

Documentación SIIGS/TES



Contenido

Paquete default Clases	1
Clase Index	1
Constructor construct	1
Método index	1
Paquete CodeIgniter Elementos procedurales	3
Form validation.php	3
Pagination.php	4
template.php	5
Paquete CodeIgniter Clases	6
Clase CI Form validation	6
Var \$CI	6
Var \$error_string	6
Var \$ config_rules	7
Var \$ error_array	7
Var \$ error_messages	7
Var \$ error_prefix	7
Var \$ error_suffix	7
Var \$ field_data	8
Var \$ safe form data	8
Constructor construct	8
Método alpha	8
Método alpha_dash	9
Método alpha_numeric	9
Método decimal	9
Método encode_php_tags	10
Método error	10
Método error_string	10
Método exact_length	11
Método greater_than	11
Método integer	12
Método is_natural	12
Método is_natural_no_zero	12
Método is_numeric	13
Método is_unique	13
Método less_than	13
Método matches	14
Método max_length	14
Método min_length	14
Método numeric	15
Método prep_for_form	15
Método prep_url	15
Método regex_match	16
Método required	16

Método run	17
Método set_checkbox	17
Método set_error_delimiters	17
Método set_message	18
Método set_radio	18
Método set_rules	19
Método set_select	19
Método set_value	19
Método strip_image_tags	20
Método valid_base64	20
Método valid_email	21
Método valid_emails	21
Método valid_ip	21
Método xss_clean	22
Método execute	22
Método reduce_array	22
Método reset_post_array	23
Método translate_fieldname	23
Clase CI_Pagination	23
Var \$anchor_class	24
Var \$base_url	24
Var \$cur_page	24
Var \$cur_tag_close	24
Var \$cur_tag_open	24
Var \$display_pages	24
Var \$first_link	24
Var \$first_tag_close	24
Var \$first_tag_open	24
Var \$first_url	24
Var \$full_tag_close	25
Var \$full_tag_open	25
Var \$last_link	25
Var \$last_tag_close	25
Var \$last_tag_open	25
Var \$next_link	25
Var \$next_tag_close	25
Var \$next_tag_open	25
Var \$num_links	25
Var \$num_tag_close	25
Var \$num_tag_open	25
Var \$page_query_string	25
Var \$per_page	25
Var \$prefix	25
Var \$prev_link	26
Var \$prev_tag_close	26
Var \$prev_tag_open	26
Var \$query_string_segment	26
Var \$suffix	26
Var \$total_rows	26

Var \$uri_segment	26
Var \$use_page_numbers	26
Constructor construct	26
Método create_links	26
Método initialize	27
Clase CI_Template	27
Var \$CI	28
Var \$config	28
Var \$css	28
Var \$js	28
Var \$master	28
Var \$output	28
Var \$parser	28
Var \$parser_method	28
Var \$parse_template	28
Var \$regions	28
Var \$template	28
Constructor CI_Template	28
Método add_css	29
Método add_js	29
Método add_region	29
Método add_template	30
Método empty_region	30
Método initialize	30
Método load	31
Método parse_view	31
Método render	32
Método set_master_template	32
Método set_parser	32
Método set_parser_method	33
Método set_regions	33
Método set_template	33
Método write	34
Método write_view	34
Paquete Libreria Clases	35
Clase Menubuilder	35
Constructor construct	35
Método build	35
Método crearMenu	36
Método isGranted	36
Clase Graph	37
Constructor construct	37
Método graph_init	37
Método map	37
Clase Obtenercurp	38
Var \$estados	39
Constructor construct	39
Método calculacurp	39
Método calcular_curp	40

<u>Método curp</u>	40
<u>Clase Tree</u>	41
<u>Constructor</u> <u>construct</u>	41
<u>Método create</u>	41
<u>Paquete Session Clases</u>	43
<u>Clase Session</u>	43
<u>Var \$ci</u>	43
<u>Var \$flashdata_key</u>	43
<u>Var \$sess_expiration</u>	44
<u>Var \$sess_namespace</u>	44
<u>Var \$store</u>	44
<u>Constructor</u> <u>construct</u>	44
<u>Método all_userdata</u>	44
<u>Método flashdata</u>	45
<u>Método is_expired</u>	45
<u>Método keep_flashdata</u>	45
<u>Método sess_create</u>	46
<u>Método sess_destroy</u>	46
<u>Método set_flashdata</u>	46
<u>Método set_userdata</u>	46
<u>Método unset_userdata</u>	47
<u>Método userdata</u>	47
<u>Paquete SIIGS Clases</u>	48
<u>Clase Accion</u>	48
<u>Constructor</u> <u>construct</u>	48
<u>Método delete</u>	48
<u>Método index</u>	49
<u>Método insert</u>	49
<u>Método update</u>	49
<u>Método view</u>	50
<u>Clase Ayuda</u>	50
<u>Constructor</u> <u>construct</u>	50
<u>Método index</u>	50
<u>Clase Bitacora</u>	51
<u>Constructor</u> <u>construct</u>	51
<u>Método index</u>	51
<u>Método validateExistUsuario</u>	52
<u>Método view</u>	52
<u>Clase Catalogo</u>	53
<u>Constructor</u> <u>construct</u>	53
<u>Método checkpk</u>	53
<u>Método checkTypeData</u>	53
<u>Método delete</u>	54
<u>Método index</u>	54
<u>Método insert</u>	54
<u>Método load</u>	55
<u>Método loadupdate</u>	55
<u>Método update</u>	55

Método view	56
Método array unique recursive	56
Clase CatalogoCsv	57
Constructor construct	57
Método ActivaEnCatalogo	57
Método checkpk	58
Método createTableAgeb	58
Método createTableGeo	58
Método createTableHemoGlobina	58
Método createTablePob	59
Método index	59
Método loadupdate	59
Método update	59
Método view	60
Método array unique recursive	60
Clase Catalogo x raiz	61
Constructor construct	61
Método check	61
Método delete	62
Método insert	62
Método view	62
Clase Cie10	63
Constructor construct	63
Método ActivaEnCatalogo	63
Método AgregaEnCatalogo	64
Método index	64
Método insert	64
Método load	65
Método update	65
Método view	66
Método array unique recursive	66
Clase Controlador	67
Constructor construct	67
Método accion	67
Método delete	67
Método getGroupPermissions	68
Método help	68
Método index	68
Método insert	69
Método update	69
Método view	70
Clase Entorno	70
Constructor construct	70
Método delete	70
Método index	71
Método insert	71
Método update	71
Método view	72
Método ExistEntorno	72

Método ExistEntornoUpdate	72
Clase Errorlog	73
Constructor construct	73
Método index	73
Método view	74
Clase Grupo	74
Constructor construct	74
Método delete	75
Método index	75
Método insert	75
Método update	76
Método view	76
Método ifGroupExists	76
Clase Menu	77
Constructor construct	77
Método delete	77
Método index	77
Método insert	78
Método update	78
Método view	79
Clase Permiso	79
Constructor construct	79
Método index	80
Clase Raiz	80
Constructor construct	80
Método createasu	80
Método delete	81
Método getChildrenFromLevel	81
Método getDataKeyValue	82
Método getDataTreeFromId	82
Método getTreeBlock	82
Método index	83
Método iniciarasu	83
Método insert	83
Método update	84
Método updateasu	84
Método view	84
Clase ReglaVacuna	85
Constructor construct	85
Método delete	85
Método index	86
Método insert	86
Método update	86
Método view	87
Clase Usuario	87
Var \$mas	87
Constructor construct	88
Método automatic access	88
Método cerrar etab	88

Método delete	88
Método form_init	88
Método getActivesByGroup	88
Método get_galleta	89
Método get_token	89
Método index	89
Método insert	89
Método load_update	90
Método login	90
Método logout	90
Método remember	90
Método reset	90
Método send_mail	91
Método token	91
Método update	91
Método update_info	92
Método view	92
Método ifUserExists	92
Clase Accion_model	93
Constructor_construct	93
Método delete	93
Método getAll	93
Método getByld	94
Método getDescripcion	94
Método getId	94
Método getMetodo	94
Método getMsgError	94
Método getNombre	95
Método getNumRows	95
Método insert	95
Método setDescripcion	95
Método setId	96
Método setMetodo	96
Método setNombre	96
Método setOffset	97
Método setRows	97
Método update	97
Clase Ageb_model	97
Constructor_construct	98
Método getMsgError	98
Método process	98
Método searchageb	98
Método searchUM	99
Clase ArbolSegmentacion_model	99
Constructor_construct	100
Método convertType	100
Método getByld	100
Método getChildrenFromId	100
Método getChildrenFromLevel	101

Método getCluesFromId	101
Método getDataKeyValue	102
Método getDescripcionById	102
Método getListChildrenLevel	103
Método getMsgError	103
Método getTree	103
Método getTreeBlock	104
Método getTreeBlockData	104
Método getUMParentsById	105
Método addSelectedItems	105
Método addSelectedItems	105
Clase Bitacora model	106
Constructor construct	106
Método addFilter	106
Método delete	107
Método deleteByFilter	107
Método getAll	107
Método getById	108
Método getFecha_hora	108
Método getId	108
Método getId_controlador_accion	108
Método getId_usuario	108
Método getMsgError	109
Método getNumRows	109
Método getParametros	109
Método insert	109
Método resetFilter	110
Método setFecha_hora	110
Método setId	110
Método setId_accion	111
Método setId_controlador	111
Método setId_controlador_accion	111
Método setId_usuario	111
Método setParametros	112
Método update	112
Clase CatalogoCsv model	112
Constructor construct	113
Método activaEnCatalogo	113
Método checkPk	113
Método getAll	114
Método getAllData	114
Método getByName	114
Método getCampos	115
Método getId	115
Método getLLave	115
Método getMsgError	115
Método getNombre	116
Método getNumRows	116
Método setCampos	116

Método setId	116
Método setLlave	117
Método setNombre	117
Método setOffset	117
Método setRows	117
Clase Catalogo_model	118
Constructor construct	118
Método checkPk	118
Método checkTypeData	119
Método delete	119
Método getAll	119
Método getAllData	119
Método getByName	120
Método getCampos	120
Método getId	120
Método getLLave	121
Método getMsgError	121
Método getNombre	121
Método getNumRows	121
Método insert	122
Método setCampos	122
Método setId	122
Método setLlave	123
Método setNombre	123
Método setOffset	123
Método setRows	123
Método updateComentario	124
Clase Catalogo_x_raiz_model	124
Constructor construct	124
Método check	125
Método delete	125
Método getByArbol	125
Método getById	126
Método getByNivel	126
Método getColumnaDescripcion	126
Método getColumnaLLave	126
Método getGrado	127
Método getId	127
Método getIdRaiz	127
Método getMsgError	127
Método getNivel	128
Método getRelacionHijo	128
Método getRelacionPadre	128
Método getRelations	128
Método getTablaCatalogo	129
Método insert	129
Método setColumnaDescripcion	129
Método setColumnaLlave	129
Método setGrado	130

Método setId	130
Método setIdRaiz	130
Método setRelacionHijo	130
Método setRelacionPadre	131
Método setTablaCatalogo	131
Clase Cie10_model	131
Constructor construct	132
Método activaEnCatalogo	132
Método agregaEnCatalogo	132
Método checkPk	133
Método getAll	133
Método getAllData	133
Método getById	134
Método getCatalogoByName	134
Método getData	134
Método getDescripcion	135
Método getId	135
Método getMsgError	135
Método getNumRows	135
Método setDescription	136
Método setId	136
Método setOffset	136
Método setRows	136
Método update	137
Clase ControladorAccion_model	137
Constructor construct	137
Método getById	137
Método getId	138
Método getIdByPath	138
Método getMsgError	139
Método setHelp	139
Clase Controlador_model	139
Constructor construct	140
Método accionesUpdate	140
Método delete	140
Método getAccion	140
Método getAcciones	140
Método getAll	141
Método getByEntorno	141
Método getById	141
Método getClase	142
Método getDescripcion	142
Método getId	142
Método getIdEntorno	142
Método getMsgError	142
Método getNombre	143
Método getNumRows	143
Método getPermisos	143
Método insert	144

Método setAccion	144
Método setClase	144
Método setDescription	145
Método setId	145
Método setIdEntorno	145
Método setName	145
Método setOffset	146
Método setRows	146
Método update	146
Clase Entorno_model	147
Constructor construct	147
Método delete	147
Método getAll	147
Método getByld	147
Método getName	148
Método getDescription	148
Método getDirectorio	148
Método getHostname	148
Método getId	149
Método getInfo	149
Método getIp	149
Método getMsgError	149
Método getName	150
Método getPermissionsByGroup	150
Método insert	150
Método setDescription	150
Método setDirectorio	151
Método setHostname	151
Método setId	151
Método setIp	151
Método setName	152
Método update	152
Clase Errorlog_model	152
Constructor construct	153
Método addFilter	153
Método getAll	153
Método getByld	154
Método getDescription	154
Método getFecha_hora	154
Método getId	154
Método getId_controlador_accion	154
Método getId_usuario	155
Método getNumRows	155
Método insert	155
Método resetFilter	155
Método save	156
Método setDescription	156
Método setId	156
Método setId_accion	157

Método setId controlador	157
Método setId controlador accion	157
Método setId usuario	157
Clase Georeferencia model	158
Constructor construct	158
Método getMsgError	158
Método process	159
Clase Grupo model	159
Constructor construct	159
Método delete	159
Método getAll	160
Método getById	160
Método getByName	160
Método getDescripcion	161
Método getEntornosById	161
Método getId	161
Método getMsgError	161
Método getNombre	162
Método getNumRows	162
Método insert	162
Método setDescripcion	162
Método setId	163
Método setNombre	163
Método update	163
Clase Hemoglobina model	164
Constructor construct	164
Método getMsgError	164
Método process	164
Clase Menu model	165
Constructor construct	165
Método addFilter	165
Método delete	166
Método deleteByFilter	166
Método getAll	166
Método getAtributo	167
Método getById	167
Método getByPadre	167
Método getId	167
Método getId controlador	168
Método getId padre	168
Método getId raiz	168
Método getMsgError	168
Método getNombre	168
Método getNumRows	169
Método getRuta	169
Método hasChild	169
Método insert	169
Método resetFilter	170
Método setAtributo	170

Método setId	170
Método setId controlador	170
Método setId padre	171
Método setId raiz	171
Método setName	171
Método setRuta	171
Método update	172
Clase Permiso_model	172
Constructor construct	172
Método deletePermissions	173
Método getFecha	173
Método getId	173
Método getIdControladorAccion	173
Método getIdGrupo	173
Método getMsgError	174
Método getPermission	174
Método insertBatch	174
Método setFecha	175
Método setId	175
Método setIdControladorAccion	175
Método setIdGrupo	175
Clase Poblacion_model	176
Constructor construct	176
Método getMsgError	176
Método process	177
Clase Raiz_model	177
Constructor construct	177
Método delete	177
Método ExistInArbol	178
Método getAll	178
Método getById	178
Método getDescripcion	179
Método getId	179
Método getMsgError	179
Método insert	179
Método setDescripcion	180
Método setId	180
Método update	180
Clase ReglaVacuna_model	181
Constructor construct	181
Método delete	181
Método getAlergias	181
Método getAll	181
Método getById	182
Método getDiaFinNacido	182
Método getDiaFinPrevio	182
Método getDiaInicioNacido	182
Método getDiaInicioPrevio	182
Método getDosis	183

Método getEsqComp	183
Método getForzarAplicacion	183
Método getId	183
Método getIdVacuna	183
Método getIdVacunaPrevia	183
Método getIdViaVacuna	183
Método getMsgError	184
Método getObservacionRegion	184
Método getOrdenEsqComp	184
Método getRegion	184
Método insert	185
Método setAlergias	185
Método setDiaFinNacido	185
Método setDiaFinPrevia	185
Método setDiaInicioNacido	186
Método setDiaInicioPrevia	186
Método setDosis	186
Método setEsqComp	186
Método setForzarAplicacion	187
Método setId	187
Método setIdVacuna	187
Método setIdVacunaPrevia	188
Método setIdViaVacuna	188
Método setObservacionRegion	188
Método setOrdenEsqComp	188
Método setRegion	189
Método update	189
Clase Usuario_model	189
Constructor construct	190
Método authenticate	190
Método checkCredentials	190
Método check_data	190
Método check_token	191
Método delete	191
Método getActivesByGroup	191
Método getActivo	192
Método getApellidoMaterno	192
Método getApellidoPaterno	192
Método getById	192
Método getByUsername	193
Método getClave	193
Método getCorreo	193
Método getgrupo	193
Método getId	193
Método getIdGrupo	194
Método getMsgError	194
Método getNombre	194
Método getNombreUsuario	194
Método getNumRows	194

Método getOnlyActives	195
Método getuser	195
Método get_grupo_ entorno	196
Método get_permiso_ entorno	196
Método get_usuario_ entorno	196
Método insert	196
Método setActivo	197
Método setApellidoMaterno	197
Método setApellidoPaterno	197
Método setClave	197
Método setCorreo	198
Método setId	198
Método setIdGrupo	198
Método setName	199
Método setNameUsuario	199
Método update	199
Método update_pass	199
Método update_user	200
Paquete TES Clases	201
Clase Enrolamiento	201
Constructor construct	201
Método addForm	201
Método autocomplete	201
Método brothers_search	202
Método brother_found	202
Método catalog_check	202
Método catalog_select	203
Método categoriacie10_select	203
Método checar_session	203
Método cie10_select	204
Método comparar_view	204
Método data_tutor	204
Método file_to_card	205
Método ifCarpExists	205
Método ifCarpTExists	206
Método index	206
Método insert	206
Método paciente_similar	207
Método print_card	207
Método searchgeb	207
Método searchum	208
Método tratamiento_select	208
Método update	209
Método update_card	209
Método vacunacion	209
Método validarForm	210
Método validarisum	210
Método validate_card	211
Método view	211

