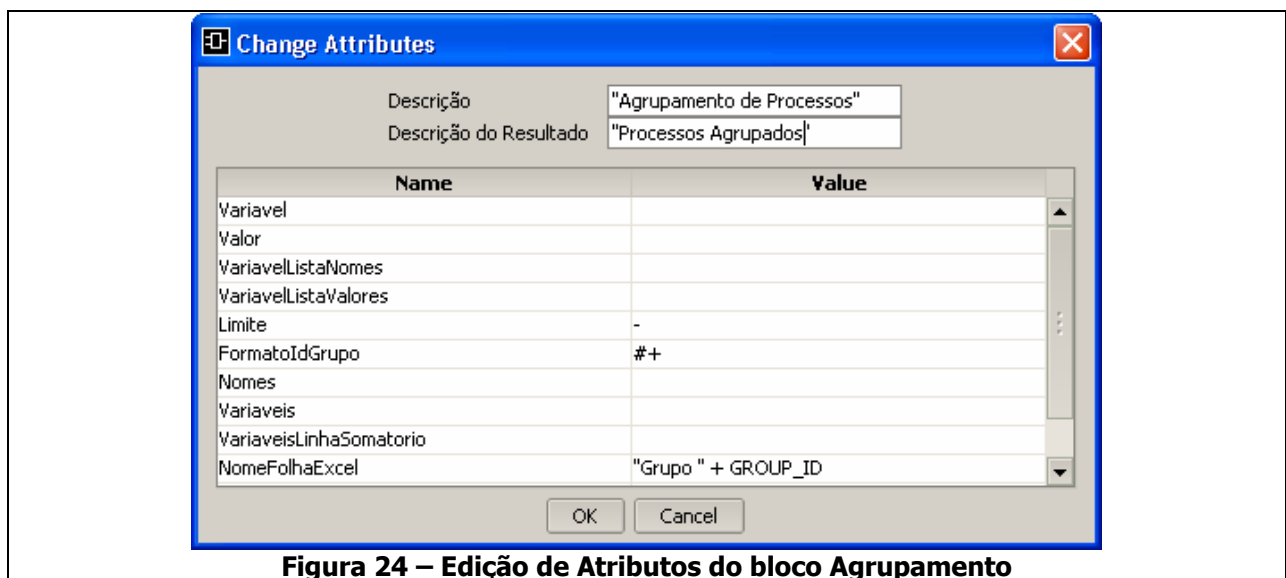


Figura 24 – Edição de Atributos do bloco Agrupamento

Além da “Descrição” e “Descrição do Resultado” (já descritos anteriormente), é possível configurar a variável que se pretende analisar para efeitos de agrupamento, o valor segundo o qual é efectuado o agrupamento (como string de Java), a variável com a lista dos nomes possíveis, a variável com a lista dos valores possíveis, o limite (se aplicável) de valores a agrupar, o formato do identificador do grupo (guardado na variável “GROUP_ID”), os nomes das colunas para exportação (separados por “,”), os nomes das variáveis associadas a essas colunas para exportação (igualmente separados por “,”) e o nome a atribuir à folha de cálculo de exportação (como string de Java). Importa salientar alguns aspectos:

- a Variável e o Valor podem ambos conter o nome da mesma variável, o que faz com que o agrupamento se faça em processos em que essa variável tenha o mesmo valor;

- o formato para o identificador do grupo é representado num formato em que cada “#” representa 1 e 1 só dígito, “#+” representa 1 ou mais dígitos e “{n}” representa exactamente “n” ocorrências o caracter que o antecede. Exemplificando, se o formato for “###/##” significa que o formato aceite para o identificador do grupo é composto por 3 dígitos, o caracter “/” e mais 2 dígitos. Qualquer outro formato irá originar um erro na página de Agrupamento.



O bloco **SaveToDB** permite que em certo ponto do fluxo o processo seja guardado na Base de Dados do Iflow. Não tem qualquer tipo de configuração ou output.



O bloco **OpenProc** permite



O bloco **SwitchProc** permite



O bloco **CleanProcError** permite

1.5. Biblioteca Base Dados

→ conjunto de blocos que permite efectuar operações SQL sobre uma base de dados (da aplicação ou outra).



Os blocos **SQLSelect**, **SQLInsert**, **SQLUpdate** e **SQLDelete** (toolbox Base Dados) permitem efectuar operações SQL sobre uma base de dados. Em todos eles é necessário configurar o atributo “JNDIName” com o nome JNDI da datasource a utilizar (caso este atributo não seja configurado, é utilizada a datasource da aplicação por default), a “Descrição” e a “Descrição do Resultado”. Outra característica comum a todos eles é o facto de todos os atributos serem configurados como strings de Java e a maior parte deles em linguagem SQL. De seguida descrevem-se os blocos mencionados, juntamente com os seus atributos:

- Bloco SQLSelect: permite efectuar pesquisas. Os atributos a configurar são:
 - Select: valores a pesquisar (SQL, com a alteração das funções, que terão de ter o prefixo “F_”, tal como ilustrado na figura de edição de atributos deste bloco);
 - From: tabelas para a pesquisa (SQL);
 - Where: condições para a pesquisa, se aplicável/pretendido (SQL);
 - Group: critério de agrupamento, se aplicável/pretendido;
 - Order: critério de ordenação dos resultados, se aplicável/pretendido(SQL);
 - ResultadoUnitário: indica se o resultado da pesquisa é unitário, caso no qual os valores pesquisados (definidos no campo “Select”) são guardados como variáveis “normais” do processo; caso contrário, os valores pesquisados são guardados em lista;
- Bloco SQLInsert: permite inserir dados do processo numa tabela em base de dados. Ao contrário do bloco SQLSelect, que permite a pesquisa em tabelas da aplicação, este bloco não permite a inserção de dados nessas tabelas, pois tal poderia causar inconsistências na aplicação. Os atributos a configurar são:
 - Into: tabela onde se vão inserir os dados (SQL);
 - Names: nomes das colunas (SQL);
 - Values: valores a inserir, respeitando a ordem definida no atributo anterior;
- Bloco SQLUpdate: permite actualizar dados numa determinada tabela em base de dados com dados do processo. Tal como no bloco anterior, a actualização de dados não é permitida em tabelas da aplicação. Os atributos a configurar são:
 - Tabela: tabela a actualizar (SQL);
 - Set: local para a atribuição dos pares nome-valor (SQL);
 - Where: critério de actualização (SQL). De notar que se não for definido nenhum critério, a actualização dos dados definidos no atributo anterior irá ser feita em toda a tabela.

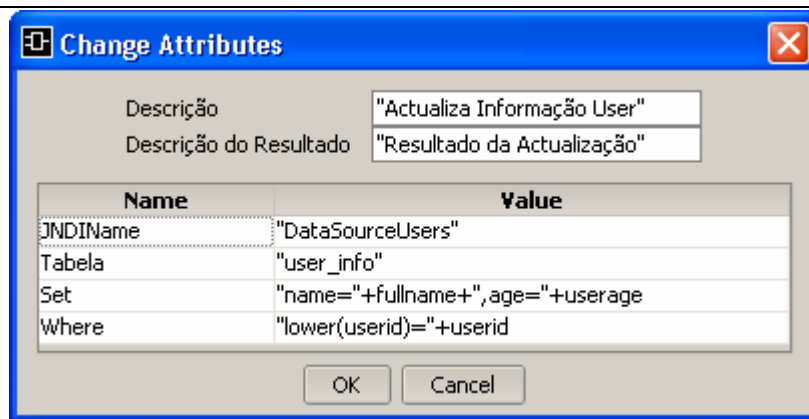
- Bloco SQLDelete: permite apagar dados de uma tabela em base de dados. Tal como nos dois blocos anteriores, não é permitido apagar dados em tabelas da aplicação. Os atributos a configurar são:
 - Tabela: tabela onde se vão eliminar dados (SQL);
 - Where: critério de remoção (SQL). De notar que se não for definido nenhum critério, todos os dados na tabela serão apagados.

