

Documentacion SIIGS/TES



Contenido

<u>Paquete Libreria Clases</u>	1
<u>Clase Graph</u>	1
<u>Constructor</u> <u>construct</u>	1
<u>Método graph_init</u>	1
<u>Método map</u>	2
<u>Clase Obtenercurp</u>	2
<u>Var \$estados</u>	3
<u>Constructor</u> <u>construct</u>	3
<u>Método calculacurp</u>	4
<u>Método calcular_curp</u>	4
<u>Método curp</u>	5
<u>Clase Tree</u>	5
<u>Constructor</u> <u>construct</u>	5
<u>Método create</u>	6
<u>Paquete SIIGS Clases</u>	7
<u>Clase Accion</u>	7
<u>Constructor</u> <u>construct</u>	7
<u>Método delete</u>	7
<u>Método index</u>	8
<u>Método insert</u>	8
<u>Método update</u>	8
<u>Método view</u>	9
<u>Clase Ayuda</u>	9
<u>Constructor</u> <u>construct</u>	9
<u>Método index</u>	9
<u>Clase Bitacora</u>	10
<u>Constructor</u> <u>construct</u>	10
<u>Método index</u>	10
<u>Método validateExistUsuario</u>	11
<u>Método view</u>	11
<u>Clase Catalogo</u>	12
<u>Constructor</u> <u>construct</u>	12
<u>Método checkpk</u>	12
<u>Método checkTypeData</u>	12
<u>Método delete</u>	13
<u>Método index</u>	13
<u>Método insert</u>	13
<u>Método load</u>	14
<u>Método loadupdate</u>	14
<u>Método update</u>	14
<u>Método view</u>	15
<u>Método array_unique_recursive</u>	15
<u>Clase CatalogoCsv</u>	16

<u>Constructor</u>	<u>construct</u>	16
<u>Método ActivaEnCatalogo</u>		16
<u>Método checkpk</u>		17
<u>Método createTableAgeb</u>		17
<u>Método createTableGeo</u>		17
<u>Método createTableHemoGlobina</u>		17
<u>Método createTablePob</u>		17
<u>Método index</u>		18
<u>Método loadupdate</u>		18
<u>Método update</u>		18
<u>Método view</u>		19
<u>Método array unique recursive</u>		19
<u>Clase Catalogo x raiz</u>		20
<u>Constructor</u>	<u>construct</u>	20
<u>Método check</u>		20
<u>Método delete</u>		21
<u>Método insert</u>		21
<u>Método view</u>		21
<u>Clase Cie10</u>		22
<u>Constructor</u>	<u>construct</u>	22
<u>Método ActivaEnCatalogo</u>		22
<u>Método AgregaEnCatalogo</u>		23
<u>Método index</u>		23
<u>Método insert</u>		23
<u>Método load</u>		24
<u>Método update</u>		24
<u>Método view</u>		25
<u>Método array unique recursive</u>		25
<u>Clase Controlador</u>		25
<u>Constructor</u>	<u>construct</u>	26
<u>Método accion</u>		26
<u>Método delete</u>		26
<u>Método getGroupPermissions</u>		27
<u>Método help</u>		27
<u>Método index</u>		27
<u>Método insert</u>		28
<u>Método update</u>		28
<u>Método view</u>		28
<u>Clase Entorno</u>		29
<u>Constructor</u>	<u>construct</u>	29
<u>Método delete</u>		29
<u>Método index</u>		30
<u>Método insert</u>		30
<u>Método update</u>		30
<u>Método view</u>		31
<u>Método ExistEntorno</u>		31
<u>Método ExistEntornoUpdate</u>		31
<u>Clase Errorlog</u>		32
<u>Constructor</u>	<u>construct</u>	32

Método index	32
Método view	33
Clase Grupo	33
Constructor construct	33
Método delete	34
Método index	34
Método insert	34
Método update	34
Método view	35
Método ifGroupExists	35
Clase Menu	36
Constructor construct	36
Método delete	36
Método index	36
Método insert	37
Método update	37
Método view	38
Clase Permiso	38
Constructor construct	38
Método index	38
Clase Raiz	39
Constructor construct	39
Método createasu	39
Método delete	40
Método getChildrenFromLevel	40
Método getDataKeyValue	41
Método getDataTreeFromId	41
Método getTreeBlock	41
Método index	42
Método iniciarasu	42
Método insert	42
Método update	43
Método updateasu	43
Método view	43
Clase ReglaVacuna	44
Constructor construct	44
Método delete	44
Método index	45
Método insert	45
Método update	45
Método view	46
Clase Usuario	46
Var \$mas	46
Constructor construct	47
Método automatic access	47
Método cerrar etab	47
Método delete	47
Método form init	47
Método getActivesByGroup	47

Método get_galleta	48
Método get_token	48
Método index	48
Método insert	48
Método load_update	49
Método login	49
Método logout	49
Método remember	49
Método reset	49
Método send_mail	50
Método token	50
Método update	50
Método update_info	51
Método view	51
Método ifUserExists	51
Clase Accion_model	52
Constructor construct	52
Método delete	52
Método getAll	52
Método getByld	52
Método getDescripcion	53
Método getId	53
Método getMetodo	53
Método getMsgError	53
Método getNombre	54
Método getNumRows	54
Método insert	54
Método setDescripcion	54
Método setId	55
Método setMetodo	55
Método setNombre	55
Método setOffset	55
Método setRows	56
Método update	56
Clase Ageb_model	56
Constructor construct	57
Método getMsgError	57
Método process	57
Método searchageb	57
Método searchUM	58
Clase ArbolSegmentacion_model	58
Constructor construct	58
Método convertType	59
Método getByld	59
Método getChildrenFromId	59
Método getChildrenFromLevel	60
Método getCluesFromId	60
Método getDataKeyValue	61
Método getDescripcionByld	61

Método getListChildrenLevel	62
Método getMsgError	62
Método getTree	62
Método getTreeBlock	63
Método getTreeBlockData	63
Método getUMParentsByld	64
Método addSelectedItems	64
Método addSelectedItems	64
Clase Bitacora_model	65
Constructor construct	65
Método addFilter	65
Método delete	66
Método deleteByFilter	66
Método getAll	66
Método getByld	67
Método getFecha_hora	67
Método getld	67
Método getld_controlador_accion	67
Método getld_usuario	67
Método getMsgError	67
Método getNumRows	68
Método getParametros	68
Método insert	68
Método resetFilter	69
Método setFecha_hora	69
Método setld	69
Método setld_accion	69
Método setld_controlador	70
Método setld_controlador_accion	70
Método setld_usuario	70
Método setParametros	71
Método update	71
Clase CatalogoCsv_model	71
Constructor construct	72
Método activaEnCatalogo	72
Método checkPk	72
Método getAll	73
Método getAllData	73
Método getByName	73
Método getCampos	73
Método getld	74
Método getLLave	74
Método getMsgError	74
Método getNombre	74
Método getNumRows	75
Método setCampos	75
Método setld	75
Método setLlave	75
Método setNombre	76

Método setOffset	76
Método setRows	76
Clase Catalogo_model	77
Constructor construct	77
Método checkPk	77
Método checkTypeData	77
Método delete	78
Método getAll	78
Método getAllData	78
Método getByName	79
Método getCampos	79
Método getId	79
Método getLLave	79
Método getMsgError	80
Método getNombre	80
Método getNumRows	80
Método insert	80
Método setCampos	81
Método setId	81
Método setLlave	81
Método setNombre	82
Método setOffset	82
Método setRows	82
Método updateComentario	83
Clase Catalogo_x_raiz_model	83
Constructor construct	83
Método check	83
Método delete	84
Método getByArbol	84
Método getById	84
Método getByNivel	85
Método getColumnaDescripcion	85
Método getColumnaLLave	85
Método getGrado	85
Método getId	86
Método getIdRaiz	86
Método getMsgError	86
Método getNivel	86
Método getRelacionHijo	87
Método getRelacionPadre	87
Método getRelations	87
Método getTablaCatalogo	87
Método insert	88
Método setColumnaDescripcion	88
Método setColumnaLlave	88
Método setGrado	88
Método setId	89
Método setIdRaiz	89
Método setRelacionHijo	89

Método setRelacionPadre	89
Método setTablaCatalogo	90
Clase Cie10_model	90
Constructor construct	90
Método activaEnCatalogo	91
Método agregaEnCatalogo	91
Método checkPk	91
Método getAll	92
Método getAllData	92
Método getByld	92
Método getCatalogoByName	93
Método getData	93
Método getDescripcion	93
Método getId	93
Método getMsgError	94
Método getNumRows	94
Método setDescripcion	94
Método setId	95
Método setOffset	95
Método setRows	95
Método update	95
Clase ControladorAccion_model	96
Constructor construct	96
Método getByld	96
Método getId	97
Método getIdByPath	97
Método getMsgError	97
Método setHelp	98
Clase Controlador_model	98
Constructor construct	98
Método accionesUpdate	99
Método delete	99
Método getAccion	99
Método getAcciones	99
Método getAll	100
Método getByEntorno	100
Método getByld	100
Método getClase	101
Método getDescripcion	101
Método getId	101
Método getIdEntorno	101
Método getMsgError	101
Método getNombre	102
Método getNumRows	102
Método getPermisos	102
Método insert	102
Método setAccion	103
Método setClase	103
Método setDescripcion	103

Método setId	104
Método setIdEntorno	104
Método setNombre	104
Método setOffset	104
Método setRows	105
Método update	105
Clase Entorno_model	105
Constructor construct	106
Método delete	106
Método getAll	106
Método getById	106
Método getName	107
Método getDescripcion	107
Método getDirectorio	107
Método getHostname	107
Método getId	107
Método getInfo	108
Método getIp	108
Método getMsgError	108
Método getNombre	108
Método getPermissionsByGroup	108
Método insert	109
Método setDescripcion	109
Método setDirectorio	109
Método setHostname	110
Método setId	110
Método setIp	110
Método setNombre	110
Método update	111
Clase Errorlog_model	111
Constructor construct	111
Método addFilter	112
Método getAll	112
Método getById	112
Método getDescripcion	113
Método getFecha_hora	113
Método getId	113
Método getId_controlador_accion	113
Método getId_usuario	113
Método getNumRows	113
Método insert	114
Método resetFilter	114
Método save	114
Método setDescripcion	115
Método setId	115
Método setId_accion	115
Método setId_controlador	116
Método setId_controlador_accion	116
Método setId_usuario	116

<u>Clase Georeferencia_model</u>	117
<u>Constructor_construct</u>	117
<u>Método getMsgError</u>	117
<u>Método process</u>	117
<u>Clase Grupo_model</u>	118
<u>Constructor_construct</u>	118
<u>Método delete</u>	118
<u>Método getAll</u>	118
<u>Método getByld</u>	119
<u>Método getByName</u>	119
<u>Método getDescripcion</u>	120
<u>Método getEntornosByld</u>	120
<u>Método getld</u>	120
<u>Método getMsgError</u>	120
<u>Método getNombre</u>	121
<u>Método getNumRows</u>	121
<u>Método insert</u>	121
<u>Método setDescription</u>	121
<u>Método setld</u>	122
<u>Método setNombre</u>	122
<u>Método update</u>	122
<u>Clase Hemoglobina_model</u>	122
<u>Constructor_construct</u>	123
<u>Método getMsgError</u>	123
<u>Método process</u>	123
<u>Clase Menu_model</u>	124
<u>Constructor_construct</u>	124
<u>Método addFilter</u>	124
<u>Método delete</u>	124
<u>Método deleteByFilter</u>	125
<u>Método getAll</u>	125
<u>Método getAtributo</u>	125
<u>Método getByld</u>	126
<u>Método getByPadre</u>	126
<u>Método getld</u>	126
<u>Método getld_controlador</u>	126
<u>Método getld_padre</u>	127
<u>Método getld_raiz</u>	127
<u>Método getMsgError</u>	127
<u>Método getNombre</u>	127
<u>Método getNumRows</u>	127
<u>Método getRuta</u>	128
<u>Método hasChild</u>	128
<u>Método insert</u>	128
<u>Método resetFilter</u>	128
<u>Método setAtributo</u>	129
<u>Método setld</u>	129
<u>Método setld_controlador</u>	129
<u>Método setld_padre</u>	129

Método setId raiz	130
Método setNombre	130
Método setRuta	130
Método update	131
Clase Permiso_model	131
Constructor construct	131
Método deletePermissions	131
Método getFecha	132
Método getId	132
Método getIdControladorAccion	132
Método getIdGrupo	132
Método getMsgError	132
Método getPermission	133
Método insertBatch	133
Método setFecha	133
Método setId	134
Método setIdControladorAccion	134
Método setIdGrupo	134
Clase Poblacion_model	135
Constructor construct	135
Método getMsgError	135
Método process	135
Clase Raiz_model	136
Constructor construct	136
Método delete	136
Método ExistInArbol	136
Método getAll	137
Método getById	137
Método getDescripcion	137
Método getId	138
Método getMsgError	138
Método insert	138
Método setDescripcion	138
Método setId	139
Método update	139
Clase ReglaVacuna_model	139
Constructor construct	140
Método delete	140
Método getAlergias	140
Método getAll	140
Método getById	140
Método getDiaFinNacido	141
Método getDiaFinPrevia	141
Método getDiaInicioNacido	141
Método getDiaInicioPrevia	141
Método getDosis	141
Método getEsqComp	141
Método getForzarAplicacion	142
Método getId	142

Método getIdVacuna	142
Método getIdVacunaPrevia	142
Método getIdViaVacuna	142
Método getMsgError	142
Método getObservacionRegion	143
Método getOrdenEsqComp	143
Método getRegion	143
Método insert	143
Método setAlergias	143
Método setDiaFinNacido	144
Método setDiaFinPrevia	144
Método setDiaInicioNacido	144
Método setDiaInicioPrevia	145
Método setDosis	145
Método setEsqComp	145
Método setForzarAplicacion	145
Método setId	146
Método setIdVacuna	146
Método setIdVacunaPrevia	146
Método setIdViaVacuna	147
Método setObservacionRegion	147
Método setOrdenEsqComp	147
Método setRegion	147
Método update	148
Clase Usuario_model	148
Constructor construct	148
Método authenticate	148
Método checkCredentials	149
Método check_data	149
Método check_token	150
Método delete	150
Método getActivesByGroup	150
Método getActivo	150
Método getApellidoMaterno	151
Método getApellidoPaterno	151
Método getById	151
Método getByUsername	151
Método getClave	152
Método getCorreo	152
Método getgrupo	152
Método getId	152
Método getIdGrupo	152
Método getMsgError	152
Método getNombre	153
Método getNombreUsuario	153
Método getNumRows	153
Método getOnlyActives	153
Método getuser	154
Método get_grupo_ entorno	154

Método get permiso entorno	155
Método get usuario entorno	155
Método insert	155
Método setActivo	155
Método setApellidoMaterno	156
Método setApellidoPaterno	156
Método setClave	156
Método setCorreo	156
Método setId	157
Método setIdGrupo	157
Método setName	157
Método setNameUsuario	158
Método update	158
Método update pass	158
Método update user	158
Paquete default Clases	160
Clase Index	160
Constructor construct	160
Método index	160
Paquete TES Clases	161
Clase Enrolamiento	161
Constructor construct	161
Método addForm	161
Método autocomplete	161
Método brothers search	162
Método brother found	162
Método catalog check	162
Método catalog select	163
Método categoriacie10 select	163
Método checar session	163
Método cie10 select	164
Método comparar view	164
Método data tutor	164
Método file to card	165
Método ifCarpExists	165
Método ifCarpTExists	166
Método index	166
Método insert	166
Método paciente similar	167
Método print card	167
Método searchgeb	167
Método searchum	168
Método tratamiento select	168
Método update	169
Método update card	169
Método vacunacion	169
Método validarForm	170
Método validarisum	170

Método validate_card	170
Método view	171
Clase Notificacion	171
Constructor construct	172
Método delete	172
Método index	172
Método insert	172
Método update	173
Método view	173
Clase Reporteador	173
Constructor construct	174
Método index	174
Método view	174
Clase Reporte sincronizacion	175
Constructor construct	175
Método index	175
Método lote	175
Método lote view	175
Método view	176
Clase Semana nacional	176
Constructor construct	177
Método delete	177
Método getAll	177
Método index	177
Método insert	178
Método update	178
Método view	178
Clase Servicios	179
Constructor construct	179
Método actualiza_estado_tableta	179
Método catalogos_relevantes	180
Método esquema_incompleto	180
Método is_step_0	180
Método is_step_1	181
Método is_step_2	181
Método is_step_3	182
Método is_step_4	182
Método prueba2	183
Método ss_step_5	183
Método ss_step_6	183
Método Synchronization	184
Clase Tableta	184
Constructor construct	185
Método delete	185
Método index	185
Método insert	186
Método setUM	186
Método update	186
Método uploadFile	186

Método view	187
Método validateMac	187
Clase Usuario tableta	187
Constructor construct	188
Método delete	188
Método index	188
Método insert	189
Clase Enrolamiento model	189
Constructor construct	189
Método autocomplete tutor	190
Método cns insert	190
Método cns update	190
Método cns update visita	191
Método data tutor	191
Método entorno x persona	191
Método getaccion nutricional	192
Método getafiliacion	192
Método getAfiliaciones	192
Método getageb	192
Método getAlergia	193
Método getalergias	193
Método getaltura	193
Método getByCurp	193
Método getById	194
Método getcalle	194
Método getCategoryCIE10	194
Método getcelular	194
Método getcelularT	195
Método getCIE10	195
Método getcodigo barras	195
Método getcolonia	195
Método getcompania	195
Método getcompaniaT	195
Método getconsulta	196
Método getControlConsultas	196
Método getcp	196
Método getcurp	196
Método getcurpT	196
Método getestimulacion capacitado	197
Método getestimulacion fecha	197
Método getfaccion nutricional	197
Método getfconsulta	197
Método getfechacivil	197
Método getfecha peri cefa	197
Método getfnacimiento	198
Método getfnutricion	198
Método getfolio	198
Método getfvacuna	198
Método gethemoglobina	198

<u>Método getld</u>	198
<u>Método getidtutor</u>	199
<u>Método getListEnrolamiento</u>	199
<u>Método getlnacimiento</u>	199
<u>Método getlocalidad</u>	199
<u>Método getlugarcivil</u>	199
<u>Método getmanzana</u>	200
<u>Método getmaterno</u>	200
<u>Método getmaternoT</u>	200
<u>Método getMsgError</u>	200
<u>Método getnacionalidad</u>	200
<u>Método getnombre</u>	201
<u>Método getnombreT</u>	201
<u>Método getnumero</u>	201
<u>Método getNumRows</u>	201
<u>Método getparto</u>	201
<u>Método getpaterno</u>	201
<u>Método getpaternoT</u>	202
<u>Método getperi_cefa</u>	202
<u>Método getpeso</u>	202
<u>Método getprecurp</u>	202
<u>Método getreferencia</u>	202
<u>Método getRegistro_civil</u>	202
<u>Método getsales_cantidad</u>	203
<u>Método getsales_fecha</u>	203
<u>Método getsangre</u>	203
<u>Método getsector</u>	203
<u>Método getsexo</u>	203
<u>Método getsexoT</u>	204
<u>Método gettalla</u>	204
<u>Método gettamiz</u>	204
<u>Método gettbeneficiario</u>	204
<u>Método gettconsulta</u>	204
<u>Método gettelefono</u>	204
<u>Método gettelefonoT</u>	204
<u>Método getumt</u>	205
<u>Método getvacuna</u>	205
<u>Método get_catalog</u>	205
<u>Método get_catalog2</u>	205
<u>Método get_catalog_count</u>	206
<u>Método get_catalog_relevante</u>	206
<u>Método get_catalog_tratamiento</u>	207
<u>Método get_catalog_view</u>	207
<u>Método get_cns_cat_persona</u>	207
<u>Método get_cns_cat_persona_count</u>	208
<u>Método get_cns_persona</u>	208
<u>Método get_control_nutricional</u>	209
<u>Método get_datos_grafica</u>	209
<u>Método get_estimulacion</u>	209

<u>Método get_notificacion</u>	210
<u>Método get_pacientes</u>	210
<u>Método get_peri_cefa</u>	210
<u>Método get_persona_x_tutor</u>	211
<u>Método get_sales</u>	211
<u>Método get_transaction_relevante</u>	211
<u>Método get_version</u>	212
<u>Método insert</u>	212
<u>Método setaccion_nutricional</u>	212
<u>Método setafiliacion</u>	212
<u>Método setageb</u>	213
<u>Método setalergias</u>	213
<u>Método setaltura</u>	213
<u>Método setcalle</u>	213
<u>Método setcelular</u>	214
<u>Método setcelularT</u>	214
<u>Método setcodigo_barras</u>	214
<u>Método setcolonia</u>	215
<u>Método setcompania</u>	215
<u>Método setcompaniaT</u>	215
<u>Método setconsulta</u>	215
<u>Método setcp</u>	216
<u>Método setcurp</u>	216
<u>Método setcurpT</u>	216
<u>Método setestimulacion_capacitado</u>	217
<u>Método setestimulacion_fecha</u>	217
<u>Método setfaccion_nutricional</u>	217
<u>Método setfconsulta</u>	217
<u>Método setfechacivil</u>	218
<u>Método setfecha_peri_cefa</u>	218
<u>Método setfnacimiento</u>	218
<u>Método setfnutricion</u>	218
<u>Método setfvacuna</u>	219
<u>Método sethemoglobina</u>	219
<u>Método setld</u>	219
<u>Método setidtutor</u>	220
<u>Método setlnacimiento</u>	220
<u>Método setlocalidad</u>	220
<u>Método setlugarcivil</u>	220
<u>Método setmanzana</u>	221
<u>Método setmaterno</u>	221
<u>Método setmaternoT</u>	221
<u>Método setnacionalidad</u>	222
<u>Método setnombre</u>	222
<u>Método setnombreT</u>	222
<u>Método setnumero</u>	222
<u>Método setparto</u>	223
<u>Método setpaterno</u>	223
<u>Método setpaternoT</u>	223

<u>Método setperi_cefa</u>	224
<u>Método setpeso</u>	224
<u>Método setprecurp</u>	224
<u>Método setreferencia</u>	224
<u>Método setsales_cantidad</u>	225
<u>Método setsales_fecha</u>	225
<u>Método setsangre</u>	225
<u>Método setsector</u>	225
<u>Método setsexo</u>	226
<u>Método setsexoT</u>	226
<u>Método settalla</u>	226
<u>Método settamiz</u>	227
<u>Método settbeneficiario</u>	227
<u>Método settconsulta</u>	227
<u>Método settelefono</u>	227
<u>Método settelefonoT</u>	228
<u>Método setumt</u>	228
<u>Método setvacuna</u>	228
<u>Método tes_pendientes_tarjeta_delete</u>	229
<u>Método update_accion</u>	229
<u>Método update_alergia</u>	229
<u>Método update_basico</u>	229
<u>Método update_beneficiario</u>	229
<u>Método update_consulta</u>	230
<u>Método update_direccion</u>	230
<u>Método update_estimulacion</u>	230
<u>Método update_nutricion</u>	230
<u>Método update_peri_cefa</u>	231
<u>Método update_regcivil</u>	231
<u>Método update_sales</u>	231
<u>Método update_status_tableta</u>	231
<u>Método update_tutor</u>	232
<u>Método update_umt</u>	232
<u>Método update_vacuna</u>	232
<u>Método valid_card</u>	232
<u>Clase Estado_tableta_model</u>	233
<u>Constructor_construct</u>	233
<u>Método getAll</u>	233
<u>Método getById</u>	233
<u>Método getDescripcion</u>	234
<u>Método getId</u>	234
<u>Método getMsgError</u>	234
<u>Método setDescripcion</u>	234
<u>Método setId</u>	235
<u>Clase Notificacion_model</u>	235
<u>Constructor_construct</u>	235
<u>Método addFilter</u>	235
<u>Método delete</u>	236
<u>Método getAll</u>	236

Método getById	236
Método getContenido	237
Método getFechaFin	237
Método getFechaInicio	237
Método getId	237
Método getIdsTabletas	237
Método getMsgError	238
Método getNumRows	238
Método getTitulo	238
Método insert	238
Método setContenido	239
Método setFechaFin	239
Método setFechaInicio	239
Método setId	239
Método setIdsTabletas	240
Método setTitulo	240
Método update	240
Clase Reporteador_model	241
Constructor construct	241
Método getCensoNominal	241
Método getCoberturaBiologicoListado	241
Método getConcentradoActividades	242
Método getEsquemasIncompletos	242
Método getGrupoVacunas	242
Método getMsgError	242
Método getSeguimientoRV1RV5	243
Método getVacunas	243
Método getVacunasByGrupo	243
Método object to array	243
Clase Reporte_censo_nominal	244
Var \$apellido_materno	244
Var \$apellido_paterno	244
Var \$curp	244
Var \$domicilio	244
Var \$fecha_nacimiento	245
Var \$nombre	245
Var \$parto_multiple	245
Var \$sexo	245
Var \$vacunas	245
Constructor construct	246
Clase Reporte_sincronizacion_model	246
Método getCount	246
Método getListado	247
Método getMsgError	247
Método get_version	247
Clase Semana_nacional_model	248
Constructor construct	248
Método delete	248
Método getAll	248

Método getByld	249
Método getDescripcion	249
Método getFecha_fin	249
Método getFecha_inicio	249
Método getId	250
Método getMsgError	250
Método getNumRows	250
Método insert	250
Método setDescripcion	250
Método setFecha_fin	251
Método setFecha_inicio	251
Método update	251
Clase Tableta_model	252
Constructor_construct	252
Método delete	252
Método getAll	253
Método getByld	253
Método getByMac	253
Método getId	254
Método getIdVersion	254
Método getId_asu_um	254
Método getId_tes_estado_tableta	254
Método getId_tipo_censo	254
Método getMac	254
Método getMsgError	255
Método getNumRows	255
Método getPeriodo_esq_inc	255
Método getUltima_actualizacion	255
Método getUsers_asignados	255
Método insert	256
Método setId	256
Método setIdVersion	256
Método setId_asu_um	256
Método setId_tes_estado_tableta	257
Método setId_tipo_censo	257
Método setMac	257
Método setPeriodo_esq_inc	258
Método setUltima_actualizacion	258
Método setUsuarios_asignados	258
Método update	258
Clase Tipo_censo_model	259
Constructor_construct	259
Método getAll	259
Método getByld	259
Método getDescripcion	260
Método getId	260
Método getMsgError	260
Método setDescripcion	261
Método setId	261

Clase Usuario tableta model	261
Constructor construct	261
Método delete	262
Método getMsgError	262
Método getTableta	262
Método getTabletasByUsuario	263
Método getUsuario	263
Método getUsuariosByTableta	263
Método insert	263
Método setTableta	264
Método setUsuario	264
Appendices	265
Appendix A - Class Trees	266
SIIGS	266
default	274
TES	274
Libreria	278

Paquete Libreria Clases

Clase Graph *[line 11]*

Controlador Objetos

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Graph::__construct() *[line 13]*

- **Access** public

void función Graph::graph_init(\$title, \$titulo, \$array, \$label, [\$grafica = ""], [\$nacimiento = ""])
[line 36]

Parámetros de la función:

- *string* **\$title** Titulo de la pagina en el navegador
- *string* **\$titulo** titulo o nombre a mostrar en la vista al crear el grafico
- *array* **\$array** datos que se graficaran ver ejemplo
- *array* **\$label** datos con las etiquetas que se muestran en cada grafica
- *array* **\$grafica** tipo de grafiva puede ser :time, basic, axis, bars, bars-h, stacked, horizontal ó pie
- *string* **\$nacimiento** si se necesita mostrar fechas enviar la fecha inicial

Crea una grafica en el lugar que se llame

- **Access** public

void función Graph::map([\$lugar = "Chiapas"], [\$zoom = 6], [\$rewrite = 0], [\$datos = ""]) [*line 93*]

Parámetros de la función:

- *string* **\$lugar** Especifica el lugar donde se centra el mapa
- *int* **\$zoom** Especifica el zoom de acercamiento en el mapa
- *boolean* **\$rewrite** Si se desea que sea una pagina o estar enbebida en otra 0=enbebido 1=pagina
- *array* **\$datos** datos a mostrar en el mapa

crea un objeto mapa con la ayuda de la api de google

- **Access** public

Clase Obtenercurp

[*line 11*]

Controlador Objeto

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Obtenercurp::\$estados

```

$estados = array(
    array(
        "AGUASCALIENTES"=>"AS",
        "BAJA CALIFORNIA NTE"=>"BC",
        "BAJA CALIFORNIA NORTE"=>"BC",
        "BAJA CALIFORNIA"=>"BC",
        "BAJA CALIFORNIA SUR"=>"BS",
        "CAMPECHE"=>"CC",
        "COAHUILA"=>"CL",
        "COLIMA"=>"CM",
        "CHIAPAS"=>"CS",
        "CHIHUAHUA"=>"CH",
        "DISTRITO FEDERAL"=>"DF",
        "DURANGO"=>"DG",
        "GUANAJUATO"=>"GT",
        "GUERRERO"=>"GR",
        "HIDALGO"=>"HG",
        "JALISCO"=>"JC",
        "MEXICO"=>"MC",
        "MICHOACAN"=>"MN",
        "MORELOS"=>"MS",
        "NAYARIT"=>"NT",
        "NUEVO LEON"=>"NL",
        "OAXACA"=>"OC",
        "PUEBLA"=>"PL",
        "QUERETARO"=>"QT",
        "QUINTANA ROO"=>"QR",
        "SAN LUIS POTOSI"=>"SP",
        "SINALOA"=>"SL",
        "SONORA"=>"SR",
        "TABASCO"=>"TC",
        "TAMAULIPAS"=>"TS",
        "TLAXCALA"=>"TL",
        "VERACRUZ"=>"VZ",
        "ZACATECAS"=>"ZS",
        "EXTERIOR MEXICANO"=>"SM",
        "NACIDO EN EL EXTRANJERO"=>"NE"
    )
); [line 17]