<u>Clase Notificacion</u>	211
<u>Constructor</u> <u>construct</u>	212
<u>Método delete</u>	212
<u>Método index</u>	212
<u>Método insert</u>	212
<u>Método update</u>	213
<u>Método view</u>	213
<u>Clase Reporteador</u>	214
<u>Constructor</u> <u>construct</u>	214
<u>Método index</u>	214
<u>Método view</u>	214
<u>Clase Reporte sincronizacion</u>	215
<u>Constructor</u> <u>construct</u>	215
<u>Método index</u>	215
<u>Método lote</u>	215
<u>Método lote</u> <u>view</u>	216
<u>Método view</u>	216
<u>Clase Semana nacional</u>	216
<u>Constructor</u> <u>construct</u>	217
<u>Método delete</u>	217
<u>Método getAll</u>	217
<u>Método index</u>	217
<u>Método insert</u>	218
<u>Método update</u>	218
<u>Método view</u>	218
<u>Clase Servicios</u>	219
<u>Constructor</u> <u>construct</u>	219
<u>Método actualiza estado tableta</u>	219
<u>Método catalogos relevantes</u>	220
<u>Método esquema incompleto</u>	220
<u>Método is_step 0</u>	220
<u>Método is_step 1</u>	221
<u>Método is_step 2</u>	221
<u>Método is_step 3</u>	222
<u>Método is_step 4</u>	222
<u>Método prueba2</u>	223
<u>Método ss_step 5</u>	223
<u>Método ss_step 6</u>	223
<u>Método Synchronization</u>	224
<u>Clase Tableta</u>	224
<u>Constructor</u> <u>construct</u>	225
<u>Método delete</u>	225
<u>Método index</u>	225
<u>Método insert</u>	226
<u>Método setUM</u>	226
<u>Método update</u>	226
<u>Método uploadFile</u>	227
<u>Método view</u>	227
<u>Método validateMac</u>	227

<u>Clase Usuario</u>	228
<u>Constructor</u>	228
<u>Método delete</u>	228
<u>Método index</u>	228
<u>Método insert</u>	229
<u>Clase Enrolamiento</u>	229
<u>Constructor</u>	229
<u>Método autocomplete</u>	230
<u>Método cns</u>	230
<u>Método cns</u>	230
<u>Método cns</u>	231
<u>Método data</u>	231
<u>Método entorno</u>	231
<u>Método getaccion</u>	232
<u>Método getafiliacion</u>	232
<u>Método getAfiliaciones</u>	232
<u>Método getageb</u>	233
<u>Método getAlergia</u>	233
<u>Método getalergias</u>	233
<u>Método getaltura</u>	233
<u>Método getByCurp</u>	233
<u>Método getById</u>	234
<u>Método getcalle</u>	234
<u>Método getCategoryCIE10</u>	234
<u>Método getcelular</u>	234
<u>Método getcelularT</u>	235
<u>Método getCIE10</u>	235
<u>Método getcodigo</u>	235
<u>Método getcolonia</u>	235
<u>Método getcompania</u>	235
<u>Método getcompaniaT</u>	236
<u>Método getconsulta</u>	236
<u>Método getControlConsultas</u>	236
<u>Método getcp</u>	236
<u>Método getcurp</u>	236
<u>Método getcurpT</u>	236
<u>Método getestimulacion</u>	237
<u>Método getestimulacion</u>	237
<u>Método getfaccion</u>	237
<u>Método getfconsulta</u>	237
<u>Método getfechacivil</u>	237
<u>Método getfecha</u>	237
<u>Método getfnacimiento</u>	238
<u>Método getfnutricion</u>	238
<u>Método getfolio</u>	238
<u>Método getfvacuna</u>	238
<u>Método gethemoglobina</u>	238
<u>Método getId</u>	239
<u>Método getIdtutor</u>	239

<u>Método getListEnrolamiento</u>	239
<u>Método getlnacimiento</u>	239
<u>Método getlocalidad</u>	239
<u>Método getlugarcivil</u>	240
<u>Método getmanzana</u>	240
<u>Método getmaterno</u>	240
<u>Método getmaternoT</u>	240
<u>Método getMsgError</u>	240
<u>Método getnacionalidad</u>	240
<u>Método getnombre</u>	241
<u>Método getnombreT</u>	241
<u>Método getnumero</u>	241
<u>Método getNumRows</u>	241
<u>Método getparto</u>	241
<u>Método getpaterno</u>	242
<u>Método getpaternoT</u>	242
<u>Método getperi_cefa</u>	242
<u>Método getpeso</u>	242
<u>Método getprecurp</u>	242
<u>Método getreferencia</u>	242
<u>Método getRegistro_civil</u>	242
<u>Método getsales_cantidad</u>	243
<u>Método getsales_fecha</u>	243
<u>Método getsangre</u>	243
<u>Método getsector</u>	243
<u>Método getsexo</u>	243
<u>Método getsexoT</u>	244
<u>Método gettalla</u>	244
<u>Método gettamiz</u>	244
<u>Método gettbbeneficiario</u>	244
<u>Método gettconsulta</u>	244
<u>Método gettelefono</u>	244
<u>Método gettelefonoT</u>	244
<u>Método getumt</u>	245
<u>Método getvacuna</u>	245
<u>Método get_catalog</u>	245
<u>Método get_catalog2</u>	245
<u>Método get_catalog_count</u>	246
<u>Método get_catalog_relevante</u>	246
<u>Método get_catalog_tratamiento</u>	247
<u>Método get_catalog_view</u>	247
<u>Método get_cns_cat_persona</u>	247
<u>Método get_cns_cat_persona_count</u>	248
<u>Método get_cns_persona</u>	248
<u>Método get_control_nutricional</u>	249
<u>Método get_datos_grafica</u>	249
<u>Método get_estimulacion</u>	249
<u>Método get_notificacion</u>	250
<u>Método get_pacientes</u>	250

Método get_peri_cefa	250
Método get_persona_x_tutor	251
Método get_sales	251
Método get_transaction_relevante	251
Método get_version	252
Método insert	252
Método setaccion_nutricional	252
Método setafiliacion	252
Método setageb	253
Método setalergias	253
Método setaltura	253
Método setcalle	253
Método setcelular	254
Método setcelularT	254
Método setcodigo_barras	254
Método setcolonia	255
Método setcompania	255
Método setcompaniaT	255
Método setconsulta	255
Método setcp	256
Método setcurp	256
Método setcurpT	256
Método setestimulacion_capacitado	257
Método setestimulacion_fecha	257
Método setfaccion_nutricional	257
Método setfconsulta	257
Método setfechacivil	258
Método setfecha_peri_cefa	258
Método setfnacimiento	258
Método setfnutricion	258
Método setfvacuna	259
Método sethemoglobina	259
Método setld	259
Método setidtutor	260
Método setlnacimiento	260
Método setlocalidad	260
Método setlugarcivil	260
Método setmanzana	261
Método setmaterno	261
Método setmaternoT	261
Método setnacionalidad	262
Método setnombre	262
Método setnombreT	262
Método setnumero	262
Método setparto	263
Método setpaterno	263
Método setpaternoT	263
Método setperi_cefa	264
Método setpeso	264

Método setprecurp	264
Método setreferencia	264
Método setsales cantidad	265
Método setsales fecha	265
Método setsangre	265
Método setsector	266
Método setsexo	266
Método setsexoT	266
Método settalla	266
Método settamiz	267
Método settbeneficiario	267
Método settconsulta	267
Método settelefono	267
Método settelefonoT	268
Método setumt	268
Método setvacuna	268
Método tes pendientes tarjeta delete	269
Método update accion	269
Método update alergia	269
Método update basico	269
Método update beneficiario	269
Método update consulta	270
Método update direccion	270
Método update estimulacion	270
Método update nutricion	270
Método update peri cefa	271
Método update regcivil	271
Método update sales	271
Método update status tableta	271
Método update tutor	272
Método update umt	272
Método update vacuna	272
Método valid card	272
Clase Estado tableta model	273
Constructor construct	273
Método getAll	273
Método getByld	273
Método getDescripcion	274
Método getId	274
Método getMsgError	274
Método setDescripcion	274
Método setId	275
Clase Notificacion model	275
Constructor construct	275
Método addFilter	275
Método delete	276
Método getAll	276
Método getByld	277
Método getContenido	277

Método <u>getFechaFin</u>	277
Método <u>getFechaInicio</u>	277
Método <u>getId</u>	277
Método <u>getIdsTabletas</u>	277
Método <u>getMsgError</u>	278
Método <u>getNumRows</u>	278
Método <u>getTitulo</u>	278
Método <u>insert</u>	278
Método <u>setContenido</u>	279
Método <u>setFechaFin</u>	279
Método <u>setFechaInicio</u>	279
Método <u>setId</u>	280
Método <u>setIdsTabletas</u>	280
Método <u>setTitulo</u>	280
Método <u>update</u>	280
Clase <u>Reporteador_model</u>	281
Constructor <u>construct</u>	281
Método <u>getCensoNominal</u>	281
Método <u>getCoberturaBiologicoListado</u>	281
Método <u>getConcentradoActividades</u>	282
Método <u>getEsquemasIncompletos</u>	282
Método <u>getGrupoVacunas</u>	282
Método <u>getMsgError</u>	282
Método <u>getSeguimientoRV1RV5</u>	283
Método <u>getVacunas</u>	283
Método <u>getVacunasByGrupo</u>	283
Método <u>object to array</u>	283
Clase <u>Reporte_censo_nominal</u>	284
Var <u>\$apellido_materno</u>	284
Var <u>\$apellido_paterno</u>	284
Var <u>\$curp</u>	284
Var <u>\$domicilio</u>	285
Var <u>\$fecha_nacimiento</u>	285
Var <u>\$nombre</u>	285
Var <u>\$parto_multiple</u>	285
Var <u>\$sexo</u>	285
Var <u>\$vacunas</u>	286
Constructor <u>construct</u>	286
Clase <u>Reporte_sincronizacion_model</u>	286
Método <u>getCount</u>	286
Método <u>getListado</u>	287
Método <u>getMsgError</u>	287
Método <u>get_version</u>	287
Clase <u>Semana_nacional_model</u>	288
Constructor <u>construct</u>	288
Método <u>delete</u>	288
Método <u>getAll</u>	288
Método <u>getById</u>	289
Método <u>getDescripcion</u>	289

Método getFecha_fin	289
Método getFecha_inicio	289
Método getId	290
Método getMsgError	290
Método getNumRows	290
Método insert	290
Método setDescription	291
Método setFecha_fin	291
Método setFecha_inicio	291
Método update	291
Clase Tableta_model	292
Constructor construct	292
Método delete	292
Método getAll	293
Método getById	293
Método getByMac	293
Método getId	294
Método getIdVersion	294
Método getId_asu_um	294
Método getId_tes_estado_tableta	294
Método getId_tipo_censo	294
Método getMac	294
Método getMsgError	295
Método getNumRows	295
Método getPeriodo_esq_inc	295
Método getUltima_actualizacion	295
Método getUsers_asignados	296
Método insert	296
Método setId	296
Método setIdVersion	296
Método setId_asu_um	296
Método setId_tes_estado_tableta	297
Método setId_tipo_censo	297
Método setMac	297
Método setPeriodo_esq_inc	298
Método setUltima_actualizacion	298
Método setUsers_asignados	298
Método update	298
Clase Tipo_censo_model	299
Constructor construct	299
Método getAll	299
Método getById	300
Método getDescription	300
Método getId	300
Método getMsgError	300
Método setDescription	301
Método setId	301
Clase Usuario_tableta_model	301
Constructor construct	302

Método delete	302
Método getMsgError	302
Método getTableta	302
Método getTabletasByUsuario	303
Método getUsuario	303
Método getUsuariosByTableta	303
Método insert	303
Método setTableta	304
Método setUsuario	304
Appendices	305
Appendix A - Class Trees	306
SIIGS	306
default	314
TES	314
Libreria	318
CodeIgniter	319
Session	319
Appendix C - Source Code	321
Package TES	322
source code: Form_validation.php	323
source code: Pagination.php	341
source code: template.php	346

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 CodeIgniter Elementos procedurales

Form_validation.php

CodeIgniter

An open source application development framework for PHP 5.1.6 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2008 - 2011, EllisLab, Inc.
- **Link** <http://codeigniter.com>
- **Since** Version 1.0
- **Filesource** [Source Code for this file](#)
- **License** http://codeigniter.com/user_guide/license.html

Pagination.php

CodeIgniter

An open source application development framework for PHP 5.1.6 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2008 - 2011, EllisLab, Inc.
- **Link** <http://codeigniter.com>
- **Since** Version 1.0
- **Filesource** [Source Code for this file](#)
- **License** http://codeigniter.com/user_guide/license.html

template.php

CodeIgniter

An open source application development framework for PHP 4.3.2 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2006, EllisLab, Inc.
- **Link** <http://codeigniter.com>
- **Since** Version 1.0
- **Filesource** [Source Code for this file](#)
- **License** http://codeigniter.com/user_guide/license.html

Paquete CodeIgniter Clases

Clase CI_Form_validation *[line 27]*

Form Validation Class

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** ExpressionEngine Dev Team
- **Link** http://codeigniter.com/user_guide/libraries/form_validation.html

CI_Form_validation::\$CI

mixed = [line 29]

- **Access** protected

CI_Form_validation::\$error_string

mixed = " [line 36]

- **Access** protected

CI_Form_validation::\$_config_rules

mixed = array() [line 31]

- **Access** protected

CI_Form_validation::\$_error_array

mixed = array() [line 32]

- **Access** protected

CI_Form_validation::\$_error_messages

mixed = array() [line 33]

- **Access** protected

CI_Form_validation::\$_error_prefix

mixed = '<div class="error">' [line 34]

- **Access** protected

CI_Form_validation::\$_error_suffix

mixed = '</div>' [line 35]

- **Access** protected

CI_Form_validation::\$_field_data

mixed = array() [line 30]

- **Access** protected

CI_Form_validation::\$_safe_form_data

mixed = FALSE [line 37]

- **Access** protected

Constructor *void* función CI_Form_validation::__construct([\$rules = array()]) [line 42]

Parámetros de la función:

- **\$rules**

Constructor

- **Access** public

bool función CI_Form_validation::alpha(\$str) [line 1099]

Parámetros de la función:

- *string* **\$str**

Alpha

- **Access** public

bool función CI_Form_validation::alpha_dash(\$str) [line 1127]

Parámetros de la función:

- *string* \$str

Alpha-numeric with underscores and dashes

- **Access** public

bool función CI_Form_validation::alpha_numeric(\$str) [line 1113]

Parámetros de la función:

- *string* \$str

Alpha-numeric

- **Access** public

bool función CI_Form_validation::decimal(\$str) [line 1184]

Parámetros de la función:

- *string* \$str

Decimal number

- **Access** public

string función CI_Form_validation::encode_php_tags(\$str) [line 1373]

Parámetros de la función:

- *string* **\$str**

Convert PHP tags to entities

- **Access** public

void función CI_Form_validation::error([\$field = "], [\$prefix = "], [\$suffix = "]) [line 208]

Parámetros de la función:

- *string* **\$field** the field name
- **\$prefix**
- **\$suffix**

Get Error Message

Gets the error message associated with a particular field

- **Access** public

str función CI_Form_validation::error_string([\$prefix = "], [\$suffix = "]) [line 240]

Parámetros de la función:

- *string* **\$prefix**

- *string* **\$suffix**

Error String

Returns the error messages as a string, wrapped in the error delimiters

- **Access** public

bool función CI_Form_validation::exact_length(\$str, \$val) [*line 1019*]

Parámetros de la función:

- *string* **\$str**
- *value* **\$val**

Exact Length

- **Access** public

bool función CI_Form_validation::greater_than(\$str, \$min) [*line 1198*]

Parámetros de la función:

- *string* **\$str**
- **\$min**

Greather than

- **Access** public

bool función CI_Form_validation::integer(\$str) [*line 1170*]

Parámetros de la función:

- *string* \$str

Integer

- **Access** public

bool función CI_Form_validation::is_natural(\$str) [*line 1234*]

Parámetros de la función:

- *string* \$str

Is a Natural number (0,1,2,3, etc.)

- **Access** public

bool función CI_Form_validation::is_natural_no_zero(\$str) [*line 1248*]

Parámetros de la función:

- *string* \$str

Is a Natural number, but not a zero (1,2,3, etc.)

- **Access** public

bool función CI_Form_validation::is_numeric(\$str) [line 1156]

Parámetros de la función:

- *string* \$str

Is Numeric

- **Access** public

bool función CI_Form_validation::is_unique(\$str, \$field) [line 951]

Parámetros de la función:

- *string* \$str
- *field* \$field

Match one field to another

- **Access** public

bool función CI_Form_validation::less_than(\$str, \$max) [line 1216]

Parámetros de la función:

- *string* \$str
- \$max

Less than

- **Access** public

bool función CI_Form_validation::matches(\$str, \$field) [line 929]

Parámetros de la función:

- *string* **\$str**
- *field* **\$field**

Match one field to another

- **Access** public

bool función CI_Form_validation::max_length(\$str, \$val) [line 994]

Parámetros de la función:

- *string* **\$str**
- *value* **\$val**

Max Length

- **Access** public

bool función CI_Form_validation::min_length(\$str, \$val) [line 969]

Parámetros de la función:

- *string* **\$str**
- *value* **\$val**

Minimum Length

- **Access** public

bool función CI_Form_validation::numeric(\$str) [*line 1141*]

Parámetros de la función:

- *string* \$str

Numeric

- **Access** public

string función CI_Form_validation::prep_for_form([\$data = ""]) [*line 1292*]

Parámetros de la función:

- *string* \$data

Prep data for form

This function allows HTML to be safely shown in a form. Special characters are converted.

- **Access** public

string función CI_Form_validation::prep_url([\$str = ""]) [*line 1321*]

Parámetros de la función:

- *string* **\$str**

Prep URL

- **Access** public

bool función CI_Form_validation::regex_match(\$str, \$regex) [line 909]

Parámetros de la función:

- *string* **\$str**
- *regex* **\$regex**

Performs a Regular Expression match test.

- **Access** public

bool función CI_Form_validation::required(\$str) [line 887]

Parámetros de la función:

- *string* **\$str**

Required

- **Access** public

bool función CI_Form_validation::run([\$group = "]) [*line 281*]

Parámetros de la función:

- **\$group**

Run the Validator

This function does all the work.

- **Access public**

string función CI_Form_validation::set_checkbox([\$field = "], [\$value = "], [\$default = FALSE]) [*line 847*]

Parámetros de la función:

- *string* **\$field**
- *string* **\$value**
- **\$default**

Set Checkbox

Enables checkboxes to be set to the value the user selected in the event of an error

- **Access public**

void función CI_Form_validation::set_error_delimiters([\$prefix = '<p>'], [\$suffix = '</p>']) [*line 189*]

Parámetros de la función:

- *string* **\$prefix**
- *string* **\$suffix**

Set The Error Delimiter

Permits a prefix/suffix to be added to each error message

- **Access** public

string función CI_Form_validation::set_message(\$lang, [\$val = "]) [*line 165*]

Parámetros de la función:

- *string* **\$lang**
- *string* **\$val**

Set Error Message

Lets users set their own error messages on the fly. Note: The key name has to match the function name that it corresponds to.