Name	Value
JNDIName	
Select	"F_to_char(sysdate, 'DD-MM-YYYY') as data_actual"
From	"dual"
Where	
Group	
Order	
ResultadoUnitario	"Sim"

Figura 25 - Edição de Atributos do bloco SQLSelect

Name	Value
JNDIName	"DataSourceUsers"
Into	"user_info"
Names	"userid, username, name, age"
Values	userid+, "+user+", "+nome+", null"

Figura 26 - Edição de Atributos do bloco SQLInsert



Change Attributes

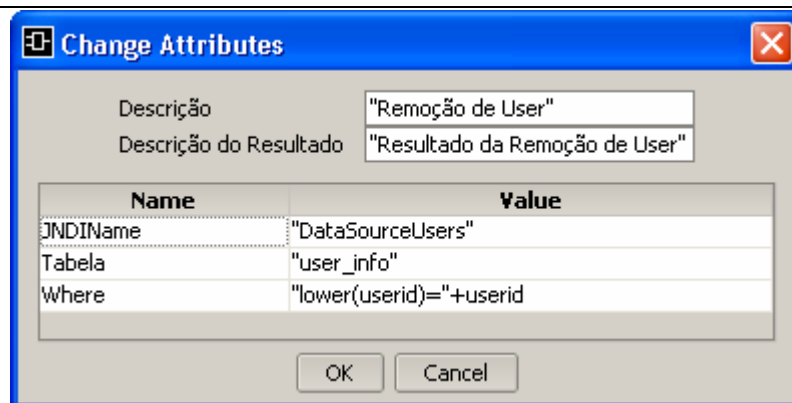
Descrição: "Atualiza Informação User"

Descrição do Resultado: "Resultado da Atualização"

Name	Value
JNDIName	"DataSourceUsers"
Tabela	"user_info"
Set	"name="+fullname+", age="+userage
Where	"lower(userid)="+userid

OK Cancel

Figura 27 - Edição de Atributos do bloco SQLUpdate



Change Attributes

Descrição: "Remoção de User"

Descrição do Resultado: "Resultado da Remoção de User"

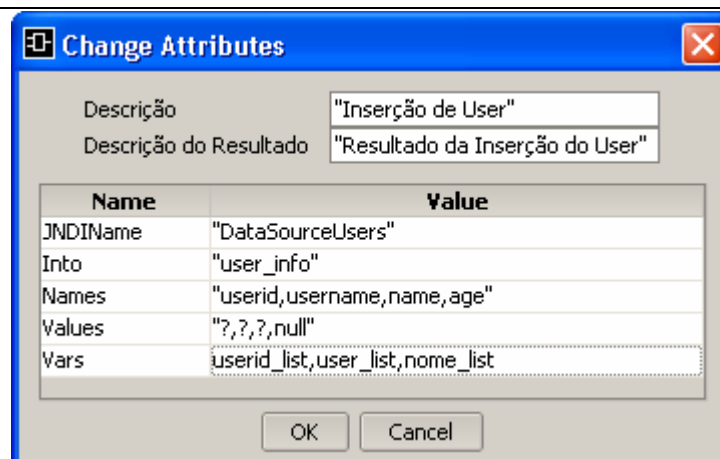
Name	Value
JNDIName	"DataSourceUsers"
Tabela	"user_info"
Where	"lower(userid)="+userid

OK Cancel

Figura 28 - Edição de Atributos do bloco SQLDelete



Os blocos **SQLBatchInsert** permite, tal como o SQLInsert, inserir dados do processo numa tabela em base de dados. Mas invés de apenas ser introduzida uma linha numa tabela, permite efectuar a inserção de uma ou mais linhas numa dada tabela de Base de Dados. A sua janela de configuração é, obviamente, um pouco diferente do bloco SQLInsert e encontra-se ilustrada na figura seguinte:



Change Attributes

Descrição: "Inserção de User"

Descrição do Resultado: "Resultado da Inserção do User"

Name	Value
JNDIName	"DataSourceUsers"
Into	"user_info"
Names	"userid,username,name,age"
Values	"?, ?, ?, null"
Vars	userid_list,user_list,nome_list

OK Cancel

Figura 29 - Edição de Atributos do bloco SQLBatchInsert

Os atributos do Bloco SQLBatchInsert a configurar são:

- Into: tabela onde se vão inserir os dados (SQL);
- Names: nomes das colunas (SQL);
- Values: valores a inserir, respeitando a ordem definida no atributo anterior, os pontos de interrogação para cada valor que será introduzido por lista;
- Vars: variáveis (listas) pela ordem que irão ser introduzidos nos pontos de interrogação no atributo Values, as listas poderão ter um ou mais elementos mas todas deverão ser do mesmo tamanho (mesmo número de elementos).

1.6. Biblioteca Notificação

→ conjunto de blocos que permite efectuar notificações, ou por correio electrónico (email) ou por sistema de mensagens (sms), tanto para users específicos como para grupos pré definidos de users (exemplo, por unidade orgânica).



Email

O bloco de **Email** serve para enviar uma mensagem de correio electrónico a um determinado utilizador, configurando o endereço de quem o está a enviar, o endereço para quem vai, o assunto e a mensagem (todos eles no formato string de Java), como ilustrado nas figuras seguintes.

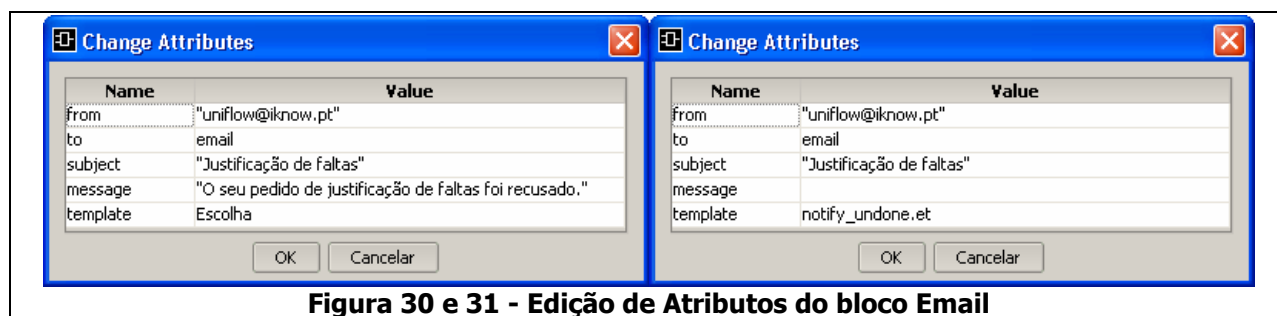


Figura 30 e 31 - Edição de Atributos do bloco Email

De salientar que a edição dos atributos message e template pode ser feita de duas formas, sendo o preenchimento dos mesmos atributos mutuamente exclusivo. É possível preencher o atributo message com a mensagem que se pretende e nesse caso não se preenche o atributo template. Ao invés quando se procede à escolha do atributo template, que no fundo se trata de uma mensagem de correio electrónico pré definida, não se deve preencher o campo message.



EmailPerfil

O bloco de **EmailPerfil** tem um funcionamento similar ao bloco Perfil, excepto para o(s) destinatário(s) da notificação por correio electrónico. Deverá ser escolhido o tipo de destinatário no atributo Type of To: Perfil para em seguida usar o atributo To Perfil, escolhendo o perfil desejado dentro dos existentes para a aplicação; PerfilTexto para em seguida usar o atributo To PerfilTexto, preenchendo como uma String Java qual o perfil pretendido.