```

Arreglo con los estados y sus abreviaturas, sirven para el cálculo de la CURP

- **Access** public

Constructor *void* función `Obtenercurp::__construct()` [line 58]

- **Access** public

echo función Obtenercurp::calculacurp(\$paterno, \$materno, \$nombre, \$dia, \$mes, \$year, \$sexo, \$estado, [\$regresar = ""]) [line 280]

Parámetros de la función:

- *string* **\$paterno** Apellido paterno de la persona
- *string* **\$materno** Apellido materno
- *string* **\$nombre** Nombre o nombres
- *int* **\$dia** Día de nacimiento
- *int* **\$mes** Mes de nacimiento
- *int* **\$year** Año de nacimiento
- *string* **\$sexo** Sexo
- *string* **\$estado** Lugar de nacimiento
- *string* **\$regresar** Tipo de retorno =1 return array !=1 json

Calcula la curp y el rfc con los datos proporcionados

- **Access** public

echo función Obtenercurp::calcular_curp(\$paterno, \$materno, \$nombre, \$dia, \$mes, \$year, \$sexo, \$estado, [\$regresar = ""]) [line 207]

Parámetros de la función:

- *string* **\$paterno** Apellido paterno de la persona
- *string* **\$materno** Apellido materno
- *string* **\$nombre** Nombre o nombres
- *int* **\$dia** Día de nacimiento
- *int* **\$mes** Mes de nacimiento
- *int* **\$year** Año de nacimiento
- *string* **\$sexo** Sexo
- *string* **\$estado** Lugar de nacimiento
- *string* **\$regresar** Tipo de retorno =1 return array !=1 json

Calcula la curp y el rfc con los datos proporcionados

- **Access** public

echo función Obtenercurp::curp(\$paterno, \$materno, \$nombre, \$dia, \$mes, \$year, \$sexo, \$estado, [\$regresar =
""])

Parámetros de la función:

- **string \$paterno** Apellido paterno de la persona
- **string \$materno** Apellido materno
- **string \$nombre** Nombre o nombres
- **int \$dia** Día de nacimiento
- **int \$mes** Mes de nacimiento
- **int \$year** Año de nacimiento
- **string \$sexo** Sexo
- **string \$estado** Lugar de nacimiento
- **string \$regresar** Tipo de retorno =1 return array !=1 json

Consulta si la curp existe en la base de datos de la condusef

- **Access** public

Clase Tree

[line 11]

Controlador Objeto

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Tree::__construct() [line 13]

- **Access public**

void función Tree::create(\$title, \$titulo, \$seleccion, \$tipo, \$menu, \$id, \$text, [\$idarbol = 1], [\$nivel = 1], [\$omitidos = array(NULL)], [\$seleccionable = ""]) [line 42]

Parámetros de la función:

- *string* **\$title** Titulo de la pagina en el navegador
- *string* **\$titulo** titulo o nombre a mostrar en la vista al crear el arbol
- *int* **\$seleccion** tipo de seleccion 1=select. 2=multiselect. 3=multiselect Parcial (Marcan los padres del hijo seleccionado)
- *string* **\$tipo** tipo de control radio o check
- *boolean* **\$menu** si se desea mostrar el menu
- *string* **\$id** id del campo oculto donde se guarda el id del elemento seleccionado
- *string* **\$text** id del campo donde se muestra la descripcion del elemento seleccionado
- *string* **\$idarbol** id del arbol donde comenzara la creacion
- *string* **\$nivel** nivel en el que se empezara a mostrar informacion
- *string* **\$omitidos** nodos que no se deben mostrar en la vista al crear el arbol
- *string* **\$seleccionable** determina si un nodo se puede o no seleccionar

Crea el arbol y lo muestra en la view

- **Access public**

Paquete SIIGS Clases

Clase Accion

[line 10]

Controlador Accion

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Accion::__construct() *[line 12]*

- **Access** public

void función Accion::delete(\$id) *[line 246]*

Parámetros de la función:

- *int* **\$id** Este parámetro no puede ser nulo

Acción para eliminar una acción, recibe el id de la acción a eliminar

- **Access** public

void función Accion::index([\$pag = 0]) [line 35]

Parámetros de la función:

- ***int \$pag*** Numero de registro para el paginador

Acción por default del controlador, carga la lista
de acciones disponibles y una lista de opciones

- **Access public**

void función Accion::insert() [line 112]

Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente

- **Access public**

void función Accion::update(\$id) [line 170]

Parámetros de la función:

- ***int \$id*** Este parámetro no puede ser nulo

Acción para preparar la actualización de una acción ya existente,
recibe un ID para obtener los valores de esa acción y mostrarlos en la vista update ,
realiza la validación del formulario del lado del cliente

- **Access** public

void función Accion::view(\$id) *[line 83]*

Parámetros de la función:

- *int* **\$id** Este parametro no puede ser nulo

Acción para visualizar información de una acción específica, obtiene el objeto acción por medio del id proporcionado.

- **Access** public

Clase Ayuda

[line 11]

Controlador Ayuda

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Ayuda::__construct() *[line 13]*

- **Access** public

void función Ayuda::index([\$id_controlador_accion = null]) *[line 25]*

Parámetros de la función:

- **int \$id_controlador_accion** Id del controlador accion, es opcional

Función que renderiza el contenido de la ayuda dependiendo de la sección donde se encuentre

- **Access public**

Clase Bitacora

[line 11]

Controlador Bitacora

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Bitacora::__construct() [line 13]

- **Access public**

void función Bitacora::index([\$pag = 0]) [line 35]

Parámetros de la función:

- **int \$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la bitacora, con su correspondiente paginación

permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico

- **Access public**

void función Bitacora::validateExistUsuario(\$id_usuario) [line 367]

Parámetros de la función:

- ***int \$id_usuario*** ID del usuario

callback utilizado por las acciones create y update para validar la existencia de un usuario

- **Access public**

void función Bitacora::view(\$id) [line 301]

Parámetros de la función:

- ***int \$id*** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access public**

Clase Catalogo

[line 10]

Controlador Catalogo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Catalogo::__construct() [line 12]

- **Access** public

void función Catalogo::checkpk(\$campos) [line 745]

Parámetros de la función:

- *string* **\$campos**

Acción para revisar registros repetidos en las columnas designadas como primary key

- **Access** public

void función Catalogo::checkTypeData(\$campo, \$type) [line 776]

Parámetros de la función:

- *string* **\$campo** Nombre del campo a revisar
- *string* **\$type** Define el tipo de dato del campo

Acción para revisar si los tipos de datos coinciden con los datos contenidos en la tabla temporal que fueron tomados del CSV

- **Access public**

void función Catalogo::delete(\$nombre) [line 805]

Parámetros de la función:

- *string \$nombre*

Acción para eliminar un catálogo, recibe el nombre del catalogo a eliminar

- **Access public**

void función Catalogo::index() [line 35]

Acción por default del controlador, carga la lista de catálogos disponibles y una lista de opciones No recibe parámetros

- **Access public**

void función Catalogo::insert() [line 548]

Acción para preparar la inserción de nuevos catálogos , realiza la validación del formulario del lado del servidor y crea la estructura para el catálogo, crea la tabla y obtiene los datos a partir de la tabla tmp_catalogo

- **Access public**

void función Catalogo::load(0) [line 130]

Parámetros de la función:

- **\$_FILES[] 0** archivocsv Variable pasada por POST con el archivo csv para cargar datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP Guarda en la tabla tmp_catalogos toda la estructura del CSV e imprime las columnas del archivo. Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Catalogo::loadupdate(\$nombrecat, [\$update = false]) [line 230]

Parámetros de la función:

- **\$_FILES[] \$nombrecat** archivocsv Variable pasada por POST con el archivo csv para cargar datos
- **\$update**

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar. Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Catalogo::update(\$nombre, [\$pag = 0]) [line 681]

Parámetros de la función:

- *string* **\$nombre**
- *int* **\$pag** Numero de registro para el paginador

Acción para preparar la actualizacion de un catálogo ya existente,

recibe un string para obtener los valores del catalogo y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y servidor

- **Access** public

void función Catalogo::view(\$nombre, [\$pag = 0]) *[line 68]*

Parámetros de la función:

- *string* **\$nombre** Este parámetro no puede ser nulo
- *int* **\$pag** Numero de registro para el paginador

Acción para visualizar información de un catálogo específico, obtiene el objeto catálogo por medio del nombre proporcionado

- **Access** public

void función Catalogo::_array_unique_recursive(\$arr) *[line 531]*

Parámetros de la función:

- *array* **\$arr** Arreglo multidimensional

_array_unique_recursive Revisa valores duplicados en arreglos multidimensionales

- **Access** public

Clase CatalogoCsv

[line 10]

Controlador CatalogoCsv

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función CatalogoCsv::__construct() [line 12]

- **Access** public

Boolean función CatalogoCsv::ActivaEnCatalogo(\$id, \$catalogo, \$activo) [line 403]

Parámetros de la función:

- *Int* **\$id** Es el id del registro en el catalogo
- *String* **\$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- *Boolean* **\$activo** False para quitar del catalogo, true para agregarlo

* **Accion para activar o desactivar elementos en los catalogos indicados en el parametro Solo se permite su acceso por medio de peticiones AJAX**

- **Access** public

void función CatalogoCsv::checkpk(\$campos) [line 494]

Parámetros de la función:

- *string* \$campos

Acción para revisar registros repetidos en las columnas designadas como primary key

- **Access** public

void función CatalogoCsv::createTableAgeb() [line 587]

Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros

- **Access** public

void función CatalogoCsv::createTableGeo() [line 555]

Acción para ejecutar la creación de la tabla poblacional No recibe parámetros

- **Access** public

void función CatalogoCsv::createTableHemoGlobina() [line 620]

Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros

- **Access** public

void función CatalogoCsv::createTablePob() [line 522]

Acción para ejecutar la creación de la tabla poblacional No recibe parámetros

- **Access** public

void función CatalogoCsv::index() [*line 35*]

Acción por default del controlador, carga la lista de catálogoscsv disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función CatalogoCsv::loadupdate(\$nombrecat, [\$update = false]) [*line 119*]

Parámetros de la función:

- *\$_FILES[]* **\$nombrecat** archivocsv Variable pasada por POST con el archivo csv para cargar datos
- *boolean* **\$update** Define si se actualizará la DB o solo se hará la primera revisión de datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP

compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función CatalogoCsv::update(\$nombre, [\$pag = 0]) [*line 447*]

Parámetros de la función:

- *string* **\$nombre**

- **int \$pag** Numero de registro para el paginador

Acción para preparar la actualización de un catálogo ya existente,

recibe un string para obtener los valores del catalogo y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y servidor

- **Access public**

void función CatalogoCsv::view(\$nombre, [\$pag = 0]) [line 68]

Parámetros de la función:

- **string \$nombre** Este parametro no puede ser nulo
- **int \$pag** Numero de registro para el paginador

Acción para visualizar información de un catálogo específico, obtiene el objeto catalogocsv por medio del nombre proporcionado

- **Access public**

void función CatalogoCsv::_array_unique_recursive(\$arr) [line 377]

Parámetros de la función:

- **array \$arr**

_array_unique_recursive **Revisa valores duplicados en arreglos que contienen arreglos**

- **Access** public

Clase Catalogo_x_raiz

[line 10]

Controlador Raiz_x_Catalogo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Catalogo_x_raiz::__construct() [line 12]

- **Access** public

boolean función Catalogo_x_raiz::check(\$id) [line 174]

Parámetros de la función:

- *type* **\$id** Id del catalogo_x_raiz

Acción que sirve para revisar inconsistencias en el arbol de segmentacion Recibe como parámetro el catálogo x raiz y revisa que todos los registros tengan un correspondiente en el catálogo padre.

Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Catalogo_x_raiz::delete(\$id) [line 226]

Parámetros de la función:

- *int \$id*

Acción para eliminar un catálogo en el arbol, recibe el id del catálogo en la raiz a eliminar

- **Access public**

void función Catalogo_x_raiz::insert([\$id = 0]) [line 63]

Parámetros de la función:

- *\$id*

Acción para preparar la inserción de nuevas raices para catálogos , realiza la validación del formulario del lado cliente

- **Access public**

void función Catalogo_x_raiz::view(\$id) [line 36]

Parámetros de la función:

- *int \$id* Este parametro no puede ser nulo

Acción para visualizar información de una raiz_x_catalogo específica, obtiene el objeto raiz_x_catalogo por medio del id proporcionado.

- **Access** public

Clase Cie10

[line 10]

Controlador Cie10

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Cie10::__construct() [line 12]

- **Access** public

Boolean función Cie10::ActivaEnCatalogo(\$id, \$catalogo, \$activo) [line 166]

Parámetros de la función:

- *Int* **\$id** Es el id del registro en el catalogo
- *String* **\$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- *Boolean* **\$activo** False para quitar del catalogo, true para agregarlo

* **Accion para activar o desactivar elementos en los catalogos IRA EDA Consultas**
Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

Boolean función Cie10::AgregaEnCatalogo(\$id, \$catalogo, \$activo) [line 121]

Parámetros de la función:

- **Int \$id** Es el id del registro en el catalogo de CIE10
- **String \$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- **Boolean \$activo** False para quitar del catalogo, true para agregarlo

* **Accion para agregar elementos del catalogo cie10 a los catalogos de EDA, IRA y Consultas dependiendo de los Solo se permite su acceso por medio de peticiones AJAX**

- **Access public**

void función Cie10::index([\$pag = 0]) [line 36]

Parámetros de la función:

- **int \$pag** Numero de registro para el paginador

Acción por default del controlador, carga la lista de datos disponibles en el cie10 y una lista de opciones

- **Access public**

void función Cie10::insert([\$update = false]) [line 346]

Parámetros de la función:

- **\$_FILES[] \$update** archivocsv Variable pasada por POST con el archivo csv para cargar datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables

PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar
Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Cie10::load(0) [line 209]

Parámetros de la función:

- **`$_FILES[] 0`** archivocsv Variable pasada por POST con el archivo csv para cargar datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP

Guarda en la tabla tmp_catalogos toda la estructura del CSV e imprime las columnas del archivo Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Cie10::update(\$id) [line 266]

Parámetros de la función:

- **`int $id`**

Acción para preparar la actualización de un registro del CIE10,

recibe un id para obtener los valores del catálogo y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y servidor

- **Access public**

void función Cie10::view(\$cat) [line 84]

Parámetros de la función:

- *string* **\$cat** Nombre del catalogo a mostrar

*** Accion para mostrar información de los catalogos IDE , ERA y Consultas**

- **Access public**

void función Cie10::_array_unique_recursive(\$arr) [line 578]

Parámetros de la función:

- *array* **\$arr**

_array_unique_recursive **Revisa valores duplicados en arreglos que contienen arreglos**

- **Access public**

Clase Controlador

[line 10]

Controlador Controlador

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Controlador::__construct() [line 12]

- **Access** public

void función Controlador::accion(\$id) [line 300]

Parámetros de la función:

- *int* \$id

Acción para preparar la actualización de acciones asignadas a un controlador, recibe un ID para obtener las acciones asignadas a ese controlador y mostrarlos en la vista update

- **Access** public

void función Controlador::delete(\$id) [line 365]

Parámetros de la función:

- *int* \$id

Acción para eliminar un controlador, recibe el id del controlador a eliminar

- **Access** public

Object función Controlador::getGroupPermissions(\$entorno, \$grupo) [line 400]

Parámetros de la función:

- *int* \$entorno
- *int* \$grupo

Acción para servir un array de objetos con los permisos asignados a

un entorno y grupo determinados, esta accion solo es accedida por peticiones AJAX y devuelve un objeto JSON Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Controlador::help(\$idControladorAccion, \$id_controlador_accion) [line 422]

Parámetros de la función:

- *int* \$id_controlador_accion
- \$idControladorAccion

Establece el texto de ayuda para el controlador accion

- **Access** public

void función Controlador::index([\$pag = 0]) [line 36]

Parámetros de la función:

- *int* \$pag Numero de registro para el paginador

Acción por default del controlador, carga la lista de controladores disponibles y una

lista de opciones Recibe un parametro en caso de filtrado por entornos

- **Access public**

void función Controlador::insert([\$id = FALSE]) [line 138]

Parámetros de la función:

- **\$id**

Acción para preparar la insercion de nuevos controladores , realiza la validacion del formulario del lado cliente

- **Access public**

void función Controlador::update(\$id) [line 205]

Parámetros de la función:

- **int \$id**

Acción para preparar la actualización de un controlador ya existente,
recibe un ID para obtener los valores de ese controlador y mostrarlos en la vista update ,
realiza la validación del formulario del lado del cliente

- **Access public**

void función Controlador::view(\$id) [line 109]

Parámetros de la función:

- **int \$id** Este parametro no puede ser nulo

Acción para visualizar información de un controlador específico, obtiene el objeto controlador por medio del id proporcionado.

- **Access** public

Clase Entorno

[line 11]

Controlador Entorno

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Entorno::__construct() [line 13]

- **Access** public

void función Entorno::delete(\$id) [line 290]

Parámetros de la función:

- **int \$id**

Acción para eliminar un entorno, recibe el id del entorno a eliminar

- **Access public**

void función Entorno::index() [line 35]

Acción por default del controlador, carga la lista de entornos disponibles y una lista de opciones No recibe parámetros

- **Access public**

void función Entorno::insert() [line 94]

Acción para preparar la inserción de nuevos entornos , realiza la validación del formulario del lado cliente y del lado servidor para evitar entornos duplicados

- **Access public**

void función Entorno::update(\$id) [line 208]

Parámetros de la función:

- **int \$id**

Acción para preparar la actualización de un entorno ya existente,

recibe un ID para obtener los valores de ese entorno y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y del servidor para evitar datos duplicados

- **Access public**

void función Entorno::view(\$id) [line 65]

Parámetros de la función:

- *int* **\$id** Este parametro no puede ser nulo

Acción para visualizar de un entorno específico, obtiene el objeto entorno por medio del id proporcionado.

- **Access public**

boolean función Entorno::_ExistEntorno(\$nombre_entorno) [line 153]

Parámetros de la función:

- *string* **\$nombre_entorno** Revisa si este valor ya existe como un entorno

Acción para validar que no exista previamente el entorno a insertar (Esta acción no puede ser accedida desde el navegador)

- **Access public**

boolean función Entorno::_ExistEntornoUpdate(\$nombre_entorno) [line 178]

Parámetros de la función:

- *string* **\$nombre_entorno** Revisa si este valor ya existe como un entorno

Acción para validar que no exista previamente el entorno a actualizar
esta acción revisa si el nombre a usar ya existe en la base excepto el mismo objeto a

actualizar (Esta acción no puede ser accedida desde el navegador)

- **Access** public

Clase Errorlog

[line 11]

Controlador Errorlog

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Errorlog::__construct() [line 13]

- **Access** public

void función Errorlog::index([\$pag = 0]) [line 34]

Parámetros de la función:

- **int \$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la tabla error, con su correspondiente paginación permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, muestra enlaces para ver detalles de un elemento específico

- **Access** public

void función Errorlog::view(\$id) *[line 141]*

Parámetros de la función:

- *int* **\$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Grupo

[line 10]

Controlador Grupo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor void función Grupo::__construct() *[line 12]*

- **Access** public

void función Grupo::delete(\$id) [line 195]

Parámetros de la función:

- *int \$id* id del grupo a eliminar

Solicita la eliminación del grupo recibido

- **Access** public

void función Grupo::index([\$pag = 0]) [line 33]

Parámetros de la función:

- *int \$pag* número de página a visualizar (paginación)

1) Visualiza los grupos existentes para su interacción CRUD 2) En caso de detectar un texto a buscar se filtran los grupos existentes acorde a la búsqueda

- **Access** public

void función Grupo::insert() [line 109]

1) Prepara el formulario para la inserción de un grupo nuevo 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Grupo::update(\$id) [line 152]

Parámetros de la función:

- *int \$id* id del grupo a modificar

1) Prepara el formulario para la modificación de un grupo existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access public**

void función Grupo::view(\$id) [line 81]

Parámetros de la función:

- *int \$id* id del grupo a visualizar

Visualiza los datos del grupo recibido

- **Access public**

boolean función Grupo::_ifGroupExists(\$name) [line 222]

Parámetros de la función:

- *string \$name* nombre del grupo a validar

Callback para validar que un nombre de grupo no se duplique

- **Access public**

Clase Menu

[line 11]

Controlador Menu

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Menu::__construct() [line 21]

- **Access** public

void función Menu::delete(\$id) [line 362]

Parámetros de la función:

- *int* **\$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

void función Menu::index([\$pag = 0]) [line 47]

Parámetros de la función:

- *int* **\$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la menu, con su correspondiente paginación

permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico

- **Access public**

void función Menu::insert([\$id = null]) [line 130]

Parámetros de la función:

- **\$id**

Muestra el formulario para crear un nuevo registro en la menu, las variables se obtienen por el metodo POST

- **Access public**

void función Menu::update(\$id) [line 225]

Parámetros de la función:

- **int \$id** ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access public**

void función Menu::view(\$id) [line 326]

Parámetros de la función:

- *int* **\$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Permiso

[line 10]

Controlador Permiso

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Permiso::__construct() [line 12]

- **Access** public

void función Permiso::index(\$id) [line 34]

Parámetros de la función:

- *int* **\$id** id del grupo del cual se obtendrán (y actualizar si aplica) los permisos asignados

1) Visualiza los entornos existentes para su selección

2) Al seleccionar un entorno se obtienen los controladores_x_accion existentes y se indica sobre cuales el grupo tiene permisos asignados 3) Elimina los permisos asignados al grupo anteriormente e inserta los asignados recientemente

- **Access** public

Clase Raiz

[line 11]

Controlador Raiz

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Giovanni

Constructor *void* función Raiz::__construct() [line 13]

- **Access** public

void función Raiz::createasu(\$id) [line 316]

Parámetros de la función:

- *int* \$id

Acción para crear el ASU a partir de una raíz Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Raiz::delete(\$id) [line 284]

Parámetros de la función:

- ***int \$id***

Acción para eliminar una raíz, recibe el id de la raíz a eliminar

- **Access public**

Object función Raiz::getChildrenFromLevel(\$idarbol, \$nivel, \$claves) [line 632]

Parámetros de la función:

- ***Int \$idarbol*** parametro pasado por POST y determina el arbol a consultar
- ***Int \$nivel*** parametro pasado por POST y determina el nivel superior a desglosar en el arbol
- ***Array \$claves*** Este parametro es pasado por POST y es la lista de valores a preseleccionar en el arbol

Accion para regresar el arbol de segmentacion determinado, el objeto regresado contiene estructura de arbol y es consumida solamente por peticiones AJAX

- **Access public**

Object función Raiz::getDataKeyValue(\$idarbol, \$nivel, [\$filtro = 0]) [line 738]

Parámetros de la función:

- *int* **\$idarbol**
- *Int* **\$nivel** Nivel de desglose de información requerida
- *Int* **\$filtro** (Opcional) filtrar por un valor determinado

Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

Object función Raiz::getDataTreeFromId(\$claves, \$desglose) [line 596]

Parámetros de la función:

- *Array* **\$claves** Este parametro es pasado por POST y es la lista de valores a consultar
- *Int* **\$desglose** parametro pasado por POST y determina si se requiere información adicional

Accion para regresar la descripción e informacion adicional de un arreglo de ID's desde el arbol de segmentacion

- **Access** public

Object función Raiz::getTreeBlock(\$idarbol, \$nivel, \$seleccionados, \$seleccionable, \$seleccionables, \$omitidos) [line 678]

Parámetros de la función:

- *int* **\$idarbol** ID del arbol (el arbol usado por la TES es el 1)
- *int* **\$nivel** nivel del arbol que se desea obtener
- *array* **\$seleccionados** se especifica si dentro del arreglo de retorno, hay valores preseleccionados
- *bool* **\$seleccionable** especifica si los elementos del arbol pueden ser seleccionados
- *array* **\$seleccionables** especifica que niveles del arbol pueden ser seleccionados

- **array \$omitidos** especifica niveles omitidos dentro del arbol (Si hay un nivel intermedio omitido, los hijos de este nivel son agregados como hijos de su nivel inmediato superior)

Sirve para obtener bloques del arbol de segmentación única ASU

solo se puede acceder por peticiones AJAX, los parametros son pasados por GET

- **Access public**

void función Raiz::index() [line 62]

Acción por default del controlador, carga la lista de Raices disponibles y una lista de opciones No recibe parámetros

- **Access public**

void función Raiz::iniciarasu(\$id) [line 38]

Parámetros de la función:

- **type \$id** ID del asu

Crea los archivos JSON necesarios para iniciar el ASU en caché y agilizar su carga

- **Access public**

void función Raiz::insert() [line 134]

Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente

- **Access public**

void función Raiz::update(\$id) [line 188]

Parámetros de la función:

- ***int \$id***

Acción para preparar la actualización de una raiz ya existente,

recibe un ID para obtener los valores de esa raiz y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access public**

void función Raiz::updateasu(\$id) [line 439]

Parámetros de la función:

- ***int \$id***

Acción para actualizar el ASU a partir de una raiz Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

void función Raiz::view(\$id) [line 94]

Parámetros de la función:

- ***int \$id*** Este parametro no puede ser nulo

Acción para visualizar información de una raíz específica, obtiene el objeto raíz por medio del id proporcionado.

- **Access** public

Clase ReglaVacuna

[line 10]

Controlador ReglaVacuna

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función ReglaVacuna::__construct() *[line 12]*

- **Access** public

void función ReglaVacuna::delete(\$id) *[line 333]*

Parámetros de la función:

- *int* \$id

Acción para eliminar una regla, recibe el id de la regla a eliminar

- **Access public**

void función ReglaVacuna::index() [line 35]

Acción por default del controlador, carga la lista de reglas de vacunas disponibles y una lista de opciones No recibe parámetros

- **Access public**

void función ReglaVacuna::insert() [line 95]

Acción para preparar la insercion de nuevas reglas , realiza la validación del formulario del lado cliente

- **Access public**

void función ReglaVacuna::update(\$id) [line 203]

Parámetros de la función:

- *int \$id*

Acción para preparar la actualización de una regla ya existente,

recibe un ID para obtener los valores de esa regla y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access public**

void función ReglaVacuna::view(\$id) [line 66]

Parámetros de la función:

- **int \$id** Este parametro no puede ser nulo

Acción para visualizar información de una regla específica, obtiene el objeto regla_vacuna por medio del id proporcionado.