- **Access** public

string función CI_Form_validation::set_radio([\$field = "], [\$value = "], [\$default = FALSE]) [*line 803*]

Parámetros de la función:

- *string* **\$field**
- *string* **\$value**
- **\$default**

Set Radio

Enables radio buttons to be set to the value the user selected in the event of an error

- **Access** public

void función CI_Form_validation::set_rules(\$field, [\$label = "], [\$rules = "]) [*line 74*]

Parámetros de la función:

- *mixed* **\$field**
- *string* **\$label**
- **\$rules**

Set Rules

This function takes an array of field names and validation rules as input, validates the info, and stores it

- **Access** public

string función CI_Form_validation::set_select([\$field = "], [\$value = "], [\$default = FALSE]) [*line 759*]

Parámetros de la función:

- *string* **\$field**
- *string* **\$value**
- **\$default**

Set Select

Enables pull-down lists to be set to the value the user selected in the event of an error

- **Access** public

void función CI_Form_validation::set_value([\$field = "], [\$default = "]) [*line 729*]

Parámetros de la función:

- *string* **\$field** the field name

- *string* **\$default**

Get the value from a form

Permits you to repopulate a form field with the value it was submitted with, or, if that value doesn't exist, with the default

- **Access** public

string función CI_Form_validation::strip_image_tags(\$str) [line 1345]

Parámetros de la función:

- *string* **\$str**

Strip Image Tags

- **Access** public

bool función CI_Form_validation::valid_base64(\$str) [line 1275]

Parámetros de la función:

- *string* **\$str**

Valid Base64

Tests a string for characters outside of the Base64 alphabet as defined by RFC 2045 <http://www.faqs.org/rfcs/rfc2045>

- **Access public**

bool función CI_Form_validation::valid_email(\$str) [line 1043]

Parámetros de la función:

- *string* **\$str**

Valid Email

- **Access public**

bool función CI_Form_validation::valid_emails(\$str) [line 1057]

Parámetros de la función:

- *string* **\$str**

Valid Emails

- **Access public**

string función CI_Form_validation::valid_ip(\$ip, [\$which = ""]) [line 1085]

Parámetros de la función:

- *string* **\$ip**
- *string* **\$which** "ipv4" or "ipv6" to validate a specific ip format

Validate IP Address

- **Access public**

string función CI_Form_validation::xss_clean(\$str) [line 1359]

Parámetros de la función:

- *string* **\$str**

XSS Clean

- **Access public**

mixed función CI_Form_validation::_execute(\$row, \$rules, [\$postdata = NULL], [\$cycles = 0]) [line 470]

Parámetros de la función:

- *array* **\$row**
- *array* **\$rules**
- *mixed* **\$postdata**
- *integer* **\$cycles**

Executes the Validation routines

- **Access protected**

mixed función CI_Form_validation::_reduce_array(\$array, \$keys, [\$i = 0]) [line 376]

Parámetros de la función:

- *array* **\$array**
- *array* **\$keys**
- *integer* **\$i**

Traverse a multidimensional \$_POST array index until the data is found

- **Access** protected

null función CI_Form_validation::_reset_post_array() [line 408]

Re-populate the _POST array with our finalized and processed data

- **Access** protected

string función CI_Form_validation::_translate_fieldname(\$fieldname) [line 697]

Parámetros de la función:

- *string* **\$fieldname** the field name

Translate a field name

- **Access** protected

Clase CI_Pagination

[line 27]

Pagination Class

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** ExpressionEngine Dev Team
- **Link** http://codeigniter.com/user_guide/libraries/pagination.html

CI_Pagination::\$anchor_class

mixed = " [line 61]

CI_Pagination::\$base_url

mixed = " [line 29]

CI_Pagination::\$cur_page

mixed = 0 [line 36]

CI_Pagination::\$cur_tag_close

mixed = '' [line 51]

CI_Pagination::\$cur_tag_open

- mixed = '*
- *<a>' [line 50]*

CI_Pagination::\$display_pages

mixed = TRUE [line 60]

CI_Pagination::\$first_link

mixed = '‹ First' [line 38]

CI_Pagination::\$first_tag_close

mixed = '
' [line 46]

CI_Pagination::\$first_tag_open

- mixed = '*
- *' [line 45]*

CI_Pagination::\$first_url

mixed = " [line 49]

CI_Pagination::\$full_tag_close

mixed = '

' [line 44]

CI_Pagination::\$full_tag_open

mixed = '<ul class="pagination pagination-sm">' [line 43]

CI_Pagination::\$last_link

mixed = 'Last ’' [line 41]

CI_Pagination::\$last_tag_close

*mixed = '
' [line 48]*

CI_Pagination::\$last_tag_open

*mixed = '
• ' [line 47]*

CI_Pagination::\$next_link

mixed = '>' [line 39]

CI_Pagination::\$next_tag_close

*mixed = '
' [line 53]*

CI_Pagination::\$next_tag_open

*mixed = '
• ' [line 52]*

CI_Pagination::\$num_links

mixed = 2 [line 35]

CI_Pagination::\$num_tag_close

*mixed = '
' [line 57]*

CI_Pagination::\$num_tag_open

*mixed = '
• ' [line 56]*

CI_Pagination::\$page_query_string

mixed = FALSE [line 58]

CI_Pagination::\$per_page

mixed = 10 [line 34]

CI_Pagination::\$prefix

mixed = '' [line 30]

CI_Pagination::\$prev_link

mixed = '<' [line 40]

CI_Pagination::\$prev_tag_close

*mixed = '
' [line 55]*

CI_Pagination::\$prev_tag_open

*mixed = '
• ' [line 54]*

CI_Pagination::\$query_string_segment

mixed = 'per_page' [line 59]

CI_Pagination::\$suffix

mixed = " [line 31]

CI_Pagination::\$total_rows

mixed = 0 [line 33]

CI_Pagination::\$uri_segment

mixed = 3 [line 42]

CI_Pagination::\$use_page_numbers

mixed = FALSE [line 37]

Constructor *void* función CI_Pagination::__construct([\$params = array()]) [line 69]

Parámetros de la función:

- *array* **\$params** initialization parameters

Constructor

- **Access** public

string función CI_Pagination::create_links() [line 115]

Generate the pagination links

- **Access** public

void función CI_Pagination::initialize([\$params = array()]) *[line 93]*

Parámetros de la función:

- *array* **\$params** initialization parameters

Initialize Preferences

- **Access** public

Clase CI_Template

[line 33]

CodeIgniter Template Class

This class is an interface to CI's View class. It aims to improve the interaction between controllers and views. Follow @link for more info

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** Colin Williams
- **Version** 1.4.1
- **Copyright** Copyright (c) 2008, Colin Williams.
- **Link** <http://www.williamsconcepts.com/ci/libraries/template/index.html>

CI_Template::\$CI

mixed = [line 35]

CI_Template::\$config

mixed = [line 36]

CI_Template::\$css

mixed = array() [line 45]

CI_Template::\$js

mixed = array() [line 44]

CI_Template::\$master

mixed = [line 38]

CI_Template::\$output

mixed = [line 43]

CI_Template::\$parser

mixed = 'parser' [line 46]

CI_Template::\$parser_method

mixed = 'parse' [line 47]

CI_Template::\$parse_template

mixed = FALSE [line 48]

CI_Template::\$regions

*mixed = array(
 '_scripts' => array(), '_styles' => array(),) [line 39]*

CI_Template::\$template

mixed = [line 37]

Constructor *void* función CI_Template::CI_Template() [line 59]

Constructor

Loads template configuration, template regions, and validates existence of default template

- **Access** public

TRUE función CI_Template::add_css(\$style, [\$type = 'link'], [\$media = FALSE]) [line 492]

Parámetros de la función:

- *string* **\$style** CSS file to link, import or embed
- *string* **\$type** 'link', 'import' or 'embed'
- *string* **\$media** media attribute to use with 'link' type only, FALSE for none

Dynamically include CSS in the template

NOTE: This function does NOT check for existence of .css file

- **Access** public

TRUE función CI_Template::add_js(\$script, [\$type = 'import'], [\$defer = FALSE]) [line 433]

Parámetros de la función:

- *string* **\$script** script to import or embed
- *string* **\$type** 'import' to load external file or 'embed' to add as-is
- *boolean* **\$defer** TRUE to use 'defer' attribute, FALSE to exclude it

Dynamically include javascript in the template

NOTE: This function does NOT check for existence of .js file

- **Access** public

void función CI_Template::add_region(\$name, [\$props = array()]) [line 233]

Parámetros de la función:

- *string* **\$name** Name to identify the region
- *array* **\$props** Optional array with region defaults

Dynamically add region to the currently set template

- **Access** public

void función CI_Template::add_template(\$group, \$template, [\$activate = FALSE]) [line 129]

Parámetros de la función:

- *string* **\$group** array key to access template settings
- *array* **\$template** properly formed
- **\$activate**

Dynamically add a template and optionally switch to it

- **Access** public

void función CI_Template::empty_region(\$name) [line 260]

Parámetros de la función:

- *string* **\$name** Name to identify the region

Empty a region's content

- **Access** public

void función CI_Template::initialize(\$props) [line 155]

Parámetros de la función:

- **array \$props** configuration array

Initialize class settings using config settings

- **Access** public

void función CI_Template::load([\$region = NULL], [\$buffer = FALSE]) [line 602]

Parámetros de la función:

- **\$region**
- **\$buffer**

Load the master template or a single region DEPRECATED!

Use render() to compile and display your template and regions

void función CI_Template::parse_view(\$region, \$view, [\$data = NULL], [\$overwrite = FALSE]) [line 391]

Parámetros de la función:

- **string \$region** region to write to
- **string \$view** view file to parse
- **array \$data** variables to pass into view for parsing
- **boolean \$overwrite** FALSE to append to region, TRUE to overwrite region

Parse content from a View to a region with the Parser Class

- **Access** public

void función CI_Template::render([\$region = NULL], [\$buffer = FALSE], [\$parse = FALSE]) [line 548]

Parámetros de la función:

- *string* **\$region** optionally opt to render a specific region
- *boolean* **\$buffer** FALSE to output the rendered template, TRUE to return as a string. Always TRUE when \$region is supplied
- **\$parse**

Render the master template or a single region

- **Access** public

void función CI_Template::set_master_template(\$filename) [line 106]

Parámetros de la función:

- *string* **\$filename** filename of new master template file

Set master template

- **Access** public

void función CI_Template::set_parser(\$parser, [\$method = NULL]) [line 282]

Parámetros de la función:

- *string* **\$parser** name of parser class to load and use for parsing methods
- **\$method**

Set parser

- **Access** public

void función CI_Template::set_parser_method(\$method) [line 303]

Parámetros de la función:

- *string* **\$method** name of parser class member function to call when parsing

Set parser method

- **Access** public

void función CI_Template::set_regions(\$regions) [line 199]

Parámetros de la función:

- *array* **\$regions** properly formed regions array

Set regions for writing to

- **Access** public

void función CI_Template::set_template(\$group) [line 83]

Parámetros de la función:

- *string* **\$group** array key to access template settings

Use given template settings

- **Access** public

void función CI_Template::write(\$region, \$content, [\$overwrite = FALSE]) *[line 320]*

Parámetros de la función:

- *string* **\$region** region to write to
- *string* **\$content** what to write
- *boolean* **\$overwrite** FALSE to append to region, TRUE to overwrite region

Write contents to a region

- **Access** public

void función CI_Template::write_view(\$region, \$view, [\$data = NULL], [\$overwrite = FALSE]) *[line 352]*

Parámetros de la función:

- *string* **\$region** region to write to
- *string* **\$view** view file to use
- *array* **\$data** variables to pass into view
- *boolean* **\$overwrite** FALSE to append to region, TRUE to overwrite region

Write content from a View to a region. 'Views within views'

- **Access** public

Paquete Libreria Clases

Clase Menubuilder *[line 12]*

Menu Builder

- **Package** Libreria
- **Sub-Package** Clase
- **Author** Pascual

Constructor *void* función Menubuilder::__construct() *[line 23]*

- **Access** public

string función Menubuilder::build([\$todos = false]) *[line 35]*

Parámetros de la función:

- *boolean* **\$todos** Establece si se debe devolver todos los elementos del menu

Construye el menu basandose en los permisos del usuario logeado

- **Static**

- **Access** public

void función Menubuilder::crearMenu(&\$strMenu, [\$id_padre = 'NULL'], [\$todos = false], \$strMenu) [line 68]

Parámetros de la función:

- *string* **\$strMenu** Variable pasada por referencia donde se guardará la cadena de ul y li
- *string* **\$id_padre** ID del padre de los elementos del arbol de menu
- **&\$strMenu**
- **\$todos**

Función recursiva que recorre todo el árbol de elementos del menu

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

void función Menubuilder::isGranted(\$accion, \$strMenu, \$id_padre) [line 123]

Parámetros de la función:

- *string* **\$strMenu** Variable pasada por referencia donde se guardará la cadena de ul y li
- *string* **\$id_padre** ID del padre de los elementos del arbol de menu
- **\$accion**

Función que valida si se visualiza o no la accion especificada (usada en views)

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

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=embebido 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 = ""]) [line 79]

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 Session Clases

Clase Session

[line 12]

CodeIgniter Native Session Library

- **Package** Session
- **Sub-Package** Libraries
- **Author** Bo-Yi Wu (appleboy) < appleboy.tw@gmail.com>
- **Author** Marko Martinovi? < marko@techtalk.info>

Session::\$ci

mixed = [line 16]

- **Access** protected

Session::\$flashdata_key

mixed = 'flash' [line 18]

- **Access** protected

Session::\$sess_expiration

mixed = " [line 15]

- **Access** protected

Session::\$sess_namespace

mixed = " [line 14]

- **Access** protected

Session::\$store

mixed = array() [line 17]

- **Access** protected

Constructor *void* función Session::__construct([\$config = array()]) *[line 28]*

Parámetros de la función:

- *array* **\$config** config preferences

Constructor

- **Access** public

array función Session::all_userdata() *[line 190]*

Fetch all session data

- **Access** public

string función Session::flashdata(\$key) [line 245]

Parámetros de la función:

- *string* **\$key**

Fetch a specific flashdata item from the session array

- **Access** public

void función Session::is_expired() [line 106]

Check if session is expired

- **Access** public

void función Session::keep_flashdata(\$key) [line 225]

Parámetros de la función:

- *string* **\$key**

Keeps existing flashdata available to next request.

- **Access** public

void función Session::sess_create() [line 85]

Create Session

- **Access** public

void función Session::sess_destroy() [line 119]

Destroy session

- **Access** public

void función Session::set_flashdata([\$newdata = array()], [\$newval = ""]) [line 204]

Parámetros de la función:

- *mixed* **\$newdata**
- *string* **\$newval**

Add or change flashdata, only available until the next request

- **Access** public

void función Session::set_userdata([\$data = array()], [\$value = ""]) [line 151]

Parámetros de la función:

- *array* **\$data** list of data to be stored
- *object value* **\$value** to be stored if only one element is passed

Set value for specific user data element

- **Access** public

void función Session::unset_userdata([\$data = array()]) [line 169]

Parámetros de la función:

- *array* **\$data** list of data to be removed

remove array value for specific user data element

- **Access** public

object element función Session::userdata(\$value) [line 131]

Parámetros de la función:

- *string* **\$value** element key

Get specific user data element

- **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
multidimensionales**

Revisa valores duplicados en arreglos

- **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 actualizacion de un catálogo ya existente,

recibe un string para obtener los valores del catalogo y mostrarlos en la vista update
, realiza la validacion 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** Geovanni

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_catálogos 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

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 raiz específica, obtiene el objeto raiz por medio del id proporcionado.

- **Access** public

Clase ReglaVacuna

[line 10]

Controlador ReglaVacuna

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

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::getByld(\$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([\$id = null]) [line 278]

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 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** Geovanni

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(\$idarbol, \$nivel) [line 223]

Parámetros de la función:

- **int \$idarbol**
- **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::getByld(\$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::getByld(\$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::getName(\$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::setDescription(\$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::getByld(\$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::getDiaFinPrevia() [line 160]

- **Access** public

void función ReglaVacuna_model::getDiaInicioNacido() [line 139]

- **Access** public

void función ReglaVacuna_model::getDiaInicioPrevia() [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::setDiaFinPrevio(\$value) [*line 163*]

Parámetros de la función:

- **\$value**

- **Access** public

void función ReglaVacuna_model::setDialInicioNacido(\$value) [line 142]

Parámetros de la función:

- **\$value**
- **Access public**

void función ReglaVacuna_model::setDialInicioPrevia(\$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 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_session**

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 petición
- *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_session**
- **\$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 petición
- *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_session**

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_session, \$datos, \$id_session) [line 466]

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 comunicacion tableta - servidor
- **\$id_session**

Guarda en la base de datos el estado de la sincronizacion

- **Access public**

echo función Servicios::is_step_4(\$id_session, \$id_session) [line 502]

Parámetros de la función:

- *int* **\$id_session** Representa la session activa para la petición
- **\$id_session**

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_session**

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 especifico permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **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
- *string* **\$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
- *string* **\$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 = ""], [\$l1 = ""], [\$l2 = ""]) [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
- *string* **\$I1** nombre del campo para hacer el limit offset
- *string* **\$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* **\$orden** 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
- *strin* **\$array** ids de personas
- *string* **\$I1** ofsset para el limit
- *strin* **\$I2** 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
- *strin* **\$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
- *strin* **\$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
- *string* **\$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 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::setIdsTabletas(\$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](#)

Menubuilder

- [Menubuilder](#)

Obtenercurp

- CI_Controller
 - [Obtenercurp](#)

Tree

- CI_Controller
 - [Tree](#)

Paquete CodeIgniter

CI_Form_validation

- [CI_Form_validation](#)

CI_Pagination

- [CI_Pagination](#)

CI_Template

- [CI_Template](#)

Paquete Session

Session

- [Session](#)

