

# API Documentation

## API Documentation

June 24, 2011

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Package saip</b>	<b>2</b>
1.1 Modules . . . . .	2
1.2 Variables . . . . .	4
<b>2 Package saip.config</b>	<b>5</b>
2.1 Modules . . . . .	5
2.2 Variables . . . . .	5
<b>3 Module saip.config.app_cfg</b>	<b>6</b>
3.1 Variables . . . . .	6
<b>4 Module saip.config.environment</b>	<b>7</b>
4.1 Variables . . . . .	7
<b>5 Module saip.config.middleware</b>	<b>8</b>
5.1 Functions . . . . .	8
<b>6 Package saip.controllers</b>	<b>9</b>
6.1 Modules . . . . .	9
6.2 Variables . . . . .	10
<b>7 Module saip.controllers.admin_controller</b>	<b>11</b>
7.1 Class AdminController . . . . .	11
7.1.1 Methods . . . . .	11
7.1.2 Class Variables . . . . .	11
<b>8 Module saip.controllers.archivo_controller</b>	<b>12</b>
8.1 Variables . . . . .	12
8.2 Class ArchivoTable . . . . .	12
8.2.1 Class Variables . . . . .	12
8.3 Class ArchivoTableFiller . . . . .	12
8.3.1 Methods . . . . .	13
8.3.2 Class Variables . . . . .	13
8.4 Class AddArchivo . . . . .	13
8.4.1 Class Variables . . . . .	13
8.5 Class ArchivoController . . . . .	13

8.5.1	Methods . . . . .	14
8.5.2	Class Variables . . . . .	14
<b>9</b>	<b>Module saip.controllers.archivo_controller_listado</b>	<b>15</b>
9.1	Variables . . . . .	15
9.2	Class ArchivoTable . . . . .	15
9.2.1	Class Variables . . . . .	15
9.3	Class ArchivoTableFiller . . . . .	16
9.3.1	Methods . . . . .	16
9.3.2	Class Variables . . . . .	16
9.4	Class AddArchivo . . . . .	16
9.4.1	Class Variables . . . . .	16
9.5	Class ArchivoControllerListado . . . . .	17
9.5.1	Methods . . . . .	17
9.5.2	Class Variables . . . . .	17
<b>10</b>	<b>Module saip.controllers.borrado_controller</b>	<b>18</b>
10.1	Variables . . . . .	18
10.2	Class ItemTable . . . . .	18
10.2.1	Class Variables . . . . .	18
10.3	Class ItemTableFiller . . . . .	19
10.3.1	Methods . . . . .	19
10.3.2	Class Variables . . . . .	19
10.4	Class BorradoController . . . . .	19
10.4.1	Methods . . . . .	19
10.4.2	Class Variables . . . . .	20
<b>11</b>	<b>Module saip.controllers.caracteristica_controller</b>	<b>21</b>
11.1	Variables . . . . .	21
11.2	Class CaracteristicaTable . . . . .	21
11.2.1	Class Variables . . . . .	21
11.3	Class CaracteristicaTableFiller . . . . .	22
11.3.1	Methods . . . . .	22
11.3.2	Class Variables . . . . .	22
11.4	Class AddCaracteristica . . . . .	22
11.4.1	Class Variables . . . . .	22
11.5	Class CaracteristicaController . . . . .	23
11.5.1	Methods . . . . .	23
11.5.2	Class Variables . . . . .	23
<b>12</b>	<b>Module saip.controllers.desarrollo_controller</b>	<b>24</b>
12.1	Class DesarrolloController . . . . .	24
12.1.1	Methods . . . . .	24
12.1.2	Class Variables . . . . .	24
<b>13</b>	<b>Module saip.controllers.desarrollo_fase_controller</b>	<b>25</b>
13.1	Variables . . . . .	25
13.2	Class FaseTable . . . . .	25
13.2.1	Class Variables . . . . .	25
13.3	Class FaseTableFiller . . . . .	25
13.3.1	Methods . . . . .	26
13.3.2	Class Variables . . . . .	26

13.4 Class DesarrolloFaseController . . . . .	26
13.4.1 Methods . . . . .	26
13.4.2 Class Variables . . . . .	26
<b>14 Module saip.controllers.desarrollo_proyecto_controller</b>	<b>27</b>
14.1 Variables . . . . .	27
14.2 Class ProyectoTable . . . . .	27
14.2.1 Class Variables . . . . .	27
14.3 Class ProyectoTableFiller . . . . .	27
14.3.1 Methods . . . . .	28
14.3.2 Class Variables . . . . .	28
14.4 Class DesarrolloProyectoController . . . . .	28
14.4.1 Methods . . . . .	28
14.4.2 Class Variables . . . . .	28
<b>15 Module saip.controllers.error</b>	<b>30</b>
15.1 Class ErrorController . . . . .	30
15.1.1 Methods . . . . .	30
15.1.2 Properties . . . . .	30
<b>16 Module saip.controllers.fase_controller</b>	<b>31</b>
16.1 Variables . . . . .	31
16.2 Class ValidarExpresion . . . . .	31
16.2.1 Class Variables . . . . .	31
16.3 Class Unico . . . . .	32
16.4 Class FaseTable . . . . .	32
16.4.1 Class Variables . . . . .	32
16.5 Class FaseTableFiller . . . . .	32
16.5.1 Methods . . . . .	32
16.5.2 Class Variables . . . . .	32
16.6 Class OrdenFieldNew . . . . .	33
16.6.1 Methods . . . . .	33
16.7 Class OrdenFieldEdit . . . . .	33
16.7.1 Methods . . . . .	33
16.8 Class AddFase . . . . .	33
16.8.1 Class Variables . . . . .	34
16.9 Class EditFase . . . . .	34
16.9.1 Class Variables . . . . .	34
16.10 Class FaseEditFiller . . . . .	34
16.10.1 Class Variables . . . . .	34
16.11 Class FaseController . . . . .	35
16.11.1 Methods . . . . .	35
16.11.2 Class Variables . . . . .	36
<b>17 Module saip.controllers.fase_controller_2</b>	<b>37</b>
17.1 Variables . . . . .	37
17.2 Class FaseTable . . . . .	37
17.2.1 Class Variables . . . . .	37
17.3 Class FaseTableFiller . . . . .	37
17.3.1 Methods . . . . .	38
17.3.2 Class Variables . . . . .	38
17.4 Class FaseControllerNuevo . . . . .	38

17.4.1	Methods . . . . .	38
17.4.2	Class Variables . . . . .	39
<b>18</b>	<b>Module saip.controllers.ficha_fase_controller</b>	<b>40</b>
18.1	Variables . . . . .	40
18.2	Class FichaTable . . . . .	40
18.2.1	Class Variables . . . . .	40
18.3	Class FichaTableFiller . . . . .	41
18.3.1	Methods . . . . .	41
18.3.2	Class Variables . . . . .	41
18.4	Class RolesField . . . . .	41
18.5	Class AddFicha . . . . .	42
18.5.1	Class Variables . . . . .	42
18.6	Class FichaFaseController . . . . .	42
18.6.1	Methods . . . . .	42
18.6.2	Class Variables . . . . .	43
<b>19</b>	<b>Module saip.controllers.ficha_proyecto_controller</b>	<b>44</b>
19.1	Variables . . . . .	44
19.2	Class FichaTable . . . . .	44
19.2.1	Class Variables . . . . .	44
19.3	Class FichaTableFiller . . . . .	45
19.3.1	Methods . . . . .	45
19.3.2	Class Variables . . . . .	45
19.4	Class RolesField . . . . .	45
19.5	Class AddFicha . . . . .	46
19.5.1	Class Variables . . . . .	46
19.6	Class FichaProyectoController . . . . .	46
19.6.1	Methods . . . . .	46
19.6.2	Class Variables . . . . .	47
<b>20</b>	<b>Module saip.controllers.ficha_sistema_controller</b>	<b>48</b>
20.1	Variables . . . . .	48
20.2	Class FichaTable . . . . .	48
20.2.1	Class Variables . . . . .	48
20.3	Class FichaTableFiller . . . . .	49
20.3.1	Methods . . . . .	49
20.3.2	Class Variables . . . . .	49
20.4	Class RolesField . . . . .	49
20.5	Class AddFicha . . . . .	50
20.5.1	Class Variables . . . . .	50
20.6	Class FichaSistemaController . . . . .	50
20.6.1	Methods . . . . .	50
20.6.2	Class Variables . . . . .	51
<b>21</b>	<b>Module saip.controllers.ficha_usuario_controller</b>	<b>52</b>
21.1	Variables . . . . .	52
21.2	Class FichaTable . . . . .	52
21.2.1	Class Variables . . . . .	52
21.3	Class FichaTableFiller . . . . .	52
21.3.1	Methods . . . . .	53
21.3.2	Class Variables . . . . .	53

21.4 Class FichaUsuarioController . . . . .	53
21.4.1 Methods . . . . .	53
21.4.2 Class Variables . . . . .	54
<b>22 Module saip.controllers.gestion_controller</b>	<b>55</b>
22.1 Class GestionController . . . . .	55
22.1.1 Methods . . . . .	55
22.1.2 Class Variables . . . . .	55
<b>23 Module saip.controllers.gestion_fase_controller</b>	<b>56</b>
23.1 Variables . . . . .	56
23.2 Class FaseTable . . . . .	56
23.2.1 Class Variables . . . . .	56
23.3 Class FaseTableFiller . . . . .	56
23.3.1 Methods . . . . .	57
23.3.2 Class Variables . . . . .	57
23.4 Class GestionFaseController . . . . .	57
23.4.1 Methods . . . . .	57
23.4.2 Class Variables . . . . .	57
<b>24 Module saip.controllers.gestion_proyecto_controller</b>	<b>59</b>
24.1 Variables . . . . .	59
24.2 Class ProyectoTable . . . . .	59
24.2.1 Class Variables . . . . .	59
24.3 Class ProyectoTableFiller . . . . .	59
24.3.1 Methods . . . . .	60
24.3.2 Class Variables . . . . .	60
24.4 Class GestionProyectoController . . . . .	60
24.4.1 Methods . . . . .	61
24.4.2 Class Variables . . . . .	61
<b>25 Module saip.controllers.item_controller</b>	<b>62</b>
25.1 Variables . . . . .	62
25.2 Class ItemTable . . . . .	62
25.2.1 Class Variables . . . . .	62
25.3 Class ItemTableFiller . . . . .	63
25.3.1 Methods . . . . .	63
25.3.2 Class Variables . . . . .	63
25.4 Class AddItem . . . . .	63
25.4.1 Class Variables . . . . .	63
25.5 Class EditItem . . . . .	64
25.5.1 Class Variables . . . . .	64
25.6 Class ItemEditFiller . . . . .	64
25.6.1 Class Variables . . . . .	64
25.7 Class ItemController . . . . .	64
25.7.1 Methods . . . . .	65
25.7.2 Class Variables . . . . .	66
<b>26 Module saip.controllers.item_controller_listado</b>	<b>68</b>
26.1 Variables . . . . .	68
26.2 Class ItemTable . . . . .	68
26.2.1 Class Variables . . . . .	68

26.3	Class ItemTableFiller . . . . .	69
26.3.1	Methods . . . . .	69
26.3.2	Class Variables . . . . .	69
26.4	Class AddItem . . . . .	69
26.4.1	Class Variables . . . . .	69
26.5	Class EditItem . . . . .	70
26.5.1	Class Variables . . . . .	70
26.6	Class ItemEditFiller . . . . .	70
26.6.1	Class Variables . . . . .	70
26.7	Class ItemControllerListado . . . . .	70
26.7.1	Methods . . . . .	71
26.7.2	Class Variables . . . . .	71
<b>27</b>	<b>Module saip.controllers.linea_base_controller</b>	<b>72</b>
27.1	Functions . . . . .	72
27.2	Variables . . . . .	72
27.3	Class LineaBaseTable . . . . .	72
27.3.1	Class Variables . . . . .	72
27.4	Class LineaBaseTableFiller . . . . .	73
27.4.1	Methods . . . . .	73
27.4.2	Class Variables . . . . .	73
27.5	Class ItemsField . . . . .	73
27.5.1	Methods . . . . .	73
27.5.2	Class Variables . . . . .	73
27.6	Class AddLineaBase . . . . .	74
27.6.1	Class Variables . . . . .	74
27.7	Class LineaBaseController . . . . .	74
27.7.1	Methods . . . . .	74
27.7.2	Class Variables . . . . .	75
<b>28</b>	<b>Module saip.controllers.proyecto_controller</b>	<b>76</b>
28.1	Variables . . . . .	76
28.2	Class Unico . . . . .	76
28.3	Class ValidarExpresion . . . . .	76
28.3.1	Class Variables . . . . .	77
28.4	Class ProyectoTable . . . . .	77
28.4.1	Class Variables . . . . .	77
28.5	Class ProyectoTableFiller . . . . .	77
28.5.1	Methods . . . . .	77
28.5.2	Class Variables . . . . .	78
28.6	Class NroValido . . . . .	78
28.7	Class AddProyecto . . . . .	78
28.7.1	Class Variables . . . . .	78
28.8	Class CantidadFasesField . . . . .	79
28.8.1	Methods . . . . .	79
28.9	Class EditProyecto . . . . .	79
28.9.1	Class Variables . . . . .	79
28.10	Class ProyectoEditFiller . . . . .	80
28.10.1	Class Variables . . . . .	80
28.11	Class ProyectoController . . . . .	80
28.11.1	Methods . . . . .	80
28.11.2	Class Variables . . . . .	81

<b>29</b>	<b>Module saip.controllers.proyecto_controller_2</b>	<b>82</b>
29.1	Variables . . . . .	82
29.2	Class ProyectoTable . . . . .	82
29.2.1	Class Variables . . . . .	82
29.3	Class ProyectoTableFiller . . . . .	82
29.3.1	Methods . . . . .	83
29.3.2	Class Variables . . . . .	83
29.4	Class ProyectoControllerNuevo . . . . .	83
29.4.1	Methods . . . . .	83
29.4.2	Class Variables . . . . .	84
<b>30</b>	<b>Module saip.controllers.relacion_controller</b>	<b>85</b>
30.1	Variables . . . . .	85
30.2	Class RelacionTable . . . . .	85
30.2.1	Class Variables . . . . .	85
30.3	Class RelacionTableFiller . . . . .	85
30.3.1	Methods . . . . .	86
30.3.2	Class Variables . . . . .	86
30.4	Class AddRelacion . . . . .	86
30.4.1	Class Variables . . . . .	86
30.5	Class RelacionController . . . . .	87
30.5.1	Methods . . . . .	87
30.5.2	Class Variables . . . . .	87
<b>31</b>	<b>Module saip.controllers.relacion_controller_listado</b>	<b>89</b>
31.1	Variables . . . . .	89
31.2	Class RelacionTable . . . . .	89
31.2.1	Class Variables . . . . .	89
31.3	Class RelacionTableFiller . . . . .	90
31.3.1	Methods . . . . .	90
31.3.2	Class Variables . . . . .	90
31.4	Class AddRelacion . . . . .	90
31.4.1	Class Variables . . . . .	90
31.5	Class RelacionControllerListado . . . . .	91
31.5.1	Methods . . . . .	91
31.5.2	Class Variables . . . . .	91
<b>32</b>	<b>Module saip.controllers.revision_controller</b>	<b>92</b>
32.1	Variables . . . . .	92
32.2	Class RevisionTable . . . . .	92
32.2.1	Class Variables . . . . .	92
32.3	Class RevisionTableFiller . . . . .	92
32.3.1	Methods . . . . .	93
32.3.2	Class Variables . . . . .	93
32.4	Class RevisionController . . . . .	93
32.4.1	Methods . . . . .	93
32.4.2	Class Variables . . . . .	94
<b>33</b>	<b>Module saip.controllers.rol_controller</b>	<b>95</b>
33.1	Variables . . . . .	95
33.2	Class ValidarExpresion . . . . .	95
33.2.1	Class Variables . . . . .	95

33.3	Class RolTable . . . . .	96
33.3.1	Class Variables . . . . .	96
33.4	Class RolTableFiller . . . . .	96
33.4.1	Methods . . . . .	96
33.4.2	Class Variables . . . . .	96
33.5	Class AddRol . . . . .	97
33.5.1	Class Variables . . . . .	97
33.6	Class PermisosField . . . . .	97
33.6.1	Methods . . . . .	97
33.6.2	Class Variables . . . . .	97
33.7	Class EditRol . . . . .	98
33.7.1	Class Variables . . . . .	98
33.8	Class RolEditFiller . . . . .	98
33.8.1	Class Variables . . . . .	98
33.9	Class RolController . . . . .	98
33.9.1	Methods . . . . .	98
33.9.2	Class Variables . . . . .	99
<b>34</b>	<b>Module saip.controllers.root</b>	<b>100</b>
34.1	Class RootController . . . . .	100
34.1.1	Methods . . . . .	100
34.1.2	Class Variables . . . . .	101
<b>35</b>	<b>Module saip.controllers.secure</b>	<b>102</b>
35.1	Class SecureController . . . . .	102
35.1.1	Methods . . . . .	102
35.1.2	Class Variables . . . . .	102
<b>36</b>	<b>Module saip.controllers.template</b>	<b>103</b>
36.1	Class TemplateController . . . . .	103
36.1.1	Methods . . . . .	103
<b>37</b>	<b>Module saip.controllers.tipo_item_controller</b>	<b>104</b>
37.1	Variables . . . . .	104
37.2	Class ValidarExpresion . . . . .	104
37.2.1	Class Variables . . . . .	104
37.3	Class Unico . . . . .	105
37.4	Class CodigoUnico . . . . .	105
37.5	Class TipoItemTable . . . . .	105
37.5.1	Class Variables . . . . .	105
37.6	Class TipoItemTableFiller . . . . .	105
37.6.1	Methods . . . . .	106
37.6.2	Class Variables . . . . .	106
37.7	Class AddTipoItem . . . . .	106
37.7.1	Class Variables . . . . .	106
37.8	Class EditTipoItem . . . . .	107
37.8.1	Class Variables . . . . .	107
37.9	Class TipoItemEditFiller . . . . .	107
37.9.1	Class Variables . . . . .	107
37.10	Class TipoItemController . . . . .	107
37.10.1	Methods . . . . .	108
37.10.2	Class Variables . . . . .	108



<b>38 Module saip.controllers.tipo_item_controller_nuevo</b>	<b>109</b>
38.1 Variables . . . . .	109
38.2 Class TipoItemTable . . . . .	109
38.2.1 Class Variables . . . . .	109
38.3 Class TipoItemTableFiller . . . . .	109
38.3.1 Methods . . . . .	110
38.3.2 Class Variables . . . . .	110
38.4 Class TipoItemControllerNuevo . . . . .	110
38.4.1 Methods . . . . .	110
38.4.2 Class Variables . . . . .	111
<b>39 Module saip.controllers.usuario_controller</b>	<b>112</b>
39.1 Variables . . . . .	112
39.2 Class UsuarioTable . . . . .	112
39.2.1 Class Variables . . . . .	112
39.3 Class UsuarioTableFiller . . . . .	113
39.3.1 Methods . . . . .	113
39.3.2 Class Variables . . . . .	113
39.4 Class Unico . . . . .	113
39.5 Class AddUsuario . . . . .	113
39.5.1 Class Variables . . . . .	113
39.6 Class EditUsuario . . . . .	114
39.6.1 Class Variables . . . . .	114
39.7 Class UsuarioEditFiller . . . . .	114
39.7.1 Class Variables . . . . .	114
39.8 Class UsuarioController . . . . .	115
39.8.1 Methods . . . . .	115
39.8.2 Class Variables . . . . .	115
<b>40 Module saip.controllers.version_controller</b>	<b>117</b>
40.1 Variables . . . . .	117
40.2 Class ItemTable . . . . .	117
40.2.1 Class Variables . . . . .	117
40.3 Class ItemTableFiller . . . . .	118
40.3.1 Methods . . . . .	118
40.3.2 Class Variables . . . . .	118
40.4 Class VersionController . . . . .	118
40.4.1 Methods . . . . .	118
40.4.2 Class Variables . . . . .	120
<b>41 Package saip.lib</b>	<b>121</b>
41.1 Modules . . . . .	121
41.2 Variables . . . . .	121
<b>42 Module saip.lib.app_globals</b>	<b>122</b>
42.1 Class Globals . . . . .	122
42.1.1 Methods . . . . .	122
42.1.2 Properties . . . . .	122
<b>43 Module saip.lib.auth</b>	<b>123</b>
43.1 Class TieneAlgúnPermiso . . . . .	123
43.1.1 Methods . . . . .	123

43.1.2 Class Variables . . . . .	123
43.2 Class TienePermiso . . . . .	123
43.2.1 Methods . . . . .	124
43.2.2 Class Variables . . . . .	124
<b>44 Module saip.lib.base</b>	<b>125</b>
44.1 Class BaseController . . . . .	125
44.1.1 Methods . . . . .	125
<b>45 Module saip.lib.func</b>	<b>126</b>
45.1 Functions . . . . .	126
<b>46 Module saip.lib.helpers</b>	<b>131</b>
<b>47 Package saip.model</b>	<b>132</b>
47.1 Modules . . . . .	132
47.2 Functions . . . . .	132
47.3 Variables . . . . .	132
<b>48 Module saip.model.app</b>	<b>133</b>
48.1 Class Proyecto . . . . .	133
48.1.1 Class Variables . . . . .	133
48.2 Class Fase . . . . .	134
48.2.1 Class Variables . . . . .	134
48.3 Class TipoItem . . . . .	134
48.3.1 Class Variables . . . . .	134
48.4 Class Caracteristica . . . . .	135
48.4.1 Class Variables . . . . .	135
48.5 Class LineaBase . . . . .	135
48.5.1 Class Variables . . . . .	135
48.6 Class Item . . . . .	136
48.6.1 Class Variables . . . . .	136
48.7 Class Archivo . . . . .	137
48.7.1 Class Variables . . . . .	137
48.8 Class Item_Archivo . . . . .	137
48.8.1 Class Variables . . . . .	137
48.9 Class Relacion . . . . .	138
48.9.1 Class Variables . . . . .	138
48.10 Class Revision . . . . .	139
48.10.1 Class Variables . . . . .	139
<b>49 Module saip.model.auth</b>	<b>140</b>
49.1 Class Ficha . . . . .	140
49.1.1 Methods . . . . .	140
49.1.2 Class Variables . . . . .	140
49.2 Class Rol . . . . .	141
49.2.1 Class Variables . . . . .	141
49.3 Class Usuario . . . . .	142
49.3.1 Methods . . . . .	142
49.3.2 Class Variables . . . . .	142
49.4 Class Permiso . . . . .	143
49.4.1 Class Variables . . . . .	143