- **Access** public

Clase Usuario

[line 10]

Controlador Usuario

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Usuario::\$mas

mixed = "" [line 773]

Acción para hacer autologin en otro sistema

- **Access** public

Constructor *void* función Usuario::__construct() [line 12]

- **Access public**

void función Usuario::automatic_access() [line 774]

- **Access public**

redirect función Usuario::cerrar_etab() [line 809]

Acción para cerrar el login en otro sistema

void función Usuario::delete(\$id) [line 325]

Parámetros de la función:

- *int* **\$id** id del usuario a eliminar

Solicita la eliminación del usuario recibido

- **Access public**

void función Usuario::form_init() [line 384]

Object función Usuario::getActivesByGroup(\$grupo) [line 752]

Parámetros de la función:

- *int* **\$grupo**

Acción para servir un array de objetos con los usuarios activos por grupo AJAX y devuelve un objeto JSON

- **Access public**

cookies función Usuario::get_galleta() [line 864]

Obtiene las cookies que se generan en la session

token función Usuario::get_token(\$url, \$var, [\$valor = ""], [\$num = ""], [\$count = ""], [\$par = ""]) [line 820]

Parámetros de la función:

- **\$url**
- **\$var**
- **\$valor**
- **\$num**
- **\$count**
- **\$par**

Acción para obtener el token oculto en el login

void función Usuario::index([\$pag = 0]) [line 126]

Parámetros de la función:

- **int \$pag** número de página a visualizar (paginación)

1) Visualiza los usuarios existentes para su interacción CRUD 2) En caso de detectar un texto a buscar se filtran los usuarios existentes acorde a la búsqueda

- **Access public**

void función Usuario::insert() [line 197]

1) Prepara el formulario para la inserción de un usuario nuevo 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Usuario::load_update() [*line 655*]

- **Access** public

void función Usuario::login() [*line 33*]

Ofrece el inicio de sesión

- **Access** public

void función Usuario::logout() [*line 108*]

Termina la sesión

- **Access** public

void función Usuario::remember() [*line 393*]

- **Access** public

void función Usuario::reset() [*line 588*]

- **Access** public

void función Usuario::send_mail(\$subject, \$body, \$from, \$rto, \$correo, \$CC, \$CCO, \$adj) [line 556]

Parámetros de la función:

- **\$subject**
- **\$body**
- **\$from**
- **\$rto**
- **\$correo**
- **\$CC**
- **\$CCO**
- **\$adj**

- **Access public**

void función Usuario::token(\$str) [line 518]

Parámetros de la función:

- **\$str**

- **Access public**

void función Usuario::update(\$id) [line 262]

Parámetros de la función:

- **int \$id** id del usuario a modificar

1) Prepara el formulario para la modificación de un usuario existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Usuario::update_info() [line 684]

- **Access** public

void función Usuario::view(\$id) [line 170]

Parámetros de la función:

- *int* **\$id** id del usuario a visualizar

Visualiza los datos del usuario recibido

- **Access** public

boolean función Usuario::_ifUserExists(\$username) [line 352]

Parámetros de la función:

- *string* **\$username** nombre de usuario a validar

Callback para validar que un nombre de usuario no se duplique

- **Access** public

Clase Accion_model

[line 11]

Modelo Accion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Accion_model::__construct() [line 129]

- **Access** public

boolean función Accion_model::delete() [line 268]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Accion_model::getAll() [line 145]

Devuelve todos los registros de la tabla acciones

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Accion_model::getById(\$id) [line 192]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de una accion por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Accion_model::getDescripcion() [*line 79*]

- **Access** public

void función Accion_model::getId() [*line 64*]

- **Access** public

void función Accion_model::getMetodo() [*line 87*]

- **Access** public

string/boolean función Accion_model::getMsgError([\$value = 'usr'], \$value,) [*line 114*]
Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Accion_model::getNombre() [line 72]

- **Access** public

int función Accion_model::getNumRows() [line 171]

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Accion_model::insert() [line 213]

Inserta en la tabla accion la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Accion_model::setDescripcion(\$value) [line 83]

Parámetros de la función:

- **\$value**

- **Access** public

void función Accion_model::setId(\$value) [line 68]

Parámetros de la función:

- **\$value**

- **Access public**

void función Accion_model::setMetodo(\$value) [line 91]

Parámetros de la función:

- **\$value**

- **Access public**

void función Accion_model::setNombre(\$value) [line 75]

Parámetros de la función:

- **\$value**

- **Access public**

void función Accion_model::setOffset(\$value) [line 95]

Parámetros de la función:

- **\$value**

- **Access** public

void función Accion_model::setRows(\$value) [line 98]

Parámetros de la función:

- **\$value**

- **Access** public

boolean función Accion_model::update() [line 240]

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Ageb_model

[line 11]

Modelo Ageb

- **Package** SIIGS
- **Sub-Package** Modelo

- **Author** Geovanni

Constructor *void* función Ageb_model::__construct() [line 36]

- **Access** public

boolean/string función Ageb_model::getMsgError([\$type = 'usr']) [line 60]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Ageb_model::process() [line 82]

Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion

- **Access** public

Object función Ageb_model::searchageb(\$idlocalidad, \$like) [line 193]

Parámetros de la función:

- *int* **\$idlocalidad** Id del ASU de la localidad
- **\$like**

Devuelve una lista de AGEBS de la localidad pasada como parámetro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Ageb_model::searchUM(\$idlocalidad, \$ageb) [*line 162*]

Parámetros de la función:

- *int* **\$idlocalidad** Id del ASU de la localidad
- *string* **\$ageb** Ageb

Devuelve la información de una UM de acuerdo a su localidad y ageb

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ArbolSegmentacion_model

[*line 11*]

Modelo ArbolSegmentacion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función ArbolSegmentacion_model::__construct() [*line 45*]

- **Access** public

Array función ArbolSegmentacion_model::convertType(\$arbol, \$seleccionable, \$seleccionados) [line 398]

Parámetros de la función:

- Array **\$arbol** fila array de valores
- bool **\$seleccionable** Seleccionable opcion para que este elemento sea seleccionable
- **\$seleccionados**

Convierte el tipo de arreglo para enviarlo como se debe recibir en el cliente de Javascript

- **Access** public

object con función ArbolSegmentacion_model::getById(\$item) [line 372]

Parámetros de la función:

- int **\$item** item seleccionado

Obtener el elemento del ASU por medio de su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getChildrenFromId(\$id, [\$omitidos = array()]) [line 331]

Parámetros de la función:

- int **\$id** Id del elemento en el ASU
- Array **\$omitidos** int \$omitidos Array de niveles omitidos

Accion para devolver los hijos de un elemento en el ASU a partir de su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getChildrenFromLevel(\$idarbol, \$nivel, [\$omitidos = array()], [\$seleccionados = array()], [\$seleccionables = array()]) [line 706]

Parámetros de la función:

- *Int* **\$idarbol** Id del arbol a crear
- *Int* **\$nivel** Nivel de segmentacion desde la cual se desarrolla el arbol
- *Array* **\$omitidos** int \$omitidos Array de niveles omitidos
- *Array* **\$seleccionados** int \$seleccionados Array de elementos seleccionados
- *Array* **\$seleccionables** int \$seleccionables Array de niveles que son seleccionables

Accion para devolver el esquema completo del ASU a partir de un nivel especificado, niveles omitidos y elementos preseleccionados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ArbolSegmentacion_model::getCluesFromId(\$id) [line 299]

Parámetros de la función:

- *int* **\$id** Id del elemento en el asu
- * @return Object un arreglo con la estructura del arbol

*

Obtiene las unidades medicas correspondientes a un ID de un elemento independientemente su nivel en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getDataKeyValue(\$idarbol, \$nivel, [\$filtro = 0]) [line 1031]

Parámetros de la función:

- *int* **\$idarbol** Id del arbol a buscar
- *Int* **\$nivel** Nivel de desglose de información requerida
- *Int* **\$filtro** (Opcional) filtrar por un valor determinado

Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getDescripcionById(\$claves, [\$desglose = 0]) [line 935]

Parámetros de la función:

- *Int* **\$desglose** Nivel de desglose de información requerida
- *Array* **\$claves** int \$claves arreglo de valores a recuperar

Accion para obtener la descripcion e información adicional del elemento en el ASU

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

void función ArbolSegmentacion_model::getListChildrenLevel(\$resultadotemp, [\$clave2 = 0]) [line 640]

Parámetros de la función:

- **\$resultadotemp**
- **\$clave2**

- **Access public**

string|boolean función ArbolSegmentacion_model::getMsgError([\$value = 'usr'], \$value,) [line 30]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access public**

Object función ArbolSegmentacion_model::getTree(\$idarbol, \$nivel, [\$nivelesocultos = array()]) [line 123]

Parámetros de la función:

- *int* **\$idarbol** ID del arbol a crear
- *int* **\$nivel** Nivel de segmentacion desde el cual se iniciará a desarrollar el arbol
- *\$nivelesocultos* **\$nivelesocultos** Array con los niveles que se deben ocultar

Regresa el objeto del arbol de segmentacion

- **Throws** Exception En caso de algun error al consultar la base de datos

- **Access** public

Object función ArbolSegmentacion_model::getTreeBlock(\$idarbol, \$nivel, [\$seleccionados = array()], \$seleccionable, \$elegido, [\$omitidos = array()], [\$seleccionables = array()]) [line 441]

Parámetros de la función:

- *Int \$idarbol* Id del arbol a crear
- *Int \$nivel* Nivel de segmentacion desde la cual se desarrolla el arbol
- *Array \$seleccionados* int \$omitidos Array de niveles omitidos
- *Array \$seleccionable* int \$seleccionados Array de elementos seleccionados
- *Array \$elegido* int \$seleccionables Array de niveles que son seleccionables
- *Array \$omitidos* int \$elegido Clave seleccionada para obtener sus hijos
- **\$seleccionables**

Accion para devolver un bloque del ASU a partir de un nivel especificado o una clave seleccionada, niveles omitidos y elementos preseleccionados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getTreeBlockData(\$idarbol, \$nivel, \$elegido) [line 221]

Parámetros de la función:

- *int \$idarbol* ID del arbol a crear
- *int \$nivel* Nivel de segmentacion desde el cual se iniciará a desarrollar el arbol
- *\$nivelesocultos \$elegido* Array con los niveles que se deben ocultar

Regresa el objeto del arbol de segmentacion por nivel o hijos de un elemento seleccionado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getUMPParentsById(\$clave) [line 63]

Parámetros de la función:

- **int \$clave** Clave de la unidad medica u elemento en el ASU

Regresa la información de los padres de una unidad medica en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ArbolSegmentacion_model::_addSelectedItems(\$datos, \$nivel, \$seleccionados, \$seleccionables) [line 863]

Parámetros de la función:

- **\$datos**
- **\$nivel**
- **\$seleccionados**
- **\$seleccionables**

- **Access** public

void función ArbolSegmentacion_model::_addSelectedItems_(\$datos, \$seleccionados, \$nivel, \$omitidos, \$seleccionables) [line 894]

Parámetros de la función:

- **\$datos**
- **\$seleccionados**
- **\$nivel**
- **\$omitidos**
- **\$seleccionables**

- **Access** public

Clase Bitacora_model

[line 11]

Modelo Bitacora

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Bitacora_model::__construct() [line 93]

- **Access** public

void/boolean función Bitacora_model::addFilter(\$columna, \$condicion, \$valor) [line 398]

Parámetros de la función:

- *string* **\$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros, id_controlador_accion)
- *string* **\$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **\$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

int función Bitacora_model::delete([\$sid = null]) [line 278]

Parámetros de la función:

- *int* **\$sid** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

int función Bitacora_model::deleteByFilter() [line 317]

Elimina el conjunto de registros que cumplen con el o los criterios de filtrado

- **Access** public

array función Bitacora_model::getAll([\$offset = null], [\$row_count = null]) [line 467]

Parámetros de la función:

- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

object|boolean función Bitacora_model::getByld(\$id) [line 349]

Parámetros de la función:

- **int \$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro de la bitacora que tiene el ID especificado

- **Access** public

void función Bitacora_model::getFecha_hora() [line 134]

- **Access** public

void función Bitacora_model::getId() [line 114]

- **Access** public

void función Bitacora_model::getId_controlador_accion() [line 154]

- **Access** public

void función Bitacora_model::getId_usuario() [line 124]

- **Access** public

boolean|string función Bitacora_model::getMsgError([\$type = 'usr']) [line 185]

Parámetros de la función:

- **string \$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access public**

int función Bitacora_model::getNumRows() [line 514]

Obtiene el numero total de registros en la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta

- **Access public**

void función Bitacora_model::getParametros() [line 144]

- **Access public**

boolean función Bitacora_model::insert(\$path, \$parametros) [line 208]

Parámetros de la función:

- **string \$path** Concatenación de directorio del proyeto, clase y metodo, unidos por dos dobles puntos '::'
- **string \$parametros**

Inserta a la bitacora la información porporcionada

- **Static**
- **Access** public

void función Bitacora_model::resetFilter() [*line 451*]
Elimina todos los filtros registrados

- **Access** public

void función Bitacora_model::setFecha_hora(\$fecha_hora) [*line 139*]
Parámetros de la función:

- **\$fecha_hora**

- **Access** public

void función Bitacora_model::setId(\$id) [*line 119*]
Parámetros de la función:

- **\$id**

- **Access** public

void función Bitacora_model::setId_accion(\$id_accion) [*line 169*]
Parámetros de la función:

- **\$id_accion**

- **Access public**

void función Bitacora_model::setId_controlador(\$id_controlador) [line 164]

Parámetros de la función:

- **\$id_controlador**

- **Access public**

void función Bitacora_model::setId_controlador_accion(\$id_controlador_accion) [line 159]

Parámetros de la función:

- **\$id_controlador_accion**

- **Access public**

void función Bitacora_model::setId_usuario(\$id_usuario) [line 129]

Parámetros de la función:

- **\$id_usuario**

- **Access public**

void función Bitacora_model::setParametros(\$parametros) [line 149]

Parámetros de la función:

- **\$parametros**

- **Access** public

boolean función Bitacora_model::update([\$id = null]) [line 241]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase CatalogoCsv_model

[line 11]

Modelo CatalogoCsv

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función CatalogoCsv_model::__construct() [line 128]

- **Access** public

boolean función CatalogoCsv_model::activaEnCatalogo(\$id, \$catalogo, \$valor) [line 255]

Parámetros de la función:

- *int* **\$id** el id del registro en el catalogo
- *string* **\$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **\$valor** agregar o eliminar el registro del catalogo

Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos
- **Access** public

boolean función CatalogoCsv_model::checkPk(\$campo) [line 282]

Parámetros de la función:

- *string* **\$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función CatalogoCsv_model::getAll() [*line 165*]

Devuelve una lista con los catalogos existentes en la DB

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función CatalogoCsv_model::getAllData(\$nombrecat) [*line 186*]

Parámetros de la función:

- *string* **\$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función CatalogoCsv_model::getByName(\$nombre) [*line 213*]

Parámetros de la función:

- *string* **\$nombre** (Nombre del catalogo)

Devuelve la informacion de un catalogo por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función CatalogoCsv_model::getCampos() [*line 79*]

- **Access public**

void función CatalogoCsv_model::getId() [*line 64*]

- **Access public**

void función CatalogoCsv_model::getLLave() [*line 86*]

- **Access public**

string|boolean función CatalogoCsv_model::getMsgError([\$value = 'usr'], \$value,) [*line 113*]
Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access public**

void función CatalogoCsv_model::getNombre() [*line 72*]

- **Access public**

int función CatalogoCsv_model::getNumRows(\$nombre) [*line 145*]

Parámetros de la función:

- *string* **\$nombre** Nombre del catalogo

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función CatalogoCsv_model::setCampos(\$value) [*line 82*]

Parámetros de la función:

- **\$value**

- **Access** public

void función CatalogoCsv_model::setId(\$value) [*line 68*]

Parámetros de la función:

- **\$value**

- **Access** public

void función CatalogoCsv_model::setLlave(\$value) [*line 89*]

Parámetros de la función:

- **\$value**

- **Access public**

void función CatalogoCsv_model::setNombre(\$value) [line 75]

Parámetros de la función:

- **\$value**

- **Access public**

void función CatalogoCsv_model::setOffset(\$value) [line 93]

Parámetros de la función:

- **\$value**

- **Access public**

void función CatalogoCsv_model::setRows(\$value) [line 96]

Parámetros de la función:

- **\$value**

- **Access public**

Clase Catalogo_model

[line 11]

Modelo Catalogo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Catalogo_model::__construct() [line 128]

- **Access** public

boolean función Catalogo_model::checkPk(\$campo) [line 254]

Parámetros de la función:

- *string* **\$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::checkTypeData(\$campo, \$type) [line 287]

Parámetros de la función:

- *string* **\$campo** (varios campos delimitados por |)
- **\$type**

Revisa en la base de datos por registros que no coincidan con el tipo de dato pasado como parametro en el campo indicado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::delete() [*line 386*]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_model::getAll() [*line 144*]

Devuelve una lista con los catalogos existentes en la DB

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_model::getAllData(\$nombrecat) [*line 186*]

Parámetros de la función:

- *string* **\$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Catalogo_model::getByName(\$nombre) [line 213]

Parámetros de la función:

- *string* **\$nombre** (Nombre del catalogo)

Devuelve la informacion de un catalogo por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_model::getCampos() [line 79]

- **Access** public

void función Catalogo_model::getId() [line 64]

- **Access** public

void función Catalogo_model::getLLave() [line 86]

- **Access** public

string|boolean función Catalogo_model::getMsgError([\$value = 'usr'], \$value,) [line 113]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Catalogo_model::getNombre() [line 72]

- **Access** public

int función Catalogo_model::getNumRows(\$nombre) [line 165]

Parámetros de la función:

- *string* **\$nombre** Nombre del catalogo

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::insert(\$create, \$select) [line 329]

Parámetros de la función:

- **string \$create** (la consulta para crear el catalogo)
- **string \$select** (la consulta para extraer datos de la tabla tmp_catalogo)

Inserta en la base datos el catálogo y obtiene los datos de la tabla temporal

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_model::setCampos(\$value) [line 82]

Parámetros de la función:

- **\$value**

- **Access** public

void función Catalogo_model::setId(\$value) [line 68]

Parámetros de la función:

- **\$value**

- **Access** public

void función Catalogo_model::setLlave(\$value) [line 89]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_model::setNombre(\$value) [line 75]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_model::setOffset(\$value) [line 94]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_model::setRows(\$value) [line 97]

Parámetros de la función:

- **\$value**

- **Access public**

boolean función Catalogo_model::updateComentario(\$nombre, \$comentario) [line 364]

Parámetros de la función:

- *string* \$nombre
- *string* \$comentario

Cambia el comentario de la tabla indicada

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Catalogo_x_raiz_model

[line 11]

Modelo Raiz_x_Catalogo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Catalogo_x_raiz_model::__construct() [line 161]

- **Access** public

Object función Catalogo_x_raiz_model::check(\$id) [line 290]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Revisa inconsistencias en los datos de un catalogo x raiz con respecto a su catalogo padre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_x_raiz_model::delete() [line 407]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_x_raiz_model::getByArbol(\$id) [line 178]

Parámetros de la función:

- *int* **\$id**

Devuelve todos los registros de la tabla raiz_x_catalogo de una raiz determinada

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Catalogo_x_raiz_model::getById(\$id) [line 245]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de un catalogo x accion por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_x_raiz_model::getByNivel(\$idarbols, \$nivel) [*line 223*]

Parámetros de la función:

- *int* \$idarbols
- *int* \$nivel

Devuelve el catalogo padre de un elemento raiz_x_catalogo

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::getColumnaDescripcion() [*line 113*]

- **Access** public

void función Catalogo_x_raiz_model::getColumnaLLave() [*line 106*]

- **Access** public

void función Catalogo_x_raiz_model::getGrado() [*line 92*]

- **Access public**

void función Catalogo_x_raiz_model::getId() [line 76]

- **Access public**

void función Catalogo_x_raiz_model::getIdRaiz() [line 84]

- **Access public**

string/boolean función Catalogo_x_raiz_model::getMsgError([\$value = 'usr'], \$value,) [line 146]
Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access public**

ArrayObject función Catalogo_x_raiz_model::getNivel(\$id) [line 200]
Parámetros de la función:

- *int* **\$id**

Devuelve el nivel siguiente para la tabla raiz_x_catalogo de un arbol determinado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::getRelacionHijo() [*line 127*]

- **Access** public

void función Catalogo_x_raiz_model::getRelacionPadre() [*line 120*]

- **Access** public

Object función Catalogo_x_raiz_model::getRelations(\$id) [*line 267*]
Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve las relaciones de una raiz x catalogo

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::getTablaCatalogo() [*line 99*]

- **Access** public

int función Catalogo_x_raiz_model::insert() [*line 350*]

Inserta en la base datos la información del objeto actual

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::setColumnaDescripcion(\$value) [*line 116*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Catalogo_x_raiz_model::setColumnaLlave(\$value) [*line 109*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Catalogo_x_raiz_model::setGrado(\$value) [*line 95*]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_x_raiz_model::setId(\$value) [line 80]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_x_raiz_model::setIdRaiz(\$value) [line 88]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_x_raiz_model::setRelacionHijo(\$value) [line 130]

Parámetros de la función:

- **\$value**

- **Access public**

void función Catalogo_x_raiz_model::setRelacionPadre(\$value) [line 123]

Parámetros de la función:

- **\$value**

- **Access** public

void función Catalogo_x_raiz_model::setTablaCatalogo(\$value) *[line 102]*

Parámetros de la función:

- **\$value**

- **Access** public

Clase Cie10_model

[line 11]

Modelo Cie10

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Cie10_model::__construct() *[line 106]*

- **Access** public

boolean función Cie10_model::activaEnCatalogo(\$id, \$catalogo, \$valor) [line 293]

Parámetros de la función:

- *int* **\$id** el id del registro en el catalogo
- *string* **\$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **\$valor** agregar o eliminar el registro del catalogo

Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos
- **Access** public

boolean función Cie10_model::agregaEnCatalogo(\$id, \$catalogo, \$valor) [line 243]

Parámetros de la función:

- *int* **\$id** el id del registro en el catalogo cie10
- *string* **\$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **\$valor** agregar o eliminar el registro del catalogo

Accion para agregar registros del CIE10 a otros catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos
- **Access** public

boolean función Cie10_model::checkPk(\$campo) [line 343]

Parámetros de la función:

- *string* **\$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getAll() [line 143]

Devuelve una lista con los registros existentes en el catalogo cie10

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getAllData(\$nombrecat) [line 320]

Parámetros de la función:

- *string* **\$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Cie10_model::getById(\$id) [line 220]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de un registro del catalogo cie10 por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getCatalogoByName(\$cat) [line 194]

Parámetros de la función:

- **\$cat**

Devuelve una lista con los registros existentes en el catalogo requerido

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getData() [line 170]

Devuelve una lista con los registros existentes en el catalogo cie10 omitiendo los ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Cie10_model::getDescripcion() [line 69]

- **Access** public

void función Cie10_model::getId() [line 61]

- **Access** public

string|boolean función Cie10_model::getMsgError([\$value = 'usr'], \$value,) [line 91]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función Cie10_model::getNumRows() [line 122]

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Cie10_model::setDescripcion(\$value) [line 73]

Parámetros de la función:

- **\$value**

- **Access** public

void función Cie10_model::setId(\$value) [line 65]

Parámetros de la función:

- **\$value**

- **Access public**

void función Cie10_model::setOffset(\$value) [line 51]

Parámetros de la función:

- **\$value**

- **Access public**

void función Cie10_model::setRows(\$value) [line 54]

Parámetros de la función:

- **\$value**

- **Access public**

boolean función Cie10_model::update() [line 374]

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ControladorAccion_model

[line 11]

Modelo ControladorAccion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función ControladorAccion_model::__construct() [line 79]

- **Access** public

Object función ControladorAccion_model::getById(\$id) [line 119]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de una accion por controlador de acuerdo a su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función ControladorAccion_model::getId(\$controlador, \$accion) [line 97]

Parámetros de la función:

- *int* **\$controlador** (Id del controlador)
- *int* **\$accion** (Id de la accion)

Devuelve el Id de una accion por controlador de una accion y controlador determinados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función ControladorAccion_model::getIdByPath(\$path) [line 143]

Parámetros de la función:

- *string* **\$path**

Devuelve el id de una accion por controlador de acuerdo al path

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

string/boolean función ControladorAccion_model::getMsgError([\$value = 'usr'], \$value,) [line 64]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función ControladorAccion_model::setHelp(\$id, \$textAyuda) [*line 181*]

Parámetros de la función:

- *int* **\$id**
- *string* **\$textAyuda**

Establece el mensaje de ayuda

- **Throws** Exception En caso de algun error al guardar el texto de ayuda
- **Access** public

Clase Controlador_model

[*line 11*]

Modelo Controlador

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Controlador_model::__construct() [*line 159*]

- **Access** public

boolean función Controlador_model::accionesUpdate() [line 374]

Actualiza las acciones asignadas a un controlador

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Controlador_model::delete() [line 431]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::getAccion() [line 108]

- **Access** public

ArrayObject función Controlador_model::getAcciones(\$id) [line 271]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve todas las acciones asignadas al controlador por su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getAll() [line 175]

Devuelve todos los registros de la tabla controlador

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getByEntorno(\$id) [line 244]

Parámetros de la función:

- *int* **\$id** ID (Id del entorno)

Devuelve todos los controladores que pertenecen a un entorno por su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Controlador_model::getById(\$id) [line 222]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de un controlador por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::getClase() [*line 116*]

- **Access** public

void función Controlador_model::getDescripcion() [*line 92*]

- **Access** public

void función Controlador_model::getId() [*line 77*]

- **Access** public

void función Controlador_model::getIdEntorno() [*line 100*]

- **Access** public

string/boolean función Controlador_model::getMsgError([\$value = 'usr'], \$value,) [*line 144*]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Controlador_model::getNombre() [*line 85*]

- **Access** public

int función Controlador_model::getNumRows([\$entorno = 0]) [*line 201*]

Parámetros de la función:

- *int* **\$entorno** , default 0 (Id del entorno)

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getPermisos(\$entorno, \$grupo) [*line 296*]

Parámetros de la función:

- *int* **\$entorno** (Id del entorno)
- *int* **\$grupo** (Id del grupo)

Devuelve los permisos asignados a un grupo sobre un entorno determinado (Mapea la información de las acciones asignadas a un controlador y los une con los permisos de un grupo sobre esas acciones)

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Controlador_model::insert() [*line 317*]

Inserta en la tabla controlador, la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::setAccion(\$value) [*line 112*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Controlador_model::setClase(\$value) [*line 120*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Controlador_model::setDescripcion(\$value) [*line 96*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Controlador_model::setId(\$value) [line 81]

Parámetros de la función:

- **\$value**

- **Access public**

void función Controlador_model::setIdEntorno(\$value) [line 104]

Parámetros de la función:

- **\$value**

- **Access public**

void función Controlador_model::setNombre(\$value) [line 88]

Parámetros de la función:

- **\$value**

- **Access public**

void función Controlador_model::setOffset(\$value) [line 124]

Parámetros de la función:

- **\$value**

- **Access** public

void función Controlador_model::setRows(\$value) [*line 127*]

Parámetros de la función:

- **\$value**

- **Access** public

boolean función Controlador_model::update() [*line 345*]

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Entorno_model

[*line 11*]

Modelo Entorno

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Entorno_model::__construct() [line 148]

- **Access** public

boolean función Entorno_model::delete() [line 311]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Entorno_model::getAll() [line 164]

Devuelve todos los registros de la tabla entorno

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Entorno_model::getById(\$id) [line 186]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de un entorno por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Entorno_model::getByName(\$nombre) [line 208]

Parámetros de la función:

- *string* **\$nombre**

Devuelve la información de un entorno por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Entorno_model::getDescripcion() [line 78]

- **Access** public

void función Entorno_model::getDirectorio() [line 99]

- **Access** public

void función Entorno_model::getHostname() [line 92]

- **Access** public

void función Entorno_model::getId() [line 64]

- **Access** public

string función Entorno_model::getInfo() [line 142]

Devuelve la información del objeto en forma de string

- **Access** public

void función Entorno_model::getIp() [line 85]

- **Access** public

string/boolean función Entorno_model::getMsgError([\$value = 'usr'], \$value,) [line 118]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Entorno_model::getNombre() [line 71]

- **Access** public

Object función Entorno_model::getPermissionsByGroup(\$grupo) [line 230]

Parámetros de la función:

- *string* **\$grupo**

Obtiene los permisos asignados al grupo

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Entorno_model::insert() [*line 252*]

Inserta en la tabla entorno la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Entorno_model::setDescripcion(\$value) [*line 81*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Entorno_model::setDirectorio(\$value) [*line 102*]

Parámetros de la función:

- **\$value**

- **Access public**

void función Entorno_model::setHostname(\$value) [line 95]

Parámetros de la función:

- **\$value**

- **Access public**

void función Entorno_model::setId(\$value) [line 67]

Parámetros de la función:

- **\$value**

- **Access public**

void función Entorno_model::setIp(\$value) [line 88]

Parámetros de la función:

- **\$value**

- **Access public**

void función Entorno_model::setNombre(\$value) [line 74]

Parámetros de la función:

- **\$value**

- **Access** public

boolean función Entorno_model::update() [*line 281*]

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Errorlog_model

[*line 11*]

Modelo Errorlog

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Errorlog_model::__construct() [*line 76*]

- **Access** public

void|boolean función `Errorlog_model::addFilter($columna, $condicion, $valor)` [line 243]

Parámetros de la función:

- *string* **\$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, descripcion, id_controlador_accion)
- *string* **\$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **\$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access public**

array función `Errorlog_model::getAll([$offset = null], [$row_count = null])` [line 301]

Parámetros de la función:

- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Error en caso de existir filtros, estos son aplicados a la consulta

- **Access public**

object|boolean función `Errorlog_model::getById($id)` [line 196]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro del Error que tiene el ID especificado

- **Access** public

void función Errorlog_model::getDescripcion() [*line 129*]

- **Access** public

void función Errorlog_model::getFecha_hora() [*line 124*]

- **Access** public

void función Errorlog_model::getId() [*line 94*]

- **Access** public

void función Errorlog_model::getId_controlador_accion() [*line 114*]

- **Access** public

void función Errorlog_model::getId_usuario() [*line 104*]

- **Access** public

int/boolean función Errorlog_model::getNumRows() [*line 344*]

Obtiene el numero total de registros en la tabla Error en caso de existir filtros, estos son aplicados a la consulta

- **Access public**

void función Errorlog_model::insert(\$path, \$descripcion, \$id_controlador, \$id_accion) [line 158]

Parámetros de la función:

- *int* **\$id_controlador**
- *int* **\$id_accion**
- *string* **\$descripcion**
- **\$path**

Inserta en la base de datos la informacion del error

- **Static**
- **Access public**

void función Errorlog_model::resetFilter() [line 285]

Elimina todos los filtros registrados

- **Access public**

string función Errorlog_model::save(\$model, \$method, \$modelo) [line 383]

Parámetros de la función:

- *string* **\$modelo** Nombre del modelo que lanzó la excepción
- *string* **\$method** Contiene el nombre de la clase y metodo donde se originó el error o el mensaje de error para mostrar al usuario final
- **\$model**

Guardar el mensaje de error descriptivo en la base de datos,

si no puede insertar el registro a la base de datos, el mensaje de error se guarda en el directorio logs, devuelve el mensaje de error para el usuario final

- **Static**
- **Access** public

void función Errorlog_model::setDescription(\$descripcion) [line 134]

Parámetros de la función:

- **\$descripcion**

- **Access** public

void función Errorlog_model::setId(\$id) [line 99]

Parámetros de la función:

- **\$id**

- **Access** public

void función Errorlog_model::setId_accion(\$id_accion) [line 144]

Parámetros de la función:

- **\$id_accion**

- **Access public**

void función Errorlog_model::setId_controlador(\$id_controlador) [line 139]

Parámetros de la función:

- **\$id_controlador**

- **Access public**

void función Errorlog_model::setId_controlador_accion(\$id_controlador_accion) [line 119]

Parámetros de la función:

- **\$id_controlador_accion**

- **Access public**

void función Errorlog_model::setId_usuario(\$id_usuario) [line 109]

Parámetros de la función:

- **\$id_usuario**

- **Access public**

Clase Georeferencia_model

[line 11]