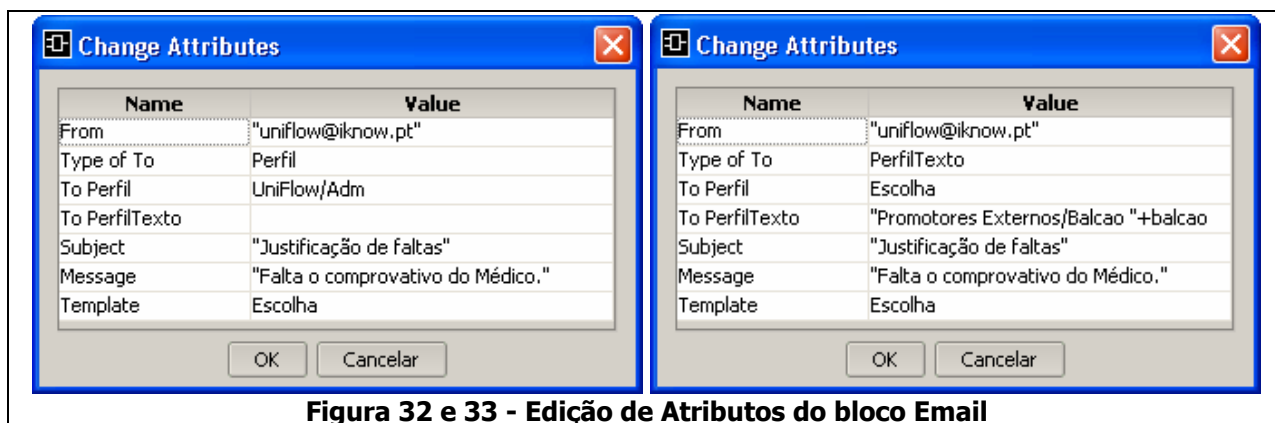


Figura 32 e 33 - Edição de Atributos do bloco Email



EmailIntervenientes

O bloco **EmailIntervenientes** permite enviar uma notificação via correio electrónico para todos os intervenientes do processo. A sua configuração é semelhante à do bloco Email. No atributo intervenientes, é possível efectuar a escolha sobre se os mesmos recebem o email directamente ou se recebem em cc (com conhecimento).

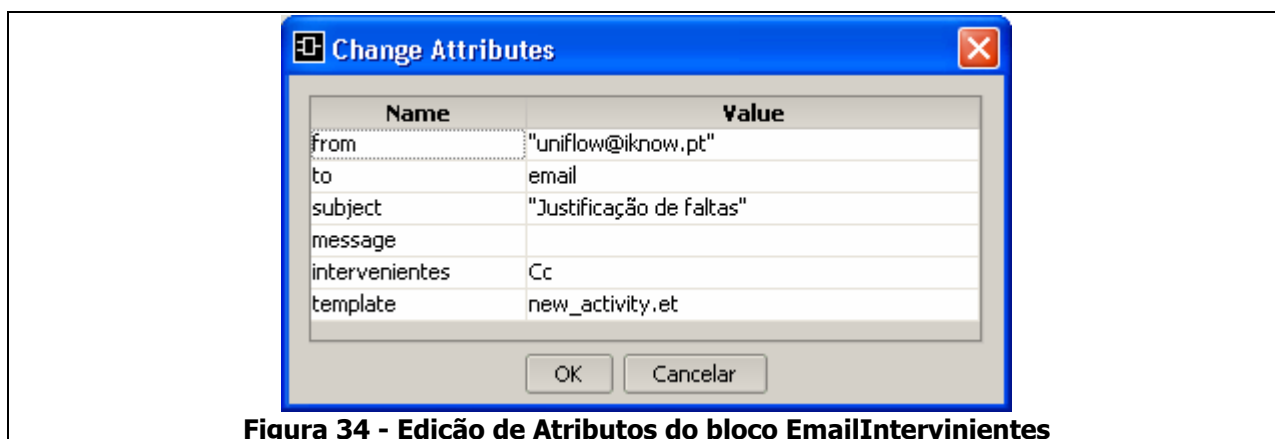


Figura 34 - Edição de Atributos do bloco EmailIntervenientes



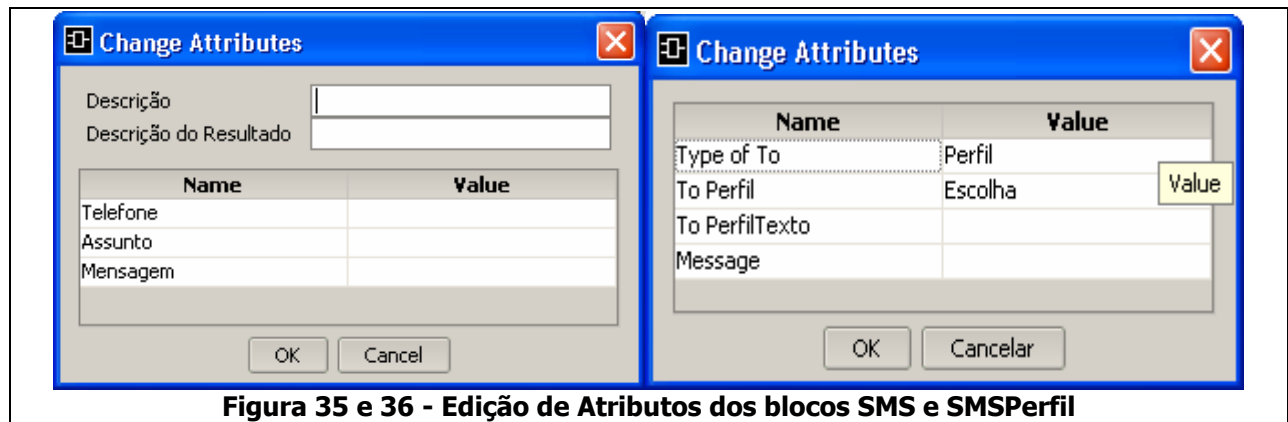
SMS



SMSPerfil

Os blocos **SMS** e **SMSPerfil** permitem enviar SMS, através da configuração do número do telefone no caso do bloco SMS ou do perfil

(pelo escolha do perfil ou atrás do perfil em texto) no caso do bloco SMSPerfil. E também pela configuração do assunto e da mensagem, todos eles como string de Java e idênticos em ambos os blocos.

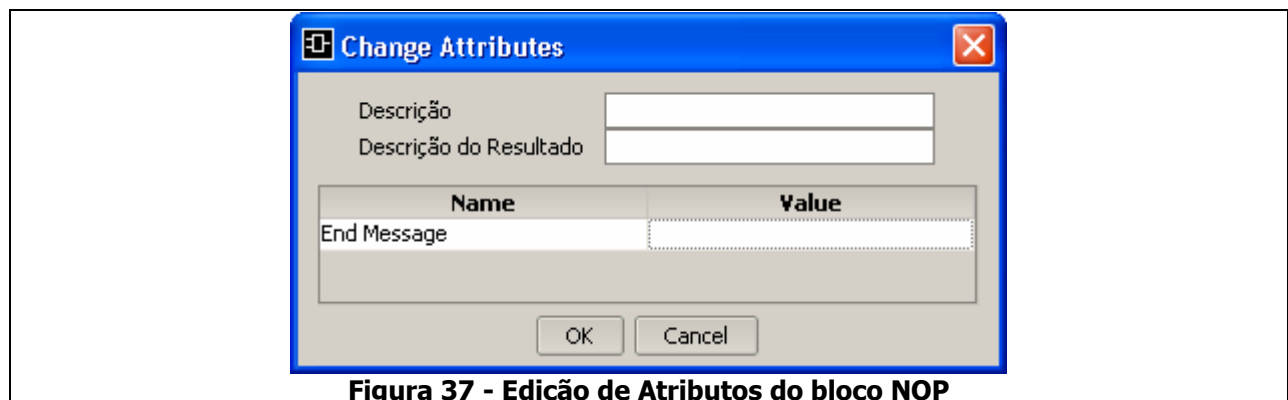


1.7. Biblioteca Eventos

→ conjunto de blocos que permite definir eventos, passando a ser possível executar certas zonas do fluxo depois do evento ser accionado (através de uma marca temporal configurável). Permite avançar um processo de forma automática sem a acção directa do utilizador.