<b>50 Package saip.templates</b>	<b>144</b>
50.1 Variables . . . . .	144
<b>51 Package saip.tests</b>	<b>145</b>
51.1 Modules . . . . .	145
51.2 Functions . . . . .	145
51.3 Variables . . . . .	145
51.4 Class TestController . . . . .	145
51.4.1 Methods . . . . .	146
51.4.2 Properties . . . . .	146
51.4.3 Class Variables . . . . .	146
<b>52 Package saip.tests.models</b>	<b>147</b>
52.1 Modules . . . . .	147
52.2 Class ModelTest . . . . .	147
52.2.1 Methods . . . . .	147
52.2.2 Properties . . . . .	148
52.2.3 Class Variables . . . . .	148
<b>53 Module saip.tests.models.test_proyecto</b>	<b>149</b>
53.1 Class TestProyecto . . . . .	149
53.1.1 Methods . . . . .	149
53.1.2 Properties . . . . .	149
53.1.3 Class Variables . . . . .	149
<b>54 Module saip.tests.models.test_rol</b>	<b>150</b>
54.1 Class TestRol . . . . .	150
54.1.1 Methods . . . . .	150
54.1.2 Properties . . . . .	150
54.1.3 Class Variables . . . . .	150
<b>55 Module saip.tests.models.test_usuario</b>	<b>151</b>
55.1 Class TestUsuario . . . . .	151
55.1.1 Methods . . . . .	151
55.1.2 Properties . . . . .	151
55.1.3 Class Variables . . . . .	151
<b>56 Package saip.websetup</b>	<b>152</b>
56.1 Modules . . . . .	152
56.2 Functions . . . . .	152
<b>57 Module saip.websetup.bootstrap</b>	<b>153</b>
57.1 Functions . . . . .	153
<b>58 Module saip.websetup.schema</b>	<b>154</b>
58.1 Functions . . . . .	154

# 1 Package saip

The SAIP package

## 1.1 Modules

- **config** (*Section 2, p. 5*)
  - **app\_cfg**: Global configuration file for TG2-specific settings in SAIP.  
(*Section 3, p. 6*)
  - **environment**: WSGI environment setup for SAIP.  
(*Section 4, p. 7*)
  - **middleware**: WSGI middleware initialization for the SAIP application.  
(*Section 5, p. 8*)
- **controllers**: Controllers for the SAIP application.  
(*Section 6, p. 9*)
  - **admin\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(*Section 7, p. 11*)
  - **archivo\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(*Section 8, p. 12*)
  - **archivo\_controller\_listado**: Módulo que define el controlador de listado de archivos de un ítem borrado o de una versión anterior.  
(*Section 9, p. 15*)
  - **borrado\_controller**: Módulo que define el controlador de ítems borrados.  
(*Section 10, p. 18*)
  - **caracteristica\_controller**: Controlador de características de un tipo de ítem en el módulo de administración.  
(*Section 11, p. 21*)
  - **desarrollo\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(*Section 12, p. 24*)
  - **desarrollo\_fase\_controller**: Módulo que define el controlador de fases del módulo de desarrollo.  
(*Section 13, p. 25*)
  - **desarrollo\_proyecto\_controller**: Controlador de proyectos en el módulo de desarrollo.  
(*Section 14, p. 27*)
  - **error**: Error controller  
(*Section 15, p. 30*)
  - **fase\_controller**: Módulo que define el controlador de fases del módulo de administración.  
(*Section 16, p. 31*)
  - **fase\_controller\_2**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(*Section 17, p. 37*)
  - **ficha\_fase\_controller**: Controlador de Fichas de fase en el módulo de administración.  
(*Section 18, p. 40*)
  - **ficha\_proyecto\_controller**: Controlador de Fichas de proyecto en el módulo de administración.  
(*Section 19, p. 44*)
  - **ficha\_sistema\_controller**: Controlador de Fichas de sistema en el módulo de administración.  
(*Section 20, p. 48*)
  - **ficha\_usuario\_controller**: Módulo que define el controlador de fichas en el menú de usuario.

- (Section 21, p. 52)
  - **gestion\_controller**: Módulo que define el controlador del módulo de gestión.  
(Section 22, p. 55)
  - **gestion\_fase\_controller**: Módulo que define el controlador de fases del módulo de gestión.  
(Section 23, p. 56)
  - **gestion\_proyecto\_controller**: Controlador de proyectos en el módulo de gestión.  
(Section 24, p. 59)
  - **item\_controller**: Controlador de ítems en el módulo de desarrollo.  
(Section 25, p. 62)
  - **item\_controller\_listado**: Módulo que define el controlador de listado de ítems pertenecientes a una línea base  
(Section 26, p. 68)
  - **linea\_base\_controller**: Módulo que define el controlador de líneas base.  
(Section 27, p. 72)
  - **proyecto\_controller**: Controlador de proyectos en el módulo de administración.  
(Section 28, p. 76)
  - **proyecto\_controller\_2**: Controlador de proyectos en el módulo de administración utilizado para la importación de fases o tipos de ítem.  
(Section 29, p. 82)
  - **relacion\_controller**: Módulo que define el controlador de relaciones del módulo de desarrollo.  
(Section 30, p. 85)
  - **relacion\_controller\_listado**: Módulo que define el controlador de listado de relaciones en el menú de versión.  
(Section 31, p. 89)
  - **revision\_controller**: Controlador de revisiones de un ítem dado en el módulo de desarrollo.  
(Section 32, p. 92)
  - **rol\_controller**: Controlador de proyectos en el módulo de administración.  
(Section 33, p. 95)
  - **root**: Main Controller  
(Section 34, p. 100)
  - **secure**: Sample controller with all its actions protected.  
(Section 35, p. 102)
  - **template**: Fallback controller.  
(Section 36, p. 103)
  - **tipo\_item\_controller**: Controlador de tipos de ítem en el módulo de administración.  
(Section 37, p. 104)
  - **tipo\_item\_controller\_nuevo**: Controlador de tipos de ítem en el módulo de administración utilizado para la importación de tipos de ítem.  
(Section 38, p. 109)
  - **usuario\_controller**: Módulo que define el controlador de usuarios.  
(Section 39, p. 112)
  - **version\_controller**: Módulo que define el controlador de versiones anteriores de un ítem.  
(Section 40, p. 117)
- **lib** (Section 41, p. 121)
  - **app\_globals**: The application's Globals object  
(Section 42, p. 122)
  - **auth**: Módulo que provee los predicates checkers para controlar la autorización de los usuarios para el acceso a los recursos del sistema.  
(Section 43, p. 123)
  - **base**: The base Controller API.  
(Section 44, p. 125)

- **func**: Módulo que provee funciones varias para la utilización en los controladores.  
(Section 45, p. 126)
- **helpers**: WebHelpers used in SAIP.  
(Section 46, p. 131)
- **model**: The application's model objects  
(Section 47, p. 132)
  - **app**: Módulo que define las clases del modelo del sistema no relacionadas a la autenticación ni a la autorización.  
(Section 48, p. 133)
  - **auth**: Módulo que define las clases del modelo del sistema relacionadas a la autenticación y a la autorización.  
(Section 49, p. 140)
- **templates**: Templates package for the application.  
(Section 50, p. 144)
- **tests**: Unit and functional test suite for SAIP.  
(Section 51, p. 145)
  - **models**: Unit test suite for the models of the application.  
(Section 52, p. 147)
    - \* **test\_proyecto** (Section 53, p. 149)
    - \* **test\_rol** (Section 54, p. 150)
    - \* **test\_usuario** (Section 55, p. 151)
- **websetup**: Setup the SAIP application  
(Section 56, p. 152)
  - **bootstrap**: Setup the SAIP application  
(Section 57, p. 153)
  - **schema**: Setup the SAIP application  
(Section 58, p. 154)

## 1.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 2 Package saip.config

### 2.1 Modules

- **app\_cfg**: Global configuration file for TG2-specific settings in SAIP.  
(Section 3, p. 6)
- **environment**: WSGI environment setup for SAIP.  
(Section 4, p. 7)
- **middleware**: WSGI middleware initialization for the SAIP application.  
(Section 5, p. 8)

### 2.2 Variables

Name	Description
__package__	<b>Value:</b> None

### 3 Module *saip.config.app\_cfg*

Global configuration file for TG2-specific settings in SAIP.

This file complements `development/deployment.ini`.

Please note that **\*\*all the argument values are strings\*\***. If you want to convert them into boolean, for example, you should use the `:func:'paste.deploy.converters.asbool'` function, as in:

```
from paste.deploy.converters import asbool
setting = asbool(global_conf.get('the_setting'))
```

#### 3.1 Variables

Name	Description
<code>base_config</code>	<b>Value:</b> <code>AppConfig()</code>



## 4 Module `saip.config.environment`

WSGI environment setup for SAIP.

### 4.1 Variables

Name	Description
<code>load_environment</code>	<b>Value:</b> <code>base.config.make_load_environment()</code>

## 5 Module *saip.config.middleware*

WSGI middleware initialization for the SAIP application.

### 5.1 Functions

```
make_app(global_conf, full_stack=True, **app_conf)
```

Set SAIP up with the settings found in the PasteDeploy configuration file used.

```
:param global_conf: The global settings for SAIP (those
    defined under the '[DEFAULT]' section).
:type global_conf: dict
:param full_stack: Should the whole TG2 stack be set up?
:type full_stack: str or bool
:return: The SAIP application with all the relevant middleware
    loaded.
```

This is the PasteDeploy factory for the SAIP application.

'app\_conf' contains all the application-specific settings (those defined under '[app:main]').

## 6 Package saip.controllers

Controllers for the SAIP application.

### 6.1 Modules

- **admin\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(Section 7, p. 11)
- **archivo\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(Section 8, p. 12)
- **archivo\_controller\_listado**: Módulo que define el controlador de listado de archivos de un ítem borrado o de una versión anterior.  
(Section 9, p. 15)
- **borrado\_controller**: Módulo que define el controlador de ítems borrados.  
(Section 10, p. 18)
- **caracteristica\_controller**: Controlador de características de un tipo de ítem en el módulo de administración.  
(Section 11, p. 21)
- **desarrollo\_controller**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(Section 12, p. 24)
- **desarrollo\_fase\_controller**: Módulo que define el controlador de fases del módulo de desarrollo.  
(Section 13, p. 25)
- **desarrollo\_proyecto\_controller**: Controlador de proyectos en el módulo de desarrollo.  
(Section 14, p. 27)
- **error**: Error controller  
(Section 15, p. 30)
- **fase\_controller**: Módulo que define el controlador de fases del módulo de administración.  
(Section 16, p. 31)
- **fase\_controller\_2**: Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.  
(Section 17, p. 37)
- **ficha\_fase\_controller**: Controlador de Fichas de fase en el módulo de administración.  
(Section 18, p. 40)
- **ficha\_proyecto\_controller**: Controlador de Fichas de proyecto en el módulo de administración.  
(Section 19, p. 44)
- **ficha\_sistema\_controller**: Controlador de Fichas de sistema en el módulo de administración.  
(Section 20, p. 48)
- **ficha\_usuario\_controller**: Módulo que define el controlador de fichas en el menú de usuario.  
(Section 21, p. 52)
- **gestion\_controller**: Módulo que define el controlador del módulo de gestión.  
(Section 22, p. 55)
- **gestion\_fase\_controller**: Módulo que define el controlador de fases del módulo de gestión.  
(Section 23, p. 56)
- **gestion\_proyecto\_controller**: Controlador de proyectos en el módulo de gestión.  
(Section 24, p. 59)
- **item\_controller**: Controlador de ítems en el módulo de desarrollo.  
(Section 25, p. 62)

- **item\_controller\_listado**: Módulo que define el controlador de listado de ítems pertenecientes a una línea base  
(Section 26, p. 68)
- **linea\_base\_controller**: Módulo que define el controlador de líneas base.  
(Section 27, p. 72)
- **proyecto\_controller**: Controlador de proyectos en el módulo de administración.  
(Section 28, p. 76)
- **proyecto\_controller\_2**: Controlador de proyectos en el módulo de administración utilizado para la importación de fases o tipos de ítem.  
(Section 29, p. 82)
- **relacion\_controller**: Módulo que define el controlador de relaciones del módulo de desarrollo.  
(Section 30, p. 85)
- **relacion\_controller\_listado**: Módulo que define el controlador de listado de relaciones en el menú de versión.  
(Section 31, p. 89)
- **revision\_controller**: Controlador de revisiones de un ítem dado en el módulo de desarrollo.  
(Section 32, p. 92)
- **rol\_controller**: Controlador de proyectos en el módulo de administración.  
(Section 33, p. 95)
- **root**: Main Controller  
(Section 34, p. 100)
- **secure**: Sample controller with all its actions protected.  
(Section 35, p. 102)
- **template**: Fallback controller.  
(Section 36, p. 103)
- **tipo\_item\_controller**: Controlador de tipos de ítem en el módulo de administración.  
(Section 37, p. 104)
- **tipo\_item\_controller\_nuevo**: Controlador de tipos de ítem en el módulo de administración utilizado para la importación de tipos de ítem.  
(Section 38, p. 109)
- **usuario\_controller**: Módulo que define el controlador de usuarios.  
(Section 39, p. 112)
- **version\_controller**: Módulo que define el controlador de versiones anteriores de un ítem.  
(Section 40, p. 117)

## 6.2 Variables

Name	Description
__package__	<b>Value:</b> None

## 7 Module saip.controllers.admin\_controller

Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.

### Author:

- Alejandro Arce<sup>1</sup>
- Gabriel Caroni<sup>2</sup>
- Rodrigo Parra<sup>3</sup>

### 7.1 Class AdminController



Controlador del módulo de administración.

#### 7.1.1 Methods

**index**(*self*)

Muestra la página del módulo de administración.

*Inherited from saip.lib.base.BaseController(Section 44.1)*

`__call__()`

#### 7.1.2 Class Variables

Name	Description
proyectos	<b>Value:</b> ProyectoController(DBSession)
roles	<b>Value:</b> RolController(DBSession)
usuarios	<b>Value:</b> UsuarioController(DBSession)
responsables	<b>Value:</b> FichaSistemaController(DBSession)

<sup>1</sup><mailto:alearce07@gmail.com>

<sup>2</sup><mailto:gabrielcaroni@gmail.com>

<sup>3</sup><mailto:rodpar07@gmail.com>

## 8 Module *saip.controllers.archivo\_controller*

Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.

### Author:

- Alejandro Arce<sup>4</sup>
- Gabriel Caroni<sup>5</sup>
- Rodrigo Parra<sup>6</sup>

### 8.1 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
archivo_table	<b>Value:</b> ArchivoTable(DBSession)
archivo_table_filler	<b>Value:</b> ArchivoTableFiller(DBSession)
add_archivo_form	<b>Value:</b> AddArchivo(DBSession)

### 8.2 Class *ArchivoTable*

sprox.tablebase.TableBase  saip.controllers.archivo\_controller.ArchivoTable

#### 8.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Archivo
__omit_fields__	<b>Value:</b> ['contenido', 'items']

### 8.3 Class *ArchivoTableFiller*

sprox.fillerbase.TableFiller  saip.controllers.archivo\_controller.ArchivoTableFiller

<sup>4</sup><mailto:alearce07@gmail.com>

<sup>5</sup><mailto:gabrielcaroni@gmail.com>

<sup>6</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas de archivos.

### 8.3.1 Methods

**`--actions--(self, obj)`**

Define las acciones posibles para cada archivo.

**`init(self, buscado, id_item, version)`**

### 8.3.2 Class Variables

Name	Description
<code>--model--</code>	<b>Value:</b> Archivo
<code>--omit_fields--</code>	<b>Value:</b> ['contenido']
<code>buscado</code>	<b>Value:</b> ""
<code>id_item</code>	<b>Value:</b> ""
<code>version</code>	<b>Value:</b> ""

## 8.4 Class AddArchivo

sprox.formbase.AddRecordForm

saip.controllers.archivo\_controller.AddArchivo

Define el formato de la tabla para agregar archivos.

### 8.4.1 Class Variables

Name	Description
<code>--model--</code>	<b>Value:</b> Archivo
<code>--omit_fields--</code>	<b>Value:</b> ['id', 'items', 'nombre']
<code>contenido</code>	<b>Value:</b> FileField('archivo')

## 8.5 Class ArchivoController

tgext.crud.CrudRestController

saip.controllers.archivo\_controller.ArchivoController

Controlador del modelo Archivo.

### 8.5.1 Methods

**get\_one**(*self*, *archivo\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista los archivos de acuerdo a lo establecido en `archivo_controller.ArchivoTableFiller._do_get_provider_count_and_objs`.

**descargar**(*self*, \**args*, \*\**kw*)

Realiza la descarga del archivo en la computadora del cliente.

**new**(*self*, \**args*, \*\**kw*)

Permite la creación de un nuevo archivo para un ítem.

**buscar**(*self*, \*\**kw*)

Lista los archivos de acuerdo a un criterio de búsqueda introducido por el usuario.

**crear\_version**(*self*, *it*)

Crea una nueva version del ítem al cual se añade el archivo.

**post**(*self*, \*\**kw*)

Registra el nuevo archivo creado.

**post\_delete**(*self*, \**args*, \*\**kw*)

Realiza la eliminación del archivo de la base de datos, creando una nueva versión del ítem correspondiente.

### 8.5.2 Class Variables

Name	Description
<code>model</code>	<b>Value:</b> Archivo
<code>table</code>	<b>Value:</b> ArchivoTable(DBSession)
<code>table_filler</code>	<b>Value:</b> ArchivoTableFiller(DBSession)
<code>new_form</code>	<b>Value:</b> AddArchivo(DBSession)



## 9 Module `saip.controllers.archivo_controller_listado`

Módulo que define el controlador de listado de archivos de un ítem borrado o de una versión anterior.

### Author:

- Alejandro Arce<sup>7</sup>
- Gabriel Caroni<sup>8</sup>
- Rodrigo Parra<sup>9</sup>

### 9.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>archivo_table</code>	<b>Value:</b> <code>ArchivoTable(DBSession)</code>
<code>archivo_table_filler</code>	<b>Value:</b> <code>ArchivoTableFiller(DBSession)</code>
<code>add_archivo_form</code>	<b>Value:</b> <code>AddArchivo(DBSession)</code>

### 9.2 Class `ArchivoTable`

`sprox.tablebase.TableBase` — `saip.controllers.archivo_controller_listado.ArchivoTable`

#### 9.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Archivo</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['contenido', 'items', '__actions__']</code>

<sup>7</sup><mailto:alearce07@gmail.com>

<sup>8</sup><mailto:gabrielcaroni@gmail.com>

<sup>9</sup><mailto:rodpar07@gmail.com>

### 9.3 Class ArchivoTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.archivo\_controller\_listado.ArchivoTableFiller**

Clase que se utiliza para llenar las tablas de listado de archivos de ítems borrados o de versiones anteriores.

#### 9.3.1 Methods

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_item</i> , <i>version</i> )
--

#### 9.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Archivo
<code>__omit_fields__</code>	<b>Value:</b> ['contenido']
<code>buscado</code>	<b>Value:</b> ""
<code>id_item</code>	<b>Value:</b> ""
<code>version</code>	<b>Value:</b> ""

### 9.4 Class AddArchivo

sprox.formbase.AddRecordForm —  
**saip.controllers.archivo\_controller\_listado.AddArchivo**

#### 9.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Archivo
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'items', 'nombre']
<code>contenido</code>	<b>Value:</b> FileField('archivo')

## 9.5 Class `ArchivoControllerListado`

`tgext.crud.CrudRestController`

`saip.controllers.archivo_controller_listado.ArchivoControllerLis`

Controlador del modelo Archivo para ítems borrados o de versiones anteriores.

### 9.5.1 Methods

**`get_one(self, archivo_id)`**

**`get_all(self, *args, **kw)`**

Lista los archivos de acuerdo a lo establecido en `archivo_controller_listado.ArchivoTableFiller._do_get_provider_count_and_objs`.

**`buscar(self, **kw)`**

Lista los archivos de acuerdo a un criterio de búsqueda introducido por el usuario.

### 9.5.2 Class Variables

Name	Description
<code>model</code>	<b>Value:</b> <code>Archivo</code>
<code>table</code>	<b>Value:</b> <code>ArchivoTable(DBSession)</code>
<code>table_filler</code>	<b>Value:</b> <code>ArchivoTableFiller(DBSession)</code>
<code>new_form</code>	<b>Value:</b> <code>AddArchivo(DBSession)</code>

## 10 Module `saip.controllers.borrado_controller`

Módulo que define el controlador de ítems borrados.