Modelo Georeferencia

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Georeferencia_model::__construct() [line 36]

- **Access** public

boolean|string función Georeferencia_model::getMsgError([\$type = 'usr']) [line 60]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Georeferencia_model::process() [line 82]

Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia

- **Access** public

Clase Grupo_model

[line 10]

Modelo Grupo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Grupo_model::__construct() [line 42]

- **Access** public

boolean función Grupo_model::delete() [line 265]

Elimina de la base de datos al grupo (id en propiedades)

- **Access** public

void/array función Grupo_model::getAll([\$keywords = "], [\$offset = null], [\$row_count = null]) [line 102]

Parámetros de la función:

- ***int \$offset*** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- ***int \$row_count*** Establece la cantidad de registros a devolver
- ***\$keywords***

Obtiene todos los grupos existentes

- **Access public**

void/object función Grupo_model::getById(\$id) [line 135]

Parámetros de la función:

- ***int \$id*** id del grupo

Obtiene el grupo solicitado

- **Access public**

void/object función Grupo_model::getByName(\$name) [line 177]

Parámetros de la función:

- ***string \$name*** nombre del grupo

Obtiene el grupo solicitado

- **Access public**

void función Grupo_model::getDescripcion() [*line 68*]

- **Access public**

void/object función Grupo_model::getEntornosById(\$id) [*line 155*]

Parámetros de la función:

- *int* **\$id** id del grupo

Obtiene el grupo solicitado con sus entornos vinculados

- **Access public**

void función Grupo_model::getId() [*line 49*]

- **Access public**

boolean función Grupo_model::getMsgError([\$value = 'usr']) [*line 85*]

Parámetros de la función:

- *string* **\$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access public**

void función Grupo_model::getNombre() [line 58]

- **Access public**

int función Grupo_model::getNumRows([\$keywords = ""]) [line 197]

Parámetros de la función:

- *boolean|string* **\$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de grupos

- **Access public**

boolean función Grupo_model::insert() [line 223]

Inserta en la base de datos los datos del grupo (datos en propiedades)

- **Access public**

void función Grupo_model::setDescription(\$descripcion) [line 73]

Parámetros de la función:

- **\$descripcion**

- **Access public**

void función Grupo_model::setId(\$value) [line 54]

Parámetros de la función:

- **\$value**

- **Access public**

void función Grupo_model::setNombre(\$nombre) [line 63]

Parámetros de la función:

- **\$nombre**

- **Access public**

boolean función Grupo_model::update() [line 244]

Actualiza en la base de datos los datos del grupo (datos en propiedades)

- **Access public**

Clase Hemoglobina_model

[line 11]

Modelo Hemoglobina

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Hemoglobina_model::__construct() [line 36]

- **Access** public

boolean/string función Hemoglobina_model::getMsgError([\$type = 'usr']) [line 60]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Hemoglobina_model::process() [line 82]

Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia

- **Access** public

Clase Menu_model

[line 11]

Modelo Menu

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Menu_model::__construct() [line 85]

- **Access** public

void/boolean función Menu_model::addFilter(\$columna, \$condicion, \$valor) [line 401]

Parámetros de la función:

- *string* **\$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros, id_controlador_accion)
- *string* **\$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **\$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

int función Menu_model::delete([\$id = null]) [line 287]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

int función Menu_model::deleteByFilter() [*line 326*]

Elimina el conjunto de registros que cumplen con el o los criterios de filtrado

- **Access** public

array función Menu_model::getAll([\$offset = null], [\$row_count = null]) [*line 469*]

Parámetros de la función:

- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Menu en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Menu_model::getAtributo() [*line 163*]

- **Access** public

object|boolean función Menu_model::getById(\$id) [line 358]

Parámetros de la función:

- **int \$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro de la menu que tiene el ID especificado

- **Access public**

boolean función Menu_model::getByPadre(\$padre) [line 566]

Parámetros de la función:

- **\$padre**

Obtiene todos los nodos hijos de un padre

- **Access public**

void función Menu_model::getId() [line 98]

- **Access public**

void función Menu_model::getId_controlador() [line 143]

- **Access public**

void función Menu_model::getId_padre() [*line 108*]

- **Access public**

void función Menu_model::getId_raiz() [*line 118*]

- **Access public**

boolean/string función Menu_model::getMsgError([\$type = 'usr']) [*line 179*]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access public**

void función Menu_model::getNombre() [*line 128*]

- **Access public**

int función Menu_model::getNumRows() [*line 511*]

Obtiene el numero total de registros en la tabla Menu en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Menu_model::getRuta() [*line 153*]

- **Access** public

boolean función Menu_model::hasChild(\$id) [*line 550*]
Parámetros de la función:

- **\$id**

Verifica si el nodo actual tiene hijos

- **Access** public

boolean función Menu_model::insert() [*line 200*]

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función Menu_model::resetFilter() [*line 453*]

Elimina todos los filtros registrados

- **Access** public

void función Menu_model::setAtributo(\$atributo) [line 158]

Parámetros de la función:

- **\$atributo**
- **Access public**

void función Menu_model::setId(\$id) [line 103]

Parámetros de la función:

- **\$id**
- **Access public**

void función Menu_model::setId_controlador(\$id_controlador) [line 138]

Parámetros de la función:

- **\$id_controlador**
- **Access public**

void función Menu_model::setId_padre(\$id_padre) [line 113]

Parámetros de la función:

- **\$id_padre**

- **Access public**

void función Menu_model::setId_raiz(\$id_raiz) [line 123]

Parámetros de la función:

- **\$id_raiz**

- **Access public**

void función Menu_model::setNombre(\$nombre) [line 133]

Parámetros de la función:

- **\$nombre**

- **Access public**

void función Menu_model::setRuta(\$ruta) [line 148]

Parámetros de la función:

- **\$ruta**

- **Access public**

boolean función Menu_model::update([\$id = null]) [line 244]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Permiso_model

[line 10]

Modelo Permiso

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Permiso_model::__construct() [line 47]

- **Access** public

boolean función Permiso_model::deletePermissions(\$entorno, \$grupo) [line 156]

Parámetros de la función:

- *int* **\$entorno** id de entorno
- *int* **\$grupo** id del grupo

Elimina de la base de datos los permisos del entorno y grupo recibidos

- **Access** public

void función Permiso_model::getFecha() [*line 73*]

- **Access** public

void función Permiso_model::getId() [*line 54*]

- **Access** public

void función Permiso_model::getIdControladorAccion() [*line 83*]

- **Access** public

void función Permiso_model::getIdGrupo() [*line 63*]

- **Access** public

boolean función Permiso_model::getMsgError([\$value = 'usr']) [*line 100*]

Parámetros de la función:

- **\$value \$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void/object función Permiso_model::getPermission(\$id) [line 114]

Parámetros de la función:

- *int* **\$id** id del controlador_x_accion

Obtiene el permiso solicitado

- **Access** public

boolean función Permiso_model::insertBatch(\$data) [line 135]

Parámetros de la función:

- *array* **\$data** object \$data array con los permisos a insertar

Inserta en la base de datos el arreglo de permisos recibido

- **Access** public

void función Permiso_model::setFecha(\$fecha) [line 78]

Parámetros de la función:

- **\$fecha**

- **Access public**

void función Permiso_model::setId(\$value) [line 59]

Parámetros de la función:

- **\$value**

- **Access public**

void función Permiso_model::setIdControladorAccion(\$id_controlador_accion) [line 88]

Parámetros de la función:

- **\$id_controlador_accion**

- **Access public**

void función Permiso_model::setIdGrupo(\$id_grupo) [line 68]

Parámetros de la función:

- **\$id_grupo**

- **Access public**

Clase Poblacion_model

[line 11]

Modelo Poblacion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Poblacion_model::__construct() [line 36]

- **Access** public

boolean/string función Poblacion_model::getMsgError([\$type = 'usr']) [line 60]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Poblacion_model::process() [line 82]

Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion

- **Access** public

Clase Raiz_model

[line 11]

Modelo Raiz

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función Raiz_model::__construct() [line 84]

- **Access** public

boolean función Raiz_model::delete() [line 216]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Boolean función Raiz_model::ExistInArbol(\$id) [line 122]

Parámetros de la función:

- *int* \$id

Revisa si la raiz pasada como parametro existe en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Raiz_model::getAll() [line 100]

Devuelve todos los registros de la tabla raiz

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Raiz_model::getById(\$id) [line 144]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de una raiz por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Raiz_model::getDescripcion() [line 48]

- **Access** public

void función Raiz_model::getId() [line 40]

- **Access** public

string/boolean función Raiz_model::getMsgError([\$value = 'usr'], \$value,) [line 69]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función Raiz_model::insert() [line 165]

Inserta en la tabla raiz, la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Raiz_model::setDescripcion(\$value) [line 52]

Parámetros de la función:

- **\$value**

- **Access** public

void función Raiz_model::setId(\$value) *[line 44]*

Parámetros de la función:

- **\$value**

- **Access** public

boolean función Raiz_model::update() *[line 190]*

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ReglaVacuna_model

[line 11]

Modelo ReglaVacuna

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Giovanni

Constructor *void* función ReglaVacuna_model::__construct() [line 251]

- **Access** public

boolean función ReglaVacuna_model::delete() [line 414]

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::getAlergias() [line 209]

- **Access** public

ArrayObject función ReglaVacuna_model::getAll() [line 267]

Devuelve todos los registros de la tabla regla_vacuna

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ReglaVacuna_model::getById(\$id) [line 288]

Parámetros de la función:

- *int* **\$id** ID (Llave primaria)

Devuelve la información de una regla de vacuna por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::getDiaFinNacido() [*line 146*]

- **Access** public

void función ReglaVacuna_model::getDiaFinPrevía() [*line 160*]

- **Access** public

void función ReglaVacuna_model::getDiaInicioNacido() [*line 139*]

- **Access** public

void función ReglaVacuna_model::getDiaInicioPrevía() [*line 153*]

- **Access** public

void función ReglaVacuna_model::getDosis() [*line 174*]

- **Access** public

void función ReglaVacuna_model::getEsqComp() [*line 195*]

- **Access** public

void función ReglaVacuna_model::getForzarAplicacion() [*line 216*]

- **Access** public

void función ReglaVacuna_model::getId() [*line 118*]

- **Access** public

void función ReglaVacuna_model::getIdVacuna() [*line 125*]

- **Access** public

void función ReglaVacuna_model::getIdVacunaPrevio() [*line 132*]

- **Access** public

void función ReglaVacuna_model::getIdViaVacuna() [*line 167*]

- **Access** public

string/boolean función ReglaVacuna_model::getMsgError([\$value = 'usr'], \$value,) [*line 236*]

Parámetros de la función:

- *string* **\$value**, default 'usr' (Tipo mensaje)
- **\$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función ReglaVacuna_model::getObservacionRegion() [*line 188*]

- **Access** public

void función ReglaVacuna_model::getOrdenEsqComp() [*line 202*]

- **Access** public

void función ReglaVacuna_model::getRegion() [*line 181*]

- **Access** public

int función ReglaVacuna_model::insert() [*line 327*]

Inserta en la tabla regla_vacuna la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::setAlergias(\$value) [*line 212*]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setDiaFinNacido(\$value) [line 149]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setDiaFinPrevía(\$value) [line 163]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setDiaInicioNacido(\$value) [line 142]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setDiaInicioPrevia(\$value) [line 156]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setDosis(\$value) [line 177]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setEsqComp(\$value) [line 198]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setForzarAplicacion(\$value) [line 219]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setId(\$value) [line 121]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setIdVacuna(\$value) [line 128]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setIdVacunaPrevia(\$value) [line 135]

Parámetros de la función:

- **\$value**

- **Access public**

void función ReglaVacuna_model::setIdViaVacuna(\$value) [line 170]

Parámetros de la función:

- **\$value**
- **Access public**

void función ReglaVacuna_model::setObservacionRegion(\$value) [line 191]

Parámetros de la función:

- **\$value**
- **Access public**

void función ReglaVacuna_model::setOrdenEsqComp(\$value) [line 205]

Parámetros de la función:

- **\$value**
- **Access public**

void función ReglaVacuna_model::setRegion(\$value) [line 184]

Parámetros de la función:

- **\$value**

- **Access** public

boolean función ReglaVacuna_model::update() [*line 370*]

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Usuario_model

[*line 10*]

Modelo Usuario

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Usuario_model::__construct() [*line 81*]

- **Access** public

null|object función Usuario_model::authenticate(\$username, \$password) [*line 423*]

Parámetros de la función:

- *string* **\$username** nombre de usuario
- *string* **\$password** clave

Valida las credenciales recibidas

- **Access** public

void función Usuario_model::checkCredentials(\$path, \$pathURL, \$group_id) [line 444]

Parámetros de la función:

- *string* **\$path** entorno::controlador::accion
- *int* **\$group_id** id del grupo a validar permisos
- **\$pathURL**

Verifica que el usuario haya iniciado sesión y además tenga permiso en la acción recibida

- **Static**
- **Access** public

void función Usuario_model::check_data(\$usuario, \$correo) [line 477]

Parámetros de la función:

- **\$usuario**
- **\$correo**

- **Access** public

void función Usuario_model::check_token(\$correo) [line 490]

Parámetros de la función:

- **\$correo**

- **Access public**

boolean función Usuario_model::delete() [line 404]

Elimina de la base de datos al usuario (id en propiedades)

- **Access public**

void/object función Usuario_model::getActivesByGroup(\$group_id, \$id, \$viewMode) [line 283]

Parámetros de la función:

- *int* **\$id** id del usuario
- *boolean* **\$viewMode** true obtiene el modo visualización, false o null obtiene el registro normal
- **\$group_id**

Obtiene los usuarios activos del grupo solicitado

- **Access public**

void función Usuario_model::getActivo() [line 161]

- **Access public**

void función Usuario_model::getApellidoMaterno() [line 141]

- **Access public**

void función Usuario_model::getApellidoPaterno() [line 131]

- **Access public**

void/object función Usuario_model::getById(\$id, [\$viewMode = FALSE]) [line 253]

Parámetros de la función:

- *int* **\$id** id del usuario
- *boolean* **\$viewMode** true obtiene el modo visualización, false o null obtiene el registro normal

Obtiene el usuario solicitado, se puede obtener el registro normal o personalizado para visualización (descripciones en tablas vinculadas)

- **Access public**

void/object función Usuario_model::getByUsername(\$username) [line 303]

Parámetros de la función:

- *string* **\$username** nombre de usuario

Obtiene el usuario solicitado

- **Access public**

void función Usuario_model::getClave() [*line 111*]

- **Access** public

void función Usuario_model::getCorreo() [*line 151*]

- **Access** public

void función Usuario_model::getgrupo(\$grupo) [*line 532*]

Parámetros de la función:

- **\$grupo**

- **Access** public

void función Usuario_model::getId() [*line 92*]

- **Access** public

void función Usuario_model::getIdGrupo() [*line 171*]

- **Access** public

boolean función Usuario_model::getMsgError([\$value = 'usr']) [*line 188*]

Parámetros de la función:

- *string* **\$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void función Usuario_model::getNombre() [*line 121*]

- **Access** public

void función Usuario_model::getNombreUsuario() [*line 101*]

- **Access** public

int función Usuario_model::getNumRows([\$keywords = ""]) [*line 323*]

Parámetros de la función:

- *boolean|string* **\$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de usuarios

- **Access** public

void/array función Usuario_model::getOnlyActives([\$keywords = "], [\$onlyActives = TRUE], [\$offset = null], [\$row_count = null]) [*line 207*]

Parámetros de la función:

- *boolean/string* **\$keywords** false no hay texto a buscar|string con texto a buscar
- *boolean* **\$onlyActives** true obtiene solo usuarios activos, false o null obtiene todos los usuarios
- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todos los usuarios existentes, se puede filtrar por: texto a buscar o solo activos si se desea

- **Access public**

void función Usuario_model::getuser(\$id) [line 519]

Parámetros de la función:

- **\$id**

- **Access public**

void función Usuario_model::get_grupo_entorno(\$nombre) [line 583]

Parámetros de la función:

- **\$nombre**

- **Access public**

```
void función Usuario_model::get_permiso_entorno($nombre) [line 560]
```

Parámetros de la función:

- **\$nombre**

- **Access** public

```
void función Usuario_model::get_usuario_entorno($nombre, [$inusuario = ""]) [line 607]
```

Parámetros de la función:

- **\$nombre**
- **\$inusuario**

- **Access** public

```
boolean función Usuario_model::insert() [line 351]
```

Inserta en la base de datos los datos del usuario (datos en propiedades)

- **Access public**

```
void función Usuario_model::setActivo($activo) [line 166]
```

Parámetros de la función:

- **\$activo**

- **Access public**

void función Usuario_model::setApellidoMaterno(\$apellido_materno) [line 146]

Parámetros de la función:

- **\$apellido_materno**

- **Access public**

void función Usuario_model::setApellidoPaterno(\$apellido_paterno) [line 136]

Parámetros de la función:

- **\$apellido_paterno**

- **Access public**

void función Usuario_model::setClave(\$clave) [line 116]

Parámetros de la función:

- **\$clave**

- **Access public**

void función Usuario_model::setCorreo(\$correo) [line 156]

Parámetros de la función:

- **\$correo**

- **Access public**

void función Usuario_model::setId(\$value) [line 97]

Parámetros de la función:

- **\$value**

- **Access public**

void función Usuario_model::setIdGrupo(\$id_grupo) [line 176]

Parámetros de la función:

- **\$id_grupo**

- **Access public**

void función Usuario_model::setNombre(\$nombre) [line 126]

Parámetros de la función:

- **\$nombre**

- **Access public**

void función Usuario_model::setNombreUsuario(\$nombre_usuario) [line 106]

Parámetros de la función:

- **\$nombre_usuario**

- **Access public**

boolean función Usuario_model::update() [line 378]

Actualiza en la base de datos los datos del usuario (datos en propiedades)

- **Access public**

void función Usuario_model::update_pass(\$pass, \$id) [line 503]

Parámetros de la función:

- **\$pass**
- **\$id**

- **Access public**

void función Usuario_model::update_user(\$id, \$clave, \$correo) [line 545]

Parámetros de la función:

- **\$id**
- **\$clave**
- **\$correo**

- **Access** public

Paquete default Clases

Clase Index *[line 3]*

- **Package** default

Constructor *void* función Index::__construct() *[line 5]*

- **Access** public

void función Index::index() *[line 10]*

- **Access** public

Paquete TES Clases

Clase Enrolamiento

[line 10]

Controlador de enrolamiento

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Enrolamiento::__construct() [line 13]

- **Access** public

echo función Enrolamiento::addForm() [line 907]

Pase de parametros para la insercion o actualizacion

- **Access** public

echo función Enrolamiento::autocomplete() [line 470]

Crea el autocomplete para facilitar la busqueda de un tutor

- **Access public**

echo función Enrolamiento::brothers_search(\$tutor) [line 1088]

Parámetros de la función:

- *string* **\$tutor** id del tutor compartido

Este metodo extrae la informacion de las personas con las que se comparte el mismo tutor si se selecciona una de estas importa los datos para el apartado direccion

- **Access public**

echo función Enrolamiento::brother_found(\$id) [line 1071]

Parámetros de la función:

- *string* **\$id** id de la persona

Este metodo verifica si un paciente comparte un mismo tutor

- **Access public**

echo función Enrolamiento::catalog_check(\$catalog, \$tipo, [\$col = 1], [\$sel = ""], [\$orden = ""]) [line 417]

Parámetros de la función:

- *string* **\$catalog** tabla de donde se extrae la informacion
- *string* **\$tipo** tipo de control radio o check
- *int* **\$col** numero de columnas en la tabla

- *string* **\$sel** identifica si un valor ya esta seleccionado
- *string* **\$orden** columna para hacer el ordenamiento

Crea un grupo de radio o check con la informacion de los catalogos

- **Access** public

echo función Enrolamiento::catalog_select(\$catalog, [\$sel = ""], [\$orden = ""], [\$campo = ""], [\$valor = ""]) [line 338]

Parámetros de la función:

- *string* **\$catalog** tabla de donde se extrae la informacion
- *string* **\$sel** identifica si un valor ya esta seleccionado
- *string* **\$orden** columna para hacer el ordenamiento
- *string* **\$campo** campo de la tabla para hacer el where
- *string* **\$valor** valor a comparar en el where

Genera los options de un campo tipo select

- **Access** public

echo función Enrolamiento::categoriacie10_select() [line 1333]

Genera los options de un campo tipo select para las categorías de CIE10

- **Access** public

bool función Enrolamiento::chechar_session() [line 1310]

Revisa si la sesión está activa

- **Access public**

echo función Enrolamiento::cie10_select(\$categoria) [line 1360]

Parámetros de la función:

- *string* **\$categoria** Categoría de la CIE10

Genera los options de un campo tipo select para los CIE10 correspondientes a una categoría

- **Access public**

echo función Enrolamiento::comparar_view(\$id, [\$prod1 = ""], [\$prod2 = ""], [\$prod3 = ""]) [line 1241]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- **\$prod1**
- **\$prod2**
- **\$prod3**

Crea la pagina para ver la infromacion de la persona y comprararla con la persona capturada

- **Access public**

echo función Enrolamiento::data_tutor(\$curp) [line 492]

Parámetros de la función:

- *string* **\$curp** curp del tutor

Obtiene inofrmacion del tutor

- **Access** public

echo función Enrolamiento::file_to_card(\$id, [\$tipo = ""]) [line 538]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- **\$tipo**

crea un archivo descargable el cual se necesita para el envio por nfc a la tarjeta del paciente

- **Access** public

echo función Enrolamiento::ifCarpExists(\$curp) [line 985]

Parámetros de la función:

- *string* **\$curp** curp de la persona

Valida que la curp del paciente no exista

- **Access** public

echo función Enrolamiento::ifCarpTExists(\$curp) [line 1029]

Parámetros de la función:

- **string \$curp** curp de la persona

valida que la curp del tutor no exista

- **Access public**

echo función Enrolamiento::index([\$pag = 0], [\$id = ""], [\$array = ""]) [line 38]

Parámetros de la función:

- **string \$pag** numero de pagina para la posicion
- **string \$id** id de una persona
- **\$array**

Este es el metodo por default, obtiene el listado de las personas
se recibe el parametro \$pag de tipo int que representa la paginacion

- **Access public**

echo función Enrolamiento::insert() [line 766]

prepara los datos para insertarlos

- **Access public**

json(\$porcentaje_similitud,\$persona) función Enrolamiento::paciente_similar(\$nombre, \$paterno, \$materno, \$curp, \$nacimiento, \$lugar, [\$calle = ""], [\$referencia = ""], [\$colonia = ""], [\$curpT = ""], \$cp, \$numero) [line 1190]

Parámetros de la función:

- *string* **\$nombre** nombre del paciente que se esta capturando
- *string* **\$paterno** apellido paterno del paciente
- *string* **\$materno** apellido materno del paciente
- *string* **\$curp** curp del paciente
- *string* **\$nacimiento** fecha de nacimiento del paciente
- *string* **\$calle** calle del domicilio del paciente
- *string* **\$referencia** referencia del domicilio del paciente
- *string* **\$colonia** colonia del paciente
- *string* **\$cp** cp de la colonia donde vive el paciente
- *string* **\$numero** numero de la vivienda
- **\$lugar**
- **\$curpT**

Comprueba la similitud de un paciente que se este capturando con los que ya existe en la base de datos, esto con la finalidad de disminuir datos repetidos

- **Access public**

echo función Enrolamiento::print_card(\$id) [line 87]

Parámetros de la función:

- *string* **\$id** identificador de la persona

Este metodo estrae la informacion del paciente que sera impreso en la tarjeta

- **Access public**

Object función Enrolamiento::searchgeb([\$idasulocalidad = ""], [\$like = ""]) [line 1158]

Parámetros de la función:

- *int* **\$idasulocalidad** Id de la localidad en el ASU
- **\$like**

Regresa un objeto JSON con la lista de agebs disponibles en la localidad Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

Object función Enrolamiento::searchum(\$idasulocalidad, \$ageb) [line 1130]

Parámetros de la función:

- *int* **\$idasulocalidad** Id de la localidad en el ASU
- *string* **\$ageb** Numero de ageb

Busca dentro del catalogo asu_ageb la unidad médica de acuerdo a la localidad y ageb Solo se permite su acceso por medio de peticiones AJAX

- **Access public**

echo función Enrolamiento::tratamiento_select([\$campo = ""], [\$valor = ""], [\$sel = ""], [\$orden = ""]) [line 371]

Parámetros de la función:

- *string* **\$campo** campo de la tabla para hacer el where
- *string* **\$valor** valor a comparar en el where
- *string* **\$sel** identifica si un valor ya esta seleccionado
- *string* **\$orden** columna para hacer el ordenamiento

Genera los options de un campo tipo select para los tratamientos de consultas

- **Access public**

echo función Enrolamiento::update(\$id) [line 194]

Parámetros de la función:

- *string \$id* identificador de la persona

Crea el formulario para editar la informacion de la persona

- **Access public**

echo función Enrolamiento::update_card(\$persona, \$impreso, [\$fecha = ""], [\$archivo = ""], [\$entorno = '4']) [line 739]

Parámetros de la función:

- *string \$persona* id de la persona
- *boolean \$impreso* identifica si el proceso de impresion fue correcto o no
- *int \$fecha* fecha del evento
- *string \$archivo* archivo generado
- *string \$entorno* tipo de entorno

Este metodo actualiza el estado del archivo descargado si fue escrito correctamente o no en la tarjeta

- **Access public**

echo función Enrolamiento::vacunacion(\$fecha, [\$id = ""]) [line 1291]

Parámetros de la función:

- *date \$fecha*

- *string* \$id

Obtiene el historial de vacunacion de un paciente cuyo id es pasado por parametro

- **Access** public

echo función Enrolamiento::validarForm([\$op = ""]) [line 833]

Parámetros de la función:

- *string* \$op bandera que identifica si una seccion entra en validacion

valida los datos de entrada en el formulario

- **Access** public

echo función Enrolamiento::validarisum(\$id) [line 1107]

Parámetros de la función:

- *string* \$id id de arbol de segmentacion

valida que el nodo seleccionado en el arbol sea una unidad medica

- **Access** public

echo función Enrolamiento::validate_card(\$persona, \$archivo) [line 755]

Parámetros de la función:

- *string* **\$persona** id de la persona
- *string* **\$archivo** archivo generado

valida que un archivo sea valido para enviar a la tarjeta por nfc

- **Access** public

echo función Enrolamiento::view(\$id) [*line 134*]

Parámetros de la función:

- *string* **\$id** identificador de la persona

Crea la pagina para ver la infromacion de la persona

- **Access** public

Clase Notificacion

[*line 10*]

Controlador Notificación

- **Package** TES
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Notificacion::__construct() [line 12]

- **Access public**

void función Notificacion::delete(\$id) [line 255]

Parámetros de la función:

- *int* **\$id** id de notificación a eliminar

Solicita la eliminación de la notificación recibida

- **Access public**

void función Notificacion::index([\$pag = 0]) [line 34]

Parámetros de la función:

- *int* **\$pag** número de página a visualizar (paginación)

- 1) Visualiza las notificaciones existentes para su interacción CRUD
- 2) En caso de detectar un texto a buscar se filtran las notificaciones existentes acorde a la búsqueda

- **Access public**

void función Notificacion::insert() [line 139]

- 1) Prepara el formulario para la inserción de una notificación nueva
- 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access public**

void función Notificacion::update(\$id) [line 197]

Parámetros de la función:

- ***int \$id*** id de la notificación a modificar

1) Prepara el formulario para la modificación de una notificación existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access public**

void función Notificacion::view(\$id) [line 103]

Parámetros de la función:

- ***int \$id*** id de notificación a visualizar

Visualiza los datos de la notificación recibida

- **Access public**

Clase Reporteador
[line 10]

Controlador Reporteador

- **Package** TES
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Reporteador::__construct() [*line 12*]

- **Access** public

void función Reporteador::index() [*line 32*]

Visualiza los reportes existentes

- **Access** public

void función Reporteador::view(\$op, \$title, \$nivel, \$id, [\$fecha = ""]) [*line 73*]

Parámetros de la función:

- *int* \$op
- *string* \$title
- *int* \$nivel
- *int* \$id
- *string* \$fecha

Renderiza la vista del reporte

- **Access** public

Clase Reporte_sincronizacion

[line 10]

Controller Usuario

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Reporte_sincronizacion::__construct() [line 13]

- **Access** public

void función Reporte_sincronizacion::index() [line 35]

Este es el metodo por default, crea el listado del reporte de sincronizacion

- **Access** public

void función Reporte_sincronizacion::lote() [line 175]

Genera el ítem del reporte de lote de vacunacion

- **Access** public

void función Reporte_sincronizacion::lote_view(\$lote, \$title, \$op, [\$lugar = "Chiapas"]) [line 344]

Parámetros de la función:

- *string* **\$lote** Valor del lote del que se esta generando el reporte
- *string* **\$title** Especifica el titulo a mostrar en el reporte
- *string* **\$op** Especifica el reporte a mostrar
- *string* **\$lugar** Especifica el lugar donde se centrara el mapa

Muestra detallada por cada list del reporte de lote de vacunacion

- **Access public**

void función Reporte_sincronizacion::view(\$op, \$title) [line 103]

Parámetros de la función:

- *string* **\$op** Especifica el reporte a mostrar
- *string* **\$title** Especifica el titulo a mostrar en el reporte

Muestra detallada por cada list del reporte de sincronizacion

- **Access public**

Clase Semana_nacional

[line 11]

Controlador Semana Nacional

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función `Semana_nacional::__construct()` [*line 13*]