Apéndice C - Código fuente

Paquete TES

Archivo fuente para Form_validation.php

La documentación para este archivo está disponible en [Form_validation.php](#)

```
1  <?php      if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2  /**
3   * CodeIgniter
4   *
5   * An open source application development framework for PHP 5.1.6 or newer
6   *
7   * @package      CodeIgniter
8   * @author       ExpressionEngine Dev Team
9   * @copyright    Copyright (c) 2008 - 2011, EllisLab, Inc.
10  * @license      http://codeigniter.com/user_guide/license.html
11  * @link         http://codeigniter.com
12  * @since        Version 1.0
13  * @filesource
14  */
15
16  // -----
17
18  /**
19   * Form Validation Class
20   *
21   * @package      CodeIgniter
22   * @subpackage   Libraries
23   * @category     Validation
24   * @author       ExpressionEngine Dev Team
25   * @link         http://codeigniter.com/user_guide/libraries/form_validation.html
26   */
27  class CI_Form_validation {
28
29      protected $CI;
30      protected $field_data      = array();
31      protected $config_rules    = array();
32      protected $error_array     = array();
33      protected $error_messages  = array();
34      protected $error_prefix    = '<div class="error">';
35      protected $error_suffix    = '</div>';
36      protected $error_string    = '';
37      protected $safe_form_data  = FALSE;
38
39      /**
40       * Constructor
41       */
42      public function __construct($rules = array())
43      {
44          $this-> CI =& get_instance();
45
46          // Validation rules can be stored in a config file.
47          $this-> config_rules = $rules;
48
49          // Automatically load the form helper
50          $this-> CI-> load-> helper('form');
51
52          // Set the character encoding in MB.
53          if (function_exists('mb_internal_encoding'))
54          {
55              mb_internal_encoding($this-> CI-> config-> item('charset'));
56          }
57
58          log_message('debug', "Form Validation Class Initialized" );
59      }
60
61      // -----
62
63      /**
64       * Set Rules
65       *
66       * This function takes an array of field names and validation
67       * rules as input, validates the info, and stores it
```

```

68 *
69 * @access      public
70 * @param       mixed
71 * @param       string
72 * @return      void
73 */
74 public function set_rules($field, $label = '', $rules = '')
75 {
76     // No reason to set rules if we have no POST data
77     if (count($_POST) == 0)
78     {
79         return $this;
80     }
81
82     // If an array was passed via the first parameter instead of individual string
83     // values we cycle through it and recursively call this function.
84     if (is_array($field))
85     {
86         foreach ($field as $row)
87         {
88             // Houston, we have a problem...
89             if ( ! isset($row['field']) OR ! isset($row['rules']))
90             {
91                 continue;
92             }
93
94             // If the field label wasn't passed we use the field name
95             $label = ( ! isset($row['label'])) ? $row['field'] : $row['label'];
96
97             // Here we go!
98             $this-> set_rules($row['field'], $label, $row['rules']);
99         }
100         return $this;
101     }
102
103     // No fields? Nothing to do...
104     if ( ! is_string($field) OR ! is_string($rules) OR $field == '')
105     {
106         return $this;
107     }
108
109     // If the field label wasn't passed we use the field name
110     $label = ($label == '') ? $field : $label;
111
112     // Is the field name an array? We test for the existence of a bracket "[" in
113     // the field name to determine this. If it is an array, we break it apart
114     // into its components so that we can fetch the corresponding POST data later
115     if (strpos($field, '[') !== FALSE AND preg_match_all('/\[([.*?])\]/', $field, $matches))
116     {
117         // Note: Due to a bug in current() that affects some versions
118         // of PHP we can not pass function call directly into it
119         $x = explode('[', $field);
120         $indexes[] = current($x);
121
122         for ($i = 0; $i < count($matches['0']); $i++)
123         {
124             if ($matches['1'][$i] != '')
125             {
126                 $indexes[] = $matches['1'][$i];
127             }
128         }
129
130         $is_array = TRUE;
131     }
132     else
133     {
134         $indexes = array();
135         $is_array = FALSE;
136     }
137
138     // Build our master array
139     $this-> _field_data[$field] = array(
140         'field'      => $field,
141         'label'      => $label,
142         'rules'      => $rules,
143         'is_array'   => $is_array,
144         'keys'       => $indexes,
145         'postdata'   => NULL,
146         'error'      => ''
147     );

```

```

148
149     return $this;
150 }
151
152 // -----
153
154 /**
155  * Set Error Message
156  *
157  * Lets users set their own error messages on the fly. Note: The key
158  * name has to match the function name that it corresponds to.
159  *
160  * @access public
161  * @param string
162  * @param string
163  * @return string
164  */
165 public function set_message($lang, $val = '')
166 {
167     if ( ! is_array($lang))
168     {
169         $lang = array($lang => $val);
170     }
171
172     $this-> _error_messages = array_merge($this-> _error_messages, $lang);
173
174     return $this;
175 }
176
177 // -----
178
179 /**
180  * Set The Error Delimiter
181  *
182  * Permits a prefix/suffix to be added to each error message
183  *
184  * @access public
185  * @param string
186  * @param string
187  * @return void
188  */
189 public function set_error_delimiters($prefix = '<p>' , $suffix = '</p>' )
190 {
191     $this-> _error_prefix = $prefix;
192     $this-> _error_suffix = $suffix;
193
194     return $this;
195 }
196
197 // -----
198
199 /**
200  * Get Error Message
201  *
202  * Gets the error message associated with a particular field
203  *
204  * @access public
205  * @param string the field name
206  * @return void
207  */
208 public function error($field = '', $prefix = '', $suffix = '')
209 {
210     if ( ! isset($this-> _field_data[$field]['error']) OR $this->
211     > _field_data[$field]['error'] == '' )
212     {
213         return '';
214     }
215
216     if ($prefix == '')
217     {
218         $prefix = $this-> _error_prefix;
219     }
220
221     if ($suffix == '')
222     {
223         $suffix = $this-> _error_suffix;
224     }
225
226     return $prefix.$this-> _field_data[$field]['error'].$suffix;

```

```

227
228 // -----
229
230 /**
231  * Error String
232  *
233  * Returns the error messages as a string, wrapped in the error delimiters
234  *
235  * @access    public
236  * @param     string
237  * @param     string
238  * @return    str
239  */
240 public function error_string($prefix = '', $suffix = '')
241 {
242     // No errors, validation passes!
243     if (count($this-> _error_array) === 0)
244     {
245         return '';
246     }
247
248     if ($prefix == '')
249     {
250         $prefix = $this-> _error_prefix;
251     }
252
253     if ($suffix == '')
254     {
255         $suffix = $this-> _error_suffix;
256     }
257
258     // Generate the error string
259     $str = '';
260     foreach ($this-> _error_array as $val)
261     {
262         if ($val != '')
263         {
264             $str .= $prefix.$val.$suffix."\n"          ;
265         }
266     }
267
268     return $str;
269 }
270
271 // -----
272
273 /**
274  * Run the Validator
275  *
276  * This function does all the work.
277  *
278  * @access    public
279  * @return    bool
280  */
281 public function run($group = '')
282 {
283     // Do we even have any data to process? Mm?
284     if (count($_POST) == 0)
285     {
286         return FALSE;
287     }
288
289     // Does the _field_data array containing the validation rules exist?
290     // If not, we look to see if they were assigned via a config file
291     if (count($this-> _field_data) == 0)
292     {
293         // No validation rules? We're done...
294         if (count($this-> _config_rules) == 0)
295         {
296             return FALSE;
297         }
298
299         // Is there a validation rule for the particular URI being accessed?
300         $uri = ($group == '') ? trim($this-> CI-> uri-> ruri_string(), '/') : $group;
301
302         if ($uri != '' AND isset($this-> _config_rules[$uri]))
303         {
304             $this-> set_rules($this-> _config_rules[$uri]);
305         }
306         else

```

```

307         {
308             $this->    set_rules($this->    _config_rules);
309         }
310
311         // We're we able to set the rules correctly?
312         if (count($this->    _field_data) == 0)
313         {
314             log_message('debug', "Unable to find validation rules"
315             );
316             return FALSE;
317         }
318
319         // Load the language file containing error messages
320         $this->    CI->    lang->    load('form_validation');
321
322         // Cycle through the rules for each field, match the
323         // corresponding $_POST item and test for errors
324         foreach ($this->    _field_data as $field =>    $row)
325         {
326             // Fetch the data from the corresponding $_POST array and cache it in the
327             // _field_data array. // Depending on whether the field name is an array or a string will determine where
328             // we get it from.
329             if ($row['is_array'] == TRUE)
330             {
331                 $this->    _field_data[$field]['postdata'] = $this->    _reduce_array($_POST,
332                 $row['keys']);
333             }
334             else
335             {
336                 if (isset($_POST[$field]) AND $_POST[$field] != ""
337                 )
338                 {
339                     $this->    _field_data[$field]['postdata'] = $_POST[$field];
340                 }
341             }
342             $this->    _execute($row, explode('|', $row['rules']), $this-
343             >    _field_data[$field]['postdata']);
344
345             // Did we end up with any errors?
346             $total_errors = count($this->    _error_array);
347
348             if ($total_errors >    0)
349             {
350                 $this->    _safe_form_data = TRUE;
351             }
352
353             // Now we need to re-set the POST data with the new, processed data
354             $this->    _reset_post_array();
355
356             // No errors, validation passes!
357             if ($total_errors == 0)
358             {
359                 return TRUE;
360             }
361
362             // Validation fails
363             return FALSE;
364         }
365
366         // -----
367
368         /**
369          * Traverse a multidimensional $_POST array index until the data is found
370          *
371          * @access    private
372          * @param    array
373          * @param    array
374          * @param    integer
375          * @return    mixed
376          */
377         protected function _reduce_array($array, $keys, $i = 0)
378         {
379             if (is_array($array))
380             {
381                 if (isset($keys[$i]))
382                 {
383                     if (isset($array[$keys[$i]]))

```

```

383         {
384             $array = $this-> _reduce_array($array[$keys[$i]], $keys, ($i+1));
385         }
386         else
387         {
388             return NULL;
389         }
390     }
391     else
392     {
393         return $array;
394     }
395 }
396
397 return $array;
398 }
399
400 // -----
401
402 /**
403  * Re-populate the _POST array with our finalized and processed data
404  *
405  * @access private
406  * @return null
407  */
408 protected function _reset_post_array()
409 {
410     foreach ($this-> _field_data as $field => $row)
411     {
412         if ( ! is_null($row['postdata']))
413         {
414             if ($row['is_array'] == FALSE)
415             {
416                 if (isset($_POST[$row['field']]))
417                 {
418                     $_POST[$row['field']] = $this-> _prep_for_form($row['postdata']);
419                 }
420             }
421             else
422             {
423                 // start with a reference
424                 $post_ref =& $_POST;
425
426                 // before we assign values, make a reference to the right POST key
427                 if (count($row['keys']) == 1)
428                 {
429                     $post_ref =& $post_ref[current($row['keys'])];
430                 }
431                 else
432                 {
433                     foreach ($row['keys'] as $val)
434                     {
435                         $post_ref =& $post_ref[$val];
436                     }
437                 }
438
439                 if (is_array($row['postdata']))
440                 {
441                     $array = array();
442                     foreach ($row['postdata'] as $k => $v)
443                     {
444                         $array[$k] = $this-> _prep_for_form($v);
445                     }
446
447                     $post_ref = $array;
448                 }
449                 else
450                 {
451                     $post_ref = $this-> _prep_for_form($row['postdata']);
452                 }
453             }
454         }
455     }
456 }
457
458 // -----
459
460 /**
461  * Executes the Validation routines
462  */

```



```

463 * @access      private
464 * @param      array
465 * @param      array
466 * @param      mixed
467 * @param      integer
468 * @return     mixed
469 */
470 protected function _execute($row, $rules, $postdata = NULL, $cycles = 0)
471 {
472     // If the $_POST data is an array we will run a recursive call
473     if (is_array($postdata))
474     {
475         foreach ($postdata as $key => $val)
476         {
477             $this-> _execute($row, $rules, $val, $cycles);
478             $cycles++;
479         }
480     }
481     return;
482 }
483
484 // -----
485
486 // If the field is blank, but NOT required, no further tests are necessary
487 $callback = FALSE;
488 if ( ! in_array('required', $rules) AND is_null($postdata))
489 {
490     // Before we bail out, does the rule contain a callback?
491     if (preg_match("/(callback_\w+(\\[.*?\])?)/", $rules), implode(' ', $rules),
492 $match))
493     {
494         $callback = TRUE;
495         $rules = (array('1' => $match[1]));
496     }
497     else
498     {
499         return;
500     }
501 }
502
503 // -----
504
505 // Isset Test. Typically this rule will only apply to checkboxes.
506 if (is_null($postdata) AND $callback == FALSE)
507 {
508     if (in_array('isset', $rules, TRUE) OR in_array('required', $rules))
509     {
510         // Set the message type
511         $type = (in_array('required', $rules)) ? 'required' : 'isset';
512
513         if ( ! isset($this-> _error_messages[$type]))
514         {
515             if (FALSE === ($line = $this-> CI-> lang-> line($type)))
516             {
517                 $line = 'The field was not set';
518             }
519         }
520         else
521         {
522             $line = $this-> _error_messages[$type];
523         }
524
525         // Build the error message
526         $message = sprintf($line, $this-> _translate_fieldname($row['label']));
527
528         // Save the error message
529         $this-> _field_data[$row['field']]['error'] = $message;
530
531         if ( ! isset($this-> _error_array[$row['field']]))
532         {
533             $this-> _error_array[$row['field']] = $message;
534         }
535     }
536     return;
537 }
538
539 // -----
540
541 // Cycle through each rule and run it

```

```

542     foreach ($rules As $rule)
543     {
544         $_in_array = FALSE;
545
546         // We set the $postdata variable with the current data in our master array so that
547         // each cycle of the loop is dealing with the processed data from the last cycle
548         if ($row['is_array'] == TRUE AND is_array($this->
549 > field_data[$row['field']]['postdata']))
550         {
551             // We shouldn't need this safety, but just in case there isn't an array index
552             // associated with this cycle we'll bail out
553             if ( ! isset($this-> field_data[$row['field']]['postdata'][$cycles]))
554             {
555                 continue;
556             }
557             $postdata = $this-> field_data[$row['field']]['postdata'][$cycles];
558             $_in_array = TRUE;
559         }
560         else
561         {
562             $postdata = $this-> field_data[$row['field']]['postdata'];
563         }
564
565         // -----
566
567         // Is the rule a callback?
568         $callback = FALSE;
569         if (substr($rule, 0, 9) == 'callback_')
570         {
571             $rule = substr($rule, 9);
572             $callback = TRUE;
573         }
574
575         // Strip the parameter (if exists) from the rule
576         // Rules can contain a parameter: max_length[5]
577         $param = FALSE;
578         if (preg_match("/(.*?)\[([.*])\]/", $rule, $match))
579         {
580             $rule = $match[1];
581             $param = $match[2];
582         }
583
584         // Call the function that corresponds to the rule
585         if ($callback == TRUE)
586         {
587             if ( ! method_exists($this-> CI, $rule))
588             {
589                 continue;
590             }
591
592             // Run the function and grab the result
593             $result = $this-> CI-> $rule($postdata, $param);
594
595             // Re-assign the result to the master data array
596             if ($_in_array == TRUE)
597             {
598                 $this-> field_data[$row['field']]['postdata'][$cycles] =
599 (is_bool($result)) ? $postdata : $result;
600             }
601             else
602             {
603                 $this-> field_data[$row['field']]['postdata'] = (is_bool($result)) ?
604 $postdata : $result;
605             }
606
607             // If the field isn't required and we just processed a callback we'll move on...
608             if ( ! in_array('required', $rules, TRUE) AND $result != FALSE)
609             {
610                 continue;
611             }
612         }
613         else
614         {
615             if ( ! method_exists($this, $rule))
616             {
617                 // If our own wrapper function doesn't exist we see if a native PHP
618                 function does.
619
620                 // Users can use any native PHP function call that has one param.
621                 if (function_exists($rule))

```

```

618         {
619             $result = $rule($postdata);
620
621             if ($_in_array == TRUE)
622             {
623                 $this-> _field_data[$row['field']][ 'postdata' ][ $cycles ] =
624                 (is_bool($result)) ? $postdata : $result;
625             }
626             else
627             {
628                 $this-> _field_data[$row['field']][ 'postdata' ] =
629                 (is_bool($result)) ? $postdata : $result;
630             }
631             else
632             {
633                 log_message('debug', "Unable to find validation rule:
634                 "
635                 . $rule);
636             }
637             continue;
638         }
639         $result = $this-> _rule($postdata, $param);
640
641         if ($_in_array == TRUE)
642         {
643             $this-> _field_data[$row['field']][ 'postdata' ][ $cycles ] =
644             (is_bool($result)) ? $postdata : $result;
645         }
646         else
647         {
648             $this-> _field_data[$row['field']][ 'postdata' ] = (is_bool($result)) ?
649             $postdata : $result;
650         }
651     }
652     // Did the rule test negatively? If so, grab the error.
653     if ($result === FALSE)
654     {
655         if ( ! isset($this-> _error_messages[$rule]))
656         {
657             if (FALSE === ($line = $this-> CI-> lang-> line($rule)))
658             {
659                 $line = 'Unable to access an error message corresponding to your field
660                 name.';
661             }
662         }
663         else
664         {
665             $line = $this-> _error_messages[$rule];
666         }
667         // Is the parameter we are inserting into the error message the name
668         // of another field? If so we need to grab its "field label"
669         if (isset($this-> _field_data[$param]) AND isset($this->
670         > _field_data[$param][ 'label' ]))
671         {
672             $param = $this-> _translate_fieldname($this->
673             > _field_data[$param][ 'label' ]);
674         }
675         // Build the error message
676         $message = sprintf($line, $this-> _translate_fieldname($row[ 'label' ]),
677         $param);
678         // Save the error message
679         $this-> _field_data[$row['field']][ 'error' ] = $message;
680
681         if ( ! isset($this-> _error_array[$row['field']]))
682         {
683             $this-> _error_array[$row['field']] = $message;
684         }
685         return;
686     }
687 }
688 // -----

```

```

689
690 /**
691  * Translate a field name
692  *
693  * @access    private
694  * @param     string    the field name
695  * @return    string
696  */
697 protected function _translate_fieldname($fieldname)
698 {
699     // Do we need to translate the field name?
700     // We look for the prefix lang: to determine this
701     if (substr($fieldname, 0, 5) == 'lang:')
702     {
703         // Grab the variable
704         $line = substr($fieldname, 5);
705
706         // Were we able to translate the field name? If not we use $line
707         if (FALSE == ($fieldname = $this-> CI-> lang-> line($line)))
708         {
709             return $line;
710         }
711     }
712
713     return $fieldname;
714 }
715
716 // -----
717
718 /**
719  * Get the value from a form
720  *
721  * Permits you to repopulate a form field with the value it was submitted
722  * with, or, if that value doesn't exist, with the default
723  *
724  * @access    public
725  * @param     string    the field name
726  * @param     string
727  * @return    void
728  */
729 public function set_value($field = '', $default = '')
730 {
731     if ( ! isset($this-> _field_data[$field]))
732     {
733         return $default;
734     }
735
736     // If the data is an array output them one at a time.
737     // E.g: form_input('name[]', set_value('name[]'));
738     if (is_array($this-> _field_data[$field]['postdata']))
739     {
740         return array_shift($this-> _field_data[$field]['postdata']);
741     }
742
743     return $this-> _field_data[$field]['postdata'];
744 }
745
746 // -----
747
748 /**
749  * Set Select
750  *
751  * Enables pull-down lists to be set to the value the user
752  * selected in the event of an error
753  *
754  * @access    public
755  * @param     string
756  * @param     string
757  * @return    string
758  */
759 public function set_select($field = '', $value = '', $default = FALSE)
760 {
761     if ( ! isset($this-> _field_data[$field]) OR ! isset($this->
762 > _field_data[$field]['postdata']))
763     {
764         if ($default == TRUE AND count($this-> _field_data) == 0)
765         {
766             return ' selected="selected" ' ;
767         }
768         return '';
769     }