### Author:

- Alejandro Arce<sup>10</sup>
- Gabriel Caroni<sup>11</sup>
- Rodrigo Parra<sup>12</sup>

### 10.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>item_table</code>	<b>Value:</b> <code>ItemTable(DBSession)</code>
<code>item_table_filler</code>	<b>Value:</b> <code>ItemTableFiller(DBSession)</code>

### 10.2 Class `ItemTable`

`sprox.tablebase.TableBase` — `saip.controllers.borrado_controller.ItemTable`

Define el formato de la tabla

#### 10.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'id_tipo_item', 'id_fase', 'id_linea_base', 'archi...</code>

<sup>10</sup><mailto:alearce07@gmail.com>

<sup>11</sup><mailto:gabrielcaroni@gmail.com>

<sup>12</sup><mailto:rodpar07@gmail.com>

### 10.3 Class ItemTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.borrado\_controller.ItemTableFiller**

Clase que se utiliza para llenar las tablas de ítems borrados a ser potencialmente recuperados.

#### 10.3.1 Methods

<b>tipo_item</b> ( <i>self</i> , <i>obj</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ítem borrado.
--

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_fase</i> )
---

#### 10.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Item
<code>buscado</code>	<b>Value:</b> ""
<code>id_fase</code>	<b>Value:</b> ""

### 10.4 Class BorradoController

tgext.crud.CrudRestController —  
**saip.controllers.borrado\_controller.BorradoController**

Controlador del modelo Item para los que se encuentran borrados.

#### 10.4.1 Methods

<b>get_one</b> ( <i>self</i> , <i>item_id</i> )
---

<b>revivir</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Permite realizar la recuperación de ítems borrados con sus respectivos atributos y las relaciones que sea posible recuperar.
--

**get\_all**(*self*, \*args, \*\*kw)

Lista los ítems borrados de acuerdo a lo establecido en  
borrado\_controller.ItemTableFiller.\_do\_get\_provider\_count\_and\_objs.

**buscar**(*self*, \*\*kw)

Lista los ítems borrados de acuerdo a un criterio de búsqueda introducido por el usuario.

**listar\_caracteristicas**(*self*, \*\*kw)

Permite al usuario visualizar las características de un ítem borrado.

#### 10.4.2 Class Variables

Name	Description
model	<b>Value:</b> Item
table	<b>Value:</b> ItemTable(DBSession)
table_filler	<b>Value:</b> ItemTableFiller(DBSession)
archivos	<b>Value:</b> ArchivoControllerListado(DBSession)

## 11 Module `saip.controllers.caracteristica_controller`

Controlador de características de un tipo de ítem en el módulo de administración.

### Author:

- Alejandro Arce<sup>13</sup>
- Gabriel Caroni<sup>14</sup>
- Rodrigo Parra<sup>15</sup>

### 11.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>caracteristica_table</code>	<b>Value:</b> <code>CaracteristicaTable(DBSession)</code>
<code>caracteristica_table_filler</code>	<b>Value:</b> <code>CaracteristicaTableFiller(DBSession)</code>
<code>add_caracteristica_form</code>	<b>Value:</b> <code>AddCaracteristica(DBSession)</code>

### 11.2 Class `CaracteristicaTable`

`sprox.tablebase.TableBase` — `saip.controllers.caracteristica_controller.CaracteristicaTable`

Define el formato de la tabla

#### 11.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Caracteristica</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'tipo_item', '__actions__', 'id_tipo_item']</code>

<sup>13</sup><mailto:alearce07@gmail.com>

<sup>14</sup><mailto:gabrielcaroni@gmail.com>

<sup>15</sup><mailto:rodpar07@gmail.com>

### 11.3 Class CharacteristicaTableFiller

sprox.fillerbase.TableFiller

saip.controllers.caracteristica\_controller.CharacteristicaTableFiller

Clase que se utiliza para llenar las tablas.

### 11.3.1 Methods

```
init(self, buscado, id_tipo_item)
```

### 11.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Característica
<code>buscado</code>	<b>Value:</b> ""
<code>id_tipo_item</code>	<b>Value:</b> ""

## 11.4 Class AddCharacteristica

sprox.formbase.AddRecordForm

saip.controllers.caracteristica\_controller.AddCaracteristica

Define el formato del formulario para crear una nueva característica de un tipo de ítem

### 11.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Característica
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'tipo_item', 'actions']
tipo	<b>Value:</b> SingleSelectField("tipo", options= ['cadena', 'entero', '...



## 11.5 Class *CaracteristicaController*

tgext.crud.CrudRestController

saip.controllers.caracteristica\_controller.CaracteristicaController

Controlador de Características

### 11.5.1 Methods

**get\_one**(*self*, *caracteristica\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista las características existentes de un tipo de ítem de acuerdo a condiciones establecidas en el `caracteristica_controller.TipoItemTableFiller.do_get_provider_count_and_objs`.

**new**(*self*, \**args*, \*\**kw*)

Despliega una página para la creación de una nueva característica de un tipo de ítem.

**edit**(*self*, \**args*, \*\**kw*)

**buscar**(*self*, \*\**kw*)

Lista las características de un tipo de ítem de acuerdo a un criterio de búsqueda introducido por el usuario.

**set\_null**(*self*, *c*)

**post**(*self*, \*\**kw*)

### 11.5.2 Class Variables

Name	Description
model	<b>Value:</b> <i>Caracteristica</i>
table	<b>Value:</b> <i>CaracteristicaTable(DBSession)</i>
table_filler	<b>Value:</b> <i>CaracteristicaTableFiller(DBSession)</i>
new_form	<b>Value:</b> <i>AddCaracteristica(DBSession)</i>
id_tipo_item	<b>Value:</b> <i>None</i>

## 12 Module *saip.controllers.desarrollo\_controller*

Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.

### Author:

- Alejandro Arce<sup>16</sup>
- Gabriel Caroni<sup>17</sup>
- Rodrigo Parra<sup>18</sup>

### 12.1 Class *DesarrolloController*



Controlador del módulo de desarrollo.

#### 12.1.1 Methods

<b>index</b> ( <i>self</i> )
Muestra la página del módulo de desarrollo.

*Inherited from saip.lib.base.BaseController(Section 44.1)*

`__call__()`

#### 12.1.2 Class Variables

Name	Description
proyectos	Value: <i>DesarrolloProyectoController()</i>

<sup>16</sup><mailto:alearce07@gmail.com>

<sup>17</sup><mailto:gabrielcaroni@gmail.com>

<sup>18</sup><mailto:rodpar07@gmail.com>

## 13 Module *saip.controllers.desarrollo\_fase\_controller*

Módulo que define el controlador de fases del módulo de desarrollo.

### Author:

- Alejandro Arce<sup>19</sup>
- Gabriel Caroni<sup>20</sup>
- Rodrigo Parra<sup>21</sup>

### 13.1 Variables

Name	Description
<code>fase_table</code>	<b>Value:</b> <code>FaseTable(DBSession)</code>
<code>fase_table_filler</code>	<b>Value:</b> <code>FaseTableFiller(DBSession)</code>

### 13.2 Class *FaseTable*

`sprox.tablebase.TableBase` — `saip.controllers.desarrollo_fase_controller.FaseTable`

#### 13.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Fase</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'proyecto', 'lineas_base', 'fichas', 'tipos_item', ...]</code>

### 13.3 Class *FaseTableFiller*

`sprox.fillerbase.TableFiller` — `saip.controllers.desarrollo_fase_controller.FaseTableFiller`

Clase que se utiliza para llenar las tablas de fase en el módulo de desarrollo.

<sup>19</sup><mailto:alearce07@gmail.com>

<sup>20</sup><mailto:gabrielcaroni@gmail.com>

<sup>21</sup><mailto:rodpar07@gmail.com>

### 13.3.1 Methods

**\_\_actions\_\_**(*self*, *obj*)

Define las acciones posibles para cada fase en el módulo de desarrollo.

**init**(*self*, *buscado*)

### 13.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>buscado</code>	<b>Value:</b> ""

## 13.4 Class *DesarrolloFaseController*

tg.controllers.RestController

saip.controllers.desarrollo\_fase\_controller.DesarrolloFaseController

Controlador del modelo Fase para el módulo de desarrollo.

### 13.4.1 Methods

**get\_one**(*self*, *proyecto\_id*)**get\_all**(*self*)Lista las fases de acuerdo a lo establecido en `desarrollo_fase_controller.FaseTableFiller._do_get_provider_count_and_objs`.**buscar**(*self*, *\*\*kw*)

Lista las fases de acuerdo a un criterio de búsqueda introducido por el usuario.

### 13.4.2 Class Variables

Name	Description
<code>items</code>	<b>Value:</b> ItemController(DBSession)
<code>table</code>	<b>Value:</b> FaseTable(DBSession)
<code>fase_filler</code>	<b>Value:</b> FaseTableFiller(DBSession)

## 14 Module saip.controllers.desarrollo\_proyecto\_controller

Controlador de proyectos en el módulo de desarrollo.

### Author:

- Alejandro Arce<sup>22</sup>
- Gabriel Caroni<sup>23</sup>
- Rodrigo Parra<sup>24</sup>

### 14.1 Variables

Name	Description
proyecto_table	<b>Value:</b> ProyectoTable(DBSession)
proyecto_table_filler	<b>Value:</b> ProyectoTableFiller(DBSession)

### 14.2 Class ProyectoTable

sprox.tablebase.TableBase  saip.controllers.desarrollo\_proyecto\_controller.ProyectoTable

Define el formato de la tabla.

#### 14.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Proyecto
__omit_fields__	<b>Value:</b> ['id', 'fases', 'fichas', 'lider']

### 14.3 Class ProyectoTableFiller

sprox.fillerbase.TableFiller  saip.controllers.desarrollo\_proyecto\_controller.ProyectoTableFiller

<sup>22</sup><mailto:alearce07@gmail.com>

<sup>23</sup><mailto:gabrielcaroni@gmail.com>

<sup>24</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas.

#### 14.3.1 Methods

**`__actions__(self, obj)`**

Define las acciones posibles para cada proyecto.

**`init(self, buscado)`**

#### 14.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>buscado</code>	<b>Value:</b> ""

### 14.4 Class *DesarrolloProyectoController*

tg.controllers.RestController — saip.controllers.desarrollo\_proyecto\_controller.DesarrolloProyectoController

Controlador de proyectos utilizado en el módulo de desarrollo del sistema

#### 14.4.1 Methods

**`get_all(self)`**

Lista los proyectos existentes de acuerdo a condiciones establecidas en el `desarrollo_proyecto_controller.ProyectoTableFiller`.  
`.do_get_provider_count_and_objs.`

**`get_one(self, id_proyecto)`**

**`buscar(self, **kw)`**

Lista los proyectos de acuerdo a un criterio de búsqueda introducido por el usuario.

#### 14.4.2 Class Variables

Name	Description
fases	<b>Value:</b> DesarrolloFaseController()
table	<b>Value:</b> ProyectoTable(DBSession)
proyecto_filler	<b>Value:</b> ProyectoTableFiller(DBSession)

## 15 Module `saip.controllers.error`

Error controller

### 15.1 Class `ErrorController`

object └─  
**`saip.controllers.error.ErrorController`**

Generates error documents as and when they are required.

The `ErrorDocuments` middleware forwards to `ErrorController` when error related status codes are returned from the application.

This behaviour can be altered by changing the parameters to the `ErrorDocuments` middleware in your `config/middleware.py` file.

#### 15.1.1 Methods

<b><code>document(self, *args, **kwargs)</code></b>
---

Render the error document
---------------------------

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 15.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	



## 16 Module saip.controllers.fase\_controller

Módulo que define el controlador de fases del módulo de administración.

### Author:

- Alejandro Arce<sup>25</sup>
- Gabriel Caroni<sup>26</sup>
- Rodrigo Parra<sup>27</sup>

### 16.1 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
fase_table	<b>Value:</b> FaseTable(DBSession)
fase_table_filler	<b>Value:</b> FaseTableFiller(DBSession)
add_fase_form	<b>Value:</b> AddFase(DBSession)
edit_fase_form	<b>Value:</b> EditFase(DBSession)
fase_edit_filler	<b>Value:</b> FaseEditFiller(DBSession)

### 16.2 Class ValidarExpresion

formencode.validators.Regex

saip.controllers.fase\_controller.ValidarExpresion

Clase para validar datos introducidos por el usuario. Recibe como parámetro una expresión regular.

#### 16.2.1 Class Variables

Name	Description
messages	<b>Value:</b> {'invalid': ("Introduzca un valor que empiece con una letr...

<sup>25</sup><mailto:alearce07@gmail.com>

<sup>26</sup><mailto:gabrielcaroni@gmail.com>

<sup>27</sup><mailto:rodpar07@gmail.com>

### 16.3 Class Unico

formencode.FancyValidator —  
**saip.controllers.fase\_controller.Unico**

Clase para verificar que el nombre de la fase introducido por el usuario sea único en el proyecto correspondiente.

### 16.4 Class FaseTable

sprox.tablebase.TableBase —  
**saip.controllers.fase\_controller.FaseTable**

#### 16.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'proyecto', 'lineas_base', 'fichas', 'tipos_item', ...]

### 16.5 Class FaseTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.fase\_controller.FaseTableFiller**

Clase que se utiliza para llenar las tablas de fase en administración.

#### 16.5.1 Methods

**`__actions__(self, obj)`**

Define las acciones posibles para cada fase en administración.

**`init(self, buscado, id_proyecto)`**

#### 16.5.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>buscado</code>	<b>Value:</b> ""
<code>id_proyecto</code>	<b>Value:</b> ""

## 16.6 Class OrdenFieldNew

sprox.widgets.PropertySingleSelectField  **saip.controllers.fase\_controller.OrdenFieldNew**

Clase para obtener los posibles números de fase (órdenes) para una nueva fase de un proyecto.

### 16.6.1 Methods

<b>obtener_fases_posibles(<i>self</i>)</b>
<b>Return Value</b> Posibles órdenes para la nueva fase de un proyecto.

## 16.7 Class OrdenFieldEdit

sprox.widgets.PropertySingleSelectField  **saip.controllers.fase\_controller.OrdenFieldEdit**

Clase para obtener los posibles números de fase (órdenes) para una fase que se edita de un proyecto.

### 16.7.1 Methods

<b>obtener_fases_posibles(<i>self</i>)</b>
<b>Return Value</b> Posibles órdenes para la fase a ser editada.

## 16.8 Class AddFase

sprox.formbase.AddRecordForm  **saip.controllers.fase\_controller.AddFase**

Define el formato de la tabla para agregar fases.

### 16.8.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'proyecto', 'lineas_base', 'fichas', 'tipos_item', ...]
<code>nombre</code>	<b>Value:</b> All(NotEmpty(), ValidarExpresion(r'^[A-Za-z] [A-Za-z0-9]*...))

## 16.9 Class EditFase

sprox.formbase.EditableForm  **saip.controllers.fase\_controller.EditFase**

Define el formato de la tabla para editar fases.

### 16.9.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>__hide_fields__</code>	<b>Value:</b> ['id', 'lineas_base', 'fichas', 'estado', 'fecha_inicio', ...]
<code>nombre</code>	<b>Value:</b> All(NotEmpty(), ValidarExpresion(r'^[A-Za-z] [A-Za-z0-9]*...))

## 16.10 Class FaseEditFiller

sprox.fillerbase.EditFormFiller  **saip.controllers.fase\_controller.FaseEditFiller**

Completa la tabla para editar fases.

### 16.10.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase

## 16.11 Class FaseController

tgext.crud.CrudRestController —  
   saip.controllers.fase\_controller.FaseController

Controlador del modelo Fase para el módulo de administración.

### 16.11.1 Methods

**get\_one**(*self*, *fase\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista las fases de acuerdo a lo establecido en  
`fase_controller.FaseTableFiller._do_get_provider_count_and_objs`.

**new**(*self*, \**args*, \*\**kw*)

Permite la creación de una nueva fase para un determinado proyecto.

**buscar**(*self*, \*\**kw*)

Lista las fases de acuerdo a un criterio de búsqueda introducido por el usuario.

**crear\_tipo\_default**(*self*, *id\_fase*)

Crea un tipo de ítem por defecto al crear una fase.

**edit**(*self*, \**args*, \*\**kw*)

Permite la creación de una nueva fase para un determinado proyecto.

**put**(*self*, \**args*, \*\**kw*)

Registra los cambios realizados en una fase.

**post**(*self*, \*\**kw*)

Registra la nueva fase creada.

**16.11.2 Class Variables**

Name	Description
proyectos	<b>Value:</b> ProyectoControllerNuevo()
tipo_item	<b>Value:</b> TipoItemController(DBSession)
responsables	<b>Value:</b> FichaFaseController(DBSession)
model	<b>Value:</b> Fase
table	<b>Value:</b> FaseTable(DBSession)
table_filler	<b>Value:</b> FaseTableFiller(DBSession)
edit_filler	<b>Value:</b> FaseEditFiller(DBSession)
edit_form	<b>Value:</b> EditFase(DBSession)
new_form	<b>Value:</b> AddFase(DBSession)
id_proyecto	<b>Value:</b> None

## 17 Module saip.controllers.fase\_controller\_2

Módulo que define el controlador de fases en el módulo de administración a la hora de importar fases o tipos de ítem.

### Author:

- Alejandro Arce<sup>28</sup>
- Gabriel Caroni<sup>29</sup>
- Rodrigo Parra<sup>30</sup>

### 17.1 Variables

Name	Description
fase_table	<b>Value:</b> FaseTable(DBSession)
fase_table_filler	<b>Value:</b> FaseTableFiller(DBSession)

### 17.2 Class FaseTable

sprox.tablebase.TableBase — saip.controllers.fase\_controller\_2.FaseTable

#### 17.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Fase
__omit_fields__	<b>Value:</b> ['id', 'proyecto', 'lineas_base', 'fichas', 'tipos_item', ...]

### 17.3 Class FaseTableFiller

sprox.fillerbase.TableFiller — saip.controllers.fase\_controller\_2.FaseTableFiller

Clase que se utiliza para llenar las tablas de fase en importación.

<sup>28</sup><mailto:alearce07@gmail.com>

<sup>29</sup><mailto:gabrielcaroni@gmail.com>

<sup>30</sup><mailto:rodpar07@gmail.com>

## 17.3.1 Methods

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada fase en importación.
---

<b>init</b> ( <i>self</i> , <i>buscado</i> )
--

## 17.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>id_proyecto</code>	<b>Value:</b> ""
<code>opcion</code>	<b>Value:</b> ""
<code>id_fase</code>	<b>Value:</b> ""

## 17.4 Class FaseControllerNuevo

tg.controllers.RestController

saip.controllers.fase\_controller\_2.FaseControllerNuevo

Controlador del modelo Fase para las importaciones.

## 17.4.1 Methods

<b>get_one</b> ( <i>self</i> , <i>proyecto_id</i> )
---

<b>get_all</b> ( <i>self</i> )
--------------------------------

Lista las fases de acuerdo a lo establecido en <code>fase_controller_2.FaseTableFiller.do_get_provider_count_and_objs</code> .
--

<b>buscar</b> ( <i>self</i> , <i>**kw</i> )
---

Lista las fases de acuerdo a un criterio de búsqueda introducido por el usuario.
--

<b>obtener_orden</b> ( <i>self</i> , <i>id_proyecto</i> )
---

Obtiene los órdenes posibles para la fase que se importa.
---



**importar\_caracteristica**(*self*, *id\_tipo\_item\_viejo*, *id\_tipo\_item\_nuevo*)

Realiza la importación de características de un tipo de ítem a otro.

**Parameters**

**id\_tipo\_item\_viejo:** id del tipo de ítem a ser importado.  
(*type=unicode*)

**id\_tipo\_item\_nuevo:** id del tipo de ítem que será creado en base al importado.  
(*type=unicode*)

**importar\_tipo\_item**(*self*, *id\_fase\_vieja*, *id\_fase\_nueva*)

Realiza la importación de un tipo de ítem de una fase a otra.

**Parameters**

**id\_fase\_vieja:** id de la fase a ser importada.  
(*type=unicode*)

**id\_fase\_nueva:** id de la fase que será creada en base a la importada.  
(*type=unicode*)

**importar\_fase**(*self*, \**args*, \*\**kw*)

Realiza la importación de una fase de un proyecto a otro.

#### 17.4.2 Class Variables

Name	Description
tipos_de_item	<b>Value:</b> TipoItemControllerNuevo()
table	<b>Value:</b> FaseTable(DBSession)
fase_filler	<b>Value:</b> FaseTableFiller(DBSession)

## 18 Module saip.controllers.ficha\_fase\_controller

Controlador de Fichas de fase en el módulo de administración.

### Author:

- Alejandro Arce<sup>31</sup>
- Gabriel Caroni<sup>32</sup>
- Rodrigo Parra<sup>33</sup>

### 18.1 Variables

Name	Description
ficha.table	<b>Value:</b> FichaTable(DBSession)
ficha.table.filler	<b>Value:</b> FichaTableFiller(DBSession)
add_ficha_form	<b>Value:</b> AddFicha(DBSession)

### 18.2 Class FichaTable

sprox.tablebase.TableBase — saip.controllers.ficha\_fase\_controller.FichaTable

Define el formato de la tabla

#### 18.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Ficha
__field_order__	<b>Value:</b> ['id', 'usuario', 'rol', 'proyecto', 'fase']
__omit_fields__	<b>Value:</b> ['id_fase', 'id_fase', 'id_usuario', 'id_rol']

<sup>31</sup><mailto:alearce07@gmail.com>

<sup>32</sup><mailto:gabrielcaroni@gmail.com>

<sup>33</sup><mailto:rodpar07@gmail.com>

### 18.3 Class FichaTableFiller

sprox.fillerbase.TableFiller —  
                                   saip.controllers.ficha\_fase\_controller.FichaTableFiller

Clase que se utiliza para llenar las tablas.