- **Access** public

void función `Semana_nacional::delete($id)` [*line 245*]
Parámetros de la función:

- *int* **\$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

json función `Semana_nacional::getAll()` [*line 278*]

Devuelve un json con todos los registros de semanas nacional

- **Access** public

void función `Semana_nacional::index([$pag = 0])` [*line 35*]

Parámetros de la función:

- *int* **\$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de semanas nacional, con su correspondiente paginación

permite eliminar un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico

- **Access public**

void función Semana_nacional::insert() [line 92]

Muestra el formulario para crear un nuevo registro en la semana_nacional, las variables se obtienen por el metodo POST

- **Access public**

void función Semana_nacional::update(\$id) [line 151]

Parámetros de la función:

- *int \$id* ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access public**

void función Semana_nacional::view(\$id) [line 205]

Parámetros de la función:

- *int \$id* ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Servicios

[line 10]

Controlador Servicios

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Servicios::__construct() *[line 11]*

- **Access** public

echo función Servicios::actualiza_estado_tableta(\$id_sesion, [\$id_tes_estado_tableta = ""], [\$version = ""], \$id_session) *[line 1025]*

Parámetros de la función:

- *int* **\$id_session** Representa la session activa para la petición
- *json* **\$id_tes_estado_tableta** Representa el status que tomara la tableta
- *int* **\$version** Representa la version de la apk instalada en la tableta
- **\$id_sesion**

Actualiza el estatus de la tableta

- **Access public**

echo función Servicios::catalogos_relevantes([\$sf = ""]) [line 1102]

Parámetros de la función:

- *string \$sf* bandera que activa el filtro de fechas segun el tipo de sincronizacion

Genera los catalogos relevantes por entorno

- **Access public**

echo función Servicios::esquema_incompleto(\$id_persona, \$fecha, \$vacunas) [line 1053]

Parámetros de la función:

- *int \$id_persona* Representa la persona a la que se le calculara su esquema
- *string \$fecha* Representa la fecha de nacimiento de la persona
- *array \$vacunas* Representa las vacunas aplicadas a la persona

Genera los esquemas incompletos de las personas que correspondan a la unidad medica de la tableta

- **Access public**

void función Servicios::is_step_0(\$id_accion, [\$id_tab = null], [\$id_sesion = null], [\$id_version = null], [\$datos = null], \$id_session, \$version_apk) [line 79]

Parámetros de la función:

- *int \$id_accion* Representa el tipo de accion que se ejecutara
- *int \$id_tab* Representa a la MAC de la tableta
- *int \$id_session* Representa la session activa para la peticion
- *int \$version_apk* Representa la version de la apk instalada en la tableta

- **json \$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor
- **\$id_sesion**
- **\$id_version**

Paso 0 se procesa las peticiones segun la accion:

Si la acci?n es 1: Valida la disponibilidad del dispositivo especificado y genera una session que se mantiene activa en toda la sincronizacion Si la acci?n es 2: Regresa la informacion de todos los catalogos Si la acci?n es 3: Recibe un mensaje si es ok actualiza el estado de la tableta si es error se crea un archivo log con la descripcion Si la acci?n es 4: Regresa la informacion de la persona que pertenescan a la unidad medica de la tableta Si la acci?n es 5: Recibe la informacion que envia la tableta y la almacena en sus respectivas tablas Si la acci?n es 6: Regresa la informacion de los catalogos y personas que se actualizaron o agregaron despues de la ultima sincronizacion de la tableta

- **Access public**

session función Servicios::is_step_1(\$id_tab, \$id_version, \$version_apk) [line 128]

Parámetros de la función:

- **int \$id_tab** Representa a la MAC de la tableta
- **int \$version_apk** Representa la version de la apk instalada en la tableta
- **\$id_version**

valida que la tableta este asignada a una unidad medica y que tenga un status valido para la sincronizacion

recibe parametros por POST

- **Access public**

echo función Servicios::is_step_2(\$id_sesion, [\$si = ""], [\$sf = ""], \$id_session) [line 207]

Parámetros de la función:

- *int* **\$id_session** Representa la session activa para la peticion
- *int* **\$si** Bandera que especifica si este paso es llamado por otro paso
- *json* **\$sf** Bandera que especifica el comportamiento del armado del json
- **\$id_sesion**

Valida que la session este activa, genera los catalogos a enviar en la sincronizacion por primera vez

- **Access public**

void función Servicios::is_step_3(\$id_sesion, \$datos, \$id_session) [line 466]

Parámetros de la función:

- *int* **\$id_session** Representa la session activa para la peticion
- *json* **\$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor
- **\$id_sesion**

Guarda en la base de datos el estado de la sincronizacion

- **Access public**

echo función Servicios::is_step_4(\$id_sesion, \$id_session) [line 502]

Parámetros de la función:

- *int* **\$id_session** Representa la session activa para la peticion
- **\$id_sesion**

Prepara la informacion de perssonas con su catalogos transaccionales de cada una

- **Access public**

void función Servicios::prueba2(\$id_accion, [\$id_tab = null], [\$id_sesion = null], [\$version = null]) *[line 1119]*

Parámetros de la función:

- **\$id_accion**
- **\$id_tab**
- **\$id_sesion**
- **\$version**

- **Access public**

echo función Servicios::ss_step_5(\$id_sesion, \$datos, \$id_session) *[line 689]*

Parámetros de la función:

- **int \$id_session** Representa la session activa para la petición
- **json \$datos** Representa el json que contiene el contenido para la comunicación tableta - servidor
- **\$id_sesion**

Recibe los datos que genero la tableta para ser almacenados en la base de datos del servidor

- **Access public**

void función Servicios::ss_step_6(\$id_sesion, \$id_session) *[line 835]*

Parámetros de la función:

- **int \$id_session** Representa la session activa para la petición
- **\$id_sesion**

prepara los datos para la sincronizacion secuencia, envia unicamente aquellos datos modificados despues de la ultima sincronizacion de la tableta

- **Access** public

void función Servicios::Synchronization(\$id_accion, \$id_tab, \$id_session, \$version_apk, \$datos) *[line 43]*

Parámetros de la función:

- *int* **\$id_accion** Representa el tipo de accion que se ejecutara
- *int* **\$id_tab** Representa a la MAC de la tableta
- *int* **\$id_session** Representa la session activa para la peticion
- *int* **\$version_apk** Representa la version de la apk instalada en la tableta
- *json* **\$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor

Metodo principal al que se le hacen las peticiones y es el que se encarga de distribuir la informacion

recibe parametros por POST

- **Access** public

Clase Tableta

[line 11]

Controlador Tableta

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Tableta::__construct() [*line 13*]

- **Access** public

void función Tableta::delete(\$id) [*line 340*]

Parámetros de la función:

- *int* **\$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

void función Tableta::index([\$pag = 0]) [*line 35*]

Parámetros de la función:

- *int* **\$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de tabletas, con su correspondiente paginación permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico

- **Access** public

void función `Tableta::insert()` [*line 163*]

Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST

- **Access public**

redirect función `Tableta::setUM($id)` [*line 461*]

Parámetros de la función:

- *int* **\$id**

Asignar unidad medica y tipo de censo

- **Access public**

void función `Tableta::update($id)` [*line 220*]

Parámetros de la función:

- *int* **\$id** ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access public**

redirect función `Tableta::uploadFile()` [*line 393*]

Registra tabletas desde un archivo csv

- **Access** public

void función Tableta::view(\$id) [line 285]

Parámetros de la función:

- *int* **\$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

boolean función Tableta::_validateMac(\$mac) [line 373]

Parámetros de la función:

- *type* **\$mac**

Valida si existe una MAC en la base de datos

- **Access** public

Clase Usuario_tableta
[line 11]

Controlador Usuario_tableta

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Usuario_tableta::__construct() [line 13]

- **Access** public

void función Usuario_tableta::delete(\$id_usuario, \$id_tableta, \$\$id_usuario) [line 143]

Parámetros de la función:

- *int* **\$\$id_usuario** ID del usuario a eliminar
- *int* **\$id_tableta** ID de la tableta que tiene asignada el usuario especificado
- **\$id_usuario**

Eliminar un usuario de una tableta especifica

- **Access** public

void función Usuario_tableta::index(\$idTableta) [line 36]

Parámetros de la función:

- *int* **\$idTableta** ID de la Tableta

Lista todos los registros de usuarios correspondientes a una tableta en específico permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico

- **Access** public

void función Usuario_tableta::insert(\$id_tableta) *[line 102]*

Parámetros de la función:

- **\$id_tableta**

Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST

- **Access** public

Clase Enrolamiento_model

[line 10]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Eliecer

Constructor *void* función Enrolamiento_model::__construct() *[line 118]*

- **Access** public

result() función Enrolamiento_model::autocomplete_tutor(\$keywords) [line 2297]

Parámetros de la función:

- *string* **\$keywords** palabras claves para hacer el filtro

obtiene informacion del tutor para genberar el autocomplete

- **Access** public

result() función Enrolamiento_model::cns_insert(\$tabla, \$array) [line 727]

Parámetros de la función:

- *string* **\$tabla** Nombre de la tabla a la que se afectara
- *array* **\$array** Datos que se guardaran en la tabla

Hace insert de las tablas **cns_control_x** que se reciben en la sincronizacion secuencial

- **Access** public

result() función Enrolamiento_model::cns_update(\$tabla, \$array, \$id, [\$campo = ""], [\$valor = ""], [\$campo2 = ""], [\$valor2 = ""]) [line 751]

Parámetros de la función:

- *string* **\$tabla** Nombre de la tabla afectada
- *array* **\$array** Datos a actualizar
- *string* **\$id** identificador para el where
- *string* **\$campo** campo si se necesitarar un segundo where
- *string* **\$valor** valor del campo a comparar
- **\$campo2**
- **\$valor2**

Actualiza la tabla especificada

- **Access public**

void función Enrolamiento_model::cns_update_visita(\$id) [line 767]

Parámetros de la función:

- **\$id**

- **Access public**

result() función Enrolamiento_model::data_tutor(\$curp) [line 2275]

Parámetros de la función:

- *string* **\$curp** Curp del tutor

obtiene informacion del tutor

- **Access public**

result() función Enrolamiento_model::entorno_x_persona(\$entorno, \$persona, \$fecha, \$archivo, \$impreso) [line 1110]

Parámetros de la función:

- *string* **\$entorno** id del entorno
- *string* **\$persona** id de la persona a la que se le asigna una tarjeta
- *string* **\$fecha** fecha en que se genera el evento

- *string* **\$archivo** nombre del archivo que se genero
- *booleana* **\$impreso** determina si el archivo fue escrita en la tarjeta o no

Este metodo actualiza o inserta los datos que permiten el envio de la informacion a la tarjeta por nfc

- **Access public**

void función Enrolamiento_model::getaccion_nutricional() [line 548]

- **Access public**

void función Enrolamiento_model::getafiliacion() [line 468]

- **Access public**

result() función Enrolamiento_model::getAfiliaciones([\$id = "], [\$order = ""]) [line 1830]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *strin* **\$order** nombre del campo para hacer el order by

Obtiene las afiliaciones asociadas a una persona

- **Access public**

void función Enrolamiento_model::getageb() [line 437]

- **Access** public

result() función Enrolamiento_model::getAlergia([\$id = "], [\$order = "]) [line 1799]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *strin* **\$order** nombre del campo para hacer el order by

Obtiene las alergias asociadas a una persona

- **Access** public

void función Enrolamiento_model::getalergias() [line 478]

- **Access** public

void función Enrolamiento_model::getaltura() [line 578]

- **Access** public

result() función Enrolamiento_model::getByCurp(\$curp, \$tabla, \$id) [line 2328]

Parámetros de la función:

- *string* **\$curp** Curp a validar
- *strin* **\$tabla** Tabla en la que se debe hacer la validacion
- *string* **\$id** id de la persona o tutor para excluirse

valida que no se repita la curp en personas y tutor

- **Access** public

result() función Enrolamiento_model::getById(\$id) [line 1739]

Parámetros de la función:

- *string* **\$id** identificado de la persona

Obtiene la informacion de la persona

- **Access** public

void función Enrolamiento_model::getcalle() [line 387]

- **Access** public

object|boolean función Enrolamiento_model::getCategoriaCIE10() [line 2717]

Obtiene el listado de categorias de CIE10

- **Access** public

void función Enrolamiento_model::getcelular() [line 637]

- **Access** public

void función Enrolamiento_model::getcelularT() [*line 337*]

- **Access public**

object|boolean función Enrolamiento_model::getCIE10(\$categoria) [*line 2746*]

Parámetros de la función:

- **\$categoria**

Obtiene el listado de CIE10 correspondientes a una CIE10

- **Access public**

void función Enrolamiento_model::getcodigo_barras() [*line 508*]

- **Access public**

void función Enrolamiento_model::getcolonia() [*line 397*]

- **Access public**

void función Enrolamiento_model::getcompania() [*line 627*]

- **Access public**

void función Enrolamiento_model::getcompaniaT() [*line 327*]

- **Access** public

void función Enrolamiento_model::getconsulta() [*line 518*]

- **Access** public

object|boolean función Enrolamiento_model::getControlConsultas(\$idPersona) [*line 2776*]

Parámetros de la función:

- **\$idPersona**

Obtiene todas las consultas asociadas a un paciente

- **Access** public

void función Enrolamiento_model::gettcp() [*line 427*]

- **Access** public

void función Enrolamiento_model::getcurp() [*line 179*]

- **Access** public

void función Enrolamiento_model::getcurpT() [*line 298*]

- **Access** public

void función Enrolamiento_model::getestimulacion_capacitado() [*line 687*]

- **Access** public

void función Enrolamiento_model::getestimulacion_fecha() [*line 677*]

- **Access** public

void función Enrolamiento_model::getfaccion_nutricional() [*line 558*]

- **Access** public

void función Enrolamiento_model::getfconsulta() [*line 528*]

- **Access** public

void función Enrolamiento_model::getfechacivil() [*line 347*]

- **Access** public

void función Enrolamiento_model::getfecha_peri_cefa() [*line 667*]

- **Access** public

void función Enrolamiento_model::getfnacimiento() [line 209]

- **Access** public

void función Enrolamiento_model::getfnutricion() [line 607]

- **Access** public

result() función Enrolamiento_model::getfolio(\$persona) [line 1158]

Parámetros de la función:

- *strin* **\$persona** id de la persona a la que se le asigna una tarjeta

Extrae el folio que se anexa en el envio para la tarjeta

- **Access** public

void función Enrolamiento_model::getfvacuna() [line 498]

- **Access** public

void función Enrolamiento_model::gethemoglobina() [line 597]

- **Access** public

void función Enrolamiento_model::getId() [line 129]

- **Access** public

void función Enrolamiento_model::getidtutor() [*line 259*]

- **Access** public

result() función Enrolamiento_model::getListEnrolamiento([\$keywords = "], [\$offset = null], [\$row_count = null]) [*line 1594*]

Parámetros de la función:

- *string* **\$keywords** palabras claves para hacer el filtro
- *strin* **\$offset** inicio del registro
- *string* **\$row_count** numero de filas a mostrar por pagina

Este metodo retorna el list de las personas enroladas

- **Access** public

void función Enrolamiento_model::getlnacimiento() [*line 169*]

- **Access** public

void función Enrolamiento_model::getlocalidad() [*line 407*]

- **Access** public

void función Enrolamiento_model::getlugarcivil() [*line 357*]

- **Access** public

void función Enrolamiento_model::getmanzana() [*line 457*]

- **Access** public

void función Enrolamiento_model::getmaterno() [*line 159*]

- **Access** public

void función Enrolamiento_model::getmaternoT() [*line 289*]

- **Access** public

void función Enrolamiento_model::getMsgError([\$value = 'usr']) [*line 2386*]

Parámetros de la función:

- **\$value**

- **Access** public

void función Enrolamiento_model::getnacionalidad() [*line 647*]

- **Access** public

void función Enrolamiento_model::getnombre() [*line 139*]

- **Access** public

void función Enrolamiento_model::getnombreT() [*line 269*]

- **Access** public

void función Enrolamiento_model::getnumero() [*line 417*]

- **Access** public

result() función Enrolamiento_model::getNumRows([\$keywords = ""]) [*line 1670*]

Parámetros de la función:

- *string* **\$keywords** palabras clave para hacer el filtro

Devuelve el numero de filas en la tabla cns_persona

- **Access** public

void función Enrolamiento_model::getparto() [*line 229*]

- **Access** public

void función Enrolamiento_model::getpaterno() [*line 149*]

- **Access** public

void función Enrolamiento_model::getpaternoT() [*line 279*]

- **Access** public

void función Enrolamiento_model::getperi_cefa() [*line 657*]

- **Access** public

void función Enrolamiento_model::getpeso() [*line 568*]

- **Access** public

void función Enrolamiento_model::getprecurp() [*line 239*]

- **Access** public

void función Enrolamiento_model::getreferencia() [*line 377*]

- **Access** public

result() función Enrolamiento_model::getRegistro_civil(\$id) [*line 1772*]

Parámetros de la función:

- *string* **\$id** identificador de la persona

obtiene informacion del registro civil

- **Access** public

void función Enrolamiento_model::getsales_cantidad() [*line 697*]

- **Access** public

void función Enrolamiento_model::getsales_fecha() [*line 707*]

- **Access** public

void función Enrolamiento_model::getsangre() [*line 199*]

- **Access** public

void función Enrolamiento_model::getsector() [*line 447*]

- **Access** public

void función Enrolamiento_model::getsexo() [*line 189*]

- **Access** public

void función Enrolamiento_model::getsexoT() [*line 308*]

- **Access** public

void función Enrolamiento_model::gettalla() [*line 587*]

- **Access** public

void función Enrolamiento_model::gettamiz() [*line 234*]

- **Access** public

void función Enrolamiento_model::gettbeneficiario() [*line 219*]

- **Access** public

void función Enrolamiento_model::gettconsulta() [*line 538*]

- **Access** public

void función Enrolamiento_model::gettelefono() [*line 617*]

- **Access** public

void función Enrolamiento_model::gettelefonoT() [*line 317*]

- **Access** public

void función Enrolamiento_model::getumt() [line 367]

- **Access** public

void función Enrolamiento_model::getvacuna() [line 488]

- **Access** public

result() función Enrolamiento_model::get_catalog(\$catalog, [\$campo = ""], [\$id = ""], [\$orden = ""], \$order) [line 1925]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- *string* **\$campo** nombre del campo para hacer el where
- *string* **\$id** valor del campo para el where
- *string* **\$order** nombre del campo para hacer el order by
- **\$orden**

Hace select de las tablas *cns_x* que representa a los catalogos

- **Access** public

result() función Enrolamiento_model::get_catalog2(\$catalog, [\$campo1 = ""], [\$id1 = ""], [\$campo2 = ""], [\$id2 = ""], [\$i1 = ""], [\$i2 = ""]) [line 2022]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- *string* **\$campo1** nombre del campo para hacer el where
- *string* **\$id1** valor del campo para el where
- *string* **\$campo2** nombre del campo para hacer el where
- *string* **\$id2** valor del campo para el where

- *strin* **\$I1** nombre del campo para hacer el limit offset
- *strin* **\$I2** nombre del campo para hacer el limit count

Obtiene los datos de una tabla

- **Access** public

count función Enrolamiento_model::get_catalog_count(\$catalog, [\$campo = ""], [\$valor = ""]) [line 1995]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- **\$campo**
- **\$valor**

Obtiene el numero de resultados de una tabla

- **Access** public

result() función Enrolamiento_model::get_catalog_relevante([\$fecha = ""]) [line 2199]

Parámetros de la función:

- **\$fecha**

obtiene los catalogos relevante x entorno para la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_catalog_tratamiento(\$catalog, \$campo, \$valor, \$orden, \$order) [line 1961]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- *string* **\$campo** nombre del campo para hacer el where
- *string* **\$valor** valor del campo para el where
- *string* **\$order** nombre del campo para hacer el order by
- **\$orden**

Hace select de los tratamientos de las consultas

- **Access public**

result() función Enrolamiento_model::get_catalog_view(\$catalog, \$id, [\$order1 = ""], [\$order2 = ""]) [line 1862]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- *string* **\$id** identificador de la persona
- *string* **\$order1** nombre del campo para hacer el order by
- *string* **\$order2** nombre del campo para hacer el order by

Hace select de los catalogos que tengan relacion con una persona para mostrarlos en el view

- **Access public**

result() función Enrolamiento_model::get_cns_cat_persona(\$catalog, \$array, [\$l1 = ""], [\$l2 = ""]) [line 2123]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla

- *string* **\$array** ids de personas
- *string* **\$l1** ofset para el limit
- *string* **\$l2** count para el limit

Este metodo obtiene los controles que le corresponde a cada persona y que seran incluidas en la sincronizacion

- **Access public**

result() función Enrolamiento_model::get_cns_cat_persona_count(\$catalog, \$persona, \$personas) [line 2158]

Parámetros de la función:

- *string* **\$catalog** Nombre de la tabla
- *string* **\$personas** personas que cumplen el requisito
- **\$persona**

Hace el count de personas que se envian en la sincronizacion

- **Access public**

result() función Enrolamiento_model::get_cns_persona(\$array, [\$fecha = ""]) [line 2089]

Parámetros de la función:

- *string* **\$array** arreglo con los ids de las personas que cumplen con los requisitos del envio
- *string* **\$fecha** fecha que determina si se envia o no una persona

Este metodo obtiene las personas que seran enviadas en la sincronizacion

- **Access public**

result() función Enrolamiento_model::get_control_nutricional(\$id, [\$order = ""]) [line 1893]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *string* **\$order** nombre del campo para hacer el order by

Obtiene los datos del control nutricional asociados a una persona

- **Access** public

void función Enrolamiento_model::get_datos_grafica(\$catalogo, \$sexo, \$edad_meses, \$id_persona, [\$asu_locali = ""]) [line 2404]

Parámetros de la función:

- *int* **\$catalogo** Determina el catalogo a consultar
- *int* **\$sexo** El sexo del paciente (F, M)
- *int* **\$edad_meses** Edad del paciente en meses
- *int* **\$id_persona** Identificador del paciente
- *int* **\$asu_locali** Identificador del asu de la localidad del domicilio

Obtiene los datos especificos de un catálogo para ser visualizados en una gráfica

- **Access** public

result() función Enrolamiento_model::get_estimulacion(\$id, [\$order = ""]) [line 2661]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *string* **\$order** nombre del campo para hacer el order by

Obtiene los registros de estimulacion temprana asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_notificacion(\$id) [line 2062]

Parámetros de la función:

- *string* **\$id** id del arbol de segmentacion

Este metodo obtiene las notificaciones que se enviaron en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_pacientes() [line 2368]

devuelve todos los pacientes de la base de datos

- **Access** public

result() función Enrolamiento_model::get_peri_cefa(\$id, [\$order = ""]) [line 2601]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *string* **\$order** nombre del campo para hacer el order by

Obtiene los registros de perimetro cefalico asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_persona_x_tutor(\$array) [line 2176]

Parámetros de la función:

- *string* **\$array** tutores que tienen asignado un paciente

obtiene los tutores de las personas que se envían en la sincronización

- **Access** public

result() función Enrolamiento_model::get_sales(\$id, [\$order = ""]) [line 2866]

Parámetros de la función:

- *string* **\$id** identificador de la persona
- *string* **\$order** nombre del campo para hacer el order by

Obtiene los registros de sales de rehidratación oral asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_transaction_relevante() [line 2225]

Obtiene las transacciones relevante para la sincronización

- **Access** public

result() función Enrolamiento_model::get_version() [line 2249]

obtiene cual es la ultima version de apk de la tableta

- **Access public**

result() función Enrolamiento_model::insert() [line 787]

Guarda la persona capturada mediante el formulario web

- **Access public**

void función Enrolamiento_model::setaccion_nutricional(\$value) [line 553]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setafiliacion(\$value) [line 473]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setageb(\$value) [line 442]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setalergias(\$value) [line 483]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setaltura(\$value) [line 583]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcalle(\$value) [line 392]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcelular(\$value) [line 642]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcelularT(\$value) [line 342]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcodigo_barras(\$value) [line 513]

Parámetros de la función:

- **\$value**

- **Access public**

```
void función Enrolamiento_model::setcolonia($value) [line 402]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setcompania($value) [line 632]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setcompaniaT($value) [line 332]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setconsulta($value) [line 523]
```

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcp(\$value) [line 432]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcurp(\$value) [line 184]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setcurpT(\$value) [line 303]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setestimulacion_capacitado(\$value) [line 692]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setestimulacion_fecha(\$value) [line 682]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setfaccion_nutricional(\$value) [line 563]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setfconsulta(\$value) [line 533]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setfechacivil(\$value) [line 352]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setfecha_peri_cefa(\$value) [line 672]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setfnacimiento(\$value) [line 214]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setfnutricion(\$value) [line 612]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setfvacuna(\$value) [line 503]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::sethemoglobina(\$value) [line 602]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setId(\$value) [line 134]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setidtutor(\$value) [line 264]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setlnacimiento(\$value) [line 174]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setlocalidad(\$value) [line 412]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setlugarcivil(\$value) [line 362]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setmanzana(\$value) [line 462]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setmaterno(\$value) [line 164]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setmaternoT(\$value) [line 293]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setnacionalidad(\$value) [line 652]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setnombre(\$value) [line 144]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setnombreT(\$value) [line 274]

Parámetros de la función:

- **\$value**
- **Access public**

void función Enrolamiento_model::setnumero(\$value) [line 422]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setparto(\$value) [line 249]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setpaterno(\$value) [line 154]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setpaternoT(\$value) [line 284]

Parámetros de la función:

- **\$value**

- **Access public**

```
void función Enrolamiento_model::setperi_cefa($value) [line 662]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setpeso($value) [line 573]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setprecurp($value) [line 244]
```

Parámetros de la función:

- **\$value**
- **Access** public

```
void función Enrolamiento_model::setreferencia($value) [line 382]
```

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsales_cantidad(\$value) [line 702]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsales_fecha(\$value) [line 712]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsangre(\$value) [line 204]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsector(\$value) [line 452]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsexo(\$value) [*line 194*]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setsexoT(\$value) [*line 313*]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::settalla(\$value) [*line 592*]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::settamiz(\$value) [line 254]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setbeneficiario(\$value) [line 224]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setconsulta(\$value) [line 543]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::settelefono(\$value) [line 622]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::settelefonoT(\$value) [line 322]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setumt(\$value) [line 372]

Parámetros de la función:

- **\$value**

- **Access public**

void función Enrolamiento_model::setvacuna(\$value) [line 493]

Parámetros de la función:

- **\$value**

- **Access public**

result() función Enrolamiento_model::tes_pendientes_tarjeta_delete() [line 2350]

Elimina los pendientes de las personas que no tengan asignado una tarjeta

- **Access public**

result() función Enrolamiento_model::update_accion() [line 1459]

Actualiza el control accion nutricional del paciente

- **Access public**

result() función Enrolamiento_model::update_alergia() [line 1360]

Actualiza los datos de las alergias del paciente

- **Access public**

result() función Enrolamiento_model::update_basico() [line 1171]

Actualiza los datos basicos del paciente

- **Access public**

result() función Enrolamiento_model::update_beneficiario() [line 1528]

Actualiza el tipo de beneficiario del paciente

- **Access** public

result() función Enrolamiento_model::update_consulta() [line 1424]

Actualiza el control consulta del paciente

- **Access** public

result() función Enrolamiento_model::update_direccion() [line 1227]

actualiza la direccion del paciente

- **Access** public

result() función Enrolamiento_model::update_estimulacion() [line 2689]

Actualiza los registros de estimulacion temprana

- **Access** public

result() función Enrolamiento_model::update_nutricion() [line 1492]

Actualiza el control nutricional del paciente

- **Access** public

result() función Enrolamiento_model::update_peri_cefa() [line 2629]

Actualiza los registros de perimetro cefalico

- **Access public**

result() función Enrolamiento_model::update_regcivil() [line 1265]

actualiza el registro civil del paciente

- **Access public**

result() función Enrolamiento_model::update_sales() [line 2834]

Actualiza los registros de sales de rehidratacion oral

- **Access public**

result() función Enrolamiento_model::update_status_tableta(\$mac, \$status, \$version, \$fecha) [line 1566]

Parámetros de la función:

- *string* **\$mac** Mac de la tableta
- *strin* **\$status** nuevo status
- *string* **\$version** version de la apk de la tableta
- *string* **\$fecha** fecha del evento

Hace update de la tableta que este sincronizando dependiendo del resultado

- **Access public**

result() función Enrolamiento_model::update_tutor() [line 1289]

Actualiza los datos del tutor del paciente

- **Access public**

result() función Enrolamiento_model::update_umt() [line 1204]

Actualiza la unidad medica tratante del paciente

- **Access public**

result() función Enrolamiento_model::update_vacuna() [line 1391]

Actualiza las vacunas del paciente

- **Access public**

result() función Enrolamiento_model::valid_card(\$persona, \$archivo) [line 1144]

Parámetros de la función:

- *string* **\$persona** id de la persona a la que se le asigna una tarjeta
- *string* **\$archivo** nombre del archivo que se genero

Este metodo valida que exista un archivo para enviar a la tarjeta por nfc

- **Access public**

Clase Estado_tableta_model

[line 11]

Modelo Estado_tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Estado_tableta_model::__construct() [line 49]

- **Access** public

array función Estado_tableta_model::getAll() [line 140]

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Estado_tableta_model::getById(\$id) [line 112]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

void función Estado_tableta_model::getDescripcion() [line 67]

- **Access** public

void función Estado_tableta_model::getId() [line 62]