```

```

768     }
769
770     $field = $this->    _field_data[$field]['postdata'];
771
772     if (is_array($field))
773     {
774         if ( ! in_array($value, $field))
775         {
776             return '';
777         }
778     }
779     else
780     {
781         if (($field == '' OR $value == '') OR ($field != $value))
782         {
783             return '';
784         }
785     }
786
787     return ' selected="selected"'          ;
788 }
789
790 // -----
791
792 /**
793  * Set Radio
794  *
795  * Enables radio buttons to be set to the value the user
796  * selected in the event of an error
797  *
798  * @access      public
799  * @param       string
800  * @param       string
801  * @return      string
802  */
803 public function set_radio($field = '', $value = '', $default = FALSE)
804 {
805     if ( ! isset($this->    _field_data[$field]) OR ! isset($this->
806 >    _field_data[$field]['postdata']))
807     {
808         if ($default === TRUE AND count($this->    _field_data) === 0)
809         {
810             return ' checked="checked"'          ;
811         }
812         return '';
813     }
814
815     $field = $this->    _field_data[$field]['postdata'];
816
817     if (is_array($field))
818     {
819         if ( ! in_array($value, $field))
820         {
821             return '';
822         }
823     }
824     else
825     {
826         if (($field == '' OR $value == '') OR ($field != $value))
827         {
828             return '';
829         }
830     }
831
832     return ' checked="checked"'          ;
833 }
834
835 // -----
836
837 /**
838  * Set Checkbox
839  *
840  * Enables checkboxes to be set to the value the user
841  * selected in the event of an error
842  *
843  * @access      public
844  * @param       string
845  * @param       string
846  * @return      string
847  */

```

```

847     public function set_checkbox($field = '', $value = '', $default = FALSE)
848     {
849         if ( ! isset($this-> __field_data[$field]) OR ! isset($this->
850 > __field_data[$field]['postdata']))
851         {
852             if ($default === TRUE AND count($this-> __field_data) === 0)
853             {
854                 return ' checked="checked" ' ;
855             }
856             return '';
857         }
858         $field = $this-> __field_data[$field]['postdata'];
859
860         if (is_array($field))
861         {
862             if ( ! in_array($value, $field))
863             {
864                 return '';
865             }
866         }
867         else
868         {
869             if (($field == '' OR $value == '') OR ($field != $value))
870             {
871                 return '';
872             }
873         }
874
875         return ' checked="checked" ' ;
876     }
877
878     // -----
879
880     /**
881     * Required
882     *
883     * @access      public
884     * @param       string
885     * @return      bool
886     */
887     public function required($str)
888     {
889         if ( ! is_array($str))
890         {
891             return (trim($str) == '') ? FALSE : TRUE;
892         }
893         else
894         {
895             return ( ! empty($str));
896         }
897     }
898
899     // -----
900
901     /**
902     * Performs a Regular Expression match test.
903     *
904     * @access      public
905     * @param       string
906     * @param       regex
907     * @return      bool
908     */
909     public function regex_match($str, $regex)
910     {
911         if ( ! preg_match($regex, $str))
912         {
913             return FALSE;
914         }
915
916         return TRUE;
917     }
918
919     // -----
920
921     /**
922     * Match one field to another
923     *
924     * @access      public
925     * @param       string

```

```

926     * @param    field
927     * @return    bool
928     */
929     public function matches($str, $field)
930     {
931         if ( ! isset($_POST[$field]))
932         {
933             return FALSE;
934         }
935
936         $field = $_POST[$field];
937
938         return ($str != $field) ? FALSE : TRUE;
939     }
940
941     // -----
942
943     /**
944     * Match one field to another
945     *
946     * @access    public
947     * @param    string
948     * @param    field
949     * @return    bool
950     */
951     public function is_unique($str, $field)
952     {
953         list($table, $field)=explode('.', $field);
954         $query = $this-> CI-> db-> limit(1)-> get_where($table, array($field =>
955 $str));
956
957         return $query-> num_rows() == 0;
958     }
959
960     // -----
961
962     /**
963     * Minimum Length
964     *
965     * @access    public
966     * @param    string
967     * @param    value
968     * @return    bool
969     */
970     public function min_length($str, $val)
971     {
972         if (preg_match("/^[^0-9]/" , $val))
973         {
974             return FALSE;
975         }
976
977         if (function_exists('mb_strlen'))
978         {
979             return (mb_strlen($str) < $val) ? FALSE : TRUE;
980         }
981
982         return (strlen($str) < $val) ? FALSE : TRUE;
983     }
984
985     // -----
986
987     /**
988     * Max Length
989     *
990     * @access    public
991     * @param    string
992     * @param    value
993     * @return    bool
994     */
995     public function max_length($str, $val)
996     {
997         if (preg_match("/^[^0-9]/" , $val))
998         {
999             return FALSE;
1000         }
1001
1002         if (function_exists('mb_strlen'))
1003         {
1004             return (mb_strlen($str) > $val) ? FALSE : TRUE;

```

```

1005         return (strlen($str) > $val) ? FALSE : TRUE;
1006     }
1007 }
1008 // -----
1009 /**
1010  * Exact Length
1011  *
1012  * @access public
1013  * @param string
1014  * @param value
1015  * @return bool
1016  */
1017 public function exact_length($str, $val)
1018 {
1019     if (preg_match("/^[0-9]/" , $val))
1020     {
1021         return FALSE;
1022     }
1023
1024     if (function_exists('mb_strlen'))
1025     {
1026         return (mb_strlen($str) != $val) ? FALSE : TRUE;
1027     }
1028
1029     return (strlen($str) != $val) ? FALSE : TRUE;
1030 }
1031 // -----
1032 /**
1033  * Valid Email
1034  *
1035  * @access public
1036  * @param string
1037  * @return bool
1038  */
1039 public function valid_email($str)
1040 {
1041     return ( ! preg_match("/^([a-z0-9\+\_\-]+)(\.[a-z0-9\+\_\-]+)*@([a-z0-9\+\_\-]+\.[a-
1042 z]{2,6})$/ix" , $str)) ? FALSE : TRUE;
1043 }
1044 // -----
1045 /**
1046  * Valid Emails
1047  *
1048  * @access public
1049  * @param string
1050  * @return bool
1051  */
1052 public function valid_emails($str)
1053 {
1054     if (strpos($str, ',') === FALSE)
1055     {
1056         return $this-> valid_email(trim($str));
1057     }
1058
1059     foreach (explode(',', $str) as $email)
1060     {
1061         if (trim($email) != '' && $this-> valid_email(trim($email)) === FALSE)
1062         {
1063             return FALSE;
1064         }
1065     }
1066
1067     return TRUE;
1068 }
1069 // -----
1070 /**
1071  * Validate IP Address
1072  *
1073  * @access public
1074  * @param string
1075  * @param string "ipv4" or "ipv6" to validate a specific ip format
1076  * @return string

```



```

1084     */
1085     public function valid_ip($ip, $which = '')
1086     {
1087         return $this-> CI-> input-> valid_ip($ip, $which);
1088     }
1089
1090     // -----
1091
1092     /**
1093      * Alpha
1094      *
1095      * @access      public
1096      * @param       string
1097      * @return      bool
1098      */
1099     public function alpha($str)
1100     {
1101         return ( ! preg_match("/^([a-z])+$/i" , $str)) ? FALSE : TRUE;
1102     }
1103
1104     // -----
1105
1106     /**
1107      * Alpha-numeric
1108      *
1109      * @access      public
1110      * @param       string
1111      * @return      bool
1112      */
1113     public function alpha_numeric($str)
1114     {
1115         return ( ! preg_match("/^([a-z0-9])+$/i" , $str)) ? FALSE : TRUE;
1116     }
1117
1118     // -----
1119
1120     /**
1121      * Alpha-numeric with underscores and dashes
1122      *
1123      * @access      public
1124      * @param       string
1125      * @return      bool
1126      */
1127     public function alpha_dash($str)
1128     {
1129         return ( ! preg_match("/^([-a-z0-9_-])+$/i" , $str)) ? FALSE : TRUE;
1130     }
1131
1132     // -----
1133
1134     /**
1135      * Numeric
1136      *
1137      * @access      public
1138      * @param       string
1139      * @return      bool
1140      */
1141     public function numeric($str)
1142     {
1143         return (bool)preg_match( '/^[\\-+]?[0-9]*\\.?[0-9]+$/', $str);
1144     }
1145
1146     // -----
1147
1148     /**
1149      * Is Numeric
1150      *
1151      * @access      public
1152      * @param       string
1153      * @return      bool
1154      */
1155     public function is_numeric($str)
1156     {
1157         return ( ! is_numeric($str)) ? FALSE : TRUE;
1158     }
1159
1160     // -----
1161
1162     /**

```

```

1164     * Integer
1165     *
1166     * @access    public
1167     * @param     string
1168     * @return    bool
1169     */
1170 public function integer($str)
1171 {
1172     return (bool) preg_match('/^[\\-+]?[0-9]+$/', $str);
1173 }
1174
1175 // -----
1176
1177 /**
1178  * Decimal number
1179  *
1180  * @access    public
1181  * @param     string
1182  * @return    bool
1183  */
1184 public function decimal($str)
1185 {
1186     return (bool) preg_match('/^[\\-+]?[0-9]+\\.?[0-9]+$/', $str);
1187 }
1188
1189 // -----
1190
1191 /**
1192  * Greather than
1193  *
1194  * @access    public
1195  * @param     string
1196  * @return    bool
1197  */
1198 public function greater_than($str, $min)
1199 {
1200     if ( ! is_numeric($str) )
1201     {
1202         return FALSE;
1203     }
1204     return $str >    $min;
1205 }
1206
1207 // -----
1208
1209 /**
1210  * Less than
1211  *
1212  * @access    public
1213  * @param     string
1214  * @return    bool
1215  */
1216 public function less_than($str, $max)
1217 {
1218     if ( ! is_numeric($str) )
1219     {
1220         return FALSE;
1221     }
1222     return $str <    $max;
1223 }
1224
1225 // -----
1226
1227 /**
1228  * Is a Natural number (0,1,2,3, etc.)
1229  *
1230  * @access    public
1231  * @param     string
1232  * @return    bool
1233  */
1234 public function is_natural($str)
1235 {
1236     return (bool) preg_match( '/^[0-9]+$/', $str);
1237 }
1238
1239 // -----
1240
1241 /**
1242  * Is a Natural number, but not a zero (1,2,3, etc.)
1243  *

```

```

1244 * @access public
1245 * @param string
1246 * @return bool
1247 */
1248 public function is_natural_no_zero($str)
1249 {
1250     if ( ! preg_match( '/^[0-9]+$/', $str))
1251     {
1252         return FALSE;
1253     }
1254
1255     if ($str == 0)
1256     {
1257         return FALSE;
1258     }
1259
1260     return TRUE;
1261 }
1262
1263 // -----
1264
1265 /**
1266 * Valid Base64
1267 *
1268 * Tests a string for characters outside of the Base64 alphabet
1269 * as defined by RFC 2045 http://www.faqs.org/rfcs/rfc2045
1270 *
1271 * @access public
1272 * @param string
1273 * @return bool
1274 */
1275 public function valid_base64($str)
1276 {
1277     return (bool) ! preg_match('/[^\a-zA-Z0-9\|\+=\/]', $str);
1278 }
1279
1280 // -----
1281
1282 /**
1283 * Prep data for form
1284 *
1285 * This function allows HTML to be safely shown in a form.
1286 * Special characters are converted.
1287 *
1288 * @access public
1289 * @param string
1290 * @return string
1291 */
1292 public function prep_for_form($data = '')
1293 {
1294     if (is_array($data))
1295     {
1296         foreach ($data as $key => $val)
1297         {
1298             $data[$key] = $this-> prep_for_form($val);
1299         }
1300
1301         return $data;
1302     }
1303
1304     if ($this-> _safe_form_data == FALSE OR $data === '')
1305     {
1306         return $data;
1307     }
1308
1309     return str_replace(array('"', '<', '>', '&#39;', '"', '<', '>', '&#39;', '"', '<', '>' ),
array("&#39;", "&quot;", "&lt;", "&gt;", "&#39;", "&quot;", "&lt;", "&gt;" , stripslashes($data));
1310 }
1311
1312 // -----
1313
1314 /**
1315 * Prep URL
1316 *
1317 * @access public
1318 * @param string
1319 * @return string
1320 */
1321 public function prep_url($str = '')
1322 {

```

```

1323     if ($str == 'http://' OR $str == '')
1324     {
1325         return '';
1326     }
1327
1328     if (substr($str, 0, 7) != 'http://' &&         substr($str, 0, 8) != 'https://')
1329     {
1330         $str = 'http://'.$str;
1331     }
1332
1333     return $str;
1334 }
1335
1336 // -----
1337
1338 /**
1339  * Strip Image Tags
1340  *
1341  * @access    public
1342  * @param    string
1343  * @return    string
1344  */
1345 public function strip_image_tags($str)
1346 {
1347     return $this-> CI-> input-> strip_image_tags($str);
1348 }
1349
1350 // -----
1351
1352 /**
1353  * XSS Clean
1354  *
1355  * @access    public
1356  * @param    string
1357  * @return    string
1358  */
1359 public function xss_clean($str)
1360 {
1361     return $this-> CI-> security-> xss_clean($str);
1362 }
1363
1364 // -----
1365
1366 /**
1367  * Convert PHP tags to entities
1368  *
1369  * @access    public
1370  * @param    string
1371  * @return    string
1372  */
1373 public function encode_php_tags($str)
1374 {
1375     return str_replace(array('<?php' , '<?PHP' , '<?' , '?>' ),
array('&lt;?php' , '&lt;?PHP' , '&lt;?' , '?&gt;' ), $str);
1376 }
1377
1378 }
1379 // END Form Validation Class
1380
1381 /* End of file Form_validation.php */
1382 /* Location: ./system/libraries/Form_validation.php */

```

Archivo fuente para Pagination.php

La documentación para este archivo está disponible en [Pagination.php](#)

```
1  <?php      if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2  /**
3   * CodeIgniter
4   *
5   * An open source application development framework for PHP 5.1.6 or newer
6   *
7   * @package      CodeIgniter
8   * @author        ExpressionEngine Dev Team
9   * @copyright      Copyright (c) 2008 - 2011, EllisLab, Inc.
10  * @license        http://codeigniter.com/user_guide/license.html
11  * @link          http://codeigniter.com
12  * @since          Version 1.0
13  * @filesource
14  */
15
16  // -----
17
18  /**
19   * Pagination Class
20   *
21   * @package      CodeIgniter
22   * @subpackage    Libraries
23   * @category      Pagination
24   * @author        ExpressionEngine Dev Team
25   * @link          http://codeigniter.com/user_guide/libraries/pagination.html
26   */
27  class CI_Pagination {
28
29      var $base_url          = ''; // The page we are linking to
30      var $prefix            = ''; // A custom prefix added to the path.
31      var $suffix            = ''; // A custom suffix added to the path.
32
33      var $total_rows        = 0; // Total number of items (database results)
34      var $per_page          = 10; // Max number of items you want shown per page
35      var $num_links         = 2; // Number of "digit" links to show before/after
36      the currently viewed page
37      var $cur_page          = 0; // The current page being viewed
38      var $use_page_numbers  = FALSE; // Use page number for segment instead of offset
39      var $first_link        = '&lsaquo; First' ;
40      var $next_link         = '&gt;' ;
41      var $prev_link         = '&lt;' ;
42      var $last_link         = 'Last &rsaquo;' ;
43      var $uri_segment       = 3;
44      var $full_tag_open     = '<ul class="pagination pagination-sm">' ;
45      var $full_tag_close    = '</ul>' ;
46      var $first_tag_open   = '<li>' ;
47      var $first_tag_close  = '</li>' ;
48      var $last_tag_open    = '<li>' ;
49      var $last_tag_close   = '</li>' ;
50      var $first_url         = ''; // Alternative URL for the First Page.
51      var $cur_tag_open     = '<li><a><strong>' ;
52      var $cur_tag_close    = '</strong></a></li>' ;
53      var $next_tag_open    = '<li>' ;
54      var $next_tag_close   = '</li>' ;
55      var $prev_tag_open    = '<li>' ;
56      var $prev_tag_close   = '</li>' ;
57      var $num_tag_open     = '<li>' ;
58      var $num_tag_close    = '</li>' ;
59      var $page_query_string = FALSE;
60      var $query_string_segment = 'per_page';
61      var $display_pages     = TRUE;
62      var $anchor_class      = '';
63
64      /**
65       * Constructor
66       *
67       * @access public
```

```

67      * @param      array      initialization parameters
68      */
69      public function __construct($params = array())
70      {
71          if (count($params) > 0)
72          {
73              $this-> initialize($params);
74          }
75
76          if ($this-> anchor_class != '')
77          {
78              $this-> anchor_class = 'class="' . $this-> anchor_class . '" ' . ' ';
79          }
80
81          log_message('debug', "Pagination Class Initialized" . ' ');
82      }
83
84      // -----
85
86      /**
87       * Initialize Preferences
88       *
89       * @access      public
90       * @param      array      initialization parameters
91       * @return      void
92       */
93      function initialize($params = array())
94      {
95          if (count($params) > 0)
96          {
97              foreach ($params as $key => $val)
98              {
99                  if (isset($this-> $key))
100                  {
101                      $this-> $key = $val;
102                  }
103              }
104          }
105      }
106
107      // -----
108
109      /**
110       * Generate the pagination links
111       *
112       * @access      public
113       * @return      string
114       */
115      function create_links()
116      {
117          // If our item count or per-page total is zero there is no need to continue.
118          if ($this-> total_rows == 0 OR $this-> per_page == 0)
119          {
120              return '';
121          }
122
123          // Calculate the total number of pages
124          $num_pages = ceil($this-> total_rows / $this-> per_page);
125
126          // Is there only one page? Hm... nothing more to do here then.
127          if ($num_pages == 1)
128          {
129              return '';
130          }
131
132          // Set the base page index for starting page number
133          if ($this-> use_page_numbers)
134          {
135              $base_page = 1;
136          }
137          else
138          {
139              $base_page = 0;
140          }
141
142          // Determine the current page number.
143          $CI =& get_instance();
144
145          if ($CI-> config-> item('enable_query_strings') === TRUE OR $this->
146      > page_query_string === TRUE)