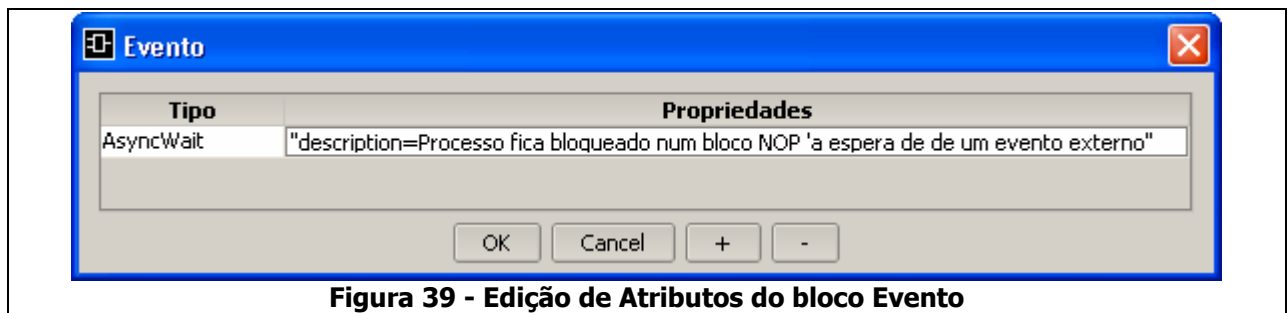
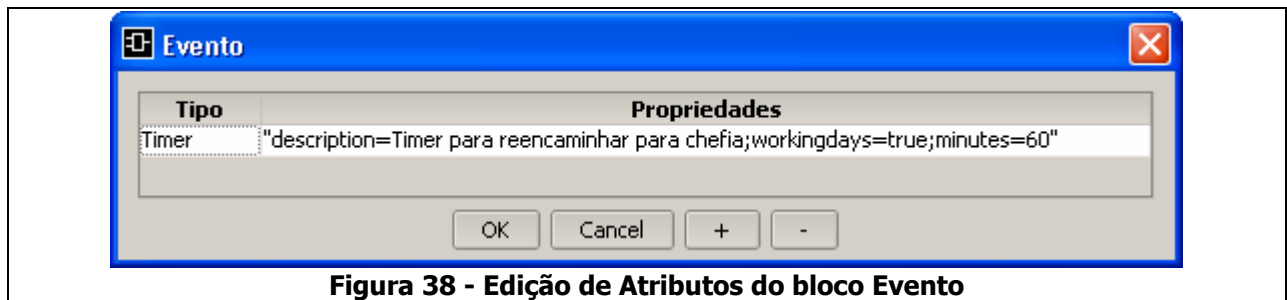


O bloco **NOP** permite suspender o processo até que o evento seja accionado.



O bloco **Evento** permite que o processo fique parado no bloco até a marca temporal expirar, voltando assim à execução normal do mesmo. Em termos de atributos temos dois por definir: Tipo, tendo como opção escolher o tipo de evento por Timer ou

AsyncWait; Propriedades, no caso do Tipo Timer é possível definir a descrição do evento que está a definir, se o timer deve contar apenas com os dias uteis da semana ou não e a marca temporal em minutos, no caso do Tipo AsyncWait (...). Nas figuras seguintes é possível ver a definição dos atributos do bloco.



1.8. Biblioteca WebServices

→ conjunto de blocos que permite (...)



O bloco **WebServiceSinc** permite (...)

Propriedades WebService Sincrono

Propriedades | Input Mapping | Output Mapping

WSDL: Novo

URL: [] [▶]

Serviço: []

Porto: []

Operação: []

Timeout: []

Num. Tentativas: []

OK Cancel

Figura 40 - Edição de Atributos do bloco WebServiceSinc

2. iFlowEditor – Particularidades de Definição dos Blocos e seus Atributos

2.1. Biblioteca Básica:

→ **Start**: As variáveis definidas neste bloco, não necessitem de estar definidas e/ou inicializadas no Catálogo de Variáveis (CatalogVars).

→ **End**: Ter em atenção que o bloco termina efectivamente o processo, deverá apenas ser usado se realmente for para acabar o processo, fechando-o.

→ **Copia**: Neste bloco o campo Destino terá que ser obrigatoriamente uma variável (seja, um double, uma string, um array de doubles ou um array de strings),. No campo Origem é possível ter variáveis, strings, números (doubles ou inteiros). Para adicionar um linha basta clicar no botão como sinal de soma (+), para remover uma linha, selecciona-se a linha que se deseja eliminar (clicando na mesma) e em seguida basta clicar no botão com o sinal de subtracção (-).

→ **Validacoes**: A coluna Condição suporta uma ou mais condições desde que obdecem à regras Java (uso dos parentesis, || (or), && (and), entre outros), onde as expressões mais usadas são: var1.equals(""); var1.equals(var2); varlist1.length(); var1 == null; var1 < num1; var1 >= num1; entre outros. A coluna Error Message terá que ser sempre preenchida com uma string, srting essa que se a(s) condição(ões) se verificar será colocada no topo do bloco formulário adjacente ao bloco validações.

→ **Condicao**: O campo condição deste bloco funciona igualmente como a coluna condição do bloco Validações.

→ **Formulário:**

- Cabeçalho – o atributo Texto é preenchido apenas com texto, não suportando strings java ou qualquer variável;
- Caixa de Texto – os atributos Expressão de Validação e Mensagem de Erro apenas estão editáveis quando seleccionado o atributo Campo Obrigatório. A Mensagem de Erro irá aparecer no topo do Formulário;
- Caixa de Password – nada a salientar;
- Tabela – na tabela é possível definir uma dada cor de fundo para cada coluna (em substituição da cor definida por default) basta para isso definir “bgcolor=#314E4E” no campo Propriedades Extra. É possível também definir o Título de uma coluna através de uma variável, para tal coloca-se a variável no campo Título e nas Propriedades Extra da mesma coluna a condição, “titulo=var”. Quando o Tipo de Dados de uma coluna se encontra definido como Check Box, é necessário de finir o valor da variável quando o campo se encontra seleccionado (checked), para essa definição é usado o campo Propriedades Extra, com a definição “checkedvalue=1”, “checkedvalue=sim” ou qualquer outra que se deseje. É também possível definir o tamanho da caixa de texto quando o Tipo de Dados está definido como tipo Caixa de Texto, para tal é necessário definir nas Propriedades Extra o seguinte “size=9”, sendo que o valor poderá ser o que desejarmos. Existe a possibilidade de mostrar ou não uma dada coluna ou colunas de uma tabela, para tal, é possível definir nas Propriedades Extra o seguinte “disable=true” ou “disable=“1”.equals(semana_temp)”, suportando assim condições Java.
- Espaço – este campo apenas introduz um espaço (linha) bastante reduzido, seria de interesse ser possível controlar o tamanho do espaço como um atributo extra;
- Lista de Selecção – quando seleccionado o atributo Texto com informação de Valor, o comportamento da lista de selecção altera-se, passando a mostrar as definições na coluna Texto e em seguida (e entre parentesis) as definições da coluna Valor;
- Saída de Texto – quando da utilização de uma variável do tipo double no Iflow e se pretende definir a mesma como saída de texto, o seu output irá ser sempre com a extensão ‘.0’ exceptuando se for escolhido no atributo Tipo de Dados o Número Inteiro;
- Área de Texto – não existe no campo Saída de Texto a opção de output do estilo da área de Texto, logo para o efeito deverá ser usado o campo de Input Área de Texto com o atributo Campo de Saída activado;
- Link – para poder editar os atributos Variável e Valor da Variável é necessário activar o atributo Link de Processo, para o mesmo efeito para o atributo Nome da Janela é necessário activar o atributo Abrir em Nova Janela;

- Mensagem de Texto – o atributo Texto é preenchido apenas com texto, não suportando strings java ou qualquer variável;
- Separador – nada a salientar;
- Caixa de Data – ao seleccionar o atributo Data Actual (se vazio) faz com que a Variável definida ficará com a data actual se não for preenchida e se avançar no formulário;
- Lista de Selecção SQL – o atributo Query funciona como uma atribuição em Java, texto entre aspas (“blabla”) e introdução de variáveis com o sinal de mais (+var+), com o atributo Texto com informação de Valor, este campo funciona como descrito no campo Lista de Selecção;
- Início de um novo bloco – nada a salientar;
- Início de uma nova coluna (mesmo bloco) – nada a salientar;
- Imagem – (...)
- Ficheiro – (...)
- Gráfico – (...)
- Sub Cabeçalho – ao contrário do campo Cabeçalho, este campo funciona como as Strings Java e permite variáveis.
- Tabela de Processo – (...)