#### 18.3.1 Methods

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_fase</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ficha.
---

<b>rol</b> ( <i>self</i> , <i>obj</i> )
---

<b>usuario</b> ( <i>self</i> , <i>obj</i> )
---

<b>proyecto</b> ( <i>self</i> , <i>obj</i> )
--

<b>fase</b> ( <i>self</i> , <i>obj</i> )
--

#### 18.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Ficha
<code>buscado</code>	<b>Value:</b> ""
<code>id_fase</code>	<b>Value:</b> ""

### 18.4 Class RolesField

sprox.widgets.PropertySingleSelectField —  
                                   saip.controllers.ficha\_fase\_controller.RolesField

Clase para obtener los roles de fase existentes.

## 18.5 Class AddFicha

sprox.formbase.AddRecordForm  saip.controllers.ficha\_fase\_controller.AddFicha

Define el formato del formulario para crear una nueva ficha

### 18.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Ficha
<code>__omit_fields__</code>	<b>Value:</b> ['proyecto', 'fase', 'id']
<code>__dropdown_field_names__</code>	<b>Value:</b> {'rol': 'nombre', 'usuario': 'nombre_usuario'}

## 18.6 Class FichaFaseController

tgext.crud.CrudRestController  saip.controllers.ficha\_fase\_controller.FichaFaseController

Controlador de fichas de fases

### 18.6.1 Methods

```
get_one(self, Ficha_id)
```

```
get_all(self, *args, **kw)
```

Lista las fichas existentes de acuerdo a condiciones establecidas en  
ficha\_fase\_controller.FichaTableFiller  
..do\_get\_provider\_count\_and\_objs.

```
new(self, *args, **kw)
```

Despliega una página para la creación de una nueva ficha de fase.

```
edit(self, *args, **kw)
```

**buscar**(*self*, \*\**kw*)

Lista las fichas de fase de acuerdo a un criterio de búsqueda introducido por el usuario.

**post**(*self*, \*\**kw*)

Registra la nueva ficha creada

### 18.6.2 Class Variables

Name	Description
model	<b>Value:</b> Ficha
table	<b>Value:</b> FichaTable(DBSession)
table_filler	<b>Value:</b> FichaTableFiller(DBSession)
new_form	<b>Value:</b> AddFicha(DBSession)

## 19 Module saip.controllers.ficha\_proyecto\_controller

Controlador de Fichas de proyecto en el módulo de administración.

### Author:

- Alejandro Arce<sup>34</sup>
- Gabriel Caroni<sup>35</sup>
- Rodrigo Parra<sup>36</sup>

### 19.1 Variables

Name	Description
ficha.table	<b>Value:</b> FichaTable(DBSession)
ficha.table.filler	<b>Value:</b> FichaTableFiller(DBSession)
add_ficha_form	<b>Value:</b> AddFicha(DBSession)

### 19.2 Class FichaTable

sprox.tablebase.TableBase  saip.controllers.ficha\_proyecto\_controller.FichaTable

Define el formato de la tabla

#### 19.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Ficha
__field_order__	<b>Value:</b> ['id', 'usuario', 'rol', 'proyecto', 'fase']
__omit_fields__	<b>Value:</b> ['id_proyecto', 'id_fase', 'id_usuario', 'id_rol']

<sup>34</sup><mailto:alearce07@gmail.com>

<sup>35</sup><mailto:gabrielcaroni@gmail.com>

<sup>36</sup><mailto:rodpar07@gmail.com>

### 19.3 Class FichaTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.ficha\_proyecto\_controller.FichaTableFiller**

Clase que se utiliza para llenar las tablas.

#### 19.3.1 Methods

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_proyecto</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ficha.
---

<b>rol</b> ( <i>self</i> , <i>obj</i> )
---

<b>usuario</b> ( <i>self</i> , <i>obj</i> )
---

<b>proyecto</b> ( <i>self</i> , <i>obj</i> )
--

<b>fase</b> ( <i>self</i> , <i>obj</i> )
--

#### 19.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Ficha
<code>buscado</code>	<b>Value:</b> ""
<code>id_proyecto</code>	<b>Value:</b> ""

### 19.4 Class RolesField

sprox.widgets.PropertySingleSelectField —  
**saip.controllers.ficha\_proyecto\_controller.RolesField**

Clase para obtener los roles de proyecto existentes.

## 19.5 Class AddFicha

sprox.formbase.AddRecordForm

saip.controllers.ficha\_proyecto\_controller.AddFicha

Define el formato del formulario para crear una nueva ficha

### 19.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Ficha</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['proyecto', 'fase', 'id']</code>
<code>__dropdown_field_names__</code>	<b>Value:</b> <code>{'rol': 'nombre', 'usuario': 'nombre_usuario'}</code>

## 19.6 Class FichaProyectoController

tgext.crud.CrudRestController

saip.controllers.ficha\_proyecto\_controller.FichaProyectoController

Controlador de fichas de Proyecto

### 19.6.1 Methods

**get\_one**(*self*, *Ficha\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista las fichas existentes de acuerdo a condiciones establecidas en `ficha_proyecto_controller.FichaTableFiller`.  
`._do_get_provider_count_and_objs.`

**new**(*self*, \**args*, \*\**kw*)

Despliega una página para la creación de una nueva ficha de proyecto.

**edit**(*self*, \**args*, \*\**kw*)



<b>buscar</b> ( <i>self</i> , ** <i>kw</i> )
--

Lista las fichas de proyecto de acuerdo a un criterio de búsqueda introducido por el usuario.
---

<b>post</b> ( <i>self</i> , ** <i>kw</i> )
--

Registra la nueva ficha creada
--------------------------------

<b>post_delete</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Elimina un registro
---------------------

### 19.6.2 Class Variables

Name	Description
model	<b>Value:</b> Ficha
table	<b>Value:</b> FichaTable(DBSession)
table_filler	<b>Value:</b> FichaTableFiller(DBSession)
new_form	<b>Value:</b> AddFicha(DBSession)

## 20 Module `saip.controllers.ficha_sistema_controller`

Controlador de Fichas de sistema en el módulo de administración.

### Author:

- Alejandro Arce<sup>37</sup>
- Gabriel Caroni<sup>38</sup>
- Rodrigo Parra<sup>39</sup>

### 20.1 Variables

Name	Description
<code>ficha.table</code>	<b>Value:</b> <code>FichaTable(DBSession)</code>
<code>ficha.table.filler</code>	<b>Value:</b> <code>FichaTableFiller(DBSession)</code>
<code>add_ficha_form</code>	<b>Value:</b> <code>AddFicha(DBSession)</code>

### 20.2 Class `FichaTable`

`sprox.tablebase.TableBase` — `saip.controllers.ficha_sistema_controller.FichaTable`

Define el formato de la tabla

#### 20.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Ficha</code>
<code>__field_order__</code>	<b>Value:</b> <code>['id', 'usuario', 'rol']</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id_proyecto', 'id_fase', 'id_usuario', 'id_rol', 'proye...</code>

<sup>37</sup><mailto:alearce07@gmail.com>

<sup>38</sup><mailto:gabrielcaroni@gmail.com>

<sup>39</sup><mailto:rodpar07@gmail.com>

## 20.3 Class FichaTableFiller

sprox.fillerbase.TableFiller —  
   saip.controllers.ficha\_sistema\_controller.FichaTableFiller

Clase que se utiliza para llenar las tablas.

### 20.3.1 Methods

<b>init</b> ( <i>self</i> , <i>buscado</i> )
--

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ficha de sistema.
--

<b>rol</b> ( <i>self</i> , <i>obj</i> )
---

<b>usuario</b> ( <i>self</i> , <i>obj</i> )
---

<b>proyecto</b> ( <i>self</i> , <i>obj</i> )
--

<b>fase</b> ( <i>self</i> , <i>obj</i> )
--

### 20.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Ficha
<code>buscado</code>	<b>Value:</b> ""

## 20.4 Class RolesField

sprox.widgets.PropertySingleSelectField —  
   saip.controllers.ficha\_sistema\_controller.RolesField

Clase para obtener los roles de sistema existentes.

## 20.5 Class AddFicha

sprox.formbase.AddRecordForm  **saip.controllers.ficha\_sistema\_controller.AddFicha**

Define el formato del formulario para crear una nueva ficha

### 20.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Ficha
<code>__omit_fields__</code>	<b>Value:</b> ['proyecto', 'fase', 'id']
<code>__dropdown_field_names__</code>	<b>Value:</b> {'rol': 'nombre', 'usuario': 'nombre_usuario'}

## 20.6 Class FichaSistemaController

tgext.crud.CrudRestController  **saip.controllers.ficha\_sistema\_controller.FichaSistemaController**

Controlador de fichas del sistema

### 20.6.1 Methods

```
get_one(self, Ficha_id)
```

```
get_all(self, *args, **kw)
```

Lista las fichas existentes de acuerdo a condiciones establecidas en `ficha_sistema_controller.FichaTableFiller`  
`..do_get_provider_count_and_objs.`

```
new(self, *args, **kw)
```

Despliega una página para la creación de una nueva ficha de sistema.

```
edit(self, *args, **kw)
```

**buscar**(*self*, \*\**kw*)

Lista las fichas de sistema de acuerdo a un criterio de búsqueda introducido por el usuario.

**post**(*self*, \*\**kw*)

Registra la nueva ficha creada

### 20.6.2 Class Variables

Name	Description
model	<b>Value:</b> Ficha
table	<b>Value:</b> FichaTable(DBSession)
table_filler	<b>Value:</b> FichaTableFiller(DBSession)
new_form	<b>Value:</b> AddFicha(DBSession)

## 21 Module saip.controllers.ficha\_usuario\_controller

Módulo que define el controlador de fichas en el menú de usuario.

### Author:

- Alejandro Arce<sup>40</sup>
- Gabriel Caroni<sup>41</sup>
- Rodrigo Parra<sup>42</sup>

### 21.1 Variables

Name	Description
ficha.table	<b>Value:</b> FichaTable(DBSession)
ficha.table_filler	<b>Value:</b> FichaTableFiller(DBSession)

### 21.2 Class FichaTable

sprox.tablebase.TableBase —  
saip.controllers.ficha\_usuario\_controller.FichaTable

#### 21.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Ficha
__field_order__	<b>Value:</b> ['id', 'usuario', 'rol', 'proyecto', 'fase']
__omit_fields__	<b>Value:</b> ['id_proyecto', 'id_fase', 'id_usuario', 'id_rol', '__act...']

### 21.3 Class FichaTableFiller

sprox.fillerbase.TableFiller —  
saip.controllers.ficha\_usuario\_controller.FichaTableFiller

<sup>40</sup><mailto:alearce07@gmail.com>

<sup>41</sup><mailto:gabrielcaroni@gmail.com>

<sup>42</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas de fichas en el módulo usuario.

### 21.3.1 Methods

```
init(self, buscado, id_usuario)
```

```
rol(self, obj)
```

```
usuario(self, obj)
```

```
proyecto(self, obj)
```

```
fase(self, obj)
```

### 21.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Ficha</code>
<code>buscado</code>	<b>Value:</b> <code>""</code>
<code>id_usuario</code>	<b>Value:</b> <code>""</code>

## 21.4 Class *FichaUsuarioController*

```
tgext.crud.CrudRestController └─ saip.controllers.ficha_usuario_controller.FichaUsuarioController
```

Controlador del modelo Fichas para el menú de usuario.

### 21.4.1 Methods

```
get_one(self, Ficha_id)
```

```
get_all(self, *args, **kw)
```

Lista las fichas de acuerdo a lo establecido en `ficha_usuario_controller.FichaTableFiller.do_get_provider_count_and_objs`.

```
new(self, *args, **kw)
```

```
edit(self, *args, **kw)
```

```
buscar(self, **kw)
```

Lista las fichas de un usuario de acuerdo a un criterio de búsqueda.

```
post(self, **kw)
```

```
put(self, *args, **kw)
```

```
post_delete(self, *args, **kw)
```

#### 21.4.2 Class Variables

Name	Description
model	<b>Value:</b> Ficha
table	<b>Value:</b> FichaTable(DBSession)
table_filler	<b>Value:</b> FichaTableFiller(DBSession)



## 22 Module `saip.controllers.gestion_controller`

Módulo que define el controlador del módulo de gestión.

### Author:

- Alejandro Arce<sup>43</sup>
- Gabriel Caroni<sup>44</sup>
- Rodrigo Parra<sup>45</sup>

### 22.1 Class `GestionController`



Controlador del módulo de gestión.

#### 22.1.1 Methods

<b><code>index(self)</code></b>
---------------------------------

Muestra la página del módulo de gestión.
--

*Inherited from `saip.lib.base.BaseController` (Section 44.1)*

`__call__()`

#### 22.1.2 Class Variables

Name	Description
proyectos	Value: <code>GestionProyectoController()</code>

---

<sup>43</sup><mailto:alearce07@gmail.com>

<sup>44</sup><mailto:gabrielcaroni@gmail.com>

<sup>45</sup><mailto:rodpar07@gmail.com>

## 23 Module `saip.controllers.gestion_fase_controller`

Módulo que define el controlador de fases del módulo de gestión.

### Author:

- Alejandro Arce<sup>46</sup>
- Gabriel Caroni<sup>47</sup>
- Rodrigo Parra<sup>48</sup>

### 23.1 Variables

Name	Description
<code>fase_table</code>	<b>Value:</b> <code>FaseTable(DBSession)</code>
<code>fase_table_filler</code>	<b>Value:</b> <code>FaseTableFiller(DBSession)</code>

### 23.2 Class `FaseTable`

`sprox.tablebase.TableBase` — `saip.controllers.gestion_fase_controller.FaseTable`

#### 23.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Fase</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'proyecto', 'lineas_base', 'fichas', 'tipos_item', ...]</code>

### 23.3 Class `FaseTableFiller`

`sprox.fillerbase.TableFiller` — `saip.controllers.gestion_fase_controller.FaseTableFiller`

Clase que se utiliza para llenar las tablas de fase en el módulo de gestión.

<sup>46</sup><mailto:alearce07@gmail.com>

<sup>47</sup><mailto:gabrielcaroni@gmail.com>

<sup>48</sup><mailto:rodpar07@gmail.com>

### 23.3.1 Methods

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_proyecto</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada fase en el módulo de gestión.
--

### 23.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Fase
<code>buscado</code>	<b>Value:</b> ""
<code>id_proyecto</code>	<b>Value:</b> ""

## 23.4 Class *GestionFaseController*

tg.controllers.RestController

saip.controllers.gestion\_fase\_controller.GestionFaseController

Controlador del modelo Fase para el módulo de gestión.

### 23.4.1 Methods

<b>get_one</b> ( <i>self</i> , <i>proyecto_id</i> )
---

<b>get_all</b> ( <i>self</i> )
--------------------------------

Lista las fases de acuerdo a lo establecido en
--

<code>gestion_fase_controller.FaseTableFiller.do_get_provider_count_and_objs.</code>
--

<b>buscar</b> ( <i>self</i> , <b>**kw</b> )
---

Lista las fases de acuerdo a un criterio de búsqueda introducido por el usuario, en el módulo de gestión.
---

### 23.4.2 Class Variables

Name	Description
<code>lineas_base</code>	<b>Value:</b> LineaBaseController(DBSession)

*continued on next page*

Name	Description
table	<b>Value:</b> FaseTable(DBSession)
fase_filler	<b>Value:</b> FaseTableFiller(DBSession)

## 24 Module saip.controllers.gestion\_proyecto\_controller

Controlador de proyectos en el módulo de gestión.

### Author:

- Alejandro Arce<sup>49</sup>
- Gabriel Caroni<sup>50</sup>
- Rodrigo Parra<sup>51</sup>

### 24.1 Variables

Name	Description
proyecto_table	<b>Value:</b> ProyectoTable(DBSession)
proyecto_table_filler	<b>Value:</b> ProyectoTableFiller(DBSession)

### 24.2 Class ProyectoTable

sprox.tablebase.TableBase  saip.controllers.gestion\_proyecto\_controller.ProyectoTable

Define el formato de la tabla.

#### 24.2.1 Class Variables

Name	Description
__model__	<b>Value:</b> Proyecto
__omit_fields__	<b>Value:</b> ['id', 'fases', 'fichas', 'lider']

### 24.3 Class ProyectoTableFiller

sprox.fillerbase.TableFiller  saip.controllers.gestion\_proyecto\_controller.ProyectoTableFiller

<sup>49</sup><mailto:alearce07@gmail.com>

<sup>50</sup><mailto:gabrielcaroni@gmail.com>

<sup>51</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas.

### 24.3.1 Methods

**\_\_actions\_\_**(*self*, *obj*)

Define las acciones posibles para cada proyecto.

**init**(*self*, *buscado*)

**fase\_apta**(*self*, *proyecto*)

Verifica si existe alguna fase en el proyecto que cumpla con los requisitos para poder gestionar las líneas base.

**Parameters**

**proyecto:** Tiene el proyecto a verificar.

(*type=Proyecto.*)

**Return Value**

Retorna True o False de acuerdo a si la fase es o no apta.

(*type=Boolean*)

### 24.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>buscado</code>	<b>Value:</b> ""

## 24.4 Class *GestionProyectoController*

tg.controllers.RestController

saip.controllers.gestion\_proyecto\_controller.GestionProyectoCont

Controlador de proyectos utilizado en el módulo de gestión del sistema

**24.4.1 Methods****get\_all**(*self*)

Lista los proyectos existentes de acuerdo a condiciones establecidas en el `gestion_proyecto_controller.ProyectoTableFiller`  
`._do_get_provider_count_and_objs`.

**get\_one**(*self*, *id\_proyecto*)**buscar**(*self*, *\*\*kw*)

Lista los proyectos de acuerdo a un criterio de búsqueda introducido por el usuario.

**24.4.2 Class Variables**

Name	Description
fases	<b>Value:</b> <code>GestionFaseController()</code>
table	<b>Value:</b> <code>ProyectoTable(DBSession)</code>
proyecto_filler	<b>Value:</b> <code>ProyectoTableFiller(DBSession)</code>

## 25 Module `saip.controllers.item_controller`

Controlador de ítems en el módulo de desarrollo.

### Author:

- Alejandro Arce<sup>52</sup>
- Gabriel Caroni<sup>53</sup>
- Rodrigo Parra<sup>54</sup>

### 25.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>item_table</code>	<b>Value:</b> <code>ItemTable(DBSession)</code>
<code>item_table.filler</code>	<b>Value:</b> <code>ItemTableFiller(DBSession)</code>
<code>add_item_form</code>	<b>Value:</b> <code>AddItem(DBSession)</code>
<code>edit_item_form</code>	<b>Value:</b> <code>EditItem(DBSession)</code>
<code>item_edit.filler</code>	<b>Value:</b> <code>ItemEditFiller(DBSession)</code>

### 25.2 Class `ItemTable`

`sprox.tablebase.TableBase` — `saip.controllers.item_controller.ItemTable`

Define el formato de la tabla.

#### 25.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'id_tipo_item', 'id_fase', 'id_linea_base', 'archi...]</code>

<sup>52</sup><mailto:alearce07@gmail.com>

<sup>53</sup><mailto:gabrielcaroni@gmail.com>

<sup>54</sup><mailto:rodpar07@gmail.com>



## 25.3 Class ItemTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.item\_controller.ItemTableFiller**

Clase que se utiliza para llenar las tablas.

### 25.3.1 Methods

<b>tipo_item</b> ( <i>self</i> , <i>obj</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ítem.
--

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_fase</i> )
---

### 25.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Item
<code>buscado</code>	<b>Value:</b> ""
<code>id_fase</code>	<b>Value:</b> ""

## 25.4 Class AddItem

sprox.formbase.AddRecordForm —  
**saip.controllers.item\_controller.AddItem**

Define el formato del formulario para crear un nuevo ítem

### 25.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Item
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'codigo', 'archivos', 'fichas', 'revisiones', 'id...]
<code>complejidad</code>	<b>Value:</b> SingleSelectField('complejidad', options= range(11) [1:])

*continued on next page*

Name	Description
prioridad	<b>Value:</b> SingleSelectField('prioridad', options= range(11) [1:])

## 25.5 Class EditItem

sprox.formbase.EditableForm —  
**saip.controllers.item\_controller.EditItem**

Define el formato del formulario para la modificación de un ítem

### 25.5.1 Class Variables

Name	Description
__model__	<b>Value:</b> Item
__omit_fields__	<b>Value:</b> ['id', 'codigo', 'archivos', 'fichas', 'revisiones', 'id...]

## 25.6 Class ItemEditFiller

sprox.fillerbase.EditFormFiller —  
**saip.controllers.item\_controller.ItemEditFiller**

Se utiliza para llenar el formulario de modificación de un ítem con los valores recuperados de la base de datos.

### 25.6.1 Class Variables

Name	Description
__model__	<b>Value:</b> Item

## 25.7 Class ItemController

tgext.crud.CrudRestController —  
**saip.controllers.item\_controller.ItemController**

## Controlador de ítem

## 25.7.1 Methods

<b>get_one</b> ( <i>self</i> , <i>item_id</i> )
---

<b>costo</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Calcula el costo de impacto de un ítem dado y despliega una representación gráfica del mismo.
---

<b>get_all</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Lista los ítems existentes de acuerdo a condiciones establecidas en el <code>item_controller.ItemTableFiller .do_get_provider_count_and_objs</code> .
---

<b>new</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Despliega una página para la creación de un nuevo ítem.
---

<b>edit</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

Despliega una página para la modificación de un ítem.
---

<b>buscar</b> ( <i>self</i> , ** <i>kw</i> )
--

Lista los ítems de acuerdo a un criterio de búsqueda introducido por el usuario.
--

<b>post</b> ( <i>self</i> , ** <i>kw</i> )
--

Registra el nuevo ítem creado.
--------------------------------

<b>put</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Registra los cambios realizados a un ítem.
--

**crear\_version**(*self*, *it*, *borrado*=None)

Crea una nueva versión de un ítem dado. Permite también borrar una relación de esta nueva versión si se especifica.