- **Access** public

boolean/string función Estado_tableta_model::getMsgError([\$type = 'usr']) [line 91]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

void función Estado_tableta_model::setDescripcion(\$descripcion) [line 76]

Parámetros de la función:

- **\$descripcion**

- **Access** public

void función Estado_tableta_model::setId(\$id) [line 71]

Parámetros de la función:

- **\$id**

- **Access** public

Clase Notificacion_model

[line 10]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Notificacion_model::__construct() [line 74]

- **Access** public

void/boolean función Notificacion_model::addFilter(\$columna, \$condicion, \$valor) [line 324]

Parámetros de la función:

- *string* **\$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros,

id_controlador_accion)

- *string* **\$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **\$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access public**

boolean función Notificacion_model::delete() [line 304]

Elimina de la base de datos la notificaci?n (id en propiedades)

- **Access public**

void/array función Notificacion_model::getAll([\$keywords = ''], [\$offset = null], [\$row_count = null]) [line 170]

Parámetros de la función:

- *boolean/string* **\$keywords** false no hay texto a buscar|string con texto a buscar
- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todas las notificaciones existentes, se puede filtrar por: texto a buscar si se desea

- **Access public**

void/object función Notificacion_model::getById(\$id) [line 209]

Parámetros de la función:

- ***int \$id*** id de notificación

Obtiene la notificación solicitada

- **Access** public

void función Notificacion_model::getContenido() [*line 105*]

- **Access** public

void función Notificacion_model::getFechaFin() [*line 125*]

- **Access** public

void función Notificacion_model::getFechaInicio() [*line 115*]

- **Access** public

void función Notificacion_model::getId() [*line 86*]

- **Access** public

void función Notificacion_model::getIdsTabletas() [*line 135*]

- **Access** public

boolean función Notificacion_model::getMsgError([\$value = 'usr']) [line 152]

Parámetros de la función:

- *string* **\$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

int función Notificacion_model::getNumRows([\$keywords = ""]) [line 229]

Parámetros de la función:

- *boolean|string* **\$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de notificaciones

- **Access** public

void función Notificacion_model::getTitulo() [line 95]

- **Access** public

boolean función Notificacion_model::insert() [line 255]

Inserta en la base de datos los datos de la notificaci?n (datos en propiedades)

- **Access public**

void función Notificacion_model::setContenido(\$contenido) [line 110]

Parámetros de la función:

- **\$contenido**

- **Access public**

void función Notificacion_model::setFechaFin(\$fecha_fin) [line 130]

Parámetros de la función:

- **\$fecha_fin**

- **Access public**

void función Notificacion_model::setFechaInicio(\$fecha_inicio) [line 120]

Parámetros de la función:

- **\$fecha_inicio**

- **Access public**

void función Notificacion_model::setId(\$value) [line 91]

Parámetros de la función:

- **\$value**

- **Access public**

void función Notificacion_model::setIdTabletas(\$id_arr_asu) [line 140]

Parámetros de la función:

- **\$id_arr_asu**

- **Access public**

void función Notificacion_model::setTitulo(\$titulo) [line 100]

Parámetros de la función:

- **\$titulo**

- **Access public**

boolean función Notificacion_model::update() [line 279]

Actualiza en la base de datos los datos de la notificación (datos en propiedades)

- **Access public**

Clase Reporteador_model

[line 10]

Modelo Reporteador

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Reporteador_model::__construct() [line 27]

- **Access** public

void función Reporteador_model::getCensoNominal(\$nivel, \$id, &\$th) [line 319]

Parámetros de la función:

- **\$nivel**
- **\$id**
- **&\$th**

- **Access** public

void función Reporteador_model::getCoberturaBiologicoListado(\$nivel, \$id, \$fecha, [\$fechaFin = null]) [line 59]

Parámetros de la función:

- *int* **\$nivel** Nivel del elemento del arbol ASU
- *int* **\$id** Identificador del elemento ASU

- *date* **\$fecha** Fecha de corte de elemento
- **\$fechaFin**

Obtiene el reporte de cobertura por tipo de biologico

- **Access** public

void función Reporteador_model::getConcentradoActividades(\$nivel, \$id, \$fecha) [line 289]

Parámetros de la función:

- **\$nivel**
- **\$id**
- **\$fecha**

- **Access** public

void función Reporteador_model::getEsquemasIncompletos(\$nivel, \$id, &\$th) [line 414]

Parámetros de la función:

- **\$nivel**
- **\$id**
- **&\$th**

- **Access** public

void función Reporteador_model::getGrupoVacunas() [line 598]

boolean función Reporteador_model::getMsgError([\$value = 'usr']) [line 43]

Parámetros de la función:

- *string* **\$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void función Reporteador_model::getSeguimientoRV1RV5(\$nivel, \$id, \$fecha) [line 304]

Parámetros de la función:

- **\$nivel**
- **\$id**
- **\$fecha**

- **Access** public

void función Reporteador_model::getVacunas() [line 547]

void función Reporteador_model::getVacunasByGrupo(\$grupo) [line 573]

Parámetros de la función:

- **\$grupo**

void función Reporteador_model::object_to_array(\$data, \$campo1) [line 532]

Parámetros de la función:

- **\$data**
- **\$campo1**

Clase Reporte_censo_nominal

[line 11]

Modelo Reporte_censo_nominal

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Reporte_censo_nominal::\$apellido_materno

varchar = [line 21]

- **Access** public

Reporte_censo_nominal::\$apellido_paterno

varchar = [line 16]

- **Access** public

Reporte_censo_nominal::\$curp

varchar = [line 36]

- **Access** public

Reporte_censo_nominal::\$domicilio

varchar = [line 31]

- **Access** public

Reporte_censo_nominal::\$fecha_nacimiento

varchar = [line 41]

- **Access** public

Reporte_censo_nominal::\$nombre

varchar = [line 26]

- **Access** public

Reporte_censo_nominal::\$parto_multiple

varchar = [line 46]

- **Access** public

Reporte_censo_nominal::\$sexo

varchar = [line 51]

- **Access** public

Reporte_censo_nominal::\$vacunas

array = [line 56]

- **Access** public

Constructor *void* función Reporte_censo_nominal::__construct() [*line 58*]

- **Access** public

Clase Reporte_sincronizacion_model

[*line 10*]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Eliecer

num_rows() función Reporte_sincronizacion_model::getCount(\$tabla, [\$sentencia = ""], \$sentencia) [*line 55*]

Parámetros de la función:

- *string* **\$tabla** nombre de una tabla en la base de datos
- *string* **\$sentencia** consulta sql
- **\$sentencia**

obtiene el numero de registros de una tabla o consulta

- **Access public**

result() función Reporte_sincronizacion_model::getListado(\$sql) [line 31]

Parámetros de la función:

- *string \$sql* consulta sql a ejecutar

Obtiene el resultado de una consulta

- **Access public**

void función Reporte_sincronizacion_model::getMsgError([\$value = 'usr']) [line 88]

Parámetros de la función:

- **\$value**

- **Access public**

result() función Reporte_sincronizacion_model::get_version() [line 71]

obtiene la ultima version de la apk de las tabletas

- **Access public**

Clase Semana_nacional_model

[line 11]

Modelo Tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Semana_nacional_model::__construct() [line 61]

- **Access** public

int función Semana_nacional_model::delete([\$id = null]) [line 203]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

array función Semana_nacional_model::getAll([\$offset = null], [\$row_count = null]) [line 281]

Parámetros de la función:

- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función `Semana_nacional_model::getById($id)` [line 243]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

void función `Semana_nacional_model::getDescripcion()` [line 79]

- **Access** public

void función `Semana_nacional_model::getFecha_fin()` [line 89]

- **Access** public

void función `Semana_nacional_model::getFecha_inicio()` [line 84]

- **Access** public

void función `Semana_nacional_model::getId()` [*line 74*]

- **Access** public

boolean/string función `Semana_nacional_model::getMsgError([$type = 'usr'])` [*line 120*]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

int función `Semana_nacional_model::getNumRows()` [*line 316*]

Obtiene el numero total de registros en la tabla

- **Access** public

boolean función `Semana_nacional_model::insert()` [*line 140*]

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función `Semana_nacional_model::setDescripcion($descripcion)` [*line 94*]

Parámetros de la función:

- **\$descripcion**

- **Access public**

void función Semana_nacional_model::setFecha_fin(\$fecha_fin) [line 104]

Parámetros de la función:

- **\$fecha_fin**

- **Access public**

void función Semana_nacional_model::setFecha_inicio(\$fecha_inicio) [line 99]

Parámetros de la función:

- **\$fecha_inicio**

- **Access public**

boolean función Semana_nacional_model::update([\$id = null]) [line 171]

Parámetros de la función:

- **int \$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Tableta_model

[line 11]

Modelo Tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Tableta_model::__construct() [line 90]

- **Access** public

int función Tableta_model::delete([\$id = null]) [line 292]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

array función Tableta_model::getAll([\$offset = null], [\$row_count = null], \$filtro) [line 419]

Parámetros de la función:

- *int* **\$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **\$row_count** Establece la cantidad de registros a devolver
- **\$filtro**

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Tableta_model::getById(\$id) [line 332]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

object|boolean función Tableta_model::getByMac(\$mac) [line 374]

Parámetros de la función:

- *int* **\$mac** Direccion MAC

Obtiene los datos del registro la MAC especificada

- **Access** public

void función Tableta_model::getId() [*line 103*]

- **Access** public

void función Tableta_model::getIdVersion() [*line 112*]

- **Access** public

void función Tableta_model::getId_asu_um() [*line 132*]

- **Access** public

void función Tableta_model::getId_tes_estado_tableta() [*line 124*]

- **Access** public

void función Tableta_model::getId_tipo_censo() [*line 128*]

- **Access** public

void función Tableta_model::getMac() [*line 108*]

- **Access** public

boolean|string función `Tableta_model::getMsgError([$type = 'usr'])` [line 188]

Parámetros de la función:

- **string \$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

int función `Tableta_model::getNumRows()` [line 518]

Obtiene el numero total de registros en la tabla en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función `Tableta_model::getPeriodo_esq_inc()` [line 136]

- **Access** public

void función `Tableta_model::getUltima_actualizacion()` [line 116]

- **Access** public

void función `Tableta_model::getUsuarios_asignados()` [line 120]

- **Access public**

boolean función Tableta_model::insert() [line 208]

Inserta en la base de datos, la informacion contenida en el objeto

- **Access public**

void función Tableta_model::setId(\$id) [line 140]

Parámetros de la función:

- **\$id**

- **Access public**

void función Tableta_model::setIdVersion(\$id_version) [line 149]

Parámetros de la función:

- **\$id_version**

- **Access public**

void función Tableta_model::setId_asu_um(\$id_asu_um) [line 169]

Parámetros de la función:

- **\$id_asu_um**

- **Access public**

void función Tableta_model::setId_tes_estado_tableta(\$id_tes_estado_tableta) [line 161]

Parámetros de la función:

- **\$id_tes_estado_tableta**

- **Access public**

void función Tableta_model::setId_tipo_censo(\$id_tipo_censo) [line 165]

Parámetros de la función:

- **\$id_tipo_censo**

- **Access public**

void función Tableta_model::setMac(\$mac) [line 145]

Parámetros de la función:

- **\$mac**

- **Access public**

void función Tableta_model::setPeriodo_esq_inc(\$periodo_esq_inc) [line 173]

Parámetros de la función:

- **\$periodo_esq_inc**

- **Access public**

void función Tableta_model::setUltima_actualizacion(\$ultima_actualizacion) [line 153]

Parámetros de la función:

- **\$ultima_actualizacion**

- **Access public**

void función Tableta_model::setUsuarios_asignados(\$usuarios_asignados) [line 157]

Parámetros de la función:

- **\$usuarios_asignados**

- **Access public**

boolean función Tableta_model::update([\$id = null]) [line 239]

Parámetros de la función:

- *int* **\$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Tipo_censo_model

[line 11]

Modelo Tipo_censo

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Tipo_censo_model::__construct() [line 49]

- **Access** public

array función Tipo_censo_model::getAll() [line 141]

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Tipo_censo_model::getById(\$id) [line 113]

Parámetros de la función:

- **int \$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access public**

void función Tipo_censo_model::getDescripcion() [line 67]

- **Access public**

void función Tipo_censo_model::getId() [line 62]

- **Access public**

boolean/string función Tipo_censo_model::getMsgError([\$type = 'usr']) [line 92]

Parámetros de la función:

- **string \$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access public**

void función Tipo_censo_model::setDescripcion(\$descripcion) [line 77]

Parámetros de la función:

- **\$descripcion**

- **Access public**

void función Tipo_censo_model::setId(\$id) [line 72]

Parámetros de la función:

- **\$id**

- **Access public**

Clase Usuario_tableta_model

[line 11]

Modelo Estado_tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void función Usuario_tableta_model::__construct() [line 49]*

- **Access public**

boolean función Usuario_tableta_model::delete(\$id_usuario, [\$id_tableta = null]) [line 161]

Parámetros de la función:

- *int* **\$id_usuario**
- *int* **\$id_tableta** Si no se proporciona el parametro, se toma el id del objeto actual

Elimina la relacion entre usuario y tableta

- **Access public**

boolean/string función Usuario_tableta_model::getMsgError([\$type = 'usr']) [line 91]

Parámetros de la función:

- *string* **\$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access public**

void función Usuario_tableta_model::getTableta() [line 62]

- **Access public**

object|boolean función Usuario_tableta_model::getTabletasByUsuario([\$id_usuario = null]) [line 136]

Parámetros de la función:

- **int \$id_usuario** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene el id de todas las tabletas relacionadas con el usuario especificado

- **Access public**

void función Usuario_tableta_model::getUsuario() [line 67]

- **Access public**

object|boolean función Usuario_tableta_model::getUsuariosByTableta([\$id_tes_tableta = null]) [line 112]

Parámetros de la función:

- **int \$id_tes_tableta** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene el id de todos los usuarios asignados a la tableta especificada

- **Access public**

boolean función Usuario_tableta_model::insert([\$id_usuario = null], [\$id_tableta = null]) [line 204]

Parámetros de la función:

- **\$id_usuario**
- **\$id_tableta**

Inserta en la base de datos, la informacion contenida en el objeto

- **Access public**

void función Usuario_tableta_model::setTableta(\$id_tes_tableta) [line 71]

Parámetros de la función:

- **\$id_tes_tableta**

- **Access public**

void función Usuario_tableta_model::setUsuario(\$id_usuario) [line 76]

Parámetros de la función:

- **\$id_usuario**

- **Access public**

Apéndices

Apéndice A - Arbol de clases

Paquete SIIGS

Accion

- CI_Controller
 - [Accion](#)

Accion_model

- CI_Model
 - [Accion_model](#)

Ageb_model

- CI_Model
 - [Ageb_model](#)

ArbolSegmentacion_model

- CI_Model
 - [ArbolSegmentacion_model](#)

Ayuda

- CI_Controller
 - [Ayuda](#)

Bitacora

- CI_Controller
 - [Bitacora](#)

Bitacora_model

- CI_Model
 - [Bitacora_model](#)

Catalogo

- CI_Controller
 - [Catalogo](#)

CatalogoCsv

- CI_Controller
 - [CatalogoCsv](#)

CatalogoCsv_model

- CI_Model
 - [CatalogoCsv_model](#)

Catalogo_model

- CI_Model
 - [Catalogo_model](#)

Catalogo_x_raiz

- CI_Controller
 - [Catalogo_x_raiz](#)

Catalogo_x_raiz_model

- CI_Model
 - [Catalogo_x_raiz_model](#)

Cie10

- CI_Controller
 - [Cie10](#)

Cie10_model

- CI_Model
 - [Cie10_model](#)

Controlador

- CI_Controller
 - [Controlador](#)

ControladorAccion_model

- CI_Model
 - [ControladorAccion_model](#)

Controlador_model

- CI_Model
 - [Controlador_model](#)

Entorno

- CI_Controller
 - [Entorno](#)

Entorno_model

- CI_Model
 - [Entorno_model](#)

Errorlog

- CI_Controller
 - [Errorlog](#)

Errorlog_model

- CI_Model
 - [Errorlog_model](#)

Georeferencia_model

- CI_Model
 - [Georeferencia_model](#)

Grupo

- CI_Controller
 - [Grupo](#)

Grupo_model

- CI_Model
 - [Grupo_model](#)

Hemoglobina_model

- CI_Model
 - [Hemoglobina_model](#)

Menu

- CI_Controller
 - [Menu](#)

Menu_model

- CI_Model
 - [Menu_model](#)

Permiso

- CI_Controller
 - [Permiso](#)

Permiso_model

- CI_Model
 - [Permiso_model](#)

Poblacion_model

- CI_Model
 - [Poblacion_model](#)

Raiz

- CI_Controller
 - [Raiz](#)

Raiz_model

- CI_Model
 - [Raiz_model](#)

ReglaVacuna

- CI_Controller
 - [ReglaVacuna](#)

ReglaVacuna_model

- CI_Model
 - [ReglaVacuna_model](#)

Usuario

- CI_Controller
 - [Usuario](#)

Usuario_model

- CI_Model
 - [Usuario_model](#)

Paquete default

Index

- CI_Controller
 - [Index](#)

Index

- CI_Controller
 - [Index](#)

Paquete TES

Enrolamiento

- CI_Controller
 - [Enrolamiento](#)

Enrolamiento_model

- CI_Model
 - [Enrolamiento_model](#)

Estado_tableta_model

- CI_Model
 - [Estado_tableta_model](#)

Notificacion

- CI_Controller
 - [Notificacion](#)

Notificacion_model

- CI_Model
 - [Notificacion_model](#)

Reporteador

- CI_Controller
 - [Reporteador](#)

Reporteador_model

- CI_Model
 - [Reporteador_model](#)

Reporte_censo_nominal

- CI_Model
 - [Reporte_censo_nominal](#)

Reporte_sincronizacion

- CI_Controller
 - [Reporte_sincronizacion](#)

Reporte_sincronizacion_model

- CI_Model
 - [Reporte_sincronizacion_model](#)

Semana_nacional

- CI_Controller
 - [Semana nacional](#)

Semana_nacional_model

- CI_Model
 - [Semana nacional_model](#)

Servicios

- CI_Controller
 - [Servicios](#)

Tableta

- CI_Controller
 - [Tableta](#)

Tableta_model

- CI_Model
 - [Tableta_model](#)

Tipo_censo_model

- CI_Model
 - [Tipo_censo_model](#)

Usuario_tableta

- CI_Controller
 - [Usuario_tableta](#)

Usuario_tableta_model

- CI_Model
 - [Usuario_tableta_model](#)

Paquete Libreria

Graph

- CI_Controller
 - [Graph](#)

Obtenercurp

- CI_Controller
 - [Obtenercurp](#)

Tree

- CI_Controller
 - [Tree](#)

Indice

A

ArbolSegmentacion_model	58
<i>Modelo ArbolSegmentacion</i>	
ArbolSegmentacion_model::convertType()	59
<i>Convierte el tipo de arreglo para enviarlo como se debe recibir en el cliente de Javascript</i>	
ArbolSegmentacion_model::getByld()	59
<i>Obtener el elemento del ASU por medio de su ID</i>	
ArbolSegmentacion_model::getChildrenFromld()	59
<i>Accion para devolver los hijos de un elemento en el ASU a partir de su ID</i>	
Ageb_model::searchUM()	58
<i>Devuelve la información de una UM de acuerdo a su localidad y ageb</i>	
Ageb_model::searchageb()	57
<i>Devuelve una lista de AGEBS de la localidad pasada como parámetro</i>	
Accion_model::update()	56
<i>Actualiza el objeto actual en la base de datos</i>	
Ageb_model	56
<i>Modelo Ageb</i>	
Ageb_model::getMsgError()	57
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Ageb_model::process()	57
<i>Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion</i>	
ArbolSegmentacion_model::getChildrenFromLevel()	60
<i>Accion para devolver el esquema completo del ASU a partir de un nivel especificado, niveles omitidos y elementos preseleccionados</i>	
ArbolSegmentacion_model::getCluesFromld()	60
*	
ArbolSegmentacion_model::getTreeBlockData()	63
<i>Regresa el objeto del arbol de segmentacion por nivel o hijos de un elemento seleccionado</i>	
ArbolSegmentacion_model::getUMPParentsByld()	64
<i>Regresa la información de los padres de una unidad medica en el ASU</i>	
ArbolSegmentacion_model::addSelectedItems()	64
ArbolSegmentacion_model::addSelectedItems()	64
ArbolSegmentacion_model::getTreeBlock()	63
<i>Accion para devolver un bloque del ASU a partir de un nivel especificado o una clave seleccionada, niveles omitidos y elementos preseleccionados</i>	
ArbolSegmentacion_model::getTree()	62
<i>Regresa el objeto del arbol de segmentacion</i>	
ArbolSegmentacion_model::getDataKeyValue()	61
<i>Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro</i>	
ArbolSegmentacion_model::getDescripcionByld()	61
<i>Accion para obtener la descripcion e información adicional del elemento en el ASU</i>	
ArbolSegmentacion_model::getListChildrenLevel()	62

<u>ArbolSegmentacion_model::getMsgError()</u>	62
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
<u>Accion_model::setRows()</u>	56
<u>Accion_model::setOffset()</u>	55
<u>Ayuda::index()</u>	9
<i>Función que renderiza el contenido de la ayuda dependiendo de la sección donde se encuentre</i>	
<u>Accion_model</u>	52
<i>Modelo Accion</i>	
<u>Accion_model::delete()</u>	52
<i>Elimina el registro actual de la base de datos</i>	
<u>Accion_model::getAll()</u>	52
<i>Devuelve todos los registros de la tabla acciones</i>	
<u>Ayuda</u>	9
<i>Controlador Ayuda</i>	
<u>Accion::view()</u>	9
<i>Acción para visualizar información de una acción específica, obtiene el objeto acción por medio del id proporcionado.</i>	
<u>Accion::delete()</u>	7
<i>Acción para eliminar una acción, recibe el id de la acción a eliminar</i>	
<u>Accion::index()</u>	8
<i>Acción por default del controlador, carga la lista</i>	
<u>Accion::insert()</u>	8
<i>Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente</i>	
<u>Accion::update()</u>	8
<i>Acción para preparar la actualización de una acción ya existente,</i>	
<u>Accion_model::getById()</u>	52
<i>Devuelve la información de una accion por su ID</i>	
<u>Accion_model::getDescripcion()</u>	53
<u>Accion_model::setDescripcion()</u>	54
<u>Accion_model::setId()</u>	55
<u>Accion_model::setMetodo()</u>	55
<u>Accion_model::setNombre()</u>	55
<u>Accion_model::insert()</u>	54
<i>Inserta en la tabla accion la información contenida en el objeto</i>	
<u>Accion_model::getNumRows()</u>	54
<i>Devuelve el numero de registros</i>	
<u>Accion_model::getId()</u>	53

<u>Accion_model::getMetodo()</u>	53
<u>Accion_model::getMsgError()</u>	53
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
<u>Accion_model::getNombre()</u>	54
<u>Accion</u>	7
<i>Controlador Accion</i>	

B

Bitacora_model::resetFilter()	69
<i>Elimina todos los filtros registrados</i>	
Bitacora_model::setFecha_hora()	69
Bitacora_model::insert()	68
<i>Inserta a la bitacora la información porporcionada</i>	
Bitacora_model::getParametros()	68
Bitacora_model::getNumRows()	68
<i>Obtiene el numero total de registros en la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta</i>	
Bitacora_model::setId()	69
Bitacora_model::setId_accion()	69
Bitacora_model::setParametros()	71
Bitacora_model::update()	71
<i>Actualiza los datos del objeto actual</i>	
Bitacora_model::setId_usuario()	70
Bitacora_model::setId_controlador_accion()	70
Bitacora_model::setId_controlador()	70
Bitacora_model::getMsgError()	67
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Bitacora_model::getId_usuario()	67
Bitacora_model	65
<i>Modelo Bitacora</i>	
Bitacora_model::addFilter()	65
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Bitacora::view()	11
<i>Muestra los datos del registro especificado por el id</i>	
Bitacora::validateExistUsuario()	11
<i>callback utilizado por las acciones create y update para validar la existencia de un usuario</i>	
Bitacora::index()	10
<i>Lista todos los registros de la bitacora, con su correspondiente paginación</i>	
Bitacora_model::delete()	66
<i>Elimina el registro actual de la base de datos</i>	
Bitacora_model::deleteByFilter()	66
<i>Elimina el conjunto de registros que cumplen con el o los criterios de filtrado</i>	
Bitacora_model::getId()	67
Bitacora_model::getId_controlador_accion()	67
Bitacora_model::getFecha_hora()	67
Bitacora_model::getById()	67
<i>Obtiene los datos del registro de la bitacora que tiene el ID especificado</i>	
Bitacora_model::getAll()	66
<i>Obtiene todos los registros de la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta</i>	
Bitacora	10
<i>Controlador Bitacora</i>	

C

Cie10_model::getData()	93
<i>Devuelve una lista con los registros existentes en el catalogo cie10 omitiendo los ID</i>	
Cie10_model::getCatalogoByName()	93

<i>Devuelve una lista con los registros existentes en el catalogo requerido</i>	
Cie10_model::getDescripcion()	93
Cie10_model::getId()	93

Cie10_model::getMsgError()	94
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
Cie10_model::getById()	92
<i>Devuelve la información de un registro del catalogo cie10 por su ID</i>	
Cie10_model::getAllData()	92
<i>Devuelve los datos de un catalogo pasado como parametro</i>	
Cie10_model::activaEnCatalogo()	91
<i>Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas</i>	
constructor Cie10_model::construct()	90
Cie10_model::agregaEnCatalogo()	91
<i>Accion para agregar registros del CIE10 a otros catalogos como el de EDA, IRA y Consultas</i>	
Cie10_model::checkPk()	91
<i>Revisa en la base de datos por registros duplicados en los campos pasados por parametro</i>	
Cie10_model::getAll()	92
<i>Devuelve una lista con los registros existentes en el catalogo cie10</i>	
Cie10_model::getNumRows()	94
<i>Devuelve el numero de registros</i>	
Cie10_model::setDescription()	94
ControladorAccion_model::getIdByPath()	97
<i>Devuelve el id de una accion por controlador de acuerdo al path</i>	
ControladorAccion_model::getId()	97
<i>Devuelve el Id de una accion por controlador de una accion y controlador determinados</i>	
ControladorAccion_model::getMsgError()	97
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
ControladorAccion_model::setHelp()	98
<i>Establece el mensaje de ayuda</i>	
Controlador_model	98
<i>Modelo Controlador</i>	
ControladorAccion_model::getById()	96
<i>Devuelve la información de una accion por controlador de acuerdo a su Id</i>	
constructor ControladorAccion_model::construct()	96
Cie10_model::setOffset()	95
Cie10_model::setId()	95
Cie10_model::setRows()	95
Cie10_model::update()	95
<i>Actualiza el objeto actual en la base de datos</i>	
ControladorAccion_model	96
<i>Modelo ControladorAccion</i>	
Cie10_model	90
<i>Modelo Cie10</i>	
Catalogo_x_raiz_model::setTablaCatalogo()	90
Catalogo_x_raiz_model::getGrado()	85
Catalogo_x_raiz_model::getColumnaLLave()	85
Catalogo_x_raiz_model::getId()	86

Catalogo x raiz model::getIdRaiz()	86
Catalogo x raiz model::getMsgError()	86
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
Catalogo x raiz model::getColumnaDescripcion()	85
Catalogo x raiz model::getByNivel()	85
<i>Devuelve el catalogo padre de un elemento raiz_x_catalogo</i>	
Catalogo x raiz model::check()	83
<i>Revisa inconsistencias en los datos de un catalogo x raiz con respecto a su catalogo padre</i>	
constructor Catalogo x raiz model:: construct()	83
Catalogo x raiz model::delete()	84
<i>Elimina el registro actual de la base de datos</i>	
Catalogo x raiz model::getByArbol()	84
<i>Devuelve todos los registros de la tabla raiz_x_catalogo de una raiz determinada</i>	
Catalogo x raiz model::getById()	84
<i>Devuelve la información de un catalogo x accion por su ID</i>	
Catalogo x raiz model::getNivel()	86
<i>Devuelve el nivel siguiente para la tabla raiz_x_catalogo de un arbol determinado</i>	
Catalogo x raiz model::getRelacionHijo()	87
Catalogo x raiz model::setId()	89
Catalogo x raiz model::setGrado()	88
Catalogo x raiz model::setIdRaiz()	89
Catalogo x raiz model::setRelacionHijo()	89
Catalogo x raiz model::setRelacionPadre()	89
Catalogo x raiz model::setColumnaLlave()	88
Catalogo x raiz model::setColumnaDescripcion()	88
Catalogo x raiz model::getRelacionPadre()	87
Catalogo x raiz model::getRelations()	87
<i>Devuelve las relaciones de una raiz x catalogo</i>	
Catalogo x raiz model::getTablaCatalogo()	87
Catalogo x raiz model::insert()	88
<i>Inserta en la base datos la información del objeto actual</i>	
constructor Controlador model:: construct()	98
Controlador model::accionesUpdate()	99
<i>Actualiza las acciones asignadas a un controlador</i>	
constructor Index:: construct()	160
constructor Usuario model:: construct()	148
constructor Enrolamiento:: construct()	161
constructor Notificacion:: construct()	172
constructor Reporteador:: construct()	174
constructor ReglaVacuna model:: construct()	140
constructor Raiz model:: construct()	136
constructor Hemoglobina model:: construct()	123
constructor Grupo model:: construct()	118
constructor Menu model:: construct()	124
constructor Permiso model:: construct()	131
constructor Poblacion model:: construct()	135
constructor Reporte sincronizacion:: construct()	175
constructor Semana nacional:: construct()	177
constructor Semana nacional model:: construct()	248

constructor Reporte censo nominal:: construct()	246
constructor Tableta model:: construct()	252
constructor Tipo censo model:: construct()	259
constructor Usuario tableta model:: construct()	261
constructor Reporteador model:: construct()	241
constructor Notificacion model:: construct()	235
constructor Tableta:: construct()	185
constructor Servicios:: construct()	179
constructor Usuario tableta:: construct()	188
constructor Enrolamiento model:: construct()	189
constructor Estado tableta model:: construct()	233
constructor Georeferencia model:: construct()	117
constructor Errorlog model:: construct()	111
Controlador model::getId()	101