2.2. Biblioteca Perfis:

→ **GetUserInfo**: não deve ser utilizado o campo de nome codigoeestrutura, pois encontra-se descontinuado. Na coluna de Input apenas um dos campos pode estar com o seu valor a true, pois apenas poderá ser uma a chave de pesquisa que o bloco vais usar para obter as informações desejadas.

→ **FowardUp**: nada a salientar.

→ **FowardTo**: apenas é possível passar o processo para um user específico ou para um perfil específico.

→ **GetUserUp**: o valor do atributo UserUpVarName é por default userup, mas poderá ser alterado para qualquer variável que se desejar.

→ **IsInProfiles**: o campo Condition pode ser preenchido com uma condição ou se não fizer sentido existir uma, deve colocar-se true, no campo False Message deverá constar uma String (“blabla”).

→ **IsInProfilesText**: nada a salientar.

→ **GetOrganicalUnitInfo**: não deve ser utilizado o campo de nome codigoestrutura, pois encontra-se descontinuado. Na coluna de Input apenas um dos campos pode estar com o seu valor a true, pois apenas poderá ser uma a chave de pesquisa que o bloco vais usar para obter as informações desejadas

→ **GetOrganicalUnitParent**: não deve ser utilizado o campo de nome codigoestrutura, pois encontra-se descontinuado. Na coluna de Input apenas um dos campos pode estar com o seu valor a true, pois apenas poderá ser uma a chave de pesquisa que o bloco vais usar para obter as informações desejadas

→ **CheckAuthentication**: em ambos os atributos do bloco (LoginVar e PasswordVar) deverá ser definida uma variável.

2.3. Biblioteca Utils:

→ **BeanShell**: de ter em atenção que ao existir um erro de codificação no bloco, o mesmo termina a execução e entra no próximo bloco, não dando qualquer output de erro na execução do fluxo, apenas nos logs.

→ **Bifurcacao**: (...)

→ **Sincronizacao**: (...)

→ **JuncaoExclusiva**: (...)

→ **SubFlow**: nada a salientar.

→ **Date**: o atributo Data Input por default usa a data actual, se não lhe for definido nenhum valor.

2.4. Biblioteca Processo:

→ **DetalheProcesso**: (...)

→ **PesquisaProcesso**: (...)

→ **Agrupamento**: (...)

→ **SaveToDB**: nada a salientar.

→ **OpenProc**: (...)

→ **SwitchProc**: (...)

→ **CleanProcError**: (...)

2.5. Biblioteca Base Dados:

→ **SQLSelect**: todos os atributos são de definição do tipo String Java e é possível o uso de variáveis;

→ **SQLInsert**: todos os atributos são de definição do tipo String Java e é possível o uso de variáveis; quando se faz a inserção de um campo do tipo Date deverá ser efectuado da seguinte forma, `to_date("+var+", "DD/MM/YYYY")`;

→ **SQLUpdate**: todos os atributos são de definição do tipo String Java e é possível o uso de variáveis, quando se faz uma actualização de um campo do tipo Date deverá ser efectuado da seguinte forma, `to_date("+var+", "DD/MM/YYYY")`;

→ **SQLDelete**: todos os atributos são de definição do tipo String Java e é possível o uso de variáveis;

→ **SQLBatchInsert**: todos os atributos são de definição do tipo String Java e é possível o uso de variáveis, excepto o campo Vars que se define, como exemplo (nome_list, idade_list, user_list)

2.6. Biblioteca Notificação:

→ **Email**: os atributos from e subject tem de estar entre parentesis (exemplos, “iflow@iknow.pt” “Reunião de Direcção”); o campo to, recebe variáveis; o campo message é do tipo String Java e aceita variáveis.

→ **EmailPerfil**: os atributos from e subject tem de estar entre parentesis (exemplos, “iflow@iknow.pt” “Reunião de Direcção”); o campo message é do tipo String Java e aceita variáveis.

→ **EmailIntervinientes**: os atributos from e subject tem de estar entre parentesis (exemplos, “iflow@iknow.pt” “Reunião de Direcção”); o campo to, recebe variáveis; o campo message é do tipo String Java e aceita variáveis.

→ **SMS**: (...)

→ **SMSPerfil**: (...)

2.7. Biblioteca Evento:

→ **NOP**: (...)

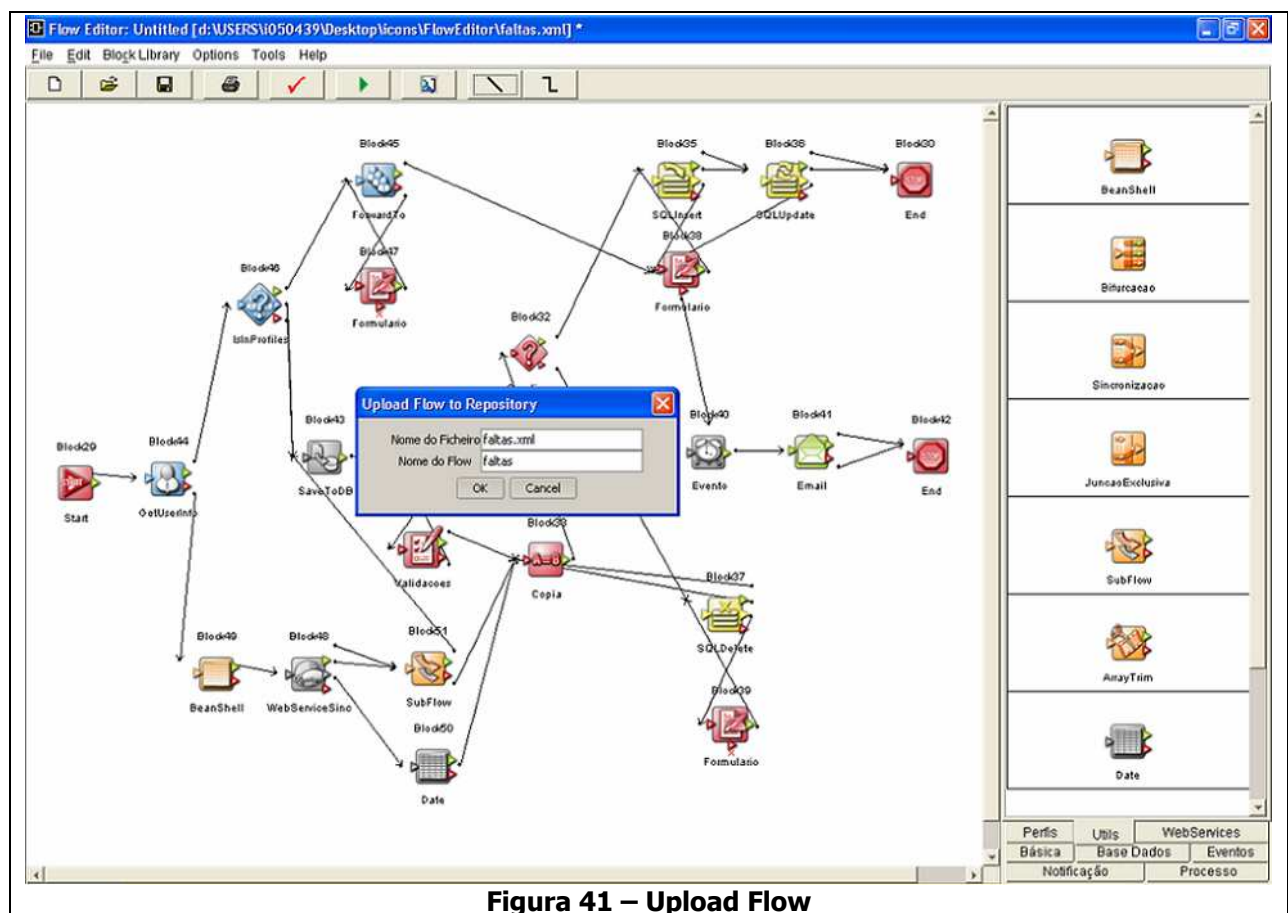
→ **Evento**: nada a salientar.