```

```

146 {
147     if ($CI-> input-> get($this-> query_string_segment) != $base_page)
148     {
149         $this-> cur_page = $CI-> input-> get($this-> query_string_segment);
150
151         // Prep the current page - no funny business!
152         $this-> cur_page = (int) $this-> cur_page;
153     }
154 }
155 else
156 {
157     if ($CI-> uri-> segment($this-> uri_segment) != $base_page)
158     {
159         $this-> cur_page = $CI-> uri-> segment($this-> uri_segment);
160
161         // Prep the current page - no funny business!
162         $this-> cur_page = (int) $this-> cur_page;
163     }
164 }
165
166 // Set current page to 1 if using page numbers instead of offset
167 if ($this-> use_page_numbers AND $this-> cur_page == 0)
168 {
169     $this-> cur_page = $base_page;
170 }
171
172 $this-> num_links = (int)$this-> num_links;
173
174 if ($this-> num_links < 1)
175 {
176     show_error('Your number of links must be a positive number.');
```

```

222     }
223
224     // And here we go...
225     $output = '';
226
227     // Render the "First" link
228     if ($this-> first_link !== FALSE AND $this-> cur_page > ( $this-> num_links +
229 1))
230     {
231         $first_url = ($this-> first_url == '') ? $this-> base_url : $this-> first_url;
232         $output .= $this-> first_tag_open.'<a ' . $this->
233         > anchor_class.'href="' . $first_url.'">' . $this-> first_link.'</a>' . $this->
234         > first_tag_close;
235     }
236
237     // Render the "previous" link
238     if ($this-> prev_link !== FALSE AND $this-> cur_page != 1)
239     {
240         if ($this-> use_page_numbers)
241         {
242             $i = $uri_page_number - 1;
243         }
244         else
245         {
246             $i = $uri_page_number - $this-> per_page;
247         }
248
249         if ($i == 0 && $this-> first_url != '')
250         {
251             $output .= $this-> prev_tag_open.'<a ' . $this->
252             > anchor_class.'href="' . $this-> first_url.'">' . $this->
253             > prev_link.'</a>' . $this-> prev_tag_close;
254         }
255         else
256         {
257             $i = ($i == 0) ? '' : $this-> prefix.$i.$this-> suffix;
258             $output .= $this-> prev_tag_open.'<a ' . $this->
259             > anchor_class.'href="' . $this-> base_url.$i.'">' . $this->
260             > prev_link.'</a>' . $this-> prev_tag_close;
261         }
262     }
263
264     // Render the pages
265     if ($this-> display_pages !== FALSE)
266     {
267         // Write the digit links
268         for ($loop = $start - 1; $loop <= $end; $loop++)
269         {
270             if ($this-> use_page_numbers)
271             {
272                 $i = $loop;
273             }
274             else
275             {
276                 $i = ($loop * $this-> per_page) - $this-> per_page;
277             }
278
279             if ($i >= $base_page)
280             {
281                 if ($this-> cur_page == $loop)
282                 {
283                     $output .= $this-> cur_tag_open.$loop.$this-> cur_tag_close; //
284                     Current page
285                 }
286                 else
287                 {
288                     $n = ($i == $base_page) ? '' : $i;
289
290                     if ($n == '' && $this-> first_url != '')
291                     {
292                         $output .= $this-> num_tag_open.'<a ' . $this->
293                         > anchor_class.'href="' . $this-> first_url.'">' . $loop.'</a>' . $this->
294                         > num_tag_close;
295                     }
296                     else
297                     {
298                         $n = ($n == '') ? '' : $this-> prefix.$n.$this-> suffix;
299                         $output .= $this-> num_tag_open.'<a ' . $this->

```



```

> anchor_class.'href="' . $this-> base_url.$n.">" . $loop.'</a>' . $this->
> num_tag_close;
292     }
293     }
294     }
295     }
296 }
297
298 // Render the "next" link
299 if ($this-> next_link !== FALSE AND $this-> cur_page < $num_pages)
300 {
301     if ($this-> use_page_numbers)
302     {
303         $i = $this-> cur_page + 1;
304     }
305     else
306     {
307         $i = ($this-> cur_page * $this-> per_page);
308     }
309
310     $output .= $this-> next_tag_open.'<a ' . $this->
> anchor_class.'href="' . $this-> base_url.$this-> prefix.$i.$this->
> suffix.">" . $this-> next_link.'</a>' . $this-> next_tag_close;
311 }
312
313 // Render the "Last" link
314 if ($this-> last_link !== FALSE AND ($this-> cur_page + $this-> num_links) <
$num_pages)
315 {
316     if ($this-> use_page_numbers)
317     {
318         $i = $num_pages;
319     }
320     else
321     {
322         $i = (($num_pages * $this-> per_page) - $this-> per_page);
323     }
324     $output .= $this-> last_tag_open.'<a ' . $this->
> anchor_class.'href="' . $this-> base_url.$this-> prefix.$i.$this->
> suffix.">" . $this-> last_link.'</a>' . $this-> last_tag_close;
325 }
326
327 // Kill double slashes. Note: Sometimes we can end up with a double slash
328 // in the penultimate link so we'll kill all double slashes.
329 $output = preg_replace("#([^\:]+)/+#" , "\\1/" , $output);
330
331 // Add the wrapper HTML if exists
332 $output = $this-> full_tag_open.$output.$this-> num_tag_open.'<span> Total de
registros: '.number_format($this-> total_rows).'</span>' . $this-> num_tag_close.$this->
> full_tag_close;
333
334     return $output;
335 }
336 }
337 // END Pagination Class
338
339 /* End of file Pagination.php */
340 /* Location: ./system/libraries/Pagination.php */

```

Archivo fuente para template.php

La documentación para este archivo está disponible en [template.php](#)

```
1  <?php      if (!defined('BASEPATH')) exit('No direct script access allowed');
2  /**
3   * CodeIgniter
4   *
5   * An open source application development framework for PHP 4.3.2 or newer
6   *
7   * @package CodeIgniter
8   * @author ExpressionEngine Dev Team
9   * @copyright Copyright (c) 2006, EllisLab, Inc.
10  * @license http://codeigniter.com/user_guide/license.html
11  * @link http://codeigniter.com
12  * @since Version 1.0
13  * @filesource
14  */
15
16  // -----
17
18  /**
19   * CodeIgniter Template Class
20   *
21   * This class is an interface to CI's View class. It aims to improve the
22   * interaction between controllers and views. Follow @link for more info
23   *
24   * @package CodeIgniter
25   * @author Colin Williams
26   * @subpackage Libraries
27   * @category Libraries
28   * @link http://www.williamsconcepts.com/ci/libraries/template/index.html
29   * @copyright Copyright (c) 2008, Colin Williams.
30   * @version 1.4.1
31   */
32
33  class CI_Template {
34
35      var $CI;
36      var $config;
37      var $template;
38      var $master;
39      var $regions = array(
40          '_scripts' => array(),
41          '_styles' => array(),
42      );
43      var $output;
44      var $js = array();
45      var $css = array();
46      var $parser = 'parser';
47      var $parser_method = 'parse';
48      var $parse_template = FALSE;
49
50      /**
51       * Constructor
52       *
53       * Loads template configuration, template regions, and validates existence of
54       * default template
55       *
56       * @access public
57       */
58
59      function CI_Template()
60      {
61          // Copy an instance of CI so we can use the entire framework.
62          $this->CI =& get_instance();
63
64          // Load the template config file and setup our master template and regions
65          include(APPPATH.'config/template'.EXT);
66          if (isset($template))
67          {
```

```

68     $this-> config = $template;
69     $this-> set_template($template['active_template']);
70 }
71 }
72
73 // -----
74
75 /**
76  * Use given template settings
77  *
78  * @access public
79  * @param string array key to access template settings
80  * @return void
81  */
82
83 function set_template($group)
84 {
85     if (isset($this-> config[$group]))
86     {
87         $this-> template = $this-> config[$group];
88     }
89     else
90     {
91         show_error('The "' . $group . '" template group does not exist. Provide a
valid group name or add the group first.');
```

```

144
145 // -----
146
147 /**
148  * Initialize class settings using config settings
149  *
150  * @access public
151  * @param array configuration array
152  * @return void
153  */
154
155 function initialize($props)
156 {
157     // Set master template
158     if (isset($props['template'])
159         && (file_exists(APPPATH . 'views/' . $props['template']) or file_exists(APPPATH
160     . 'views/' . $props['template'] . EXT)))
161     {
162         $this-> master = $props['template'];
163     }
164     else
165     {
166         // Master template must exist. Throw error.
167         show_error('Either you have not provided a master template or the one provided does
168         not exist in <strong>' . APPPATH . 'views</strong>. Remember to include the extension if
169         other than ".php"');
170     }
171
172     // Load our regions
173     if (isset($props['regions']))
174     {
175         $this-> set_regions($props['regions']);
176     }
177
178     // Set parser and parser method
179     if (isset($props['parser']))
180     {
181         $this-> set_parser($props['parser']);
182     }
183     if (isset($props['parser_method']))
184     {
185         $this-> set_parser_method($props['parser_method']);
186     }
187
188     // Set master template parser instructions
189     $this-> parse_template = isset($props['parse_template']) ? $props['parse_template'] :
190 FALSE;
191 }
192
193 // -----
194
195 /**
196  * Set regions for writing to
197  *
198  * @access public
199  * @param array properly formed regions array
200  * @return void
201  */
202
203 function set_regions($regions)
204 {
205     if (count($regions))
206     {
207         $this-> regions = array(
208             '_scripts' => array(),
209             '_styles' => array(),
210         );
211         foreach ($regions as $key => $region)
212         {
213             // Regions must be arrays, but we take the burden off the template
214             // developer and insure it here
215             if ( ! is_array($region))
216             {
217                 $this-> add_region($region);
218             }
219             else {
220                 $this-> add_region($key, $region);
221             }
222         }
223     }
224 }

```

```

220     }
221
222     // -----
223
224     /**
225      * Dynamically add region to the currently set template
226      *
227      * @access public
228      * @param string Name to identify the region
229      * @param array Optional array with region defaults
230      * @return void
231      */
232
233     function add_region($name, $props = array())
234     {
235         if ( ! is_array($props))
236         {
237             $props = array();
238         }
239
240         if ( ! isset($this-> regions[$name]))
241         {
242             $this-> regions[$name] = $props;
243         }
244         else
245         {
246             show_error('The " ' . $name . '" region has already been defined.' );
247         }
248     }
249
250     // -----
251
252     /**
253      * Empty a region's content
254      *
255      * @access public
256      * @param string Name to identify the region
257      * @return void
258      */
259
260     function empty_region($name)
261     {
262         if (isset($this-> regions[$name]['content']))
263         {
264             $this-> regions[$name]['content'] = array();
265         }
266         else
267         {
268             show_error('The " ' . $name . '" region is undefined.' );
269         }
270     }
271
272     // -----
273
274     /**
275      * Set parser
276      *
277      * @access public
278      * @param string name of parser class to load and use for parsing methods
279      * @return void
280      */
281
282     function set_parser($parser, $method = NULL)
283     {
284         $this-> parser = $parser;
285         $this-> CI-> load-> library($parser);
286
287         if ($method)
288         {
289             $this-> set_parser_method($method);
290         }
291     }
292
293     // -----
294
295     /**
296      * Set parser method
297      *
298      * @access public
299      * @param string name of parser class member function to call when parsing

```

```

300     * @return void
301     */
302
303     function set_parser_method($method)
304     {
305         $this-> parser_method = $method;
306     }
307
308     // -----
309
310     /**
311      * Write contents to a region
312      *
313      * @access public
314      * @param string region to write to
315      * @param string what to write
316      * @param boolean FALSE to append to region, TRUE to overwrite region
317      * @return void
318      */
319
320     function write($region, $content, $overwrite = FALSE)
321     {
322         if (isset($this-> regions[$region]))
323         {
324             if ($overwrite === TRUE) // Should we append the content or overwrite it
325             {
326                 $this-> regions[$region]['content'] = array($content);
327             } else {
328                 $this-> regions[$region]['content'][] = $content;
329             }
330         }
331
332         // Regions MUST be defined
333         else
334         {
335             show_error(" Cannot write to the '{$region}' region. The region is
336             undefined." );
337         }
338
339         // -----
340
341         /**
342          * Write content from a View to a region. 'Views within views'
343          *
344          * @access public
345          * @param string region to write to
346          * @param string view file to use
347          * @param array variables to pass into view
348          * @param boolean FALSE to append to region, TRUE to overwrite region
349          * @return void
350          */
351
352         function write_view($region, $view, $data = NULL, $overwrite = FALSE)
353         {
354             $args = func_get_args();
355
356             // Get rid of non-views
357             unset($args[0], $args[2], $args[3]);
358
359             // Do we have more view suggestions?
360             if (count($args) > 1)
361             {
362                 foreach ($args as $suggestion)
363                 {
364                     if (file_exists(APPPATH . 'views/' . $suggestion . EXT) or file_exists(APPPATH
365                     . 'views/' . $suggestion))
366                     {
367                         // Just change the $view arg so the rest of our method works as normal
368                         $view = $suggestion;
369                         break;
370                     }
371                 }
372             }
373
374             $content = $this-> CI-> load-> view($view, $data, TRUE);
375             $this-> write($region, $content, $overwrite);
376         }
377

```

```

378 // -----
379
380 /**
381  * Parse content from a View to a region with the Parser Class
382  *
383  * @access public
384  * @param string region to write to
385  * @param string view file to parse
386  * @param array variables to pass into view for parsing
387  * @param boolean FALSE to append to region, TRUE to overwrite region
388  * @return void
389  */
390
391 function parse_view($region, $view, $data = NULL, $overwrite = FALSE)
392 {
393     $this-> CI-> load-> library('parser');
394
395     $args = func_get_args();
396
397     // Get rid of non-views
398     unset($args[0], $args[2], $args[3]);
399
400     // Do we have more view suggestions?
401     if (count($args) > 1)
402     {
403         foreach ($args as $suggestion)
404         {
405             if (file_exists(APPPATH . 'views/' . $suggestion . EXT) or file_exists(APPPATH
406             . 'views/' . $suggestion))
407             {
408                 // Just change the $view arg so the rest of our method works as normal
409                 $view = $suggestion;
410                 break;
411             }
412         }
413     }
414
415     $content = $this-> CI->{ $this-> parser}->{ $this-> parser_method}($view,
416     $data, TRUE);
417     $this-> write($region, $content, $overwrite);
418 }
419 // -----
420
421 /**
422  * Dynamically include javascript in the template
423  *
424  * NOTE: This function does NOT check for existence of .js file
425  *
426  * @access public
427  * @param string script to import or embed
428  * @param string 'import' to load external file or 'embed' to add as-is
429  * @param boolean TRUE to use 'defer' attribute, FALSE to exclude it
430  * @return TRUE on success, FALSE otherwise
431  */
432
433 function add_js($script, $type = 'import', $defer = FALSE)
434 {
435     $success = TRUE;
436     $js = NULL;
437
438     $this-> CI-> load-> helper('url');
439
440     switch ($type)
441     {
442         case 'import':
443             $filepath = base_url() . $script;
444             $js = '<script type="text/javascript" src="' . $filepath . '"';
445             if ($defer)
446             {
447                 $js .= ' defer="defer"';
448             }
449             $js .= "></script>";
450             break;
451
452         case 'embed':
453             $js = '<script type="text/javascript"' . $script . '>';
454             if ($defer)
455             {

```

```

456         $js .= ' defer="defer" ' ;
457     }
458     $js .= ">" ;
459     $js .= $script;
460     $js .= '</script>' ;
461     break;
462
463     default:
464         $success = FALSE;
465         break;
466 }
467
468 // Add to js array if it doesn't already exist
469 if ($js != NULL && ! in_array($js, $this-> js))
470 {
471     $this-> js[] = $js;
472     $this-> write('_scripts', $js);
473 }
474
475 return $success;
476 }
477
478 // -----
479
480 /**
481  * Dynamically include CSS in the template
482  *
483  * NOTE: This function does NOT check for existence of .css file
484  *
485  * @access public
486  * @param string CSS file to link, import or embed
487  * @param string 'link', 'import' or 'embed'
488  * @param string media attribute to use with 'link' type only, FALSE for none
489  * @return TRUE on success, FALSE otherwise
490  */
491
492 function add_css($style, $type = 'link', $media = FALSE)
493 {
494     $success = TRUE;
495     $css = NULL;
496
497     $this-> CI-> load-> helper('url');
498     $filepath = base_url() . $style;
499
500     switch ($type)
501     {
502         case 'link':
503
504             $css = '<link type="text/css" rel="stylesheet" href="'
505 $filepath . '"';
506             if ($media)
507             {
508                 $css .= ' media="' . $media . '"';
509             }
510             $css .= ' />' ;
511             break;
512
513         case 'import':
514             $css = '<style type="text/css">@import url(' . $filepath
515 . ');</style>' ;
516             break;
517
518         case 'embed':
519             $css = '<style type="text/css">' ;
520             $css .= $style;
521             $css .= '</style>' ;
522             break;
523
524         default:
525             $success = FALSE;
526             break;
527     }
528
529 // Add to js array if it doesn't already exist
530 if ($css != NULL && ! in_array($css, $this-> css))
531 {
532     $this-> css[] = $css;
533     $this-> write('_styles', $css);
534 }
535