**Parameters**

**it**: Ítem a partir del cual se creará una nueva versión.  
(*type*=*Item*)

**borrado**: Relación a eliminar.  
(*type*=*Relacion*)

**crear\_revision**(*self*, *item*, *msg*)

Crea una nueva revisión y la agrega a un ítem dado.

**Parameters**

**item**: Ítem al cual se agregará la revisión  
(*type*=*Item*)

**msg**: Mensaje de la revisión  
(*type*=*String*)

**post\_delete**(*self*, \**args*, \*\**kw*)

Elimina un ítem.

**listo**(*self*, \*\**kw*)

Asigna el estado 'Listo' a un ítem dado.

**aprobar**(*self*, \*\**kw*)

Asigna el estado 'Aprobado' a un ítem dado.

**listar\_caracteristicas**(*self*, \*\**kw*)

Despliega el valor de las características propias del tipo de ítem de un ítem dado, siempre que el ítem no sea del tipo 'Default'

### 25.7.2 Class Variables

Name	Description
relaciones	<b>Value:</b> RelacionController(DBSession)
archivos	<b>Value:</b> ArchivoController(DBSession)
borrados	<b>Value:</b> BorradoController(DBSession)

*continued on next page*

Name	Description
versiones	<b>Value:</b> VersionController(DBSession)
revisiones	<b>Value:</b> RevisionController(DBSession)
model	<b>Value:</b> Item
table	<b>Value:</b> ItemTable(DBSession)
table_filler	<b>Value:</b> ItemTableFiller(DBSession)
edit_filler	<b>Value:</b> ItemEditFiller(DBSession)
edit_form	<b>Value:</b> EditItem(DBSession)
new_form	<b>Value:</b> AddItem(DBSession)

## 26 Module `saip.controllers.item_controller_listado`

Módulo que define el controlador de listado de ítems pertenecientes a una línea base

### Author:

- Alejandro Arce<sup>55</sup>
- Gabriel Caroni<sup>56</sup>
- Rodrigo Parra<sup>57</sup>

### 26.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>item_table</code>	<b>Value:</b> <code>ItemTable(DBSession)</code>
<code>item_table.filler</code>	<b>Value:</b> <code>ItemTableFiller(DBSession)</code>
<code>add_item_form</code>	<b>Value:</b> <code>AddItem(DBSession)</code>
<code>edit_item_form</code>	<b>Value:</b> <code>EditItem(DBSession)</code>
<code>item_edit.filler</code>	<b>Value:</b> <code>ItemEditFiller(DBSession)</code>

### 26.2 Class `ItemTable`

`sprox.tablebase.TableBase` — `saip.controllers.item_controller_listado.ItemTable`

#### 26.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'id_tipo_item', 'id_fase', 'id_linea_base', 'archi...</code>

<sup>55</sup><mailto:alearce07@gmail.com>

<sup>56</sup><mailto:gabrielcaroni@gmail.com>

<sup>57</sup><mailto:rodpar07@gmail.com>

## 26.3 Class *ItemTableFiller*

sprox.fillerbase.TableFiller —  
**saip.controllers.item\_controller\_listado.ItemTableFiller**

Clase que se utiliza para llenar las tablas de ítems pertenecientes a una línea base.

### 26.3.1 Methods

<b>tipo_item</b> ( <i>self</i> , <i>obj</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ítem (ver características).
--

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_fase</i> )
---

### 26.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>buscado</code>	<b>Value:</b> <code>" "</code>
<code>id_fase</code>	<b>Value:</b> <code>" "</code>

## 26.4 Class *AddItem*

sprox.formbase.AddRecordForm —  
**saip.controllers.item\_controller\_listado.AddItem**

### 26.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'codigo', 'archivos', 'fichas', 'revisiones', 'id...]</code>
<code>complejidad</code>	<b>Value:</b> <code>SingleSelectField('complejidad', options= range(11) [1:])</code>

*continued on next page*

Name	Description
prioridad	<b>Value:</b> <code>SingleSelectField('prioridad', options= range(11) [1:])</code>

## 26.5 Class `EditItem`

`sprox.formbase.EditableForm` — `saip.controllers.item_controller_listado.EditItem`

### 26.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'codigo', 'archivos', 'fichas', 'revisiones', 'id...]</code>

## 26.6 Class `ItemEditFiller`

`sprox.fillerbase.EditFormFiller` — `saip.controllers.item_controller_listado.ItemEditFiller`

### 26.6.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>

## 26.7 Class `ItemControllerListado`

`tgext.crud.CrudRestController` — `saip.controllers.item_controller_listado.ItemControllerListado`

Controlador de ítem para el listado de los pertenecientes a una línea base.



**26.7.1 Methods**

**get\_all**(*self*, \**args*, \*\**kw*)

Lista los ítems de acuerdo a lo establecido en  
*ItemTableFiller.\_do\_get\_provider\_count\_and\_objs*.

**new**(*self*, \**args*, \*\**kw*)

**edit**(*self*, \**args*, \*\**kw*)

**buscar**(*self*, \*\**kw*)

Lista los ítems pertenecientes a una línea base de acuerdo a un criterio de  
 búsqueda introducido por el usuario.

**listar\_caracteristicas**(*self*, \*\**kw*)

Muestra las características de un ítem seleccionado que pertenece a una línea  
 base.

**26.7.2 Class Variables**

Name	Description
model	<b>Value:</b> Item
table	<b>Value:</b> ItemTable(DBSession)
table_filler	<b>Value:</b> ItemTableFiller(DBSession)
edit_filler	<b>Value:</b> ItemEditFiller(DBSession)
edit_form	<b>Value:</b> EditItem(DBSession)
new_form	<b>Value:</b> AddItem(DBSession)

## 27 Module `saip.controllers.linea_base_controller`

Módulo que define el controlador de líneas base.

### Author:

- Alejandro Arce<sup>58</sup>
- Gabriel Caroni<sup>59</sup>
- Rodrigo Parra<sup>60</sup>

### 27.1 Functions

<b>UnificarItem</b> ( <i>items</i> )
--------------------------------------

Obtiene todos los ítems, cada uno con su mayor versión.
---

### 27.2 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
linea_base_table	<b>Value:</b> LineaBaseTable(DBSession)
linea_base_table_filler	<b>Value:</b> LineaBaseTableFiller(DBSession)
add_linea_base_form	<b>Value:</b> AddLineaBase(DBSession)

### 27.3 Class `LineaBaseTable`

sprox.tablebase.TableBase  saip.controllers.linea\_base\_controller.LineaBaseTable

#### 27.3.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> LineaBase
<code>__omit_fields__</code>	<b>Value:</b> ['fase', 'items', 'id.fase']

<sup>58</sup><mailto:alearce07@gmail.com>

<sup>59</sup><mailto:gabrielcaroni@gmail.com>

<sup>60</sup><mailto:rodpar07@gmail.com>



Name	Description
template	<b>Value:</b> <code>'saip.templates.selectshuttle'</code>

## 27.6 Class AddLineaBase

sprox.formbase.AddRecordForm —  
**saip.controllers.linea\_base\_controller.AddLineaBase**

Define el formato de la tabla para agregar líneas base.

### 27.6.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>LineaBase</code>
<code>__hide_fields__</code>	<b>Value:</b> <code>['fase', 'consistente', 'cerrado']</code>
<code>__dropdown_field_names__</code>	<b>Value:</b> <code>{'items': 'codigo'}</code>

## 27.7 Class LineaBaseController

tgext.crud.CrudRestController —  
**saip.controllers.linea\_base\_controller.LineaBaseController**

Controlador del modelo Línea Base, módulo de gestión.

### 27.7.1 Methods

<b>get_one</b> ( <i>self</i> , <i>linea_base_id</i> )
---

<b>get_all</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Lista las líneas base de acuerdo a lo establecido en <code>linea_base_controller.LineaBaseTableFiller._do_get_provider_count_and_objs</code> .
---

<b>new</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Permite la creación de una nueva línea base para una determinada fase de un proyecto.
---

**buscar**(*self*, \*\**kw*)

Lista las líneas base de acuerdo a un criterio de búsqueda introducido por el usuario.

**post**(*self*, \*\**kw*)

Registra la nueva línea base creada.

**abrir**(*self*, \*\**kw*)

Permite indicar que la línea se encuentra abierta.

**cerrar**(*self*, \*\**kw*)

Permite indicar que la línea base se encuentra cerrada.

**unir**(*self*, \*\**kw*)

Permite realizar la unión entre dos o más líneas base de manera que formen una sola.

**dividir**(*self*, \*\**kw*)

Permite formar una línea base a partir de dos o más líneas base existentes previamente.

### 27.7.2 Class Variables

Name	Description
model	<b>Value:</b> LineaBase
table	<b>Value:</b> LineaBaseTable(DBSession)
table_filler	<b>Value:</b> LineaBaseTableFiller(DBSession)
new_form	<b>Value:</b> AddLineaBase(DBSession)
items	<b>Value:</b> ItemControllerListado(DBSession)
id_primera_lb	<b>Value:</b> ""

## 28 Module saip.controllers.proyecto\_controller

Controlador de proyectos en el módulo de administración.

### Author:

- Alejandro Arce<sup>61</sup>
- Gabriel Caroni<sup>62</sup>
- Rodrigo Parra<sup>63</sup>

### 28.1 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
proyecto_table	<b>Value:</b> ProyectoTable(DBSession)
proyecto_table_filler	<b>Value:</b> ProyectoTableFiller(DBSession)
add_proyecto_form	<b>Value:</b> AddProyecto(DBSession)
form_validator	<b>Value:</b> Schema(nro_fases=All(NroValido(), NotEmpty(), Int(min=0...))
edit_proyecto_form	<b>Value:</b> EditProyecto(DBSession)
proyecto_edit_filler	<b>Value:</b> ProyectoEditFiller(DBSession)

### 28.2 Class Unico

formencode.FancyValidator  **saip.controllers.proyecto\_controller.Unico**

Clase correspondiente a un validador que se utiliza para controlar que el nombre ingresado en cierto proyecto añadido o modificado no se repita dentro del sistema.

### 28.3 Class ValidarExpresion

formencode.validators.Regex  **saip.controllers.proyecto\_controller.ValidarExpresion**

<sup>61</sup><mailto:alearce07@gmail.com>

<sup>62</sup><mailto:gabrielcaroni@gmail.com>

<sup>63</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para validar datos ingresados por el usuario, recibe como parámetro una expresión regular.

### 28.3.1 Class Variables

Name	Description
messages	<b>Value:</b> {'invalid':("Introduzca un valor que empiece con una letr...

## 28.4 Class ProyectoTable

sprox.tablebase.TableBase



**saip.controllers.proyecto\_controller.ProyectoTable**

Define el formato de la tabla.

### 28.4.1 Class Variables

Name	Description
__model__	<b>Value:</b> Proyecto
__omit_fields__	<b>Value:</b> ['id', 'fases', 'fichas', 'id_lider']

## 28.5 Class ProyectoTableFiller

sprox.fillerbase.TableFiller



**saip.controllers.proyecto\_controller.ProyectoTableFiller**

Clase que se utiliza para llenar las tablas.

### 28.5.1 Methods

<b>lider</b> ( <i>self</i> , <i>obj</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
Define las acciones posibles para cada proyecto.

<code>init(self, buscado)</code>
----------------------------------

### 28.5.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>buscado</code>	<b>Value:</b> ""

## 28.6 Class NroValido

formencode.FancyValidator


**saip.controllers.proyecto\_controller.NroValido**

Clase correspondiente a un validador que se utiliza para controlar que el número de fases ingresado en una modificación de un proyecto no sea menor a la cantidad de fases existentes en ese proyecto.

## 28.7 Class AddProyecto

sprox.formbase.AddRecordForm


**saip.controllers.proyecto\_controller.AddProyecto**

Define el formato del formulario para crear un nuevo proyecto

### 28.7.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'fases', 'fichas', 'estado', 'fecha_inicio', 'fech...
<code>nombre</code>	<b>Value:</b> All(NotEmpty(), ValidarExpresion(r'^[A-Za-z][A-Za-z0-9]*...
<code>nro_fases</code>	<b>Value:</b> All(NotEmpty(), Int(min= 0))
<code>__dropdown_field_names__</code>	<b>Value:</b> {'lider': 'nombre_usuario'}



## 28.8 Class CantidadFasesField

tw.forms.fields.TextField —  
**saip.controllers.proyecto\_controller.CantidadFasesField**

Clase correspondiente a un validador que se utiliza para deshabilitar la modificación del número de fases de un proyecto si el mismo ya ha iniciado.

### 28.8.1 Methods

<b>update_params(<i>self</i>, <i>d</i>)</b>
---

Realiza el control citado anteriormente.
--

## 28.9 Class EditProyecto

sprox.formbase.EditableForm —  
**saip.controllers.proyecto\_controller.EditProyecto**

Define el formato del formulario para la modificación de un proyecto

### 28.9.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>__base_validator__</code>	<b>Value:</b> Schema(nro_fases=All(NroValido(), NotEmpty(), Int(min=0...))
<code>__hide_fields__</code>	<b>Value:</b> ['id', 'fases', 'fichas', 'estado', 'fecha_inicio', 'fech...
<code>nro_fases</code>	<b>Value:</b> CantidadFasesField('nro_fases')
<code>nombre</code>	<b>Value:</b> All(NotEmpty(), ValidarExpresion(r'^[A-Za-z][A-Za-z0-9]*...'))
<code>__dropdown_field_names__</code>	<b>Value:</b> {'lider': 'nombre_usuario'}

## 28.10 Class *ProyectoEditFiller*

sprox.fillerbase.EditFormFiller —  
**saip.controllers.proyecto\_controller.ProyectoEditFiller**

Se utiliza para llenar el formulario de modificación de un proyecto con los valores recuperados de la base de datos.

### 28.10.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto

## 28.11 Class *ProyectoController*

tgext.crud.CrudRestController —  
**saip.controllers.proyecto\_controller.ProyectoController**

Controlador de proyectos

### 28.11.1 Methods

**iniciar**(*self*, *id\_proyecto*)

Método invocado desde `ProyectoTableFiller.__actions__` utilizado para setear el estado de un proyecto a en desarrollo.

**Parameters**

**id\_proyecto:** Contiene el id del proyecto a iniciar.  
*(type=Unicode.)*

**get\_one**(*self*, *proyecto\_id*)

**get\_all**(*self*, *\*args*, *\*\*kw*)

Lista los proyectos existentes de acuerdo a condiciones establecidas en el `proyecto_controller.ProyectoTableFiller`  
`.do_get_provider_count_and_objs.`

<b>new</b> ( <i>self</i> , *args, **kw)
---

Despliega una página para la creación de un nuevo proyecto.
---

<b>edit</b> ( <i>self</i> , *args, **kw)
--

Despliega una página para la modificación de un proyecto.
---

<b>buscar</b> ( <i>self</i> , **kw)
-------------------------------------

Lista los proyectos de acuerdo a un criterio de búsqueda introducido por el usuario.
--

<b>post</b> ( <i>self</i> , **kw)
-----------------------------------

Registra el nuevo proyecto creado.
------------------------------------

<b>put</b> ( <i>self</i> , *args, **kw)
---

Registra los cambios realizados a un proyecto.
--

### 28.11.2 Class Variables

Name	Description
fases	<b>Value:</b> FaseController(DBSession)
model	<b>Value:</b> Proyecto
table	<b>Value:</b> ProyectoTable(DBSession)
table_filler	<b>Value:</b> ProyectoTableFiller(DBSession)
edit_filler	<b>Value:</b> ProyectoEditFiller(DBSession)
edit_form	<b>Value:</b> EditProyecto(DBSession)
new_form	<b>Value:</b> AddProyecto(DBSession)
responsables	<b>Value:</b> FichaProyectoController(DBSession)
pr	<b>Value:</b> DBSession.query(Proyecto).get(id_proyecto)
fecha_inicio	<b>Value:</b> datetime.datetime.now()

## 29 Module *saip.controllers.proyecto\_controller\_2*

Controlador de proyectos en el módulo de administración utilizado para la importación de fases o tipos de ítem.

### Author:

- Alejandro Arce<sup>64</sup>
- Gabriel Caroni<sup>65</sup>
- Rodrigo Parra<sup>66</sup>

### 29.1 Variables

Name	Description
<code>proyecto_table</code>	<b>Value:</b> <code>ProyectoTable(DBSession)</code>
<code>proyecto_table_filler</code>	<b>Value:</b> <code>ProyectoTableFiller(DBSession)</code>

### 29.2 Class *ProyectoTable*

`sprox.tablebase.TableBase` — `saip.controllers.proyecto_controller_2.ProyectoTable`

Define el formato de la tabla.

#### 29.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Proyecto</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fases', 'fichas', 'id_lider']</code>

### 29.3 Class *ProyectoTableFiller*

`sprox.fillerbase.TableFiller` — `saip.controllers.proyecto_controller_2.ProyectoTableFiller`

<sup>64</sup><mailto:alearce07@gmail.com>

<sup>65</sup><mailto:gabrielcaroni@gmail.com>

<sup>66</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas.

### 29.3.1 Methods

**\_\_actions\_\_**(*self*, *obj*)

Define las acciones posibles para cada proyecto.

**lider**(*self*, *obj*)

**init**(*self*, *buscado*)

### 29.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Proyecto
<code>id</code>	<b>Value:</b> ""
<code>opcion</code>	<b>Value:</b> ""
<code>buscado</code>	<b>Value:</b> ""

## 29.4 Class ProyectoControllerNuevo

tg.controllers.RestController — saip.controllers.proyecto\_controller\_2.ProyectoControllerNuevo

Controlador de proyectos utilizado para la importación

### 29.4.1 Methods

**get\_all**(*self*)

Lista los proyectos existentes de acuerdo a condiciones establecidas en el `proyecto_controller_2.ProyectoTableFiller`  
`._do_get_provider_count_and_objs.`

**get\_one**(*self*, *id\_proyecto*)

**buscar**(*self*, \*\**kw*)

Lista los proyectos de acuerdo a un criterio de búsqueda introducido por el usuario.

#### 29.4.2 Class Variables

Name	Description
fases	<b>Value:</b> FaseControllerNuevo()
table	<b>Value:</b> ProyectoTable(DBSession)
proyecto_filler	<b>Value:</b> ProyectoTableFiller(DBSession)

## 30 Module `saip.controllers.relacion_controller`

Módulo que define el controlador de relaciones del módulo de desarrollo.

### Author:

- Alejandro Arce<sup>67</sup>
- Gabriel Caroni<sup>68</sup>
- Rodrigo Parra<sup>69</sup>

### 30.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>relacion_table</code>	<b>Value:</b> <code>RelacionTable(DBSession)</code>
<code>relacion_table_filler</code>	<b>Value:</b> <code>RelacionTableFiller(DBSession)</code>
<code>add_relacion_form</code>	<b>Value:</b> <code>AddRelacion(DBSession)</code>

### 30.2 Class `RelacionTable`

`sprox.tablebase.TableBase` — `saip.controllers.relacion_controller.RelacionTable`

#### 30.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Relacion</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id_item_1', 'id_item_2']</code>
<code>__xml_fields__</code>	<b>Value:</b> <code>['fase']</code>

### 30.3 Class `RelacionTableFiller`

`sprox.fillerbase.TableFiller` — `saip.controllers.relacion_controller.RelacionTableFiller`

<sup>67</sup><mailto:alearce07@gmail.com>

<sup>68</sup><mailto:gabrielcaroni@gmail.com>

<sup>69</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas de relaciones.

### 30.3.1 Methods

**\_\_actions\_\_**(*self, obj*)

Define las acciones posibles para cada relación.

**item\_1**(*self, obj*)

**item\_2**(*self, obj*)

**init**(*self, buscado, id\_item, version\_item*)

### 30.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Relacion</code>
<code>buscado</code>	<b>Value:</b> <code>""</code>
<code>id_item</code>	<b>Value:</b> <code>""</code>
<code>version_item</code>	<b>Value:</b> <code>""</code>

## 30.4 Class *AddRelacion*

sprox.formbase.AddRecordForm — **saip.controllers.relacion\_controller.AddRelacion**

Define el formato de la tabla para agregar relaciones.

### 30.4.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Relacion</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'id_item_1', 'id_item_2', 'item_1']</code>



### 30.5 Class *RelacionController*

tgext.crud.CrudRestController — **saip.controllers.relacion\_controller.RelacionController**

Controlador del modelo Relación para el módulo de desarrollo.

#### 30.5.1 Methods

**get\_one**(*self*, *relacion\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista las relaciones de acuerdo a lo establecido en `relacion_controller.RelacionTableFiller._do_get_provider_count_and_objs`.

**new**(*self*, \**args*, \*\**kw*)

Permite la creación de una nueva relación entre dos ítems de un determinado proyecto.

**buscar**(*self*, \*\**kw*)

Lista las relaciones de acuerdo a un criterio de búsqueda introducido por el usuario.

**crear\_version**(*self*, *it*, *borrado*=None)

Crea una nueva versión del ítem que es el hijo o el sucesor en la relación.

**post**(*self*, \*\**kw*)

Registra la nueva relación creada.

**crear\_revision**(*self*, *item*, *msg*)

Crea una revisión cuando al borrar una relación un ítem queda huérfano, se le genera una revisión al mismo.

**post\_delete**(*self*, \**args*, \*\**kw*)

Borra una relación de la base de datos, con las dependencias correspondientes.