Controlador model::getDescripcion()	101
Controlador model::getIdEntorno()	101
Controlador model::getMsgError()	101
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
Controlador model::getNombre()	102
Controlador model::getClase()	101
Controlador model::getById()	100
<i>Devuelve la información de un controlador por su ID</i>	
Controlador model::getAccion()	99
Controlador model::delete()	99
<i>Elimina el registro actual de la base de datos</i>	
Controlador model::getAcciones()	99
<i>Devuelve todas las acciones asignadas al controlador por su Id</i>	
Controlador model::getAll()	100
<i>Devuelve todos los registros de la tabla controlador</i>	
Controlador model::getByEntorno()	100
<i>Devuelve todos los controladores que pertenecen a un entorno por su Id</i>	
Controlador model::getNumRows()	102
<i>Devuelve el numero de registros</i>	
Controlador model::getPermisos()	102
<i>Devuelve los permisos asignados a un grupo sobre un entorno determinado</i>	
<i>(Mapea la información de las acciones asignadas a un controlador y los une con</i>	
<i>los permisos de un grupo sobre esas acciones)</i>	
Controlador model::setOffset()	104
Controlador model::setNombre()	104
Controlador model::setRows()	105
Controlador model::update()	105
<i>Actualiza el objeto actual en la base de datos</i>	
constructor Entorno model:: construct()	106
Controlador model::setIdEntorno()	104
Controlador model::setId()	104
Controlador model::insert()	102
<i>Inserta en la tabla controlador, la información contenida en el objeto</i>	
Controlador model::setAccion()	103
Controlador model::setClase()	103
Controlador model::setDescripcion()	103

Catalogo x raiz_model	83
<i>Modelo Raiz_x_Catalogo</i>	
Catalogo_model::updateComentario()	83
<i>Cambia el comentario de la tabla indicada</i>	
Catalogo x raiz::view()	21
<i>Acción para visualizar información de una raiz_x_catalogo específica, obtiene el objeto raiz_x_catalogo por medio del id proporcionado.</i>	
Catalogo x raiz::insert()	21
<i>Acción para preparar la inserción de nuevas raíces para catálogos , realiza la validación del formulario del lado cliente</i>	
Cie10	22
<i>Controlador Cie10</i>	
constructor Cie10::construct()	22
Cie10::ActivaEnCatalogo()	22
*	
<i>Accion para activar o desactivar elementos en los catalogos IRA EDA Consultas</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Catalogo x raiz::delete()	21
<i>Acción para eliminar un catálogo en el arbol, recibe el id del catálogo en la raiz a eliminar</i>	
Catalogo x raiz::check()	20
<i>Acción que sirve para revisar inconsistencias en el arbol de segmentacion</i>	
<i>Recibe como parámetro el catálogo x raiz y revisa que todos los registros tengan un correspondiente en el catálogo padre.</i>	
CatalogoCsv::view()	19
<i>Acción para visualizar información de un catálogo específico, obtiene el objeto catalogocsv por medio del nombre proporcionado</i>	
CatalogoCsv::update()	18
<i>Acción para preparar la actualizacion de un catálogo ya existente,</i>	
CatalogoCsv::array_unique_recursive()	19
<i>_array_unique_recursive</i>	
<i>Revisa valores duplicados en arreglos que contienen arreglos</i>	
Catalogo x raiz	20
<i>Controlador Raiz_x_Catalogo</i>	
constructor Catalogo x raiz::construct()	20
Cie10::AgregaEnCatalogo()	23
*	
<i>Accion para agregar elementos del catalogo cie10 a los catalogos de EDA, IRA y Consultas dependiendo de los</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Cie10::index()	23
<i>Acción por default del controlador, carga la lista de datos disponibles en el cie10 y una lista de opciones</i>	
Controlador::delete()	26
<i>Acción para eliminar un controlador, recibe el id del controlador a eliminar</i>	
Controlador::accion()	26
<i>Acción para preparar la actualización de acciones asignadas a un controlador, recibe un ID para obtener las acciones asignadas a ese controlador y mostrarlos en la vista update</i>	
Controlador::getGroupPermissions()	27
<i>Acción para servir un array de objetos con los permisos asignados a</i>	
Controlador::help()	27
<i>Establece el texto de ayuda para el controlador accion</i>	
Controlador::index()	27

Acción por default del controlador, carga la lista de controladores disponibles y una lista de opciones Recibe un parametro en caso de filtrado por entornos	
constructor Controlador::__construct()	26
Controlador	25
Controlador Controlador	
Cie10::load()	24
Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP	
Cie10::insert()	23
Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar	
Cie10::update()	24
Acción para preparar la actualización de un registro del CIE10,	
Cie10::view()	25
*	
Accion para mostrar información de los catalogos IDE , ERA y Consultas	
Cie10::array_unique_recursive()	25
_array_unique_recursive Revisa valores duplicados en arreglos que contienen arreglos	
CatalogoCsv::loadupdate()	18
Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP	
CatalogoCsv::index()	18
Acción por default del controlador, carga la lista de catálogoscsv disponibles y una lista de opciones No recibe parámetros	
Catalogo::checkTypeData()	12
Acción para revisar si los tipos de datos coinciden con los datos contenidos en la tabla temporal que fueron tomados del CSV	
Catalogo::checkpk()	12
Acción para revisar registros repetidos en las columnas designadas como primary key	
Catalogo::delete()	13
Acción para eliminar un catálogo, recibe el nombre del catalogo a eliminar	
Catalogo::index()	13
Acción por default del controlador, carga la lista de catálogos disponibles y una lista de opciones No recibe parámetros	
Catalogo::insert()	13
Acción para preparar la inserción de nuevos catálogos , realiza la validación del formulario del lado del servidor y crea la estructura para el catálogo, crea la tabla y obtiene los datos a partir de la tabla tmp_catalogo	
constructor Catalogo::__construct()	12
Catalogo	12
Controlador Catalogo	
constructor Tree::__construct()	5
constructor Obtenercurp::__construct()	3
constructor Accion::__construct()	7
constructor Ayuda::__construct()	9
constructor Bitacora::__construct()	10
Catalogo::load()	14
Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP Guarda en la tabla tmp_catalogos toda la estructura del CSV e imprime las columnas del	

archivo. Solo se permite su acceso por medio de peticiones AJAX	
Catalogo::loadupdate()	14
Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar.	
CatalogoCsv::createTableAgeb()	17
Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros	
CatalogoCsv::checkpk()	17
Acción para revisar registros repetidos en las columnas designadas como primary key	
CatalogoCsv::createTableGeo()	17
Acción para ejecutar la creación de la tabla poblacional No recibe parámetros	
CatalogoCsv::createTableHemoGlobina()	17
Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros	
CatalogoCsv::createTablePob()	17
Acción para ejecutar la creación de la tabla poblacional No recibe parámetros	
CatalogoCsv::ActivaEnCatalogo()	16
* Accion para activar o desactivar elementos en los catalogos indicados en el parametro Solo se permite su acceso por medio de peticiones AJAX	
constructor CatalogoCsv:: __construct()	16
Catalogo::update()	14
Acción para preparar la actualizacion de un catálogo ya existente,	
Catalogo::view()	15
Acción para visualizar información de un catálogo específico, obtiene el objeto catálogo por medio del nombre proporcionado	
Catalogo:: _array_unique_recursive()	15
_array_unique_recursive Revisa valores duplicados en arreglos multidimensionales	
CatalogoCsv	16
Controlador CatalogoCsv	
Controlador::insert()	28
Acción para preparar la insercion de nuevos controladores , realiza la validacion del formulario del lado cliente	
Controlador::update()	28
Acción para preparar la actualización de un controlador ya existente,	
Catalogo_model::checkTypeData()	77
Revisa en la base de datos por registros que no coincidan con el tipo de dato pasado como parametro en el campo indicado	
Catalogo_model::checkPk()	77
Revisa en la base de datos por registros duplicados en los campos pasados por parametro	
Catalogo_model::delete()	78
Elimina el registro actual de la base de datos	
Catalogo_model::getAll()	78
Devuelve una lista con los catalogos existentes en la DB	
Catalogo_model::getAllData()	78
Devuelve los datos de un catalogo pasado como parametro	
constructor Catalogo_model:: __construct()	77
Catalogo_model	77

<i>Modelo Catalogo</i>	
CatalogoCsv_model::setLlave()	75
CatalogoCsv_model::setId()	75
CatalogoCsv_model::setNombre()	76
CatalogoCsv_model::setOffset()	76
CatalogoCsv_model::setRows()	76
Catalogo_model::getByName()	79
<i>Devuelve la informacion de un catalogo por su nombre</i>	
Catalogo_model::getCampos()	79
Catalogo_model::setLlave()	81
Catalogo_model::setId()	81
Catalogo_model::setNombre()	82
Catalogo_model::setOffset()	82
Catalogo_model::setRows()	82
Catalogo_model::setCampos()	81
Catalogo_model::insert()	80
<i>Inserta en la base datos el catálogo y obtiene los datos de la tabla temporal</i>	
Catalogo_model::getLLave()	79
Catalogo_model::getId()	79

Catalogo_model::getMsgError()	80
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
Catalogo_model::getNombre()	80
Catalogo_model::getNumRows()	80
<i>Devuelve el numero de registros</i>	
CatalogoCsv_model::setCampos()	75
CatalogoCsv_model::getNumRows()	75
<i>Devuelve el numero de registros</i>	
constructor Usuario:: __construct()	47
constructor ReglaVacuna:: __construct()	44
constructor Accion_model:: __construct()	52
constructor Ageb_model:: __construct()	57
constructor ArbolSegmentacion_model:: __construct()	58
constructor Raiz:: __construct()	39
constructor Permiso:: __construct()	38
constructor Entorno:: __construct()	29
Controlador::view()	28
<i>Acción para visualizar información de un controlador específico, obtiene el objeto controlador por medio del id proporcionado.</i>	
constructor Errorlog:: __construct()	32
constructor Grupo:: __construct()	33
constructor Menu:: __construct()	36
constructor Bitacora_model:: __construct()	65
CatalogoCsv_model	71
<i>Modelo CatalogoCsv</i>	
CatalogoCsv_model::getId()	74

CatalogoCsv_model::getCampos()	73
CatalogoCsv_model::getLLave()	74
CatalogoCsv_model::getMsgError()	74

<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
CatalogoCsv_model::getNombre()	74
CatalogoCsv_model::getByName()	73
<i>Devuelve la informacion de un catalogo por su nombre</i>	
CatalogoCsv_model::getAllData()	73
<i>Devuelve los datos de un catalogo pasado como parametro</i>	
constructor CatalogoCsv_model::construct()	72
CatalogoCsv_model::activaEnCatalogo()	72
<i>Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas</i>	
CatalogoCsv_model::checkPk()	72
<i>Revisa en la base de datos por registros duplicados en los campos pasados por parametro</i>	
CatalogoCsv_model::getAll()	73
<i>Devuelve una lista con los catalogos existentes en la DB</i>	
constructor Graph::construct()	1

E

Enrolamiento_model::get_transaction_relevante()	211
<i>Obtiene las transacciones relevante para la sincronizacion</i>	
Enrolamiento_model::get_sales()	211
<i>Obtiene los registros de sales de rehidratación oral asociados a una persona</i>	
Enrolamiento_model::get_persona_x_tutor()	211
<i>obtiene los tutores de las personas que se envian en la sincronizacion</i>	
Enrolamiento_model::get_version()	212
<i>obtiene cual es la ultima version de apk de la tableta</i>	
Enrolamiento_model::insert()	212
<i>Guarda la persona capturada mediante el formulario web</i>	
Enrolamiento_model::setafiliacion()	212
Enrolamiento_model::setaccion_nutricional()	212
Enrolamiento_model::get_peri_cefa()	210
<i>Obtiene los registros de perimetro cefalico asociados a una persona</i>	
Enrolamiento_model::get_pacientes()	210
<i>devuelve todos los pacientes de la base de datos</i>	
Enrolamiento_model::get_cns_persona()	208
<i>Este metodo obtiene las personas que seran enviadas en la sincronizacion</i>	
Enrolamiento_model::get_cns_cat_persona_count()	208
<i>Hace el count de personas que se envian en la sincronizacion</i>	
Enrolamiento_model::get_control_nutricional()	209
<i>Obtiene los datos del control nutricional asociados a una persona</i>	
Enrolamiento_model::get_datos_grafica()	209
<i>Obtiene los datos especificos de un catálogo para ser visualizados en una gráfica</i>	
Enrolamiento_model::get_notificacion()	210
<i>Este metodo obtiene las notificaciones que se enviarian en la sincronizacion</i>	
Enrolamiento_model::get_estimulacion()	209
<i>Obtiene los registros de estimulacion temprana asociados a una persona</i>	
Enrolamiento_model::setageb()	213
Enrolamiento_model::setalergias()	213
Enrolamiento_model::setcp()	216
Enrolamiento_model::setconsulta()	215
Enrolamiento_model::setcurp()	216

Enrolamiento_model::setcurpT()	216
Enrolamiento_model::setestimulacion_fecha()	217
Enrolamiento_model::setestimulacion_capacitado()	217
Enrolamiento_model::setcompaniaT()	215
Enrolamiento_model::setcompania()	215
Enrolamiento_model::setcalle()	213
Enrolamiento_model::setaltura()	213
Enrolamiento_model::setcelular()	214
Enrolamiento_model::setcelularT()	214
Enrolamiento_model::setcolonia()	215
Enrolamiento_model::setcodigo_barras()	214
Enrolamiento_model::get_cns_cat_persona()	207
<i>Este metodo obtiene los controles que le corresponde a cada persona y que seran incluidas en la sincronizacion</i>	
Enrolamiento_model::get_catalog_view()	207
<i>Hace select de los catalogos que tengan relacion con una persona para mostrarlos en el view</i>	
Enrolamiento_model::getreferencia()	202
Enrolamiento_model::getprecurp()	202
Enrolamiento_model::getpeso()	202
Enrolamiento_model::getRegistro_civil()	202
<i>obtiene informacion del registro civil</i>	
Enrolamiento_model::getsales_cantidad()	203
Enrolamiento_model::getsangre()	203
Enrolamiento_model::getsales_fecha()	203
Enrolamiento_model::getperi_cefa()	202
Enrolamiento_model::getpaternoT()	202
Enrolamiento_model::getnombreT()	201
Enrolamiento_model::getnombre()	201
Enrolamiento_model::getnumero()	201
Enrolamiento_model::getNumRows()	201
<i>Devuelve el numero de filas en la tabla cns_persona</i>	
Enrolamiento_model::getpaterno()	201
Enrolamiento_model::getparto()	201
Enrolamiento_model::getsector()	203
Enrolamiento_model::getsexo()	203
Enrolamiento_model::get_catalog()	205
<i>Hace select de las tablas cns_x que representa a los catalogos</i>	
Enrolamiento_model::getvacuna()	205
Enrolamiento_model::get_catalog2()	205
<i>Obtiene los datos de una tabla</i>	
Enrolamiento_model::get_catalog_count()	206
<i>Obtiene el numero de resultados de una tabla</i>	
Enrolamiento_model::get_catalog_tratamiento()	207
<i>Hace select de los tratamientos de las consultas</i>	
Enrolamiento_model::get_catalog_relevante()	206
<i>obtiene los catalogos relevante x entorno para la sincronizacion</i>	
Enrolamiento_model::getumt()	205
Enrolamiento_model::gettelefonoT()	204
Enrolamiento_model::gettalla()	204
Enrolamiento_model::getsexoT()	204
Enrolamiento_model::gettamiz()	204
Enrolamiento_model::getbeneficiario()	204
Enrolamiento_model::gettelefono()	204

Enrolamiento_model::gettconsulta()	204
Enrolamiento_model::setfaccion_nutricional()	217
Enrolamiento_model::setfconsulta()	217
Enrolamiento_model::update_beneficiario()	229
<i>Actualiza el tipo de beneficiario del paciente</i>	
Enrolamiento_model::update_basico()	229
<i>Actualiza los datos basicos del paciente</i>	
Enrolamiento_model::update_alergia()	229
<i>Actualiza los datos de las alergias del paciente</i>	
Enrolamiento_model::update_consulta()	230
<i>Actualiza el control consulta del paciente</i>	
Enrolamiento_model::update_direccion()	230
<i>actualiza la direccion del paciente</i>	
Enrolamiento_model::update_nutricion()	230
<i>Actualiza el control nutricional del paciente</i>	
Enrolamiento_model::update_estimulacion()	230
<i>Actualiza los registros de estimulacion temprana</i>	
Enrolamiento_model::update_accion()	229
<i>Actualiza el control accion nutricional del paciente</i>	
Enrolamiento_model::tes_pendientes_tarjeta_delete()	229
<i>Elimina los pendientes de las personas que no tengan asignado una tarjeta</i>	
Enrolamiento_model::settconsulta()	227
Enrolamiento_model::settbeneficiario()	227
Enrolamiento_model::settelefono()	227
Enrolamiento_model::settelefonoT()	228
Enrolamiento_model::setvacuna()	228
Enrolamiento_model::setumt()	228
Enrolamiento_model::update_peri_cefa()	231
<i>Actualiza los registros de perimetro cefalico</i>	
Enrolamiento_model::update_regcivil()	231
<i>actualiza el registro civil del paciente</i>	
Estado_tableta_model::getDescripcion()	234
Estado_tableta_model::getById()	233
<i>Obtiene los datos del registro que tiene el ID especificado</i>	
Estado_tableta_model::getId()	234
Estado_tableta_model::getMsgError()	234
<i>Devuelve el mensaje de error,</i>	
<i>en caso de existir un error despues de ejecutar un metodo,</i>	
<i>de lo contrario false</i>	
Estado_tableta_model::setId()	235
Estado_tableta_model::setDescription()	234
Estado_tableta_model::getAll()	233
<i>Obtiene todos los registros de la tabla</i>	
Estado_tableta_model	233
<i>Modelo Estado_tableta</i>	
Enrolamiento_model::update_status_tableta()	231
<i>Hace update de la tableta que este sincronizando dependiendo del resultado</i>	
Enrolamiento_model::update_sales()	231
<i>Actualiza los registros de sales de rehidratacion oral</i>	
Enrolamiento_model::update_tutor()	232
<i>Actualiza los datos del tutor del paciente</i>	
Enrolamiento_model::update_umt()	232
<i>Actualiza la unidad medica tratante del paciente</i>	

Enrolamiento_model::valid_card()	232
<i>Este metodo valida que exista un archivo para enviar a la tarjeta por nfc</i>	
Enrolamiento_model::update_vacuna()	232
<i>Actualiza las vacunas del paciente</i>	
Enrolamiento_model::settamiz()	227
Enrolamiento_model::settalla()	226
Enrolamiento_model::setlugarcivil()	220
Enrolamiento_model::setlocalidad()	220
Enrolamiento_model::setlnacimiento()	220
Enrolamiento_model::setmanzana()	221
Enrolamiento_model::setmaterno()	221
Enrolamiento_model::setnacionalidad()	222
Enrolamiento_model::setmaternoT()	221
Enrolamiento_model::setidtutor()	220
Enrolamiento_model::setld()	219
Enrolamiento_model::setfecha_peri_cefa()	218
Enrolamiento_model::setfechacivil()	218
Enrolamiento_model::setfnacimiento()	218
Enrolamiento_model::setfnutricion()	218
Enrolamiento_model::sethemoglobina()	219
Enrolamiento_model::setfvacuna()	219
Enrolamiento_model::setnombre()	222
Enrolamiento_model::setnombreT()	222
Enrolamiento_model::setsales_fecha()	225
Enrolamiento_model::setsales_cantidad()	225
Enrolamiento_model::setsangre()	225
Enrolamiento_model::setsector()	225
Enrolamiento_model::setsexoT()	226
Enrolamiento_model::setsexo()	226
Enrolamiento_model::setreferencia()	224
Enrolamiento_model::setprecurp()	224
Enrolamiento_model::setparto()	223
Enrolamiento_model::setnumero()	222
Enrolamiento_model::setpaterno()	223
Enrolamiento_model::setpaternoT()	223
Enrolamiento_model::setpeso()	224
Enrolamiento_model::setperi_cefa()	224
Enrolamiento_model::getnacionalidad()	200
Enrolamiento_model::getMsgError()	200
Errorlog_model::resetFilter()	114
<i>Elimina todos los filtros registrados</i>	
Errorlog_model::insert()	114
<i>Inserta en la base de datos la informacion del error</i>	
Errorlog_model::getNumRows()	113
<i>Obtiene el numero total de registros en la tabla Error en caso de existir filtros, estos son aplicados a la consulta</i>	
Errorlog_model::save()	114
<i>Guardar el mensaje de error descriptivo en la base de datos,</i>	
Errorlog_model::setDescripcion()	115
Errorlog_model::setld_accion()	115
Errorlog_model::setld()	115
Errorlog_model::getld_usuario()	113
Errorlog_model::getld_controlador_accion()	113

Errorlog_model::getAll()	112
<i>Obtiene todos los registros de la tabla Error en caso de existir filtros, estos son aplicados a la consulta</i>	
Errorlog_model::addFilter()	112
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Errorlog_model::getById()	112
<i>Obtiene los datos del registro del Error que tiene el ID especificado</i>	
Errorlog_model::getDescripcion()	113
Errorlog_model::getId()	113
Errorlog_model::getFecha_hora()	113
Errorlog_model::setId_controlador()	116
Errorlog_model::setId_controlador_accion()	116
Enrolamiento::checkar_session()	163
<i>Revisa si la sesión está activa</i>	
Enrolamiento::categoriacie10_select()	163
<i>Genera los options de un campo tipo select para las categorías de CIE10</i>	
Enrolamiento::cie10_select()	164
<i>Genera los options de un campo tipo select para los CIE10 correspondientes a una categoría</i>	
Enrolamiento::comparar_view()	164
<i>Crea la pagina para ver la informacion de la persona y compararla con la persona capturada</i>	
Enrolamiento::file_to_card()	165
<i>crea un archivo descargable el cual se necesita para el envio por nfc a la tarjeta del paciente</i>	
Enrolamiento::data_tutor()	164
<i>Obtiene inofrmacion del tutor</i>	
Enrolamiento::catalog_select()	163
<i>Genera los options de un campo tipo select</i>	
Enrolamiento::catalog_check()	162
<i>Crea un grupo de radio o check con la informacion de los catalogos</i>	
Enrolamiento	161
<i>Controlador de enrolamiento</i>	
Errorlog_model::setId_usuario()	116
Enrolamiento::addForm()	161
<i>Pase de parametros para la insercion o actualizacion</i>	
Enrolamiento::autocomplete()	161
<i>Crea el autocomplete para facilitar la busqueda de un tutor</i>	
Enrolamiento::brother_found()	162
<i>Este metodo verifica si un paciente comparte un mismo tutor</i>	
Enrolamiento::brothers_search()	162
<i>Este metodo extrae la informacion de las personas con las que se comparte el mismo tutor si se selecciona una de estas importa los datos para el apartado direccion</i>	
Errorlog_model	111
<i>Modelo Errorlog</i>	
Entorno_model::update()	111
<i>Actualiza el objeto actual en la base de datos</i>	
Entorno_model	105
<i>Modelo Entorno</i>	
Errorlog::view()	33
<i>Muestra los datos del registro especificado por el id</i>	
Errorlog::index()	32
<i>Lista todos los registros de la tabla error, con su correspondiente paginación permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, muestra enlaces para ver detalles de un elemento específico</i>	
Entorno_model::delete()	106

<i>Elimina el registro actual de la base de datos</i>	106
Entorno_model::getAll()	106
<i>Devuelve todos los registros de la tabla entorno</i>	
Entorno_model::getByName()	107
<i>Devuelve la información de un entorno por su nombre</i>	
Entorno_model::getById()	106
<i>Devuelve la información de un entorno por su ID</i>	
Errorlog	32
<i>Controlador Errorlog</i>	
Entorno::ExistEntornoUpdate()	31
<i>Acción para validar que no exista previamente el entorno a actualizar</i>	
Entorno::index()	30
<i>Acción por default del controlador, carga la lista de entornos disponibles y una lista de opciones</i>	
<i>No recibe parámetros</i>	
Entorno::delete()	29
<i>Acción para eliminar un entorno, recibe el id del entorno a eliminar</i>	
Entorno::insert()	30
<i>Acción para preparar la inserción de nuevos entornos , realiza la validación del formulario del lado cliente y del lado servidor para evitar entornos duplicados</i>	
Entorno::update()	30
<i>Acción para preparar la actualización de un entorno ya existente,</i>	
Entorno::ExistEntorno()	31
<i>Acción para validar que no exista previamente el entorno a insertar (Esta acción no puede ser accedida desde el navegador)</i>	
Entorno::view()	31
<i>Acción para visualizar de un entorno específico, obtiene el objeto entorno por medio del id proporcionado.</i>	
Entorno_model::getDescripcion()	107
Entorno_model::getDirectorio()	107
Entorno_model::setDirectorio()	109
Entorno_model::setDescripcion()	109
Entorno_model::setHostname()	110
Entorno_model::setId()	110
Entorno_model::setNombre()	110
Entorno_model::setIp()	110
Entorno_model::insert()	109
<i>Inserta en la tabla entorno la información contenida en el objeto</i>	
Entorno_model::getPermissionsByGroup()	108
<i>Obtiene los permisos asignados al grupo</i>	
Entorno_model::getId()	107

Entorno_model::getHostname()	107
Entorno_model::getInfo()	108
<i>Devuelve la información del objeto en forma de string</i>	
Entorno_model::getIp()	108
Entorno_model::getNombre()	108
Entorno_model::getMsgError()	108
<i>Devuelve los mensajes de error en caso de ocurrir alguna excepción</i>	
<i>'usr' devuelve el mensaje para la vista de usuario</i>	
<i>'log' devuelve el mensaje para el log de errores</i>	
Enrolamiento::ifCupExists()	165
<i>Valida que la cup del paciente no exista</i>	

Enrolamiento::ifCupTExists()	166
<i>valida que la cup del tutor no exista</i>	
Enrolamiento_model::getcupT()	196
Enrolamiento_model::getcup()	196
Enrolamiento_model::getcp()	196
Enrolamiento_model::getestimulacion_capacitado()	197
Enrolamiento_model::getestimulacion_fecha()	197
Enrolamiento_model::getfconsulta()	197
Enrolamiento_model::getfaccion_nutricional()	197
Enrolamiento_model::getControlConsultas()	196
<i>Obtiene todas las consultas asociadas a un paciente</i>	
Enrolamiento_model::getconsulta()	196
Enrolamiento_model::getCIE10()	195
<i>Obtiene el listado de CIE10 correspondientes a una CIE10</i>	
Enrolamiento_model::getcelularT()	195
Enrolamiento_model::getcodigo_barras()	195
Enrolamiento_model::getcolonia()	195
Enrolamiento_model::getcompaniaT()	195
Enrolamiento_model::getcompania()	195
Enrolamiento_model::getfechacivil()	197
Enrolamiento_model::getfecha_peri_cefa()	197
Enrolamiento_model::getlocalidad()	199
Enrolamiento_model::getlnacimiento()	199
Enrolamiento_model::getlugarcivil()	199
Enrolamiento_model::getmanzana()	200
Enrolamiento_model::getmaternoT()	200
Enrolamiento_model::getmaterno()	200
Enrolamiento_model::getListEnrolamiento()	199
<i>Este metodo retorna el list de las personas enroladas</i>	
Enrolamiento_model::getidtutor()	199
Enrolamiento_model::getfnutricion()	198
Enrolamiento_model::getfnacimiento()	198
Enrolamiento_model::getfolio()	198
<i>Extrae el folio que se anexa en el envio para la tarjeta</i>	
Enrolamiento_model::getfvacuna()	198
Enrolamiento_model::getId()	198
Enrolamiento_model::gethemoglobina()	198
Enrolamiento_model::getcelular()	194
Enrolamiento_model::getCategoriaCIE10()	194
<i>Obtiene el listado de categorias de CIE10</i>	
Enrolamiento::validarForm()	170
<i>valida los datos de entrada en el formulario</i>	
Enrolamiento::vacunacion()	169
<i>Obtiene el historial de vacunacion de un paciente cuyo id es pasado por parametro</i>	
Enrolamiento::update_card()	169
<i>Este metodo actualiza el estado del archivo descargado si fue escrito correctamente o no en la tarjeta</i>	
Enrolamiento::validarisum()	170
<i>valida que el nodo seleccionado en el arbol sea una unidad medica</i>	
Enrolamiento::validate_card()	170
<i>valida que un archivo sea valido para enviar a la tarjeta por nfc</i>	
Enrolamiento_model	189
<i>Modelo Usuario</i>	