```



```

534     return $success;
535 }
536 // -----
537
538 /**
539  * Render the master template or a single region
540  *
541  * @access      public
542  * @param      string      optionally opt to render a specific region
543  * @param      boolean     FALSE to output the rendered template, TRUE to return as a string.
544  Always TRUE when $region is supplied
545  * @return     void or string (result of template build)
546  */
547
548 function render($region = NULL, $buffer = FALSE, $parse = FALSE)
549 {
550     // Just render $region if supplied
551     if ($region) // Display a specific regions contents
552     {
553         if (isset($this-> regions[$region]))
554         {
555             $output = $this-> _build_content($this-> regions[$region]);
556         }
557         else
558         {
559             show_error("      Cannot render the '{$region}' region. The region is
undefined."
);
560         }
561     }
562
563     // Build the output array
564     else
565     {
566         foreach ($this-> regions as $name => $region)
567         {
568             $this-> output[$name] = $this-> _build_content($region);
569         }
570
571         if ($this-> parse_template === TRUE or $parse === TRUE)
572         {
573             // Use provided parser class and method to render the template
574             $output = $this-> CI->{ $this-> parser }->{ $this-> parser_method }($this->
master, $this-> output, TRUE);
575
576             // Parsers never handle output, but we need to mimick it in this case
577             if ($buffer === FALSE)
578             {
579                 $this-> CI-> output-> set_output($output);
580             }
581         }
582         else
583         {
584             // Use CI's loader class to render the template with our output array
585             $output = $this-> CI-> load-> view($this-> master, $this-> output,
$buffer);
586         }
587     }
588
589     return $output;
590 }
591 // -----
592
593 /**
594  * Load the master template or a single region
595  *
596  * @DEPRECATED!
597  *
598  * Use render() to compile and display your template and regions
599  */
600
601 function load($region = NULL, $buffer = FALSE)
602 {
603     $region = NULL;
604     $this-> render($region, $buffer);
605 }
606 // -----
607
608
609

```

```

610  /**
611  * Build a region from it's contents. Apply wrapper if provided
612  *
613  * @access    private
614  * @param     string    region to build
615  * @param     string    HTML element to wrap regions in; like '<div>'
616  * @param     array     Multidimensional array of HTML elements to apply to $wrapper
617  * @return    string    Output of region contents
618  */
619
620  function _build_content($region, $wrapper = NULL, $attributes = NULL)
621  {
622      $output = NULL;
623
624      // Can't build an empty region. Exit stage left
625      if ( ! isset($region['content']) or ! count($region['content']) )
626      {
627          return FALSE;
628      }
629
630      // Possibly overwrite wrapper and attributes
631      if ($wrapper)
632      {
633          $region['wrapper'] = $wrapper;
634      }
635      if ($attributes)
636      {
637          $region['attributes'] = $attributes;
638      }
639
640      // Open the wrapper and add attributes
641      if (isset($region['wrapper']))
642      {
643          // This just trims off the closing angle bracket. Like '<p>' to '<p'
644          $output .= substr($region['wrapper'], 0, strlen($region['wrapper']) - 1);
645
646          // Add HTML attributes
647          if (isset($region['attributes']) && is_array($region['attributes']))
648          {
649              foreach ($region['attributes'] as $name => $value)
650              {
651                  // We don't validate HTML attributes. Imagine someone using a custom XML
652                  $output .= "        $name=\""        $value "\"        ";
653              }
654          }
655
656          $output .= ">"        ;
657      }
658
659      // Output the content items.
660      foreach ($region['content'] as $content)
661      {
662          $output .= $content;
663      }
664
665      // Close the wrapper tag
666      if (isset($region['wrapper']))
667      {
668          // This just turns the wrapper into a closing tag. Like '<p>' to '</p>'
669          $output .= str_replace('<' , '</' , $region['wrapper']) . "\n"        ;
670      }
671
672      return $output;
673  }
674
675  }
676  // END Template Class
677
678  /* End of file Template.php */
679  /* Location: ./system/application/libraries/Template.php */

```

Indice

A

ArbolSegmentacion_model	99
<i>Modelo ArbolSegmentacion</i>	
ArbolSegmentacion_model::convertType()	100
<i>Convierte el tipo de arreglo para enviarlo como se debe recibir en el cliente de Javascript</i>	
ArbolSegmentacion_model::getById()	100
<i>Obtener el elemento del ASU por medio de su ID</i>	
ArbolSegmentacion_model::getChildrenFromId()	100
<i>Accion para devolver los hijos de un elemento en el ASU a partir de su ID</i>	
Ageb_model::searchUM()	99
<i>Devuelve la información de una UM de acuerdo a su localidad y ageb</i>	
Ageb_model::searchageb()	98
<i>Devuelve una lista de AGEBS de la localidad pasada como parámetro</i>	
Accion_model::update()	97
<i>Actualiza el objeto actual en la base de datos</i>	
Ageb_model	97
<i>Modelo Ageb</i>	
Ageb_model::getMsgError()	98
<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()	98
<i>Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion</i>	
ArbolSegmentacion_model::getChildrenFromLevel()	101
<i>Accion para devolver el esquema completo del ASU a partir de un nivel especificado, niveles omitidos y elementos preseleccionados</i>	
ArbolSegmentacion_model::getCluesFromId()	101
*	
ArbolSegmentacion_model::getTreeBlockData()	104
<i>Regresa el objeto del arbol de segmentacion por nivel o hijos de un elemento seleccionado</i>	
ArbolSegmentacion_model::getUMPARENTSById()	105
<i>Regresa la información de los padres de una unidad medica en el ASU</i>	
ArbolSegmentacion_model:: addSelectedItems()	105
ArbolSegmentacion_model:: addSelectedItems ()	105
ArbolSegmentacion_model::getTreeBlock()	104
<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()	103
<i>Regresa el objeto del arbol de segmentacion</i>	
ArbolSegmentacion_model::getDataKeyValue()	102
<i>Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro</i>	
ArbolSegmentacion_model::getDescripcionById()	102

<i>Accion para obtener la descripcion e información adicional del elemento en el ASU</i>	
ArbolSegmentacion_model::getListChildrenLevel()	103
ArbolSegmentacion_model::getMsgError()	103
<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>	
Accion_model::setRows()	97
Accion_model::setOffset()	97
Ayuda::index()	50
<i>Función que renderiza el contenido de la ayuda dependiendo de la sección donde se encuentre</i>	
Accion_model	93
<i>Modelo Accion</i>	
Accion_model::delete()	93
<i>Elimina el registro actual de la base de datos</i>	
Accion_model::getAll()	93
<i>Devuelve todos los registros de la tabla acciones</i>	
Ayuda	50
<i>Controlador Ayuda</i>	
Accion::view()	50
<i>Acción para visualizar información de una acción específica, obtiene el objeto acción por medio del id proporcionado.</i>	
Accion::delete()	48
<i>Acción para eliminar una acción, recibe el id de la acción a eliminar</i>	
Accion::index()	49
<i>Acción por default del controlador, carga la lista</i>	
Accion::insert()	49
<i>Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente</i>	
Accion::update()	49
<i>Acción para preparar la actualización de una acción ya existente,</i>	
Accion_model::getById()	94
<i>Devuelve la información de una accion por su ID</i>	
Accion_model::getDescripcion()	94
Accion_model::setDescripcion()	95
Accion_model::setId()	96
Accion_model::setMetodo()	96
Accion_model::setNombre()	96
Accion_model::insert()	95
<i>Inserta en la tabla accion la información contenida en el objeto</i>	
Accion_model::getNumRows()	95
<i>Devuelve el numero de registros</i>	
Accion_model::getId()	94

Accion_model::getMetodo()	94
Accion_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>	
Accion_model::getNombre()	95
Accion	48
<i>Controlador Accion</i>	

B

Bitacora_model::resetFilter()	110
<i>Elimina todos los filtros registrados</i>	
Bitacora_model::setFecha_hora()	110
Bitacora_model::insert()	109
<i>Inserta a la bitacora la información porporcionada</i>	
Bitacora_model::getParametros()	109
Bitacora_model::getNumRows()	109
<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()	110
Bitacora_model::setId_accion()	111
Bitacora_model::setParametros()	112
Bitacora_model::update()	112
<i>Actualiza los datos del objeto actual</i>	
Bitacora_model::setId_usuario()	111
Bitacora_model::setId_controlador_accion()	111
Bitacora_model::setId_controlador()	111
Bitacora_model::getMsgError()	109
<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()	108
Bitacora_model	106
<i>Modelo Bitacora</i>	
Bitacora_model::addFilter()	106
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Bitacora::view()	52
<i>Muestra los datos del registro especificado por el id</i>	
Bitacora::validateExistUsuario()	52
<i>callback utilizado por las acciones create y update para validar la existencia de un usuario</i>	
Bitacora::index()	51
<i>Lista todos los registros de la bitacora, con su correspondiente paginación</i>	
Bitacora_model::delete()	107
<i>Elimina el registro actual de la base de datos</i>	
Bitacora_model::deleteByFilter()	107
<i>Elimina el conjunto de registros que cumplen con el o los criterios de filtrado</i>	
Bitacora_model::getId()	108
Bitacora_model::getId_controlador_accion()	108
Bitacora_model::getFecha_hora()	108
Bitacora_model::getById()	108
<i>Obtiene los datos del registro de la bitacora que tiene el ID especificado</i>	
Bitacora_model::getAll()	107
<i>Obtiene todos los registros de la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta</i>	
Bitacora	51
<i>Controlador Bitacora</i>	

C

Catalogo_model::setNombre()	123
---	-----

Catalogo_model::setLlave()	123
Catalogo_model::setId()	122
Catalogo_model::setCampos()	122
Catalogo_model::setOffset()	123
Catalogo_model::setRows()	123
constructor Catalogo_x_raiz_model::construct()	124
Catalogo_x_raiz_model	124
<i>Modelo Raiz_x_Catalogo</i>	
Catalogo_model::updateComentario()	124
<i>Cambia el comentario de la tabla indicada</i>	
Catalogo_model::insert()	122
<i>Inserta en la base datos el catálogo y obtiene los datos de la tabla temporal</i>	
Catalogo_model::getNumRows()	121
<i>Devuelve el numero de registros</i>	
Catalogo_model::getByName()	120
<i>Devuelve la informacion de un catalogo por su nombre</i>	
Catalogo_model::getAllData()	119
<i>Devuelve los datos de un catalogo pasado como parametro</i>	
Catalogo_model::getAll()	119
<i>Devuelve una lista con los catalogos existentes en la DB</i>	
Catalogo_model::delete()	119
<i>Elimina el registro actual de la base de datos</i>	
Catalogo_model::getCampos()	120
Catalogo_model::getId()	120

Catalogo_model::getNombre()	121
Catalogo_model::getMsgError()	121
<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::getLlave()	121
Catalogo_x_raiz_model::check()	125
<i>Revisa inconsistencias en los datos de un catalogo x raiz con respecto a su catalogo padre</i>	
Catalogo_x_raiz_model::delete()	125
<i>Elimina el registro actual de la base de datos</i>	
Catalogo_x_raiz_model::getTablaCatalogo()	129
Catalogo_x_raiz_model::getRelations()	128
<i>Devuelve las relaciones de una raiz x catalogo</i>	
Catalogo_x_raiz_model::getRelacionPadre()	128
Catalogo_x_raiz_model::getRelacionHijo()	128
Catalogo_x_raiz_model::insert()	129
<i>Inserta en la base datos la información del objeto actual</i>	
Catalogo_x_raiz_model::setColumnaDescripcion()	129
Catalogo_x_raiz_model::setId()	130
Catalogo_x_raiz_model::setGrado()	130
Catalogo_x_raiz_model::setColumnaLlave()	129
Catalogo_x_raiz_model::getNivel()	128
<i>Devuelve el nivel siguiente para la tabla raiz_x_catalogo de un arbol determinado</i>	
Catalogo_x_raiz_model::getMsgError()	127
<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::getByNivel()	126
<i>Devuelve el catalogo padre de un elemento raiz_x_catalogo</i>	
Catalogo x raiz_model::getById()	126
<i>Devuelve la información de un catalogo x accion por su ID</i>	
Catalogo x raiz_model::getByArbol()	125
<i>Devuelve todos los registros de la tabla raiz_x_catalogo de una raiz determinada</i>	
Catalogo x raiz_model::getColumnaDescripcion()	126
Catalogo x raiz_model::getColumnaLLave()	126
Catalogo x raiz_model::getIdRaiz()	127
Catalogo x raiz_model::getId()	127

Catalogo x raiz_model::getGrado()	127
Catalogo_model::checkTypeData()	119
<i>Revisa en la base de datos por registros que no coincidan con el tipo de dato pasado como parametro en el campo indicado</i>	
Catalogo_model::checkPk()	118
<i>Revisa en la base de datos por registros duplicados en los campos pasados por parametro</i>	
constructor ReglaVacuna::construct()	85
constructor Raiz::construct()	80
constructor Permiso::construct()	79
constructor Menu::construct()	77
constructor Usuario::construct()	88
constructor Accion_model::construct()	93
constructor Bitacora_model::construct()	106
constructor ArbolSegmentacion_model::construct()	100
constructor Ageb_model::construct()	98
constructor Grupo::construct()	74
constructor Errorlog::construct()	73
Controlador::help()	68
<i>Establece el texto de ayuda para el controlador accion</i>	
Controlador::getGroupPermissions()	68
<i>Acción para servir un array de objetos con los permisos asignados a</i>	
Controlador::delete()	67
<i>Acción para eliminar un controlador, recibe el id del controlador a eliminar</i>	
Controlador::index()	68
<i>Acción por default del controlador, carga la lista de controladores disponibles y una lista de opciones</i>	
<i>Recibe un parametro en caso de filtrado por entornos</i>	
Controlador::insert()	69
<i>Acción para preparar la insercion de nuevos controladores , realiza la validacion del formulario del lado cliente</i>	
constructor Entorno::construct()	70
Controlador::view()	70
<i>Acción para visualizar información de un controlador específico, obtiene el objeto controlador por medio del id proporcionado.</i>	
Controlador::update()	69
<i>Acción para preparar la actualización de un controlador ya existente,</i>	
CatalogoCsv_model	112
<i>Modelo CatalogoCsv</i>	
constructor CatalogoCsv_model::construct()	113
CatalogoCsv_model::setLlave()	117

CatalogoCsv_model::setId()	116
CatalogoCsv_model::setCampos()	116
CatalogoCsv_model::getNumRows()	116
<i>Devuelve el numero de registros</i>	
CatalogoCsv_model::setNombre()	117
CatalogoCsv_model::setOffset()	117
constructor Catalogo_model::construct()	118
Catalogo_model	118
<i>Modelo Catalogo</i>	
CatalogoCsv_model::setRows()	117
CatalogoCsv_model::getNombre()	116
CatalogoCsv_model::getMsgError()	115
<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::getAll()	114
<i>Devuelve una lista con los catalogos existentes en la DB</i>	
CatalogoCsv_model::checkPk()	113
<i>Revisa en la base de datos por registros duplicados en los campos pasados por parametro</i>	
CatalogoCsv_model::activaEnCatalogo()	113
<i>Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas</i>	
CatalogoCsv_model::getAllData()	114
<i>Devuelve los datos de un catalogo pasado como parametro</i>	
CatalogoCsv_model::getByName()	114
<i>Devuelve la informacion de un catalogo por su nombre</i>	
CatalogoCsv_model::getLLave()	115
CatalogoCsv_model::getId()	115

CatalogoCsv_model::getCampos()	115
Catalogo x raiz_model::setIdRaiz()	130
Catalogo x raiz_model::setRelacionHijo()	130
constructor Errorlog_model::construct()	153
constructor Entorno_model::construct()	147
Controlador_model::update()	146
<i>Actualiza el objeto actual en la base de datos</i>	
Controlador_model::setRows()	146
constructor Georeferencia_model::construct()	158
constructor Grupo_model::construct()	159
constructor Permiso_model::construct()	172
constructor Menu_model::construct()	165
constructor Hemoglobina_model::construct()	164
Controlador_model::setOffset()	146
Controlador_model::setNombre()	145
Controlador_model::insert()	144
<i>Inserta en la tabla controlador, la información contenida en el objeto</i>	
Controlador_model::getPermisos()	143
<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 los permisos de un grupo sobre esas acciones)</i>	
Controlador_model::getNumRows()	143
<i>Devuelve el numero de registros</i>	
Controlador_model::getNombre()	143

Controlador_model::setAccion()	144
Controlador_model::setClase()	144
Controlador_model::setIdEntorno()	145
Controlador_model::setId()	145
Controlador_model::setDescripcion()	145
constructor Poblacion_model::construct()	176
constructor Raiz_model::construct()	177
constructor Reporteador_model::construct()	281
constructor Notificacion_model::construct()	275
constructor Estado_tableta_model::construct()	273
constructor Enrolamiento_model::construct()	229
constructor Reporte_censo_nominal::construct()	286
constructor Semana_nacional_model::construct()	288
constructor Usuario_tableta_model::construct()	302
constructor Tipo_censo_model::construct()	299
constructor Tableta_model::construct()	292
constructor Usuario_tableta::construct()	228
constructor Tableta::construct()	225
constructor Enrolamiento::construct()	201
constructor Usuario_model::construct()	190
constructor ReglaVacuna_model::construct()	181
constructor Notificacion::construct()	212
constructor Reporteador::construct()	214
constructor Servicios::construct()	219
constructor Semana_nacional::construct()	217
constructor Reporte_sincronizacion::construct()	215
Controlador_model::getMsgError()	142
<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::getIdEntorno()	142
Cie10_model::getId()	135

Cie10_model::getDescripcion()	135
Cie10_model::getData()	134
<i>Devuelve una lista con los registros existentes en el catalogo cie10 omitiendo los ID</i>	
Cie10_model::getCatalogoByName()	134
<i>Devuelve una lista con los registros existentes en el catalogo requerido</i>	
Cie10_model::getMsgError()	135
<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::getNumRows()	135
<i>Devuelve el numero de registros</i>	
Cie10_model::setOffset()	136
Cie10_model::setId()	136
Cie10_model::setDescripcion()	136
Cie10_model::getById()	134
<i>Devuelve la información de un registro del catalogo cie10 por su ID</i>	
Cie10_model::getAllData()	133
<i>Devuelve los datos de un catalogo pasado como parametro</i>	
Cie10_model	131
<i>Modelo Cie10</i>	

Catalogo x raiz model::setTablaCatalogo()	131
Catalogo x raiz model::setRelacionPadre()	131
constructor Cie10 model:: construct()	132
Cie10 model::activaEnCatalogo()	132
<i>Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas</i>	
Cie10 model::getAll()	133
<i>Devuelve una lista con los registros existentes en el catalogo cie10</i>	
Cie10 model::checkPk()	133
<i>Revisa en la base de datos por registros duplicados en los campos pasados por parametro</i>	
Cie10 model::agregaEnCatalogo()	132
<i>Accion para agregar registros del CIE10 a otros catalogos como el de EDA, IRA y Consultas</i>	
Cie10 model::setRows()	136
Cie10 model::update()	137
<i>Actualiza el objeto actual en la base de datos</i>	
Controlador model::getAll()	141
<i>Devuelve todos los registros de la tabla controlador</i>	
Controlador model::getAcciones()	140
<i>Devuelve todas las acciones asignadas al controlador por su Id</i>	
Controlador model::getAccion()	140
Controlador model::delete()	140
<i>Elimina el registro actual de la base de datos</i>	
Controlador model::getByEntorno()	141
<i>Devuelve todos los controladores que pertenecen a un entorno por su Id</i>	
Controlador model::getById()	141
<i>Devuelve la información de un controlador por su ID</i>	
Controlador model::getId()	142