#### 30.5.2 Class Variables

Name	Description
fases	<b>Value:</b> FaseController(DBSession)
model	<b>Value:</b> Relacion
table	<b>Value:</b> RelacionTable(DBSession)
table_filler	<b>Value:</b> RelacionTableFiller(DBSession)
new_form	<b>Value:</b> AddRelacion(DBSession)

## 31 Module `saip.controllers.relacion_controller_listado`

Módulo que define el controlador de listado de relaciones en el menú de versión.

### Author:

- Alejandro Arce<sup>70</sup>
- Gabriel Caroni<sup>71</sup>
- Rodrigo Parra<sup>72</sup>

### 31.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>relacion_table</code>	<b>Value:</b> <code>RelacionTable(DBSession)</code>
<code>relacion_table_filler</code>	<b>Value:</b> <code>RelacionTableFiller(DBSession)</code>
<code>add_relacion_form</code>	<b>Value:</b> <code>AddRelacion(DBSession)</code>

### 31.2 Class `RelacionTable`

`sprox.tablebase.TableBase` — `saip.controllers.relacion_controller_listado.RelacionTable`

#### 31.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Relacion</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id_item_1', 'id_item_2', '__actions__']</code>
<code>__xml_fields__</code>	<b>Value:</b> <code>['fase']</code>

<sup>70</sup><mailto:alearce07@gmail.com>

<sup>71</sup><mailto:gabrielcaroni@gmail.com>

<sup>72</sup><mailto:rodpar07@gmail.com>



### 31.5 Class `RelacionControllerListado`

`tgext.crud.CrudRestController` — `saip.controllers.relacion_controller_listado.RelacionControllerL`

Controlador del modelo Relaciones para el menú de versiones.

#### 31.5.1 Methods

**`get_one(self, relacion_id)`**

**`get_all(self, *args, **kw)`**

Lista las relaciones para el menú versión de acuerdo a lo establecido en `fase_controller.FaseTableFiller.do_get_provider_count_and_objs`.

**`buscar(self, **kw)`**

Lista las relaciones de acuerdo a un criterio de búsqueda introducido por el usuario.

#### 31.5.2 Class Variables

Name	Description
<code>fases</code>	<b>Value:</b> <code>FaseController(DBSession)</code>
<code>model</code>	<b>Value:</b> <code>Relacion</code>
<code>table</code>	<b>Value:</b> <code>RelacionTable(DBSession)</code>
<code>table_filler</code>	<b>Value:</b> <code>RelacionTableFiller(DBSession)</code>
<code>new_form</code>	<b>Value:</b> <code>AddRelacion(DBSession)</code>

## 32 Module `saip.controllers.revision_controller`

Controlador de revisiones de un ítem dado en el módulo de desarrollo.

### Author:

- Alejandro Arce<sup>73</sup>
- Gabriel Caroni<sup>74</sup>
- Rodrigo Parra<sup>75</sup>

### 32.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>revision_table</code>	<b>Value:</b> <code>RevisionTable(DBSession)</code>
<code>revision_table_filler</code>	<b>Value:</b> <code>RevisionTableFiller(DBSession)</code>

### 32.2 Class `RevisionTable`

`sprox.tablebase.TableBase` — `saip.controllers.revision_controller.RevisionTable`

Define el formato de la tabla.

#### 32.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Revision</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id_item', 'item']</code>

### 32.3 Class `RevisionTableFiller`

`sprox.fillerbase.TableFiller` — `saip.controllers.revision_controller.RevisionTableFiller`

<sup>73</sup><mailto:alearce07@gmail.com>

<sup>74</sup><mailto:gabrielcaroni@gmail.com>

<sup>75</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas.

### 32.3.1 Methods

**\_\_actions\_\_**(*self*, *obj*)

Define las acciones posibles para cada revisión.

**init**(*self*, *buscado*, *id\_item*, *version*)

### 32.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Revision</code>
<code>id_item</code>	<b>Value:</b> <code>" "</code>
<code>buscado</code>	<b>Value:</b> <code>" "</code>
<code>version</code>	<b>Value:</b> <code>" "</code>

## 32.4 Class *RevisionController*

tgext.crud.CrudRestController — saip.controllers.revision\_controller.RevisionController

Controlador de revisiones de un ítem dado.

### 32.4.1 Methods

**get\_one**(*self*, *revision\_id*)

**get\_all**(*self*, *\*args*, *\*\*kw*)

Lista las revisiones de un ítem de acuerdo a condiciones establecidas en el `revision_controller.RevisionTableFiller`.  
`.do_get_provider_count_and_objs`.

**buscar**(*self*, *\*\*kw*)

Lista las revisiones de acuerdo a un criterio de búsqueda introducido por el usuario.

**32.4.2 Class Variables**

Name	Description
model	<b>Value:</b> Revision
table	<b>Value:</b> RevisionTable(DBSession)
table_filler	<b>Value:</b> RevisionTableFiller(DBSession)



### 33 Module saip.controllers.rol\_controller

Controlador de proyectos en el módulo de administración.

#### Author:

- Alejandro Arce<sup>76</sup>
- Gabriel Caroni<sup>77</sup>
- Rodrigo Parra<sup>78</sup>

#### 33.1 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
rol_table	<b>Value:</b> RolTable(DBSession)
rol_table_filler	<b>Value:</b> RolTableFiller(DBSession)
add_rol_form	<b>Value:</b> AddRol(DBSession)
edit_rol_form	<b>Value:</b> EditRol(DBSession)
rol_edit_filler	<b>Value:</b> RolEditFiller(DBSession)

#### 33.2 Class ValidarExpresion

formencode.validators.Regex  **saip.controllers.rol\_controller.ValidarExpresion**

Clase que se utiliza para validar datos ingresados por el usuario, recibe como parámetro una expresión regular.

##### 33.2.1 Class Variables

Name	Description
messages	<b>Value:</b> {'invalid': ("Introduzca un valor que empiece con una letr...

<sup>76</sup><mailto:alearce07@gmail.com>

<sup>77</sup><mailto:gabrielcaroni@gmail.com>

<sup>78</sup><mailto:rodpar07@gmail.com>

### 33.3 Class RolTable



Define el formato de la tabla

#### 33.3.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Rol
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'fichas', 'usuarios', 'permisos']

### 33.4 Class RolTableFiller



Clase que se utiliza para llenar las tablas.

#### 33.4.1 Methods

<b><code>__actions__(self, obj)</code></b>
Define las acciones posibles para cada proyecto.
<b><code>init(self, buscado)</code></b>

#### 33.4.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Rol
<code>buscado</code>	<b>Value:</b> ""

### 33.5 Class AddRol


sprox.formbase.AddRecordForm  saip.controllers.rol\_controller.AddRol

Define el formato del formulario para crear un nuevo rol

#### 33.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Rol
<code>__omit_fields__</code>	<b>Value:</b> ['id', 'fichas', 'usuarios', 'permisos']
<code>nombre</code>	<b>Value:</b> All(NotEmpty(), ValidarExpresion(r'^[A-Za-z][A-Za-z0-9]*...'))
<code>tipo</code>	<b>Value:</b> SingleSelectField("tipo", options= ['Sistema', 'Proyecto'...])

### 33.6 Class PermisosField

sprox.widgets.dojo.SproxDojoSelectShuttleField  saip.controllers.rol\_controller.PermisosField

Clase para obtener los permisos disponibles para la creación de un rol.

#### 33.6.1 Methods

<code>update_params(self, d)</code>
-------------------------------------

#### 33.6.2 Class Variables

Name	Description
<code>template</code>	<b>Value:</b> 'saip.templates.selectshuttle'

### 33.7 Class EditRol

sprox.dojo.formbase.DojoEditableForm —  
**saip.controllers.rol\_controller.EditRol**

Define el formato de la tabla para editar roles.

#### 33.7.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Rol
<code>__limit_fields__</code>	<b>Value:</b> ['permisos']
<code>__hide_fields__</code>	<b>Value:</b> ['id']
<code>__dropdown_field_names__</code>	<b>Value:</b> {'permisos': 'nombre'}

### 33.8 Class RolEditFiller

sprox.fillerbase.EditFormFiller —  
**saip.controllers.rol\_controller.RolEditFiller**

Completa la tabla para editar roles.

#### 33.8.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Rol

### 33.9 Class RolController

tgext.crud.CrudRestController —  
**saip.controllers.rol\_controller.RolController**

Controlador de Rol para el módulo de administración.

#### 33.9.1 Methods

<code>get_one(self, Rol_id)</code>
------------------------------------

**get\_all**(*self*, \*args, \*\*kw)

Lista los roles existentes de acuerdo a condiciones establecidas en el `rol_controller.RolTableFiller._do_get_provider_count_and_objs`.

**new**(*self*, \*args, \*\*kw)

Despliega una página para la creación de un nuevo rol.

**edit**(*self*, \*args, \*\*kw)

Despliega una página para la modificación de un rol.

**buscar**(*self*, \*\*kw)

Lista los roles de acuerdo a un criterio de búsqueda introducido por el usuario.

**post**(*self*, \*\*kw)

Registra el nuevo rol creado.

### 33.9.2 Class Variables

Name	Description
model	<b>Value:</b> Rol
table	<b>Value:</b> RolTable(DBSession)
table_filler	<b>Value:</b> RolTableFiller(DBSession)
edit_filler	<b>Value:</b> RolEditFiller(DBSession)
edit_form	<b>Value:</b> EditRol(DBSession)
new_form	<b>Value:</b> AddRol(DBSession)

## 34 Module `saip.controllers.root`

Main Controller

### 34.1 Class `RootController`



The root controller for the SAIP application.

All the other controllers and WSGI applications should be mounted on this controller. For example:

```

panel = ControlPanelController()
another_app = AnotherWSGIApplication()
  
```

Keep in mind that WSGI applications shouldn't be mounted directly: They must be wrapped around with :class:`tg.controllers.WSGIAppController`.

#### 34.1.1 Methods

<b>fase_apt</b> <i>a</i> ( <i>self</i> , <i>proyecto</i> )
--

<b>index</b> ( <i>self</i> )
------------------------------

Handle the front-page.
------------------------

<b>about</b> ( <i>self</i> )
------------------------------

Handle the 'about' page.
--------------------------

<b>environ</b> ( <i>self</i> )
--------------------------------

This method showcases TG's access to the wsgi environment.
--

<b>data</b> ( <i>self</i> , <i>**kw</i> )
---

This method showcases how you can use the same controller for a data page and a display page
--

**auth**(*self*)

Display some information about auth\* on this application.

**manage\_permission\_only**(*self*, \*\**kw*)

Illustrate how a page for managers only works.

**editor\_user\_only**(*self*, \*\**kw*)

Illustrate how a page exclusive for the editor works.

**login**(*self*, *came\_from*=url('/'))

Start the user login.

**post\_login**(*self*, *came\_from*='/')

Redirect the user to the initially requested page on successful authentication or redirect her back to the login page if login failed.

**post\_logout**(*self*, *came\_from*=url('/'))

Redirect the user to the initially requested page on logout and say goodbye as well.

*Inherited from saip.lib.base.BaseController(Section 44.1)*

`__call__()`

### 34.1.2 Class Variables

Name	Description
desarrollo	<b>Value:</b> DesarrolloController()
secc	<b>Value:</b> SecureController()
admin	<b>Value:</b> AdminController()
gestion	<b>Value:</b> GestionController()
error	<b>Value:</b> ErrorController()

## 35 Module saip.controllers.secure

Sample controller with all its actions protected.

### 35.1 Class SecureController



Sample controller-wide authorization

#### 35.1.1 Methods

<b>index</b> ( <i>self</i> )
------------------------------

Let the user know that's visiting a protected controller.
---

<b>some_where</b> ( <i>self</i> )
-----------------------------------

Let the user know that this action is protected too.
--

*Inherited from saip.lib.base.BaseController(Section 44.1)*

`--call--()`

#### 35.1.2 Class Variables

Name	Description
allow_only	<b>Value:</b> has_permission('manage', msg=1_('Only for people with th...



## 36 Module `saip.controllers.template`

Fallback controller.

### 36.1 Class `TemplateController`



The fallback controller for SAIP.

By default, the final controller tried to fulfill the request when no other routes match. It may be used to display a template when all else fails, e.g.:

```
def view(self, url):
    return render('/%s' % url)
```

Or if you're using Mako and want to explicitly send a 404 (Not Found) response code when the requested template doesn't exist:

```
import mako.exceptions

def view(self, url):
    try:
        return render('/%s' % url)
    except mako.exceptions.TopLevelLookupException:
        abort(404)
```

#### 36.1.1 Methods

<b><code>view(self, url)</code></b>
Abort the request with a 404 HTTP status code.

*Inherited from `saip.lib.base.BaseController` (Section 44.1)*

`__call__()`

## 37 Module saip.controllers.tipo\_item\_controller

Controlador de tipos de ítem en el módulo de administración.

### Author:

- Alejandro Arce<sup>79</sup>
- Gabriel Caroni<sup>80</sup>
- Rodrigo Parra<sup>81</sup>

### 37.1 Variables

Name	Description
errors	<b>Value:</b> IntegrityError, DatabaseError, ProgrammingError
tipo_item_table	<b>Value:</b> TipoItemTable(DBSession)
tipo_item_table.filler	<b>Value:</b> TipoItemTableFiller(DBSession)
add_tipo_item_form	<b>Value:</b> AddTipoItem(DBSession)
edit_tipo_item_form	<b>Value:</b> EditTipoItem(DBSession)
tipo_item_edit.filler	<b>Value:</b> TipoItemEditFiller(DBSession)

### 37.2 Class ValidarExpresion

formencode.validators.Regex

saip.controllers.tipo\_item\_controller.ValidarExpresion

Clase que se utiliza para validar datos ingresados por el usuario, recibe como parámetro una expresión regular.

#### 37.2.1 Class Variables

Name	Description
messages	<b>Value:</b> {'invalid': ("Introduzca un valor que empiece con una letr...

<sup>79</sup><mailto:alearce07@gmail.com>

<sup>80</sup><mailto:gabrielcaroni@gmail.com>

<sup>81</sup><mailto:rodpar07@gmail.com>

### 37.3 Class Unico

formencode.FancyValidator —  
**saip.controllers.tipo\_item\_controller.Unico**

Clase correspondiente a un validador que se utiliza para controlar que el nombre ingresado para cierto tipo de ítem añadido o modificado no se repita dentro de una fase.

### 37.4 Class CodigoUnico

formencode.FancyValidator —  
**saip.controllers.tipo\_item\_controller.CodigoUnico**

Clase correspondiente a un validador que se utiliza para controlar que el código ingresado para cierto tipo de ítem añadido a una fase no se repita dentro de la misma.

### 37.5 Class TipoItemTable

sprox.tablebase.TableBase —  
**saip.controllers.tipo\_item\_controller.TipoItemTable**

Define el formato de la tabla.

#### 37.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fase', 'id_fase', 'items', 'caracteristicas']</code>

### 37.6 Class TipoItemTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.tipo\_item\_controller.TipoItemTableFiller**

Clase que se utiliza para llenar las tablas.

### 37.6.1 Methods

<b>__actions__</b> ( <i>self, obj</i> )
---

Define las acciones posibles para cada tipo de ítem.
--

<b>init</b> ( <i>self, buscado, id_fase</i> )
---

### 37.6.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>buscado</code>	<b>Value:</b> <code>""</code>
<code>id_fase</code>	<b>Value:</b> <code>""</code>

## 37.7 Class *AddTipoItem*

sprox.formbase.AddRecordForm

saip.controllers.tipo\_item\_controller.AddTipoItem

Define el formato del formulario para crear un nuevo tipo de ítem

### 37.7.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fase', 'id_fase', 'items', 'caracteristicas']</code>
<code>nombre</code>	<b>Value:</b> <code>All(NotEmpty(), ValidarExpresion(r'^[A-Za-z] [A-Za-z0-9]*...'))</code>
<code>codigo</code>	<b>Value:</b> <code>All(NotEmpty(), ValidarExpresion(r'^[A-Za-z] [A-Za-z]*\$'), ...)</code>

### 37.8 Class **EditTipoItem**

sprox.formbase.EditableForm —  
**saip.controllers.tipo\_item\_controller.EditTipoItem**

Define el formato del formulario para la modificación de un tipo de ítem

#### 37.8.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>__hide_fields__</code>	<b>Value:</b> <code>['id', 'fase', 'items', 'caracteristicas', 'codigo']</code>
<code>nombre</code>	<b>Value:</b> <code>All(NotEmpty(), ValidarExpresion(r'^[A-Za-z][A-Za-z0-9]*...'))</code>

### 37.9 Class **TipoItemEditFiller**

sprox.fillerbase.EditFormFiller —  
**saip.controllers.tipo\_item\_controller.TipoItemEditFiller**

Se utiliza para llenar el formulario de modificación de un tipo de ítem con los valores recuperados de la base de datos.

#### 37.9.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>

### 37.10 Class **TipoItemController**

tgext.crud.CrudRestController —  
**saip.controllers.tipo\_item\_controller.TipoItemController**

Controlador de tipos de ítem

**37.10.1 Methods**

<b>get_one</b> ( <i>self</i> , <i>tipo_item_id</i> )
--

<b>get_all</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Lista los tipos de ítem existentes de acuerdo a condiciones establecidas en el <code>tipo_item_controller.TipoItemTableFiller.do_get_provider_count_and_objs</code> .
---

<b>new</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
---

Despliega una página para la creación de un nuevo tipo de ítem
--

<b>edit</b> ( <i>self</i> , * <i>args</i> , ** <i>kw</i> )
--

Despliega una página para la modificación de un tipo de ítem
--

<b>buscar</b> ( <i>self</i> , ** <i>kw</i> )
--

Lista los tipos de ítem de acuerdo a un criterio de búsqueda introducido por el usuario.
--

<b>post</b> ( <i>self</i> , ** <i>kw</i> )
--

Registra el nuevo tipo de ítem creado.
--

**37.10.2 Class Variables**

Name	Description
proyectos	<b>Value:</b> ProyectoControllerNuevo()
caracteristicas	<b>Value:</b> CaracteristicaController(DBSession)
model	<b>Value:</b> TipoItem
table	<b>Value:</b> TipoItemTable(DBSession)
table_filler	<b>Value:</b> TipoItemTableFiller(DBSession)
edit_filler	<b>Value:</b> TipoItemEditFiller(DBSession)
edit_form	<b>Value:</b> EditTipoItem(DBSession)
new_form	<b>Value:</b> AddTipoItem(DBSession)
id_fase	<b>Value:</b> None

## 38 Module `saip.controllers.tipo_item_controller_nuevo`

Controlador de tipos de ítem en el módulo de administración utilizado para la importación de tipos de ítem.

### Author:

- Alejandro Arce<sup>82</sup>
- Gabriel Caroni<sup>83</sup>
- Rodrigo Parra<sup>84</sup>

### 38.1 Variables

Name	Description
<code>tipo_item_table</code>	<b>Value:</b> <code>TipoItemTable(DBSession)</code>
<code>tipo_item_table_filler</code>	<b>Value:</b> <code>TipoItemTableFiller(DBSession)</code>

### 38.2 Class `TipoItemTable`

`sprox.tablebase.TableBase` — `saip.controllers.tipo_item_controller_nuevo.TipoItemTable`

Define el formato de la tabla

#### 38.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fase', 'id_fase', 'items', 'caracteristicas']</code>

### 38.3 Class `TipoItemTableFiller`

`sprox.fillerbase.TableFiller` — `saip.controllers.tipo_item_controller_nuevo.TipoItemTableFiller`

<sup>82</sup><mailto:alearce07@gmail.com>

<sup>83</sup><mailto:gabrielcaroni@gmail.com>

<sup>84</sup><mailto:rodpar07@gmail.com>

Clase que se utiliza para llenar las tablas.

### 38.3.1 Methods

**`__actions__(self, obj)`**

Define las acciones posibles para cada tipo de ítem.

**`init(self, buscado)`**

### 38.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>TipoItem</code>
<code>buscado</code>	<b>Value:</b> <code>" "</code>
<code>id_fase</code>	<b>Value:</b> <code>" "</code>
<code>id_fase_que_importa</code>	<b>Value:</b> <code>" "</code>

## 38.4 Class *TipoItemControllerNuevo*

tg.controllers.RestController

saip.controllers.tipo\_item\_controller\_nuevo.TipoItemControllerN

Controlador de tipos de ítem utilizado para la importación

### 38.4.1 Methods

**`get_one(self, fase_id)`**

**`get_all(self)`**

Lista los tipos de ítem existentes de acuerdo a condiciones establecidas en el `tipo_item_controller_nuevo.TipoItemTableFiller`  
`._do_get_provider_count_and_objs.`

**`buscar(self, **kw)`**

Lista los tipos de ítem de acuerdo a un criterio de búsqueda introducido por el usuario.



```
importar_caracteristica(self, id_tipo_item_viejo, id_tipo_item_nuevo)
```

Importa las características correspondientes al tipo de ítem a importar.

**Parameters**

**id\_tipo\_item\_viejo:** Id del tipo de ítem a importar.

(*type=Unicode.*)

**id\_tipo\_item\_nuevo:** Id del tipo de ítem nuevo o importado.

(*type=Unicode*)

```
importar_tipo_item(self, *args, **kw)
```

Importa un tipo de ítem a una fase determinada.

### 38.4.2 Class Variables

Name	Description
table	<b>Value:</b> TipoItemTable(DBSession)
tipo_item_filler	<b>Value:</b> TipoItemTableFiller(DBSession)

## 39 Module *saip.controllers.usuario\_controller*

Módulo que define el controlador de usuarios.