Enrolamiento::view()	171
<i>Crea la pagina para ver la informacion de la persona</i>	
Enrolamiento::update()	169
<i>Crea el formulario para editar la informacion de la persona</i>	
Enrolamiento::tratamiento_select()	168
<i>Genera los options de un campo tipo select para los tratamientos de consultas</i>	
Enrolamiento::insert()	166
<i>prepara los datos para insertarlos</i>	
Enrolamiento::index()	166
<i>Este es el metodo por default, obtiene el listado de las personas</i>	
Enrolamiento::paciente_similar()	167
<i>Comprueba la similitud de un paciente que se este capturando con los que ya existe en la base de datos,</i>	
<i>esto con la finalidad de disminuir datos repetidos</i>	
Enrolamiento::print_card()	167
<i>Este metodo extrae la informacion del paciente que sera impreso en la tarjeta</i>	
Enrolamiento::searchum()	168
<i>Busca dentro del catalogo asu_ageb la unidad médica de acuerdo a la localidad y ageb</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Enrolamiento::searchageb()	167
<i>Regresa un objeto JSON con la lista de agebs disponibles en la localidad</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Enrolamiento_model::autocomplete_tutor()	190
<i>obtiene informacion del tutor para genberar el autocomplete</i>	
Enrolamiento_model::cns_insert()	190
<i>Hace insert de las tablas cns_control_x que se reciben en la sincronizacion secuencial</i>	
Enrolamiento_model::getalergias()	193
Enrolamiento_model::getAlergia()	193
<i>Obtiene las alergias asociadas a una persona</i>	
Enrolamiento_model::getaltura()	193
Enrolamiento_model::getByCurp()	193
<i>valida que no se repita la curp en personas y tutor</i>	
Enrolamiento_model::getcalle()	194
Enrolamiento_model::getById()	194
<i>Obtiene la informacion de la persona</i>	
Enrolamiento_model::getageb()	192
Enrolamiento_model::getAfiliaciones()	192
<i>Obtiene las afiliaciones asociadas a una persona</i>	
Enrolamiento_model::cns_update_visita()	191
Enrolamiento_model::cns_update()	190
<i>Actualiza la tabla especificada</i>	
Enrolamiento_model::data_tutor()	191
<i>obtiene informacion del tutor</i>	
Enrolamiento_model::entorno_x_persona()	191
<i>Este metodo actualiza o inserta los datos que permiten el envio de la informacion a la tarjeta por nfc</i>	
Enrolamiento_model::getafiliacion()	192
Enrolamiento_model::getaccion_nutricional()	192
Entorno	29
<i>Controlador Entorno</i>	

G

Grupo_model::getId()	120
Grupo_model::getMsgError()	120
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Grupo_model::getEntornosByld()	120
<i>Obtiene el grupo solicitado con sus entornos vinculados</i>	
Grupo_model::getDescripcion()	120
Grupo_model::getByld()	119
<i>Obtiene el grupo solicitado</i>	
Grupo_model::getByName()	119
<i>Obtiene el grupo solicitado</i>	
Grupo_model::getNombre()	121
Grupo_model::getNumRows()	121
<i>Obtiene el numero total de grupos</i>	
Grupo_model::setNombre()	122
Grupo_model::update()	122
<i>Actualiza en la base de datos los datos del grupo (datos en propiedades)</i>	
Grupo_model::setId()	122
Grupo_model::setDescription()	121
Grupo_model::insert()	121
<i>Inserta en la base de datos los datos del grupo (datos en propiedades)</i>	
Grupo_model::getAll()	118
<i>Obtiene todos los grupos existentes</i>	
Grupo_model::delete()	118
<i>Elimina de la base de datos al grupo (id en propiedades)</i>	
Grupo::index()	34
1) <i>Visualiza los grupos existentes para su interacción CRUD</i>	
2) <i>En caso de detectar un texto a buscar se filtran los grupos existentes acorde a la búsqueda</i>	
Grupo::insert()	34
1) <i>Prepara el formulario para la inserción de un grupo nuevo</i>	
2) <i>Realiza las validaciones necesarias sobre cada campo del registro</i>	
Grupo::delete()	34
<i>Solicita la eliminación del grupo recibido</i>	
Grupo	33
<i>Controlador Grupo</i>	
Graph::graph_init()	1
<i>Crea una grafica en el lugar que se llame</i>	
Graph::map()	2
<i>crea un objeto mapa con la ayuda de la api de google</i>	
Grupo::update()	34
1) <i>Prepara el formulario para la modificación de un grupo existente</i>	
2) <i>Realiza las validaciones necesarias sobre cada campo del registro</i>	
Grupo::view()	35
<i>Visualiza los datos del grupo recibido</i>	
Georeferencia_model::process()	117
<i>Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia</i>	
Grupo_model	118
<i>Modelo Grupo</i>	
Georeferencia_model::getMsgError()	117
<i>Devuelve el mensaje de error,</i>	

en caso de existir un error despues de ejecutar un metodo, de lo contrario false	
Georeferencia_model	117
Modelo Georeferencia	
Grupo::ifGroupExists()	35
Callback para validar que un nombre de grupo no se duplique	
Graph	1
Controlador Objetos	

H

Hemoglobina_model::process()	123
Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia	
Hemoglobina_model::getMsgError()	123
Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false	
Hemoglobina_model	122
Modelo Hemoglobina	

I

Index::index()	160
Index	160

M

Menu_model::hasChild()	128
Verifica si el nodo actual tiene hijos	
Menu_model::insert()	128
Inserta en la base de datos, la informacion contenida en el objeto	
Menu_model::resetFilter()	128
Elimina todos los filtros registrados	
Menu_model::getRuta()	128
Menu_model::getNumRows()	127
Obtiene el numero total de registros en la tabla Menu en caso de existir filtros, estos son aplicados a la consulta	
Menu_model::getMsgError()	127
Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false	
Menu_model::getNombre()	127
Menu_model::setAtributo()	129
Menu_model::setId()	129
Menu_model::setRuta()	130
Menu_model::update()	131
Actualiza los datos del objeto actual	
Menu_model::setNombre()	130
Menu_model::setId_raiz()	130

Menu_model::setId controlador()	129
Menu_model::setId padre()	129
Menu_model::getId raiz()	127
Menu_model::getId padre()	127
Menu::view()	38
<i>Muestra los datos del registro especificado por el id</i>	
Menu_model	124
<i>Modelo Menu</i>	
Menu_model::addFilter()	124
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Menu::update()	37
<i>Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos</i>	
Menu::insert()	37
<i>Muestra el formulario para crear un nuevo registro en la menu, las variables se obtienen por el metodo POST</i>	
Menu::delete()	36
<i>Eliminar el registro especificado por el id</i>	
Menu::index()	36
<i>Lista todos los registros de la menu, con su correspondiente paginación</i>	
Menu_model::delete()	124
<i>Elimina el registro actual de la base de datos</i>	
Menu_model::deleteByFilter()	125
<i>Elimina el conjunto de registros que cumplen con el o los criterios de filtrado</i>	
Menu_model::getId()	126
Menu_model::getId controlador()	126
Menu_model::getByPadre()	126
<i>Obtiene todos los nodos hijos de un padre</i>	
Menu_model::getById()	126
<i>Obtiene los datos del registro de la menu que tiene el ID especificado</i>	
Menu_model::getAll()	125
<i>Obtiene todos los registros de la tabla Menu en caso de existir filtros, estos son aplicados a la consulta</i>	
Menu_model::getAtributo()	125
Menu	36
<i>Controlador Menu</i>	

N

Notificacion_model::getTitulo()	238
Notificacion_model::insert()	238
<i>Inserta en la base de datos los datos de la notificación (datos en propiedades)</i>	
Notificacion_model::getNumRows()	238
<i>Obtiene el numero total de notificaciones</i>	
Notificacion_model::getMsgError()	238
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Notificacion_model::getIdsTabletas()	237
Notificacion_model::setContenido()	239
Notificacion_model::setFechaFin()	239
Notificacion_model::setTitulo()	240
Notificacion_model::update()	240
<i>Actualiza en la base de datos los datos de la notificación (datos en propiedades)</i>	

Notificacion_model::setIdsTabletas()	240
Notificacion_model::setId()	239
Notificacion_model::setFechaInicio()	239
Notificacion_model::getId()	237
Notificacion_model::getFechaInicio()	237
Notificacion::update()	173
1) Prepara el formulario para la modificación de una notificación existente	
2) Realiza las validaciones necesarias sobre cada campo del registro	
Notificacion::view()	173
Visualiza los datos de la notificación recibida	
Notificacion::insert()	172
1) Prepara el formulario para la inserción de una notificación nueva	
2) Realiza las validaciones necesarias sobre cada campo del registro	
Notificacion::index()	172
1) Visualiza las notificaciones existentes para su interacción CRUD	
2) En caso de detectar un texto a buscar se filtran las notificaciones existentes acorde a la búsqueda	
Notificacion::delete()	172
Solicita la eliminación de la notificación recibida	
Notificacion_model	235
Modelo Usuario	
Notificacion_model::addFilter()	235
Agrega una nueva regla de filtrado al arreglo de filtros	
Notificacion_model::getContenido()	237
Notificacion_model::getFechaFin()	237
Notificacion_model::getById()	236
Obtiene la notificación solicitada	
Notificacion_model::getAll()	236
Obtiene todas las notificaciones existentes, se puede filtrar por: texto a buscar si se desea	
Notificacion_model::delete()	236
Elimina de la base de datos la notificación (id en propiedades)	
Notificacion	171
Controlador Notificación	

O

Obtenercurp::curp()	5
Consulta si la curp existe en la base de datos de la condusef	
Obtenercurp::calcular_curp()	4
Calcula la curp y el rfc con los datos proporcionados	
Obtenercurp::calcularcurp()	4
Calcula la curp y el rfc con los datos proporcionados	
Obtenercurp::\$estados	3
Arreglo con los estados y sus abreviaturas, sirven para el cálculo de la CURP	
Obtenercurp	2
Controlador Objeto	

P

Permiso_model::setId()	134
Permiso_model::setFecha()	133

Permiso_model::insertBatch()	133
<i>Inserta en la base de datos el arreglo de permisos recibido</i>	
Permiso_model::setIdControladorAccion()	134
Permiso_model::setIdGrupo()	134
Poblacion_model::process()	135
<i>Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion</i>	
Poblacion_model::getMsgError()	135
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Poblacion_model	135
<i>Modelo Poblacion</i>	
Permiso_model::getPermission()	133
<i>Obtiene el permiso solicitado</i>	
Permiso_model::getMsgError()	132
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Permiso_model::deletePermissions()	131
<i>Elimina de la base de datos los permisos del entorno y grupo recibidos</i>	
Permiso_model	131
<i>Modelo Permiso</i>	
Permiso::index()	38
<i>1) Visualiza los entornos existentes para su selección</i>	
Permiso_model::getFecha()	132
Permiso_model::getId()	132
Permiso_model::getIdGrupo()	132
Permiso_model::getIdControladorAccion()	132
Permiso	38
<i>Controlador Permiso</i>	

R

Reporteador	173
<i>Controlador Reporteador</i>	
Reporteador::index()	174
<i>Visualiza los reportes existentes</i>	
ReglaVacuna_model::update()	148
<i>Actualiza el objeto actual en la base de datos</i>	
ReglaVacuna_model::setRegion()	147
ReglaVacuna_model::setOrdenEsqComp()	147
Reporteador::view()	174
<i>Renderiza la vista del reporte</i>	
Reporte_sincronizacion	175
<i>Controller Usuario</i>	
Reporte_sincronizacion::view()	176
<i>Muestra detallada por cada list del reporte de sincronizacion</i>	
Reporte_sincronizacion::lote_view()	175
<i>Muestra detallada por cada list del reporte de lote de vacunacion</i>	
Reporte_sincronizacion::lote()	175
<i>Genera el item del reporte de lote de vacunacion</i>	
Reporte_sincronizacion::index()	175
<i>Este es el metodo por default, crea el listado del reporte de sincronizacion</i>	

ReglaVacuna_model::setObservacionRegion()	147
ReglaVacuna_model::setIdViaVacuna()	147
ReglaVacuna_model::setDiaInicioNacido()	144
ReglaVacuna_model::setDiaInicioPrevia()	145
ReglaVacuna_model::setDiaFinPrevia()	144
ReglaVacuna_model::setDiaFinNacido()	144
ReglaVacuna_model::setAlergias()	143
ReglaVacuna_model::setDosis()	145
ReglaVacuna_model::setEsqComp()	145
ReglaVacuna_model::setIdVacunaPrevia()	146
ReglaVacuna_model::setIdVacuna()	146
ReglaVacuna_model::setId()	146
ReglaVacuna_model::setForzarAplicacion()	145
Reporteador_model	241
<i>Modelo Reporteador</i>	
Reporteador_model::getCensoNominal()	241
Reporte_censo_nominal::\$parto_multiple	245
Reporte_censo_nominal::\$sexo	245
Reporte_censo_nominal::\$nombre	245
Reporte_censo_nominal::\$fecha_nacimiento	245
Reporte_censo_nominal::\$domicilio	244
Reporte_censo_nominal::\$vacunas	245
Reporte_sincronizacion_model	246
<i>Modelo Usuario</i>	
Reporte_sincronizacion_model::get_version()	247
<i>obtiene la ultima version de la apk de las tabletas</i>	
Reporte_sincronizacion_model::getMsgError()	247
Reporte_sincronizacion_model::getListado()	247
<i>Obtiene el resultado de una consulta</i>	
Reporte_sincronizacion_model::getCount()	246
<i>obtiene el numero de registros de una tabla o consulta</i>	
Reporte_censo_nominal::\$sculp	244
Reporte_censo_nominal::\$apellido_paterno	244
Reporteador_model::getGrupoVacunas()	242
Reporteador_model::getMsgError()	242
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Reporteador_model::getEsquemasIncompletos()	242
Reporteador_model::getConcentradoActividades()	242
Reporteador_model::getCoberturaBiologicoListado()	241
<i>Obtiene el reporte de cobertura por tipo de biologico</i>	
Reporteador_model::getSeguimientoRV1RV5()	243
Reporteador_model::getVacunas()	243
Reporte_censo_nominal::\$apellido_materno	244
Reporte_censo_nominal	244
<i>Modelo Reporte_censo_nominal</i>	
Reporteador_model::object to array()	243
Reporteador_model::getVacunasByGrupo()	243
ReglaVacuna_model::insert()	143
<i>Inserta en la tabla regla_vacuna la información contenida en el objeto</i>	
ReglaVacuna_model::getRegion()	143
ReglaVacuna::update()	45
<i>Acción para preparar la actualización de una regla ya existente,</i>	
ReglaVacuna::view()	46

Acción para visualizar información de una regla específica, obtiene el objeto regla_vacuna por medio del id proporcionado.	
ReglaVacuna::insert()	45
Acción para preparar la insercion de nuevas reglas , realiza la validación del formulario del lado cliente	
ReglaVacuna::index()	45
Acción por default del controlador, carga la lista de reglas de vacunas disponibles y una lista de opciones	
No recibe parámetros	
ReglaVacuna::delete()	44
Acción para eliminar una regla, recibe el id de la regla a eliminar	
Raiz_model	136
Modelo Raiz	
Raiz_model::delete()	136
Elimina el registro actual de la base de datos	
Raiz_model::getDescripcion()	137
Raiz_model::getById()	137
Devuelve la información de una raiz por su ID	
Raiz_model::getAll()	137
Devuelve todos los registros de la tabla raiz	
Raiz_model::ExistInArbol()	136
Revisa si la raiz pasada como parametro existe en el ASU	
ReglaVacuna	44
Controlador ReglaVacuna	
Raiz::view()	43
Acción para visualizar información de una raiz específica, obtiene el objeto raiz por medio del id proporcionado.	
Raiz::getDataKeyValue()	41
Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro	
Raiz::getDataTreeFromId()	41
Accion para regresar la descripción e informacion adicional de un arreglo de ID's desde el arbol de segmentacion	
Raiz::getChildrenFromLevel()	40
Accion para regresar el arbol de segmentacion determinado, el objeto regresado contiene estructura de arbol y es consumida solamente por peticiones AJAX	
Raiz::delete()	40
Acción para eliminar una raiz, recibe el id de la raiz a eliminar	
Raiz::createasu()	39
Acción para crear el ASU a partir de una raiz	
Solo se permite su acceso por medio de peticiones AJAX	
Raiz::getTreeBlock()	41
Sirve para obtener bloques del arbol de segmentación única ASU	
Raiz::index()	42
Acción por default del controlador, carga la lista de Raices disponibles y una lista de opciones	
No recibe parámetros	
Raiz::updateasu()	43
Acción para actualizar el ASU a partir de una raiz	
Solo se permite su acceso por medio de peticiones AJAX	
Raiz::update()	43
Acción para preparar la actualización de una raiz ya existente,	
Raiz::insert()	42
Acción para preparar la inserción de nuevas acciones , realiza la validación	

del formulario del lado cliente	
Raiz::iniciarasu()	42
Crea los archivos JSON necesarios para iniciar el ASU en caché y agilizar su carga	
Raiz_model::getId()	138

Raiz_model::getMsgError()	138
Devuelve los mensajes de error en caso de ocurrir alguna excepción	
'usr' devuelve el mensaje para la vista de usuario	
'log' devuelve el mensaje para el log de errores	
ReglaVacuna_model::getForzarAplicacion()	142
ReglaVacuna_model::getId()	142

ReglaVacuna_model::getEsqComp()	141
ReglaVacuna_model::getDosis()	141
ReglaVacuna_model::getDialInicioPrevia()	141
ReglaVacuna_model::getIdVacuna()	142
ReglaVacuna_model::getIdVacunaPrevia()	142
ReglaVacuna_model::getOrdenEsqComp()	143
ReglaVacuna_model::getObservacionRegion()	143
ReglaVacuna_model::getMsgError()	142
Devuelve los mensajes de error en caso de ocurrir alguna excepción	
'usr' devuelve el mensaje para la vista de usuario	
'log' devuelve el mensaje para el log de errores	
ReglaVacuna_model::getIdViaVacuna()	142
ReglaVacuna_model::getDialInicioNacido()	141
ReglaVacuna_model::getDiaFinPrevia()	141
Raiz_model::update()	139
Actualiza el objeto actual en la base de datos	
Raiz_model::setId()	139
Raiz_model::setDescripcion()	138
Raiz_model::insert()	138
Inserta en la tabla raiz, la información contenida en el objeto	
ReglaVacuna_model	139
Modelo ReglaVacuna	
ReglaVacuna_model::delete()	140
Elimina el registro actual de la base de datos	
ReglaVacuna_model::getDiaFinNacido()	141
ReglaVacuna_model::getByld()	140
Devuelve la información de una regla de vacuna por su ID	
ReglaVacuna_model::getAll()	140
Devuelve todos los registros de la tabla regla_vacuna	
ReglaVacuna_model::getAlergias()	140
Raiz	39
Controlador Raiz	

S

Semana_nacional_model::getByld()	249
Obtiene los datos del registro que tiene el ID especificado	
Semana_nacional_model::getDescripcion()	249
Semana_nacional_model::getFecha_fin()	249
Semana_nacional_model::getAll()	248

Obtiene todos los registros de la tabla	248
Semana_nacional_model::delete()	248
Elimina el registro actual de la base de datos	
Servicios::Synchronization()	184
Metodo principal al que se le hacen las peticiones y es el que se encarga de distribuir la informacion	
Semana_nacional_model	248
Modelo Tableta	
Semana_nacional_model::getFecha_inicio()	249
Semana_nacional_model::getId()	250
Semana_nacional_model::setFecha_fin()	251
Semana_nacional_model::setFecha_inicio()	251
Semana_nacional_model::update()	251
Actualiza los datos del objeto actual	
Semana_nacional_model::setDescripcion()	250
Semana_nacional_model::insert()	250
Inserta en la base de datos, la informacion contenida en el objeto	
Semana_nacional_model::getMsgError()	250
Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false	
Semana_nacional_model::getNumRows()	250
Obtiene el numero total de registros en la tabla	
Servicios::ss_step_6()	183
prepara los datos para la sincronizacion secuencia, envia unicamente aquellos datos modificados despues de la ultima sincronizacion de la tableta	
Servicios::ss_step_5()	183
Recibe los datos que genero la tableta para ser almacenados en la base de datos del servidor	
Semana_nacional::update()	178
Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos	
Semana_nacional::view()	178
Muestra los datos del registro especificado por el id	
Servicios	179
Controlador Servicios	
Semana_nacional::insert()	178
Muestra el formulario para crear un nuevo registro en la semana_nacional, las variables se obtienen por el metodo POST	
Semana_nacional::index()	177
Lista todos los registros de semanas nacional, con su correspondiente paginación permite eliminar un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico	
Semana_nacional::delete()	177
Eliminar el registro especificado por el id	
Semana_nacional::getAll()	177
Devuelve un json con todos los registros de semanas nacional	
Servicios::actualiza_estado_tableta()	179
Actualiza el estatus de la tableta	
Servicios::catalogos_relevantes()	180
Genera los catalogos relevantes por entorno	
Servicios::is_step_3()	182
Guarda en la base de datos el estado de la sincronizacion	
Servicios::is_step_4()	182

<i>Prepara la informacion de perssonas con su catalogos transaccionales de cada una</i>	
Servicios::prueba2()	183
Servicios::is_step_2()	181
<i>Valida que la session este activa, genera los catalogos a enviar en la sincronizacion por primera vez</i>	
Servicios::is_step_1()	181
<i>valida que la tableta este asignada a una unidad medica y que tenga un status valido para la sincronizacion</i>	
Servicios::esquema_incompleto()	180
<i>Genera los esquemas incompletos de las personas que correspondan a la unidad medica de la tableta</i>	
Servicios::is_step_0()	180
<i>Paso 0 se procesa las peticiones segun la accion:</i>	
Semana_nacional	176
<i>Controlador Semana Nacional</i>	

T

Tableta_model::setId_asu_um()	256
Tableta_model::setId_tes_estado_tableta()	257
Tableta_model::setId_tipo_censo()	257
Tableta_model::setMac()	257
Tableta_model::setIdVersion()	256
Tableta_model::setId()	256
Tableta_model::getPeriodo_esq_inc()	255
Tableta_model::getUltima_actualizacion()	255
Tableta_model::getUsuarios_asignados()	255
Tableta_model::insert()	256
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Tableta_model::setPeriodo_esq_inc()	258
Tableta_model::setUltima_actualizacion()	258
Tipo_censo_model::getId()	260
Tipo_censo_model::getMsgError()	260
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Tipo_censo_model::setDescripcion()	261
Tipo_censo_model::setId()	261
Tipo_censo_model::getDescripcion()	260
Tipo_censo_model::getById()	259
<i>Obtiene los datos del registro que tiene el ID especificado</i>	
Tableta_model::setUsuarios_asignados()	258
Tableta_model::update()	258
<i>Actualiza los datos del objeto actual</i>	
Tipo_censo_model	259
<i>Modelo Tipo_censo</i>	
Tipo_censo_model::getAll()	259
<i>Obtiene todos los registros de la tabla</i>	
Tableta_model::getNumRows()	255
<i>Obtiene el numero total de registros en la tabla en caso de existir filtros, estos son aplicados a la consulta</i>	
Tableta_model::getMsgError()	255

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false	
Tableta::update()	186
Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos	
Tableta::uploadFile()	186
Registra tabletas desde un archivo csv	
Tableta::view()	187
Muestra los datos del registro especificado por el id	
Tableta:: validateMac()	187
Valida si existe una MAC en la base de datos	
Tableta::setUM()	186
Asignar unidad medica y tipo de censo	
Tableta::insert()	186
Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST	
Tree::create()	6
Crea el arbol y lo muestra en la view	
Tableta	184
Controlador Tableta	
Tableta::delete()	185
Eliminar el registro especificado por el id	
Tableta::index()	185
Lista todos los registros de tabletas, con su correspondiente paginación permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento específico	
Tableta_model	252
Modelo Tableta	
Tableta_model::delete()	252
Elimina el registro actual de la base de datos	
Tableta_model::getId_asu_um()	254
Tableta_model::getId_tes_estado_tableta()	254
Tableta_model::getId_tipo_censo()	254
Tableta_model::getMac()	254
Tableta_model::getIdVersion()	254
Tableta_model::getId()	254
Tableta_model::getAll()	253
Obtiene todos los registros de la tabla	
Tableta_model::getById()	253
Obtiene los datos del registro que tiene el ID especificado	
Tableta_model::getByMac()	253
Obtiene los datos del registro la MAC especificada	
Tree	5
Controlador Objeto	

U

Usuario_model::setApellidoMaterno()	156
Usuario_model::setActivo()	155
Usuario_model::insert()	155
Inserta en la base de datos los datos del usuario (datos en propiedades)	

Usuario_model::setApellidoPaterno()	156
Usuario_model::setClave()	156
Usuario_model::setIdGrupo()	157
Usuario_model::setId()	157
Usuario_model::setCorreo()	156
Usuario_model::get_usuario_entorno()	155
Usuario_model::get_permiso_entorno()	155
Usuario_model::getNombreUsuario()	153
Usuario_model::getNombre()	153
Usuario_model::getMsgError()	152
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Usuario_model::getNumRows()	153
<i>Obtiene el numero total de usuarios</i>	
Usuario_model::getOnlyActives()	153
<i>Obtiene todos los usuarios existentes, se puede filtrar por: texto a buscar o solo activos si se desea</i>	
Usuario_model::get_grupo_entorno()	154
Usuario_model::getuser()	154
Usuario_model::setNombre()	157
Usuario_model::setNombreUsuario()	158
Usuario_tableta_model::getTabletasByUsuario()	263
<i>Obtiene el id de todas las tabletas relacionadas con el usuario especificado</i>	
Usuario_tableta_model::getTableta()	262
Usuario_tableta_model::getMsgError()	262
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Usuario_tableta_model::getUsuario()	263
Usuario_tableta_model::getUsuariosByTableta()	263
<i>Obtiene el id de todos los usuarios asignados a la tableta especificada</i>	
Usuario_tableta_model::setUsuario()	264
Usuario_tableta_model::setTableta()	264
Usuario_tableta_model::insert()	263
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Usuario_tableta_model::delete()	262
<i>Elimina la relacion entre usuario y tableta</i>	
Usuario_tableta_model	261
<i>Modelo Estado_tableta</i>	
Usuario_model::update_user()	158
Usuario_model::update_pass()	158
Usuario_model::update()	158
<i>Actualiza en la base de datos los datos del usuario (datos en propiedades)</i>	
Usuario_tableta	187
<i>Controlador Usuario_tableta</i>	
Usuario_tableta::delete()	188
<i>Eliminar un usuario de una tableta especifica</i>	
Usuario_tableta::insert()	189
<i>Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST</i>	
Usuario_tableta::index()	188
<i>Lista todos los registros de usuarios correspondientes a una tableta en especifico permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico</i>	

Usuario_model::getIdGrupo()	152
Usuario_model::getId()	152
Usuario::login()	49
<i>Ofrece el inicio de sesión</i>	
Usuario::load_update()	49
Usuario::insert()	48
1) <i>Prepara el formulario para la inserción de un usuario nuevo</i>	
2) <i>Realiza las validaciones necesarias sobre cada campo del registro</i>	
Usuario::logout()	49
<i>Termina la sesión</i>	
Usuario::remember()	49
Usuario::token()	50
Usuario::send_mail()	50
Usuario::reset()	49
Usuario::index()	48
1) <i>Visualiza los usuarios existentes para su interacción CRUD</i>	
2) <i>En caso de detectar un texto a buscar se filtran los usuarios existentes acorde a la búsqueda</i>	
Usuario::get_token()	48
<i>Acción para obtener el token oculto en el login</i>	
Usuario::cerrar_etab()	47
<i>Acción para cerrar el login en otro sistema</i>	
Usuario::automatic_access()	47
Usuario::\$mas	46
<i>Acción para hacer autologin en otro sistema</i>	
Usuario::delete()	47
<i>Solicita la eliminación del usuario recibido</i>	
Usuario::form_init()	47
Usuario::get_galleta()	48
<i>Obtiene las cookies que se generan en la session</i>	
Usuario::getActivesByGroup()	47
<i>Acción para servir un array de objetos con los usuarios activos por grupo</i>	
<i>AJAX y devuelve un objeto JSON</i>	
Usuario::update()	50
1) <i>Prepara el formulario para la modificación de un usuario existente</i>	
2) <i>Realiza las validaciones necesarias sobre cada campo del registro</i>	
Usuario::update_info()	51
Usuario_model::getApellidoPaterno()	151
Usuario_model::getApellidoMaterno()	151
Usuario_model::getActivo()	150
Usuario_model::getById()	151
<i>Obtiene el usuario solicitado, se puede obtener el registro normal o personalizado para visualización (descripciones en tablas vinculadas)</i>	
Usuario_model::getByUsername()	151
<i>Obtiene el usuario solicitado</i>	
Usuario_model::getgrupo()	152
Usuario_model::getCorreo()	152
Usuario_model::getClave()	152
Usuario_model::getActivesByGroup()	150
<i>Obtiene los usuarios activos del grupo solicitado</i>	
Usuario_model::delete()	150
<i>Elimina de la base de datos al usuario (id en propiedades)</i>	

<u>Usuario_model</u>	148
<i>Modelo Usuario</i>	
<u>Usuario::ifUserExists()</u>	51
<i>Callback para validar que un nombre de usuario no se duplique</i>	
<u>Usuario::view()</u>	51
<i>Visualiza los datos del usuario recibido</i>	
<u>Usuario_model::authenticate()</u>	148
<i>Valida las credenciales recibidas</i>	
<u>Usuario_model::checkCredentials()</u>	149
<i>Verifica que el usuario haya iniciado sesión y además tenga permiso en la acción recibida</i>	
<u>Usuario_model::check_token()</u>	150
<u>Usuario_model::check_data()</u>	149
<u>Usuario</u>	46
<i>Controlador Usuario</i>	