2.8. Biblioteca Evento:

→ WebServiceSinc: (...)

3. iFlowEditor – Activar um Fluxo

Após a ligação e configuração dos blocos que compõe o fluxo, é necessário carregá-lo para o repositório, de modo a que fique disponível para deployment no iFlow. Isso é feito através do menu “File” do editor e seleccionando a opção “Upload Flow”, a qual abre uma janela na qual se pode especificar o nome do ficheiro associado ao fluxo (identifica univocamente o fluxo na aplicação), bem como o seu nome, tal como se ilustra na figura seguinte para o exemplo seguido, no qual se atribuiu o nome para o ficheiro de “faltas.xml” e o nome para fluxo de “Justificação de Faltas”.



Caso se pretenda editar um fluxo já existente no repositório (para fazer alterações à lógica do processo, por exemplo), deve-se seleccionar a opção “Download Flow” do mesmo menu, escolhendo o fluxo da lista apresentada na janela de download, como ilustrado na figura seguinte.

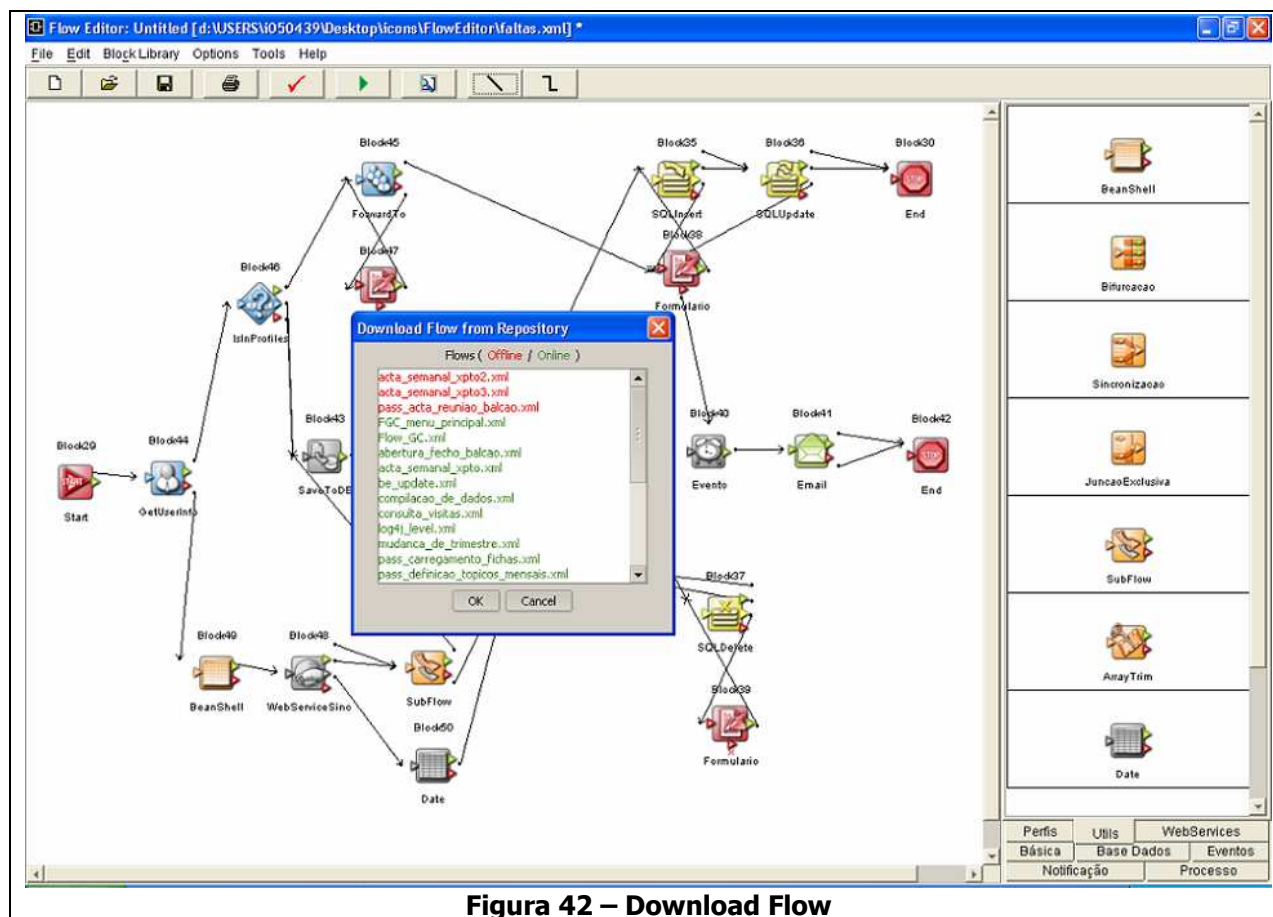


Figura 42 – Download Flow

De notar que é possível fazer download de fluxos que estejam online e offline (disponíveis e não disponíveis) no iFlow, o mesmo não acontecendo com o upload de fluxos, situação na qual só é permitido fazer upload de fluxos que se encontrem offline (não disponíveis) no iFlow.

iFlow

Após a criação do fluxo no editor e do seu upload para o repositório, o passo seguinte é criá-lo no lado da aplicação iFlow. Para tal, deve-se efectuar o login na aplicação com um utilizador que tenha privilégios de administração, procedendo de seguida para área de “Administração” (só visível para utilizadores com privilégios de administração), ilustrada na figura seguinte.