### Author:

- Alejandro Arce<sup>85</sup>
- Gabriel Caroni<sup>86</sup>
- Rodrigo Parra<sup>87</sup>

### 39.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>()</code>
<code>usuario_table</code>	<b>Value:</b> <code>UsuarioTable(DBSession)</code>
<code>usuario_table_filler</code>	<b>Value:</b> <code>UsuarioTableFiller(DBSession)</code>
<code>form_validator</code>	<b>Value:</b> <code>Schema(password= NotEmpty(), chained_validators= (FieldsM...</code>
<code>add_usuario_form</code>	<b>Value:</b> <code>AddUsuario(DBSession)</code>
<code>form_validator_2</code>	<b>Value:</b> <code>Schema(chained_validators= (FieldsMatch('nuevo_password',...</code>
<code>edit_usuario_form</code>	<b>Value:</b> <code>EditUsuario(DBSession)</code>
<code>usuario_edit_filler</code>	<b>Value:</b> <code>UsuarioEditFiller(DBSession)</code>

### 39.2 Class *UsuarioTable*

`sprox.tablebase.TableBase` — `saip.controllers.usuario_controller.UsuarioTable`

#### 39.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Usuario</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fichas', '_password', 'password', 'roles', 'proye...</code>

<sup>85</sup><mailto:alearce07@gmail.com>

<sup>86</sup><mailto:gabrielcaroni@gmail.com>

<sup>87</sup><mailto:rodpar07@gmail.com>

### 39.3 Class UsuarioTableFiller

sprox.fillerbase.TableFiller —  
**saip.controllers.usuario\_controller.UsuarioTableFiller**

Clase que se utiliza para llenar las tablas de usuario.

#### 39.3.1 Methods

<b>__actions__</b> ( <i>self, obj</i> )
---

Define las acciones posibles para cada usuario.
---

<b>init</b> ( <i>self, buscado</i> )
--------------------------------------

#### 39.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Usuario
<code>buscado</code>	<b>Value:</b> ""

### 39.4 Class Unico

formencode.FancyValidator —  
**saip.controllers.usuario\_controller.Unico**

Clase para verificar que el nombre de usuario introducido sea único.

### 39.5 Class AddUsuario

sprox.formbase.AddRecordForm —  
**saip.controllers.usuario\_controller.AddUsuario**

Define el formato de la tabla para agregar usuarios.

#### 39.5.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Usuario</code>
<code>__base_validator__</code>	<b>Value:</b> <code>Schema(password= NotEmpty(), chained_validators= (FieldsM...</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'fichas', '_password', 'roles', 'proyectos']</code>
<code>nombre_usuario</code>	<b>Value:</b> <code>All(Unico(), NotEmpty())</code>
<code>password</code>	<b>Value:</b> <code>PasswordField('password')</code>
<code>password_c</code>	<b>Value:</b> <code>PasswordField('confirmar_password')</code>

### 39.6 Class `EditUsuario`

`sprox.formbase.EditableForm` └─ **`saip.controllers.usuario_controller.EditUsuario`**

Define el formato de la tabla para editar usuarios.

#### 39.6.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Usuario</code>
<code>__base_validator__</code>	<b>Value:</b> <code>Schema(chained_validators= (FieldsMatch('nuevo_password',...</code>
<code>__hide_fields__</code>	<b>Value:</b> <code>['id', 'nombre_usuario', 'fichas', '_password', 'roles', ...</code>
<code>password_a</code>	<b>Value:</b> <code>PasswordField('nuevo_password')</code>
<code>password_c</code>	<b>Value:</b> <code>PasswordField('confirmar_password')</code>

### 39.7 Class `UsuarioEditFiller`

`sprox.fillerbase.EditFormFiller` └─ **`saip.controllers.usuario_controller.UsuarioEditFiller`**

Completa la tabla para editar usuarios.

#### 39.7.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Usuario

### 39.8 Class *UsuarioController*

tgext.crud.CrudRestController — **saip.controllers.usuario\_controller.UsuarioController**

Controlador del modelo Usuario.

#### 39.8.1 Methods

**get\_one**(*self*, *usuario\_id*)

**get\_all**(*self*, \**args*, \*\**kw*)

Lista las fases de acuerdo a lo establecido en `usuario_controller.UsuarioTableFiller._do_get_provider_count_and_objs`.

**new**(*self*, \**args*, \*\**kw*)

Permite la creación de nuevos usuarios en el sistema.

**edit**(*self*, \**args*, \*\**kw*)

Permite la edición de usuarios del sistema.

**buscar**(*self*, \*\**kw*)

Lista los usuarios de acuerdo a un criterio de búsqueda.

**post**(*self*, \*\**kw*)

Registra el nuevo usuario creado.

**put**(*self*, \**args*, \*\**kw*)

Registra los cambios realizados en un usuario.

#### 39.8.2 Class Variables

Name	Description
model	<b>Value:</b> Usuario
table	<b>Value:</b> UsuarioTable(DBSession)
table_filler	<b>Value:</b> UsuarioTableFiller(DBSession)
edit_filler	<b>Value:</b> UsuarioEditFiller(DBSession)
edit_form	<b>Value:</b> EditUsuario(DBSession)
new_form	<b>Value:</b> AddUsuario(DBSession)
responsabilidades	<b>Value:</b> FichaUsuarioController(DBSession)

## 40 Module *saip.controllers.version\_controller*

Módulo que define el controlador de versiones anteriores de un ítem.

### Author:

- Alejandro Arce<sup>88</sup>
- Gabriel Caroni<sup>89</sup>
- Rodrigo Parra<sup>90</sup>

### 40.1 Variables

Name	Description
<code>errors</code>	<b>Value:</b> <code>IntegrityError</code> , <code>DatabaseError</code> , <code>ProgrammingError</code>
<code>item_table</code>	<b>Value:</b> <code>ItemTable(DBSession)</code>
<code>item_table_filler</code>	<b>Value:</b> <code>ItemTableFiller(DBSession)</code>

### 40.2 Class *ItemTable*

`sprox.tablebase.TableBase` — `saip.controllers.version_controller.ItemTable`

Define el formato de la tabla

#### 40.2.1 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> <code>Item</code>
<code>__omit_fields__</code>	<b>Value:</b> <code>['id', 'id_tipo_item', 'id_fase', 'id_linea_base', 'archi...</code>

<sup>88</sup><mailto:alearce07@gmail.com>

<sup>89</sup><mailto:gabrielcaroni@gmail.com>

<sup>90</sup><mailto:rodpar07@gmail.com>

### 40.3 Class ItemTableFiller

sprox.fillerbase.TableFiller —  
                                   saip.controllers.version\_controller.ItemTableFiller

Clase que se utiliza para llenar las tablas.

#### 40.3.1 Methods

<b>tipo_item</b> ( <i>self</i> , <i>obj</i> )
---

<b>__actions__</b> ( <i>self</i> , <i>obj</i> )
---

Define las acciones posibles para cada ítem.
--

<b>init</b> ( <i>self</i> , <i>buscado</i> , <i>id_item</i> , <i>version</i> )
--

#### 40.3.2 Class Variables

Name	Description
<code>__model__</code>	<b>Value:</b> Item
<code>buscado</code>	<b>Value:</b> ""
<code>id_item</code>	<b>Value:</b> ""
<code>version</code>	<b>Value:</b> ""

### 40.4 Class VersionController

tgext.crud.CrudRestController —  
                                   saip.controllers.version\_controller.VersionController

Controlador de versiones anteriores de un ítem.

#### 40.4.1 Methods

<b>get_one</b> ( <i>self</i> , <i>item_id</i> )
---



**crear\_version\_sin\_relaciones**(*self*, *it*, *id\_item*)

Crea una nueva versión de un ítem dado sin sus relaciones.

**Parameters**

**it:** Item a partir del cual se creará una nueva versión.  
(*type=Item*)

**id\_item:** Id del ítem a reversionar.  
(*type=String*)

**crear\_version**(*self*, *it*, *borrado=None*)

Crea una nueva versión de un ítem dado. Permite también borrar una relación de esta nueva versión si se especifica.

**Parameters**

**it:** Item a partir del cual se creará una nueva versión.  
(*type=Item*)

**borrado:** Relación a eliminar.  
(*type=Relacion*)

**crear\_relacion**(*self*, *item\_1*, *item\_2*)

Crea una relación entre dos ítems.

**Parameters**

**item\_1:** Item antecesor/padre de la relación a ser creada.  
(*type=Item*)

**item\_2:** Item sucesor/hijo de la relación a ser creada.  
(*type=Item*)

**Return Value**

True si la relación se creó exitosamente, False en caso contrario.  
(*type=Bool*)

**crear\_revision**(*self*, *item*, *msg*)

Crea una nueva revisión y la agrega a un ítem dado.

**Parameters**

**item:** Item al cual se agregará la revisión  
(*type=Item*)

**msg:** Mensaje de la revisión  
(*type=String*)

**revertir**(*self*, \*args, \*\*kw)

Revierte un ítem a una versión especificada en los parámetros.

**get\_all**(*self*, \*args, \*\*kw)

Lista las versiones de un ítem de acuerdo a condiciones establecidas en el `version_controller.ItemTableFiller .do_get_provider_count_and_objs`.

**buscar**(*self*, \*\*kw)

Lista las versiones de un ítem de acuerdo a un criterio de búsqueda introducido por el usuario.

**listar\_caracteristicas**(*self*, \*\*kw)

Despliega el valor de las características propias del tipo de ítem de una versión de un ítem dado, siempre que el ítem no sea del tipo 'Default'

#### 40.4.2 Class Variables

Name	Description
relaciones	<b>Value:</b> RelacionControllerListado(DBSession)
archivos	<b>Value:</b> ArchivoControllerListado(DBSession)
model	<b>Value:</b> Item
table	<b>Value:</b> ItemTable(DBSession)
table_filler	<b>Value:</b> ItemTableFiller(DBSession)

## 41 Package saip.lib

### 41.1 Modules

- **app\_globals**: The application's Globals object  
(Section 42, p. 122)
- **auth**: Módulo que provee los predicates checkers para controlar la autorización de los usuarios para el acceso a los recursos del sistema.  
(Section 43, p. 123)
- **base**: The base Controller API.  
(Section 44, p. 125)
- **func**: Módulo que provee funciones varias para la utilización en los controladores.  
(Section 45, p. 126)
- **helpers**: WebHelpers used in SAIP.  
(Section 46, p. 131)

### 41.2 Variables

Name	Description
--package--	<b>Value:</b> None

## 42 Module *saip.lib.app\_globals*

The application's Globals object

### 42.1 Class Globals

object └─  
***saip.lib.app\_globals.Globals***

Container for objects available throughout the life of the application.

One instance of Globals is created during application initialization and is available during requests via the 'app\_globals' variable.

#### 42.1.1 Methods

<b><code>__init__(self)</code></b>
Do nothing, by default.
Overrides: <i>object.__init__</i>

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 42.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 43 Module *saip.lib.auth*

Módulo que provee los predicates checkers para controlar la autorización de los usuarios para el acceso a los recursos del sistema.

### Author:

- Alejandro Arce<sup>91</sup>
- Gabriel Caroni<sup>92</sup>
- Rodrigo Parra<sup>93</sup>

### 43.1 Class *TieneAlgunPermiso*

repoze.what.predicates.Predicate — **saip.lib.auth.TieneAlgunPermiso**

Predicate checker que verifica si el usuario cuenta con algún permiso de acuerdo a las condiciones que recibe como parámetro.

#### 43.1.1 Methods

```
__init__(self, **kwargs)
```

```
evaluate(self, environ, credentials)
```

#### 43.1.2 Class Variables

Name	Description
message	<b>Value:</b> "El usuario no cuenta con ningun permiso"+ "de las caract..."

### 43.2 Class *TienePermiso*

repoze.what.predicates.Predicate — **saip.lib.auth.TienePermiso**

<sup>91</sup><mailto:alearce07@gmail.com>

<sup>92</sup><mailto:gabrielcaroni@gmail.com>

<sup>93</sup><mailto:rodpar07@gmail.com>

Predicate checker que evalúa si el usuario cuenta con un permiso particular de acuerdo a las condiciones que recibe como parámetro

#### 43.2.1 Methods

<code>__init__(self, permiso, **kwargs)</code>
--

<code>evaluate(self, environ, credentials)</code>
---

#### 43.2.2 Class Variables

Name	Description
message	<b>Value:</b> "El usuario no cuenta con los permisos necesarios"+ "para..."

## 44 Module **saip.lib.base**

The base Controller API.

### 44.1 Class **BaseController**

tg.TGController  **saip.lib.base.BaseController**

Base class for the controllers in the application.

Your web application should have one of these. The root of your application is used to compute URLs used by your app.

#### 44.1.1 Methods

<code><b>__call__</b>(<i>self</i>, <i>environ</i>, <i>start_response</i>)</code>
--

Invoke the Controller
-----------------------

## 45 Module *saip.lib.func*

Módulo que provee funciones varias para la utilización en los controladores.

### Author:

- Alejandro Arce<sup>94</sup>
- Gabriel Caroni<sup>95</sup>
- Rodrigo Parra<sup>96</sup>

### 45.1 Functions

#### **es\_huerfano**(*item*)

Determina si un item es considerado huérfano o no.

#### **Parameters**

**item:** Item que se analizará  
(*type=Item*)

#### **Return Value**

True si el item es huérfano, False en caso contrario.  
(*type=Bool*)

#### **opuesto**(*arista, nodo*)

Devuelve el otro item correspondiente a una relación

#### **Parameters**

**arista:** Relación que se analizará  
(*type=Relacion*)

**nodo:** Item conocido de la relación  
(*type=Item*)

#### **Return Value**

El otro item correspondiente a la relación, es decir, el que no se recibió como parámetro  
(*type=Item*)

<sup>94</sup><mailto:alearce07@gmail.com>

<sup>95</sup><mailto:gabrielcaroni@gmail.com>

<sup>96</sup><mailto:rodpar07@gmail.com>



**relaciones\_a\_actualizadas**(*aristas*)

Provee la lista de relaciones actualizadas (con el hijo/sucesor en su versión actual) a partir de una lista de relaciones .

**Parameters**

**aristas:** lista de relaciones  
(*type=list(Relacion)*)

**Return Value**

lista filtrada de relaciones con el item hijo/sucesor en su en su versión actual  
(*type=list(Relacion)*)

**relaciones\_b\_actualizadas**(*aristas*)

Provee la lista de relaciones actualizadas (con el padre/antecesor en su versión actual) a partir de una lista de relaciones .

**Parameters**

**aristas:** lista de relaciones  
(*type=list(Relacion)*)

**Return Value**

lista filtrada de relaciones con el item hijo/sucesor en su en su versión actual  
(*type=list(Relacion)*)

**relaciones\_a\_recuperar**(*aristas*)

Provee la lista de relaciones sin duplicados (con el hijo/sucesor en la mayor versión presente en la lista recibida, por cada id\_item) a partir de una lista de relaciones .

**Parameters**

**aristas:** lista de relaciones  
(*type=list(Relacion)*)

**Return Value**

lista filtrada de relaciones con el item hijo/sucesor en la mayor versión presente en aristas  
(*type=list(Relacion)*)

**relaciones\_b\_recuperar**(*aristas*)

Provee la lista de relaciones sin duplicados (con el padre/antecesor en la mayor versión presente en la lista recibida, por cada id\_item) a partir de una lista de relaciones .

**Parameters**

**aristas:** lista de relaciones  
(*type=list(Relacion)*)

**Return Value**

lista filtrada de relaciones con el item padre/antecesor en la mayor versión presente en aristas  
(*type=list(Relacion)*)

**forma\_ciclo**(*nodo*, *nodos\_explorados*=[], *aristas\_exploradas*=[], *band*=False, *nivel*=1)

Determina recursivamente si existe un ciclo en la componente conexas del grafo de items a la que pertenece un item dado.

**Parameters**

**nodo:** Item dado.(Normalmente acaba de añadirse un relación)  
(*type={Item}*)

**nodos\_explorados:** Lista de items que ya han sido visitados.  
(*type=list({Item})*)

**aristas\_exploradas:** Lista de relaciones visitadas.  
(*type=list({Relacion})*)

**band:** Bandera que indica si ya se ha encontrado un bucle.  
(*type=Bool*)

**nivel:** Indica la cantidad de llamadas recursivas anidadas realizadas.  
(*type=Integer*)

**Return Value**

True si existe un bucle, False en caso contrario.  
(*type=Bool*)

**color**(*nodo*)

Determina el color que debe tener un item para la representación gráfica del costo de impacto basado en el orden de la fase a la que pertenece.

**Parameters**

**nodo:** Item a colorear.  
(*type*=*{Item}*)

**Return Value**

El color que debe usarse para colorear el item.  
(*type*=*String*)

**costo\_impacto**(*nodo*, *grafo*, *nodos\_explorados*=[], *aristas\_exploradas*=[], *costo*=0)

Calcula recursivamente el costo de impacto de un item determinado y genera el grafo para la representación gráfica del resultado.

**Parameters**

**nodo:** Item dado  
(*type*=*{Item}*)

**grafo:** Grafo que se utilizará para la representación gráfica de resultados.  
(*type*=*grafo Pydot*)

**nodos\_explorados:** Lista de items que ya han sido visitados.  
(*type*=*list({Item})*)

**aristas\_exploradas:** Lista de relaciones visitadas.  
(*type*=*list({Relacion})*)

**costo:** Suma de las complejidades de los items visitados.  
(*type*=*Integer*)

**Return Value**

El costo de impacto y el grafo para representar el resultado.

**estado\_proyecto**(*proyecto*)

Asigna el estado al correspondiente a un proyecto dado.

**Parameters**

**proyecto:** Proyecto cuyo estado desea actualizarse.  
(*type*=*{Proyecto}*)

**estado\_fase**(*fase*)

Asigna el estado correspondiente a una fase dada.

**Parameters**

**fase:** Fase cuyo estado desea actualizarse.  
(*type*=*{ Fase }*)

**sucesor**(*item*)

Evalúa si un item tiene o no un sucesor en la fase siguiente.

**Parameters**

**item:** Item dado  
(*type*=*{ Item }*)

**Return Value**

True si cuenta con un sucesor, False en caso contrario.

**consistencia\_lb**(*lb*)

Evalúa la consistencia de una línea base y asigna el valor correspondiente.

**Parameters**

**lb:** Línea base a evaluar.  
(*type*=*LineaBase*)

**proximo\_id**(*lista\_ids*)

Determina el siguiente id a ser utilizado para un objeto del sistema (proyecto, fase, item, etc.).

**Parameters**

**lista\_ids:** lista de ids existentes del elemento del modelo dado.  
(*type*=*list()*)

**Return Value**

proximo id a ser utilizado.

## **46    Module saip.lib.helpers**

WebHelpers used in SAIP.

## 47 Package saip.model

The application's model objects

### 47.1 Modules

- **app**: Módulo que define las clases del modelo del sistema no relacionadas a la autenticación ni a la autorización.  
(Section 48, p. 133)
- **auth**: Módulo que define las clases del modelo del sistema relacionadas a la autenticación y a la autorización.  
(Section 49, p. 140)

### 47.2 Functions

<b>init_model</b> ( <i>engine</i> )
-------------------------------------

Call me before using any of the tables or classes in the model.
---

### 47.3 Variables

Name	Description
maker	<b>Value:</b> sessionmaker(autoflush= True, autocommit= False, extensio...
DBSession	<b>Value:</b> scoped_session(maker)
DeclarativeBase	<b>Value:</b> declarative_base()
metadata	<b>Value:</b> DeclarativeBase.metadata

## 48 Module saip.model.app

Módulo que define las clases del modelo del sistema no relacionadas a la autenticación ni a la autorización.

### Author:

- Alejandro Arce<sup>97</sup>
- Gabriel Caroni<sup>98</sup>
- Rodrigo Parra<sup>99</sup>

### 48.1 Class Proyecto

declarative\_base() └─  
saip.model.app.Proyecto

Clase correspondiente a un proyecto del sistema, mapeada a la tabla proyectos de forma declarativa.

#### 48.1.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'proyectos'
id	<b>Value:</b> Column(Unicode, primary_key=True)
nombre	<b>Value:</b> Column(Unicode, nullable= False)
fecha_inicio	<b>Value:</b> Column(Date)
fecha_fin	<b>Value:</b> Column(Date)
descripcion	<b>Value:</b> Column(Unicode)
estado	<b>Value:</b> Column(Unicode, nullable= False)
nro_fases	<b>Value:</b> Column(Integer, nullable= False)
id_lider	<b>Value:</b> Column(Unicode, ForeignKey("usuarios.id"))
lider	<b>Value:</b> relation("Usuario", backref=backref('proyectos', order_b...

<sup>97</sup><mailto:alearce07@gmail.com>

<sup>98</sup><mailto:gabrielcaroni@gmail.com>

<sup>99</sup><mailto:rodpar07@gmail.com>

## 48.2 Class Fase

```
declarative_base() └─
                    saip.model.app.Fase
```

Clase correspondiente a un fase del sistema, mapeada a la tabla fases de forma declarativa.

### 48.2.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> 'fases'
<code>id</code>	<b>Value:</b> Column(Unicode, primary_key=True)
<code>nombre</code>	<b>Value:</b> Column(Unicode, nullable= False)
<code>orden</code>	<b>Value:</b> Column(Integer, nullable= False)
<code>fecha_inicio</code>	<b>Value:</b> Column(Date)
<code>fecha_fin</code>	<b>Value:</b> Column(Date)
<code>descripcion</code>	<b>Value:</b> Column(Unicode)
<code>estado</code>	<b>Value:</b> Column(Unicode, nullable= False)
<code>id_proyecto</code>	<b>Value:</b> Column(Unicode, ForeignKey("proyectos.id"))
<code>proyecto</code>	<b>Value:</b> relation("Proyecto", backref=backref('fases', cascade= "...

## 48.3 Class TipoItem

```
declarative_base() └─
                    saip.model.app.TipoItem
```

Clase correspondiente a un tipo de item del sistema, mapeada a la tabla tipos\_item de forma declarativa.

### 48.3.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> 'tipos_item'
<code>id</code>	<b>Value:</b> Column(Unicode, primary_key=True)
<code>codigo</code>	<b>Value:</b> Column(Unicode)
<code>nombre</code>	<b>Value:</b> Column(Unicode, nullable= False)

*continued on next page*



Name	Description
descripcion	<b>Value:</b> Column(Unicode)
id_fase	<b>Value:</b> Column(Unicode, ForeignKey("fases.id"))
fase	<b>Value:</b> relation("Fase", backref=backref('tipos_item', cascade= ...

## 48.4 Class Caracteristica

declarative\_base()  **saip.model.app.Caracteristica**

Clase correspondiente a una caracteristica del sistema, mapeada a la tabla caracteristicas de forma declarativa.

### 48.4.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'caracteristicas'
id	<b>Value:</b> Column(Unicode, primary_key=True)
nombre	<b>Value:</b> Column(Unicode, nullable= False)
tipo	<b>Value:</b> Column(Unicode)
descripcion	<b>Value:</b> Column(Unicode)
id_tipo_item	<b>Value:</b> Column(Unicode, ForeignKey("tipos_item.id"))
tipo_item	<b>Value:</b> relation("TipoItem", backref=backref('caracteristicas', ...

## 48.5 Class LineaBase

declarative\_base()  **saip.model.app.LineaBase**

Clase correspondiente a una linea base del sistema, mapeada a la tabla lineas\_base de forma declarativa.

### 48.5.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> <code>'lineas_base'</code>
<code>id</code>	<b>Value:</b> <code>Column(Unicode, primary_key=True)</code>
<code>descripcion</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>cerrado</code>	<b>Value:</b> <code>Column(Boolean, nullable= False)</code>
<code>id_fase</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey("fases.id"))</code>
<code>consistente</code>	<b>Value:</b> <code>Column(Boolean, nullable= False)</code>
<code>fase</code>	<b>Value:</b> <code>relation("Fase", backref=backref('lineas_base', cascade=...</code>

## 48.6 Class Item

`declarative_base()` —  
**saip.model.app.Item**

Clase correspondiente a un item del sistema, mapeada a la tabla items de forma declarativa.

### 48.6.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> <code>'items'</code>
<code>id</code>	<b>Value:</b> <code>Column(Unicode, primary_key=True)</code>
<code>version</code>	<b>Value:</b> <code>Column(Integer, primary_key=True)</code>
<code>codigo</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>nombre</code>	<b>Value:</b> <code>Column(Unicode, nullable= False)</code>
<code>descripcion</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>estado</code>	<b>Value:</b> <code>Column(Unicode, nullable= False)</code>
<code>anexo</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>observaciones</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>prioridad</code>	<b>Value:</b> <code>Column(Integer, nullable= False)</code>
<code>complejidad</code>	<b>Value:</b> <code>Column(Integer, nullable= False)</code>
<code>borrado</code>	<b>Value:</b> <code>Column(Boolean)</code>
<code>id_tipo_item</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey("tipos_item.id"))</code>
<code>id_linea_base</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey("lineas_base.id"))</code>

*continued on next page*

Name	Description
tipo_item	<b>Value:</b> relation("TipoItem", backref=backref('items', cascade= "...
linea_base	<b>Value:</b> relation("LineaBase", backref=backref('items', order_by=...

## 48.7 Class Archivo

declarative\_base() —  
**saip.model.app.Archivo**

Clase correspondiente a una archivo del sistema, mapeada a la tabla archivos de forma declarativa.

### 48.7.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'archivos'
id	<b>Value:</b> Column(Unicode, primary_key=True)
nombre	<b>Value:</b> Column(Unicode)
contenido	<b>Value:</b> Column(LargeBinary)
items	<b>Value:</b> relation("Item", secondary="item_archivo", backref= back...

## 48.8 Class Item\_Archivo

declarative\_base() —  
**saip.model.app.Item\_Archivo**

Clase que mapea items con sus archivos correspondientes. Es utilizada por Ssqlalchemy para manejar la relación muchos a muchos entre *Item* y *Archivo*

### 48.8.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'item_archivo'

*continued on next page*

Name	Description
<code>__table_args__</code>	<b>Value:</b> <code>ForeignKeyConstraint(['id_item', 'version_item'], ['items...</code>
<code>id_item</code>	<b>Value:</b> <code>Column(Unicode, primary_key=True)</code>
<code>version_item</code>	<b>Value:</b> <code>Column(Integer, primary_key=True)</code>
<code>id_archivo</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey('archivos.id', onupdate="CASCADE...</code>

## 48.9 Class Relacion

`declarative_base()` —  
**saip.model.app.Relacion**

Clase correspondiente a una relacion del sistema, mapeada a la tabla relaciones de forma declarativa.

### 48.9.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> <code>'relaciones'</code>
<code>__table_args__</code>	<b>Value:</b> <code>ForeignKeyConstraint(['id_item_1', 'version_item_1'], ['i...</code>
<code>id</code>	<b>Value:</b> <code>Column(Unicode, primary_key=True)</code>
<code>id_item_1</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>version_item_1</code>	<b>Value:</b> <code>Column(Integer)</code>
<code>id_item_2</code>	<b>Value:</b> <code>Column(Unicode)</code>
<code>version_item_2</code>	<b>Value:</b> <code>Column(Integer)</code>
<code>item_1</code>	<b>Value:</b> <code>relation("Item", primaryjoin=and_(id_item_1== Item.id, v...</code>
<code>item_2</code>	<b>Value:</b> <code>relation("Item", primaryjoin=and_(id_item_2== Item.id, v...</code>

## 48.10 Class Revision

```
declarative_base() └─
                    saip.model.app.Revision
```

Clase correspondiente a una revision del sistema, mapeada a la tabla revisiones de forma declarativa.

### 48.10.1 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> 'revisiones'
<code>id</code>	<b>Value:</b> Column(Unicode, primary_key=True)
<code>descripcion</code>	<b>Value:</b> Column(Unicode)
<code>id_item</code>	<b>Value:</b> Column(Unicode, ForeignKey("items.id", "items.version"))
<code>item</code>	<b>Value:</b> relation("Item", backref=backref('revisiones', cascade= ...

## 49 Module *saip.model.auth*

Módulo que define las clases del modelo del sistema relacionadas a la autenticación y a la autorización. Este módulo es utilizado por *repoze.what* y *repoze.who*

### Author:

- Alejandro Arce<sup>100</sup>
- Gabriel Caroni<sup>101</sup>
- Rodrigo Parra<sup>102</sup>

### 49.1 Class *Ficha*

`declarative_base()` └─  
                           ***saip.model.auth.Ficha***

Clase correspondiente a una ficha del sistema mapeada a la tabla *fichas* de forma declarativa. Relaciona un *Rol* con un {Usuario} y, si corresponde, con un {Proyecto} y/o una {Fase}

#### 49.1.1 Methods

<code>get_nombre(<i>self</i>)</code>
--------------------------------------

<code>nombre(<i>self</i>)</code>
----------------------------------

#### 49.1.2 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> <code>'fichas'</code>
<code>id</code>	<b>Value:</b> <code>Column(Unicode, primary_key=True)</code>
<code>id_usuario</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey('usuarios.id', onupdate="CASC...)</code>
<code>id_rol</code>	<b>Value:</b> <code>Column(Unicode, ForeignKey('roles.id', onupdate="CASCADE...)</code>

*continued on next page*

<sup>100</sup><mailto:alearce07@gmail.com>

<sup>101</sup><mailto:gabrielcaroni@gmail.com>

<sup>102</sup><mailto:rodpar07@gmail.com>

Name	Description
id_proyecto	<b>Value:</b> Column(Unicode, ForeignKey('proyectos.id', onupdate="CAS...
id_fase	<b>Value:</b> Column(Unicode, ForeignKey('fases.id', onupdate="CASCADE...
proyecto	<b>Value:</b> relation("Proyecto", backref=backref('fichas', order_by=...
fase	<b>Value:</b> relation("Fase", backref=backref('fichas', order_by= id,...
usuario	<b>Value:</b> relation("Usuario", backref=backref('roles', order_by= i...
rol	<b>Value:</b> relation("Rol", backref=backref('fichas', order_by= id, ...

## 49.2 Class Rol

declarative\_base()  saip.model.auth.Rol

Clase correspondiente a un Rol del sistema mapeada a la tabla roles de forma declarativa.

### 49.2.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'roles'
<i>Columns</i>	
id	<b>Value:</b> Column(Unicode, primary_key=True)
nombre	<b>Value:</b> Column(Unicode, unique= True, nullable= False)
descripcion	<b>Value:</b> Column(Unicode)
tipo	<b>Value:</b> Column(Unicode, nullable= False)
<i>Relations</i>	
usuarios	<b>Value:</b> []

### 49.3 Class Usuario

```
declarative_base() └─
                     saip.model.auth.Usuario
```

Clase correspondiente a un Usuario del sistema mapeada a la tabla usuarios de forma declarativa.

#### 49.3.1 Methods

<b>validate_password</b> ( <i>self</i> , <i>password</i> )
Valida el password ingresado por el usuario.
<b>Parameters</b>
<i>password</i> : el password (texto claro) proveído por el usuario y que será hashado con la versión almacenada en la base de datos. ( <i>type=Unicode.</i> )
<b>Return Value</b>
Si el password ingresado es válido o no ( <i>type=Bool</i> )

#### Getters and setters

<b>permissions</b> ( <i>self</i> )
<b>by_email_address</b> ( <i>cls</i> , <i>emaila</i> )
<b>by_user_name</b> ( <i>cls</i> , <i>username</i> )

#### 49.3.2 Class Variables

Name	Description
<code>__tablename__</code>	<b>Value:</b> 'usuarios'
<i>Columns</i>	
<code>id</code>	<b>Value:</b> Column(Unicode, primary_key=True)
<code>nombre_usuario</code>	<b>Value:</b> Column(Unicode, nullable=False, unique=True)
<code>nombre</code>	<b>Value:</b> Column(Unicode, nullable=False)

*continued on next page*



Name	Description
apellido	<b>Value:</b> Column(Unicode, nullable= False)
email	<b>Value:</b> Column(Unicode, nullable= False)
telefono	<b>Value:</b> Column(Unicode, nullable= False)
direccion	<b>Value:</b> Column(Unicode, nullable= False)
<i>Getters and setters</i>	
password	<b>Value:</b> synonym('_password', descriptor=property(_get_password, ...

## 49.4 Class Permiso

declarative\_base() —  
saip.model.auth.Permiso

Clase correspondiente a un Permiso del sistema mapeada a la tabla permisos de forma declarativa.

### 49.4.1 Class Variables

Name	Description
__tablename__	<b>Value:</b> 'permisos'
<i>Columns</i>	
id	<b>Value:</b> Column(Unicode, primary_key= True)
nombre	<b>Value:</b> Column(Unicode, nullable= False)
tipo	<b>Value:</b> Column(Unicode, nullable= False)
recurso	<b>Value:</b> Column(Unicode, nullable= False)
descripcion	<b>Value:</b> Column(Unicode)
<i>Relations</i>	
roles	<b>Value:</b> relation('Rol', secondary= rol_permiso, backref= 'permisos')

## 50 Package saip.templates

Templates package for the application.

### 50.1 Variables

Name	Description
--package--	Value: None

## 51 Package `saip.tests`

Unit and functional test suite for SAIP.

### 51.1 Modules

- **models**: Unit test suite for the models of the application.  
(Section 52, p. 147)
  - **test\_proyecto** (Section 53, p. 149)
  - **test\_rol** (Section 54, p. 150)
  - **test\_usuario** (Section 55, p. 151)

### 51.2 Functions

<b>setup_db()</b>
Method used to build a database

<b>teardown_db()</b>
Method used to destroy a database

### 51.3 Variables

Name	Description
<code>url_for</code>	

### 51.4 Class `TestController`

object  **`saip.tests.TestController`**

Base functional test case for the controllers.

The SAIP application instance (“`self.app`”) set up in this test case (and descendants) has authentication disabled, so that developers can test the protected areas independently of the `:mod:'repoze.who'` plugins used initially. This way, authentication can be tested once and separately.

Check `saip.tests.functional.test_authentication` for the `repoze.who` integration tests.

This is the officially supported way to test protected areas with repoze.who-testutil (<http://code.gustavonaro.net/testutil/>).

#### 51.4.1 Methods

<b>setUp</b> ( <i>self</i> )
Method called by nose before running each test

<b>tearDown</b> ( <i>self</i> )
Method called by nose after running each test

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 51.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 51.4.3 Class Variables

Name	Description
<code>application_under_test</code>	<b>Value:</b> <code>'main_without_authn'</code>

## 52 Package saip.tests.models

Unit test suite for the models of the application.

### 52.1 Modules

- **test\_proyecto** (*Section 53, p. 149*)
- **test\_rol** (*Section 54, p. 150*)
- **test\_usuario** (*Section 55, p. 151*)

### 52.2 Class ModelTest

object └─ **saip.tests.models.ModelTest**

Base unit test case for the models.

#### 52.2.1 Methods

<b>setUp</b> ( <i>self</i> )
Prepare model test fixture.

<b>tearDown</b> ( <i>self</i> )
Finish model test fixture.

<b>do_get_dependencies</b> ( <i>self</i> )
Get model test dependencies.
Use this method to pull in other objects that need to be created for this object to be build properly.

<b>test_create_obj</b> ( <i>self</i> )
Model objects can be created

<b>test_query_obj</b> ( <i>self</i> )
Model objects can be queried

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 52.2.2 Properties

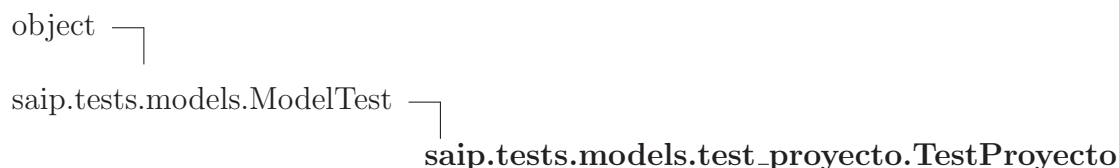
Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 52.2.3 Class Variables

Name	Description
<code>klass</code>	<b>Value:</b> None
<code>attrs</code>	<b>Value:</b> {}

## 53 Module `saip.tests.models.test_proyecto`

### 53.1 Class `TestProyecto`



Unidad de testing del modelo “Proyecto”

#### 53.1.1 Methods

<b><code>test_obj_creacion(self)</code></b>
---

Los datos de proyecto se almacenan correctamente
--

*Inherited from `saip.tests.models.ModelTest`(Section 52.2)*

`do_get_dependencies()`, `setUp()`, `tearDown()`, `test_create_obj()`, `test_query_obj()`

*Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 53.1.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

#### 53.1.3 Class Variables

Name	Description
<code>klass</code>	<b>Value:</b> Proyecto
<code>attrs</code>	<b>Value:</b> dict(id= u"PR1", nombre= u"proyecto", descripcion= u"proy...

## 54 Module saip.tests.models.test\_rol

### 54.1 Class TestRol



Unidad de testing del modelo “Rol”

#### 54.1.1 Methods

<b>test_obj_creacion(<i>self</i>)</b>
---------------------------------------

Los datos de rol se almacenan correctamente
---

*Inherited from saip.tests.models.ModelTest(Section 52.2)*

do\_get\_dependencies(), setUp(), tearDown(), test\_create\_obj(), test\_query\_obj()

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_init\_\_(), \_\_new\_\_(), \_\_reduce\_\_(),  
\_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 54.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

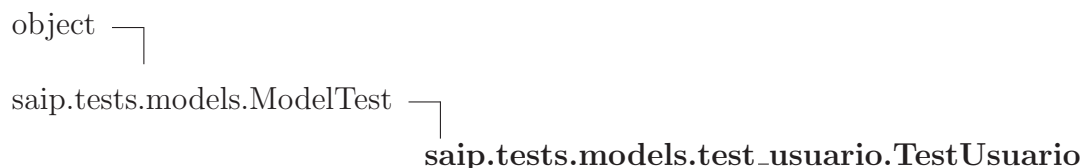
#### 54.1.3 Class Variables

Name	Description
klass	<b>Value:</b> Rol
attrs	<b>Value:</b> dict(id= u"RL1", nombre= u"rol", descripcion= u"rol de pr...



## 55 Module saip.tests.models.test\_usuario

### 55.1 Class TestUsuario



Unidad de testing del modelo “Usuario”

#### 55.1.1 Methods

<b>test_obj_creacion(<i>self</i>)</b>
---------------------------------------

Los datos de usuario se almacenan correctamente
---

*Inherited from saip.tests.models.ModelTest(Section 52.2)*

do\_get\_dependencies(), setUp(), tearDown(), test\_create\_obj(), test\_query\_obj()

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_init\_\_(), \_\_new\_\_(), \_\_reduce\_\_(),  
\_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 55.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

#### 55.1.3 Class Variables

Name	Description
klass	<b>Value:</b> Usuario
attrs	<b>Value:</b> dict(id= u"US1", nombre= u"usuario", nombre_usuario= u"no...

## 56 Package **saip.websetup**

Setup the SAIP application

### 56.1 Modules

- **bootstrap**: Setup the SAIP application  
(Section 57, p. 153)
- **schema**: Setup the SAIP application  
(Section 58, p. 154)

### 56.2 Functions

<b>setup_app</b> ( <i>command</i> , <i>conf</i> , <i>vars</i> )
Place any commands to setup saip here

## 57 Module **saip.websetup.bootstrap**

Setup the SAIP application

### 57.1 Functions

<b>bootstrap</b> ( <i>command</i> , <i>conf</i> , <i>vars</i> )
Place any commands to setup saip here

## 58 Module **saip.websetup.schema**

Setup the SAIP application

### 58.1 Functions

<b>setup_schema</b> ( <i>command</i> , <i>conf</i> , <i>vars</i> )
Place any commands to setup saip here

## Index

- saip (*package*), 2–4
  - saip.config (*package*), 5
    - saip.config.app\_cfg (*module*), 6
    - saip.config.environment (*module*), 7
    - saip.config.middleware (*module*), 8
  - saip.controllers (*package*), 9–10
    - saip.controllers.admin\_controller (*module*), 11
    - saip.controllers.archivo\_controller (*module*), 12–14
    - saip.controllers.archivo\_controller\_listado (*module*), 15–17
    - saip.controllers.borrado\_controller (*module*), 18–20
    - saip.controllers.caracteristica\_controller (*module*), 21–24
    - saip.controllers.desarrollo\_controller (*module*), 24
    - saip.controllers.desarrollo\_fase\_controller (*module*), 25–27
    - saip.controllers.desarrollo\_proyecto\_controller (*module*), 27–29
    - saip.controllers.error (*module*), 30
    - saip.controllers.fase\_controller (*module*), 31–36
    - saip.controllers.fase\_controller\_2 (*module*), 37–39
    - saip.controllers.ficha\_fase\_controller (*module*), 40–43
    - saip.controllers.ficha\_proyecto\_controller (*module*), 44–47
    - saip.controllers.ficha\_sistema\_controller (*module*), 48–51
    - saip.controllers.ficha\_usuario\_controller (*module*), 52–54
    - saip.controllers.gestion\_controller (*module*), 55
    - saip.controllers.gestion\_fase\_controller (*module*), 56–58
    - saip.controllers.gestion\_proyecto\_controller (*module*), 59–61
    - saip.controllers.item\_controller (*module*), 62–67
    - saip.controllers.item\_controller\_listado (*module*), 68–71
    - saip.controllers.linea\_base\_controller (*module*), 72–75
    - saip.controllers.proyecto\_controller (*module*), 76–81
    - saip.controllers.proyecto\_controller\_2 (*module*), 82–84
    - saip.controllers.relacion\_controller (*module*), 85–88
    - saip.controllers.relacion\_controller\_listado (*module*), 89–91
    - saip.controllers.revision\_controller (*module*), 92–94
    - saip.controllers.rol\_controller (*module*), 95–99
    - saip.controllers.root (*module*), 100–101
    - saip.controllers.secure (*module*), 102
    - saip.controllers.template (*module*), 103
    - saip.controllers.tipo\_item\_controller (*module*), 104–108
    - saip.controllers.tipo\_item\_controller\_nuevo (*module*), 109–111
    - saip.controllers.usuario\_controller (*module*), 112–116
    - saip.controllers.version\_controller (*module*), 117–120
  - saip.lib (*package*), 121
    - saip.lib.app\_globals (*module*), 122
    - saip.lib.auth (*module*), 123–124
    - saip.lib.base (*module*), 125
    - saip.lib.func (*module*), 126–130
    - saip.lib.helpers (*module*), 131
  - saip.model (*package*), 132
    - saip.model.app (*module*), 133–139
    - saip.model.auth (*module*), 140–143
    - saip.model.init\_model (*function*), 132
  - saip.templates (*package*), 144
  - saip.tests (*package*), 145–146
    - saip.tests.models (*package*), 147–148
    - saip.tests.setup\_db (*function*), 145

---

- saip.tests.teardown\_db (*function*), 145
- saip.tests.TestController (*class*), 145–146
- saip.websetup (*package*), 152
  - saip.websetup.bootstrap (*module*), 153
  - saip.websetup.schema (*module*), 154
  - saip.websetup.setup\_app (*function*), 152
- saip.websetup.bootstrap.bootstrap (*function*), 153