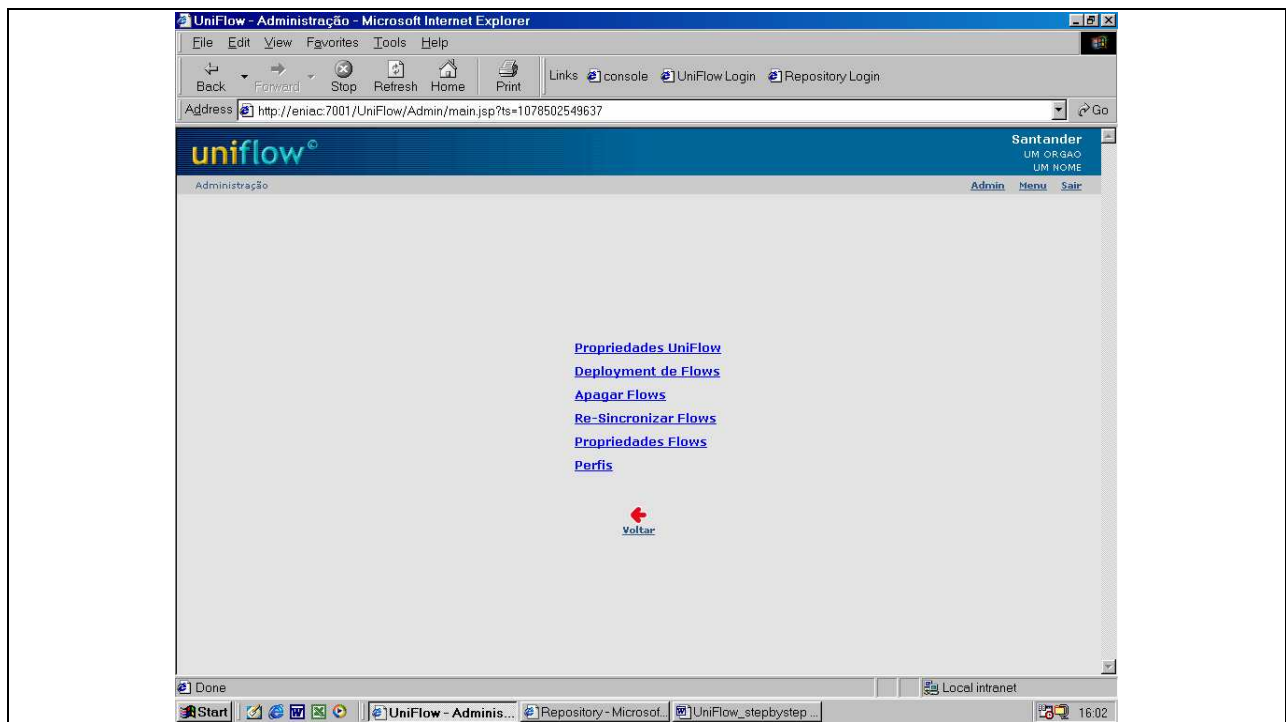


Figura XX – Área de Administração do iFlow

De seguida deve-se efectuar o deployment do fluxo, seguindo o link “Deployment de Flows”, onde encontramos a lista de fluxos que se encontram offline (undeployed) e a lista de fluxos que se encontram online (deployed) na aplicação. Sendo assim, e continuando a utilizar o fluxo de exemplo “faltas.xml”, deverá seleccionar-se esse fluxo nos fluxos offline e passá-lo para online, clicando no botão “=>”, como se pode ver na figura seguinte.

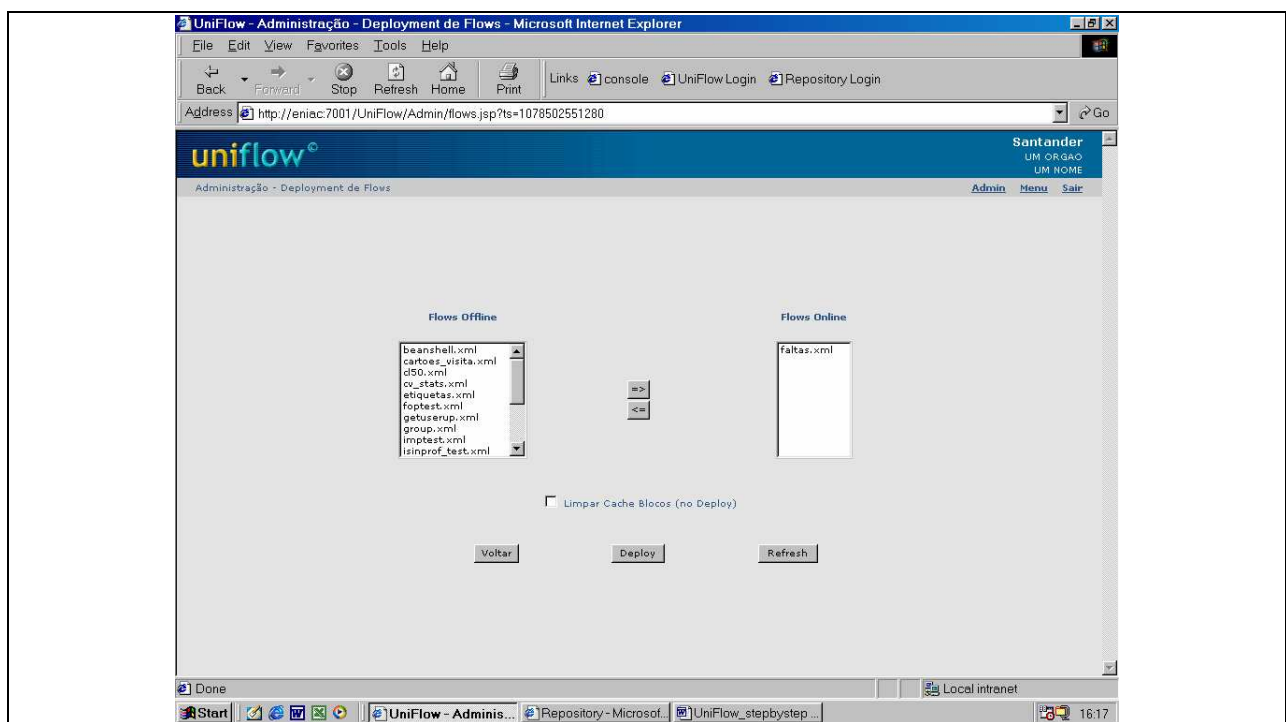


Figura XX – Área de Deployment de Flows

Para concluir o processo, deve-se ainda pressionar o botão “Deploy”, deixando a lista sem nenhum fluxo seleccionado ou então seleccionando o(s) fluxo(s) sobre o(s) qual(uais) se pretende fazer deploy (esta última solução é preferível quando já existem fluxos deployed e não se pretende efectuar novamente o seu deployment). Se o deployment não originar nenhum erro, o fluxo foi criado com sucesso no iFlow, passando a estar disponível na aplicação. Após a sua criação, é aconselhável que se proceda à passagem do fluxo para offline, por forma a torná-lo indisponível para os utilizadores enquanto se configuram as propriedades do fluxo. Para tal deve-se proceder de forma análoga ao deployment: escolher o fluxo “faltas.xml” na lista dos fluxos que se encontram online e clicar no botão “<=”. De seguida pressionar o botão “Deploy”.

Para configurar as propriedades do fluxo, voltar à área de Administração e seguir o link “Propriedades Flows”. De seguida seguir o link do fluxo que se pretende configurar (“faltas.xml” neste caso), resultando na página ilustrada na figura seguinte.

Variável	Propriedade	Tipo	Valor
@motivos_id	Id dos Motivos	Lista	1: 0 (TEXTO) Remover 2: 1 (TEXTO) Remover 3: 2 (TEXTO) Remover 4: 3 (TEXTO) Remover 5: 4 (TEXTO) Remover 6: Adicionar: Valor Query
@motivos_texto	Motivos	Lista	1: Escolha (TEXTO) Remover 2: Doença (TEXTO) Remover 3: Ida ao médico (TEXTO) Remover 4: Familiar (TEXTO) Remover 5: Outro (TEXTO) Remover 6: Adicionar: Valor Query
DATASET_LOCATION	Iniciar Processo em	Simple	Sessao
DENY_NOTIFY_FOR_PROFILE	Abortar Notificação para Perfil (quando NOTIFY_USER=Sim)	Simple	Escolha Adicionar Perfil
FLOW_DATE_FORMAT	Formato para as Datas (default: yyyy-MM-dd HH:mm)	Simple	
FLOW_ENTRY_PAGE_LINK	Link Pág. Entrada	Simple	
FLOW_ENTRY_PAGE_TITLE	Título Pág. Entrada	Simple	
FORCE_NOTIFY_FOR_PROFILE	Forçar Notificação para Perfil (quando NOTIFY_USER=Nao)	Simple	Escolha Adicionar Perfil
NOTIFY_USER	Notificar Utilizador de Nova Tarefa	Simple	Sim

Voltar Exportar Importar Guardar Propagar para

Local intranet 16:44

Figura XX – Área de Edição de Propriedades do Fluxo

Nesta página encontram-se as propriedades do fluxo seleccionado. Existem dois tipos de propriedades: as propriedades definidas no fluxo, através do bloco Start no editor, e as propriedades inerentes a todos os fluxos.

Começando pelas primeiras, podemos constatar a presença das duas propriedades do tipo lista configuradas no bloco Start. Para elas já se encontram definidos os seus valores, os quais

serão usados mais tarde no fluxo (no campo do tipo “Lista de Selecção” do bloco Formulário). Para adicionar valores à lista, deve-se preencher a caixa de texto vazia com o valor pretendido e clicar em “Valor” na zona de “Adicionar:”. Além de adicionar valores “simples”, é também possível adicionar queries de SQL (estas no formato de string de Java), bastando para isso definir a query na caixa de texto e clicar em “Query” na zona de “Adicionar:”. Caso se pretenda que a query definida utilize outra datasource que não a da aplicação, deverá ter o prefixo “{DS=<datasource>}”, em que <datasource> é o nome da datasource a utilizar. Para remover valores ou queries, deve-se clicar em “Remover”, notando que esta opção só está disponível para o último elemento da lista.

Em relação às segundas (não pela mesma ordem da figura):

- DATASET_LOCATION: indica a localização inicial para os dados do processo, sessão ou base de dados, sendo a primeira a opção seleccionada por defeito;
- NOTIFY_USER: permite configurar a notificação de novas tarefas pendentes;
- DENY_NOTIFY_FOR_PROFILE: permite criar uma lista de perfis para os quais os utilizadores que neles constem não sejam notificados de novas tarefas pendentes, quando a propriedade de notificar os utilizadores estiver a “Sim”. A adição de novos perfis faz-se seleccionando o perfil desejado e clicando em “Adicionar Perfil”. A remoção de um determinado perfil, faz-se directamente na caixa de texto, apagando o perfil desejado;
- FORCE_NOTIFY_FOR_PROFILE: de forma análoga à propriedade anterior, permite criar uma lista de perfis para os quais os utilizadores que neles constem sejam notificados de novas tarefas pendentes, quando a propriedade de notificar os utilizadores estiver a “Não”. A adição de novos perfis faz-se seleccionando o perfil desejado e clicando em “Adicionar Perfil”. A remoção de um determinado perfil, faz-se directamente na caixa de texto, apagando o perfil desejado;
- FLOW_DATE_FORMAT: permite configurar o formato das datas por defeito do fluxo;
- FLOW_ENTRY_PAGE_LINK: permite definir o texto associado ao link do fluxo no menu principal da aplicação quando esta está restrita ao fluxo. Esta restrição é feita adicionando o parâmetro “ufid” com o identificador do fluxo na página de login da aplicação, e faz com que só se possam efectuar operações, criar e consultar processos deste fluxo;

- `FLOW_ENTRY_PAGE_TITLE`: idêntica à propriedade anterior, com a diferença que o texto definido nesta propriedade irá ser usado como título no menu principal da aplicação quando esta estiver restrita ao fluxo;
- `SHOW_SCHED_USERS`: indica se a aplicação deve mostrar os utilizadores para quem o processo foi reencaminhado ou se encontra agendado.

Após a configuração das propriedades, é necessário clicar em “Guardar”, para que elas fiquem guardadas na base de dados. Caso o fluxo esteja online, é necessário ainda (caso seja o pretendido), clicar no link “Propagar para Flow”, para que as propriedades sejam propagadas para o fluxo. É ainda possível importar e exportar as propriedades de e para um ficheiro respectivamente.

O passo seguinte é a configuração das permissões por perfil de utilizador para o fluxo, dado que, quando é criado, o fluxo apenas pode ser consultado, criado e administrado por utilizadores com privilégios de administração. Para alterar isso, deve-se voltar à área de Administração e seguir para a área de Perfis. Aqui, é possível sincronizar os perfis existentes no repositório com os perfis no LDAP para a aplicação iFlow.

De seguida deve-se seleccionar o fluxo sobre o qual se pretende alterar as permissões. Aqui pode-se visualizar os perfis e permissões já existentes para o fluxo, sendo possível alterar as permissões para um determinado perfil (seleccionando ou não a respectiva permissão e clicando em “Actualizar Permissões”) e/ou remover um determinado perfil. Além disso, é possível adicionar perfis à lista existente, seleccionando o(s) perfil(s) desejado(s) da lista disponibilizada e clicando na seta à direita (ou em alternativa no link “Adicionar Perfil”). De notar que só podem aceder ao fluxo os utilizadores pertencentes aos perfis que tenham permissões para tal. De seguida apresenta-se uma figura com esta área para o exemplo em análise.

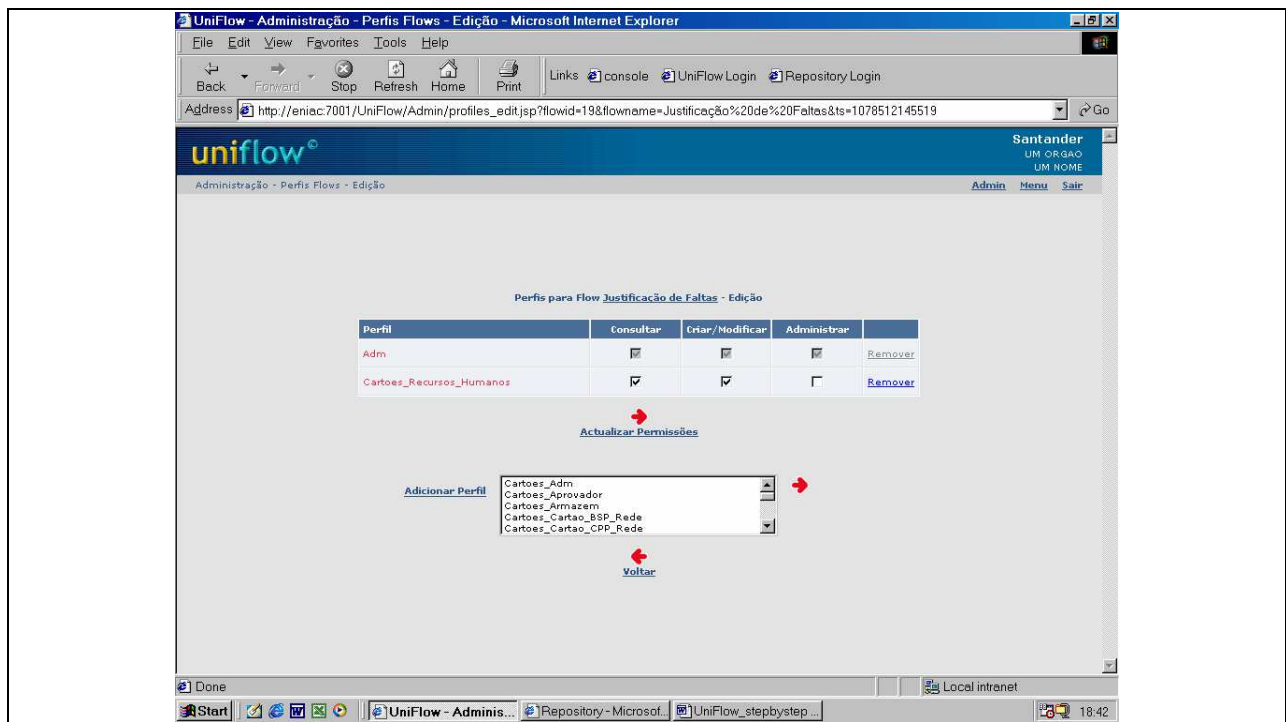


Figura XX – Área de Edição de Permissões por Perfil

De referir que a atribuição da permissão de administração nesta área apenas permite a administração da área de perfis e a administração da área das propriedades para o fluxo em questão, não sendo possível realizar outras tarefas administrativas (tais como deployment).

A partir deste momento passa a ser viável a colocação online do fluxo (deployment), pois já foram efectuadas todas as configurações necessárias, após a qual passa a ser possível a criação de processos nesse fluxo (para o caso dos utilizadores que o criem pertençam a um perfil que tenha permissões para tal). Na criação de um processo, além de todas as variáveis associadas às propriedades do fluxo estarem disponíveis, existem outras que existem por defeito, necessárias ao funcionamento do processo na aplicação, podendo ser utilizadas também no editor. São elas:

- PROCESS_CREATOR: guarda o utilizador que criou/iniciou o processo;
- PROCESS_CREATION_DATE: guarda a data de criação do processo;
- PROCESS_STATE: guarda o estado do processo (aberto ou fechado);
- PROCESS_STATE_DESC: guarda a descrição do estado do processo (aberto, fechado ou cancelado);
- flowid: guarda o identificador do fluxo;
- pid: guarda o identificador do processo;
- userid: guarda o actual ou último dono do processo.

Obviamente, estas variáveis podem ser utilizadas pelos blocos na criação do fluxo no editor, tal como acontece na situação ilustrada na **Erro! A origem da referência não foi encontrada.** com a variável “userid” no exemplo em análise.