Controlador model::getDescripcion()	142
Controlador model::getClase()	142
Controlador model::accionesUpdate()	140
<i>Actualiza las acciones asignadas a un controlador</i>	
constructor Controlador model:: construct()	140
ControladorAccion model::getById()	137
<i>Devuelve la información de una accion por controlador de acuerdo a su Id</i>	
constructor ControladorAccion model:: construct()	137
ControladorAccion model	137
<i>Modelo ControladorAccion</i>	
ControladorAccion model::getId()	138
<i>Devuelve el Id de una accion por controlador de una accion y controlador determinados</i>	
ControladorAccion model::getIdByPath()	138
<i>Devuelve el id de una accion por controlador de acuerdo al path</i>	
Controlador model	139
<i>Modelo Controlador</i>	
ControladorAccion model::setHelp()	139
<i>Establece el mensaje de ayuda</i>	
ControladorAccion model::getMsgError()	139
<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::accion()	67

*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*

constructor Controlador:: construct()	67
CI Pagination::\$cur_tag_close	24
CI Pagination::\$cur_page	24
CI Pagination::\$base_url	24
CI Pagination::\$anchor_class	24
CI Pagination::\$cur_tag_open	24
CI Pagination::\$display_pages	24
CI Pagination::\$first_tag_open	24
CI Pagination::\$first_tag_close	24
CI Pagination::\$first_link	24
CI Pagination	23
<i>Pagination Class</i>	
CI Form validation:: translate_fieldname()	23
<i>Translate a field name</i>	
CI Form validation::valid_emails()	21
<i>Valid Emails</i>	
CI Form validation::valid_email()	21
<i>Valid Email</i>	
CI Form validation::valid_base64()	20
<i>Valid Base64</i>	
CI Form validation::strip_image_tags()	20
<i>Strip Image Tags</i>	
CI Form validation::valid_ip()	21
<i>Validate IP Address</i>	
CI Form validation::xss_clean()	22
<i>XSS Clean</i>	
CI Form validation:: reset_post_array()	23
<i>Re-populate the _POST array with our finalized and processed data</i>	
CI Form validation:: reduce_array()	22
<i>Traverse a multidimensional \$_POST array index until the data is found</i>	
CI Form validation:: execute()	22
<i>Executes the Validation routines</i>	
CI Pagination::\$first_url	24
CI Pagination::\$full_tag_close	25
CI Pagination::\$prev_link	26
CI Pagination::\$prefix	25
CI Pagination::\$per_page	25
CI Pagination::\$page_query_string	25
CI Pagination::\$prev_tag_close	26
CI Pagination::\$prev_tag_open	26
CI Pagination::\$total_rows	26
CI Pagination::\$suffix	26
CI Pagination::\$query_string_segment	26
CI Pagination::\$num_tag_open	25
CI Pagination::\$num_tag_close	25
CI Pagination::\$last_tag_close	25
CI Pagination::\$last_link	25
CI Pagination::\$full_tag_open	25
CI Pagination::\$last_tag_open	25
CI Pagination::\$next_link	25

CI_Pagination::\$num_links	25
CI_Pagination::\$next_tag_open	25
CI_Pagination::\$next_tag_close	25
CI_Form_validation::set_value()	19
<i>Get the value from a form</i>	
CI_Form_validation::set_select()	19
<i>Set Select</i>	
CI_Form_validation::alpha_numeric()	9
<i>Alpha-numeric</i>	
CI_Form_validation::alpha_dash()	9
<i>Alpha-numeric with underscores and dashes</i>	
CI_Form_validation::alpha()	8
<i>Alpha</i>	
constructor CI_Form_validation::construct()	8
<i>Constructor</i>	
CI_Form_validation::decimal()	9
<i>Decimal number</i>	
CI_Form_validation::encode_php_tags()	10
<i>Convert PHP tags to entities</i>	
CI_Form_validation::exact_length()	11
<i>Exact Length</i>	
CI_Form_validation::error_string()	10
<i>Error String</i>	
CI_Form_validation::error()	10
<i>Get Error Message</i>	
CI_Form_validation::\$safe_form_data	8
CI_Form_validation::\$field_data	8
CI_Form_validation::\$error_string	6
CI_Form_validation::\$CI	6
CI_Form_validation	6
<i>Form Validation Class</i>	
CI_Form_validation::\$config_rules	7
CI_Form_validation::\$error_array	7
CI_Form_validation::\$error_suffix	7
CI_Form_validation::\$error_prefix	7
CI_Form_validation::\$error_messages	7
CI_Form_validation::greater_than()	11
<i>Greather than</i>	
CI_Form_validation::integer()	12
<i>Integer</i>	
CI_Form_validation::run()	17
<i>Run the Validator</i>	
CI_Form_validation::required()	16
<i>Required</i>	
CI_Form_validation::regex_match()	16
<i>Performs a Regular Expression match test.</i>	
CI_Form_validation::prep_url()	15
<i>Prep URL</i>	
CI_Form_validation::set_checkbox()	17
<i>Set Checkbox</i>	
CI_Form_validation::set_error_delimiters()	17
<i>Set The Error Delimiter</i>	
CI_Form_validation::set_rules()	19

Set Rules	
CI_Form_validation::set_radio()	18
Set Radio	
CI_Form_validation::set_message()	18
Set Error Message	
CI_Form_validation::prep_for_form()	15
Prep data for form	
CI_Form_validation::numeric()	15
Numeric	
CI_Form_validation::is_numeric()	13
Is Numeric	
CI_Form_validation::is_natural_no_zero()	12
Is a Natural number, but not a zero (1,2,3, etc.)	
CI_Form_validation::is_natural()	12
Is a Natural number (0,1,2,3, etc.)	
CI_Form_validation::is_unique()	13
Match one field to another	
CI_Form_validation::less_than()	13
Less than	
CI_Form_validation::min_length()	14
Minimum Length	
CI_Form_validation::max_length()	14
Max Length	
CI_Form_validation::matches()	14
Match one field to another	
CI_Pagination::\$uri_segment	26
CI_Pagination::\$use_page_numbers	26
CatalogoCsv::checkpk()	58
Acción para revisar registros repetidos en las columnas designadas como primary key	
CatalogoCsv::ActivaEnCatalogo()	57
*	
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()	57
CatalogoCsv	57
Controlador CatalogoCsv	
CatalogoCsv::createTableAgeb()	58
Acción para ejecutar la creación de la tabla Asu Ageb	
No recibe parámetros	
CatalogoCsv::createTableGeo()	58
Acción para ejecutar la creación de la tabla poblacional	
No recibe parámetros	
CatalogoCsv::index()	59
Acción por default del controlador, carga la lista	
de catálogoscsv disponibles y una lista de opciones	
No recibe parámetros	
CatalogoCsv::createTablePob()	59
Acción para ejecutar la creación de la tabla poblacional	
No recibe parámetros	
CatalogoCsv::createTableHemoGlobina()	58
Acción para ejecutar la creación de la tabla Asu Ageb	
No recibe parámetros	
Catalogo::array_unique_recursive()	56

_array_unique_recursive	
<i>Revisa valores duplicados en arreglos multidimensionales</i>	
Catalogo::view()	56
<i>Acción para visualizar información de un catálogo específico, obtiene el objeto catálogo por medio del nombre proporcionado</i>	
Catalogo::delete()	54
<i>Acción para eliminar un catálogo, recibe el nombre del catalogo a eliminar</i>	
Catalogo::checkTypeData()	53
<i>Acción para revisar si los tipos de datos coinciden con los datos contenidos en la tabla temporal que fueron tomados del CSV</i>	
Catalogo::checkpk()	53
<i>Acción para revisar registros repetidos en las columnas designadas como primary key</i>	
Catalogo::index()	54
<i>Acción por default del controlador, carga la lista de catálogos disponibles y una lista de opciones</i>	
<i>No recibe parámetros</i>	
Catalogo::insert()	54
<i>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</i>	
Catalogo::update()	55
<i>Acción para preparar la actualizacion de un catálogo ya existente,</i>	
Catalogo::loadupdate()	55
<i>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.</i>	
Catalogo::load()	55
<i>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</i>	
CatalogoCsv::loadupdate()	59
<i>Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP</i>	
CatalogoCsv::update()	59
<i>Acción para preparar la actualizacion de un catálogo ya existente,</i>	
Cie10::insert()	64
<i>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</i>	
Cie10::index()	64
<i>Acción por default del controlador, carga la lista de datos disponibles en el cie10 y una lista de opciones</i>	
Cie10::AgregaEnCatalogo()	64
<i>*</i>	
<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::ActivaEnCatalogo()	63
<i>*</i>	
<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>	
Cie10::load()	65

<i>Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP</i>	
Cie10::update()	65
<i>Acción para preparar la actualización de un registro del CIE10,</i>	
Controlador	67
<i>Controlador Controlador</i>	
Cie10::array_unique_recursive()	66
<i>_array_unique_recursive</i>	
<i>Revisa valores duplicados en arreglos que contienen arreglos</i>	
Cie10::view()	66
*	
<i>Accion para mostrar información de los catalogos IDE , ERA y Consultas</i>	
constructor Cie10::construct()	63
Cie10	63
<i>Controlador Cie10</i>	
Catalogo x raiz	61
<i>Controlador Raiz_x_Catalogo</i>	
CatalogoCsv::array_unique_recursive()	60
<i>_array_unique_recursive</i>	
<i>Revisa valores duplicados en arreglos que contienen arreglos</i>	
CatalogoCsv::view()	60
<i>Acción para visualizar información de un catálogo específico, obtiene el objeto catalogocsv por medio del nombre proporcionado</i>	
constructor Catalogo x raiz::construct()	61
Catalogo x raiz::check()	61
<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>	
Catalogo x raiz::view()	62
<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()	62
<i>Acción para preparar la inserción de nuevas raices para catálogos , realiza la validación del formulario del lado cliente</i>	
Catalogo x raiz::delete()	62
<i>Acción para eliminar un catálogo en el arbol, recibe el id del catálogo en la raiz a eliminar</i>	
constructor Catalogo::construct()	53
Catalogo	53
<i>Controlador Catalogo</i>	
CI_Template::\$regions	28
CI_Template::\$parse_template	28
CI_Template::\$parser_method	28
CI_Template::\$parser	28
CI_Template::\$template	28
constructor CI_Template::CI_Template()	28
<i>Constructor</i>	
CI_Template::add_region()	29
<i>Dynamically add region to the currently set template</i>	
CI_Template::add_js()	29
<i>Dynamically include javascript in the template</i>	
CI_Template::add_css()	29
<i>Dynamically include CSS in the template</i>	
CI_Template::\$output	28
CI_Template::\$master	28

CI_Pagination::initialize()	27
<i>Initialize Preferences</i>	
CI_Pagination::create_links()	26
<i>Generate the pagination links</i>	
constructor CI_Pagination::construct()	26
<i>Constructor</i>	
CI_Template	27
<i>CodeIgniter Template Class</i>	
CI_Template::\$CI	28
CI_Template::\$js	28
CI_Template::\$css	28
CI_Template::\$config	28
CI_Template::add_template()	30
<i>Dynamically add a template and optionally switch to it</i>	
CI_Template::empty_region()	30
<i>Empty a region's content</i>	
constructor Obtenercurp::construct()	39
constructor Graph::construct()	37
constructor Menubuilder::construct()	35
CI_Template::write_view()	34
<i>Write content from a View to a region. 'Views within views'</i>	
constructor Tree::construct()	41
constructor Session::construct()	44
<i>Constructor</i>	
constructor Bitacora::construct()	51
constructor Ayuda::construct()	50
constructor Accion::construct()	48
CI_Template::write()	34
<i>Write contents to a region</i>	
CI_Template::set_template()	33
<i>Use given template settings</i>	
CI_Template::parse_view()	31
<i>Parse content from a View to a region with the Parser Class</i>	
CI_Template::load()	31
<i>Load the master template or a single region</i>	
CI_Template::initialize()	30
<i>Initialize class settings using config settings</i>	
CI_Template::render()	32
<i>Render the master template or a single region</i>	
CI_Template::set_master_template()	32
<i>Set master template</i>	
CI_Template::set_regions()	33
<i>Set regions for writing to</i>	
CI_Template::set_parser_method()	33
<i>Set parser method</i>	
CI_Template::set_parser()	32
<i>Set parser</i>	
constructor Index::construct()	1

E

Enrolamiento_model::get_transaction_relevante()	251
---	-----

<i>Obtiene las transacciones relevante para la sincronizacion</i>	
Enrolamiento_model::get_sales()	251
<i>Obtiene los registros de sales de rehidratación oral asociados a una persona</i>	
Enrolamiento_model::get_persona_x_tutor()	251
<i>obtiene los tutores de las personas que se envian en la sincronizacion</i>	
Enrolamiento_model::get_version()	252
<i>obtiene cual es la ultima version de apk de la tableta</i>	
Enrolamiento_model::insert()	252
<i>Guarda la persona capturada mediante el formulario web</i>	
Enrolamiento_model::setafiliacion()	252
Enrolamiento_model::setaccion_nutricional()	252
Enrolamiento_model::get_peri_cefa()	250
<i>Obtiene los registros de perimetro cefalico asociados a una persona</i>	
Enrolamiento_model::get_pacientes()	250
<i>devuelve todos los pacientes de la base de datos</i>	
Enrolamiento_model::get_cns_persona()	248
<i>Este metodo obtiene las personas que seran enviadas en la sincronizacion</i>	
Enrolamiento_model::get_cns_cat_persona_count()	248
<i>Hace el count de personas que se envian en la sincronizacion</i>	
Enrolamiento_model::get_control_nutricional()	249
<i>Obtiene los datos del control nutricional asociados a una persona</i>	
Enrolamiento_model::get_datos_grafica()	249
<i>Obtiene los datos especificos de un catálogo para ser visualizados en una gráfica</i>	
Enrolamiento_model::get_notificacion()	250
<i>Este metodo obtiene las notificaciones que se enviaron en la sincronizacion</i>	
Enrolamiento_model::get_estimulacion()	249
<i>Obtiene los registros de estimulacion temprana asociados a una persona</i>	
Enrolamiento_model::setageb()	253
Enrolamiento_model::setalergias()	253
Enrolamiento_model::setcp()	256
Enrolamiento_model::setconsulta()	255
Enrolamiento_model::setcurp()	256
Enrolamiento_model::setcurpT()	256
Enrolamiento_model::setestimulacion_fecha()	257
Enrolamiento_model::setestimulacion_capacitado()	257
Enrolamiento_model::setcompaniaT()	255
Enrolamiento_model::setcompania()	255
Enrolamiento_model::setcalle()	253
Enrolamiento_model::setaltura()	253
Enrolamiento_model::setcelular()	254
Enrolamiento_model::setcelularT()	254
Enrolamiento_model::setcolonia()	255
Enrolamiento_model::setcodigo_barras()	254
Enrolamiento_model::get_cns_cat_persona()	247
<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()	247
<i>Hace select de los catalogos que tengan relacion con una persona para mostrarlos en el view</i>	
Enrolamiento_model::getreferencia()	242
Enrolamiento_model::getprecurp()	242
Enrolamiento_model::getpeso()	242
Enrolamiento_model::getRegistro_civil()	242

<i>obtiene informacion del registro civil</i>	
Enrolamiento_model::getsales_cantidad()	243
Enrolamiento_model::getsangre()	243
Enrolamiento_model::getsales_fecha()	243
Enrolamiento_model::getperi_cefa()	242
Enrolamiento_model::getpaternoT()	242
Enrolamiento_model::getnombreT()	241
Enrolamiento_model::getnombre()	241
Enrolamiento_model::getnumero()	241
Enrolamiento_model::getNumRows()	241
<i>Devuelve el numero de filas en la tabla cns_persona</i>	
Enrolamiento_model::getpaterno()	242
Enrolamiento_model::getparto()	241
Enrolamiento_model::getsector()	243
Enrolamiento_model::getsexo()	243
Enrolamiento_model::get_catalog()	245
<i>Hace select de las tablas cns_x que representa a los catalogos</i>	
Enrolamiento_model::getvacuna()	245
Enrolamiento_model::get_catalog2()	245
<i>Obtiene los datos de una tabla</i>	
Enrolamiento_model::get_catalog_count()	246
<i>Obtiene el numero de resultados de una tabla</i>	
Enrolamiento_model::get_catalog_tratamiento()	247
<i>Hace select de los tratamientos de las consultas</i>	
Enrolamiento_model::get_catalog_relevante()	246
<i>obtiene los catalogos relevante x entorno para la sincronizacion</i>	
Enrolamiento_model::getumt()	245
Enrolamiento_model::gettelefonoT()	244
Enrolamiento_model::gettalla()	244
Enrolamiento_model::getsexoT()	244
Enrolamiento_model::gettamiz()	244
Enrolamiento_model::gettbeneficiario()	244
Enrolamiento_model::gettelefono()	244
Enrolamiento_model::gettconsulta()	244
Enrolamiento_model::setfaccion_nutricional()	257
Enrolamiento_model::setfconsulta()	257
Enrolamiento_model::update_beneficiario()	269
<i>Actualiza el tipo de beneficiario del paciente</i>	
Enrolamiento_model::update_basico()	269
<i>Actualiza los datos basicos del paciente</i>	
Enrolamiento_model::update_alergia()	269
<i>Actualiza los datos de las alergias del paciente</i>	
Enrolamiento_model::update_consulta()	270
<i>Actualiza el control consulta del paciente</i>	
Enrolamiento_model::update_direccion()	270
<i>actualiza la direccion del paciente</i>	
Enrolamiento_model::update_nutricion()	270
<i>Actualiza el control nutricional del paciente</i>	
Enrolamiento_model::update_estimulacion()	270
<i>Actualiza los registros de estimulacion temprana</i>	
Enrolamiento_model::update_accion()	269
<i>Actualiza el control accion nutricional del paciente</i>	
Enrolamiento_model::tes_pendientes_tarjeta_delete()	269

<i>Elimina los pendientes de las personas que no tengan asignado una tarjeta</i>	
Enrolamiento_model::settconsulta()	267
Enrolamiento_model::settbeneficiario()	267
Enrolamiento_model::settelefono()	267
Enrolamiento_model::settelefonoT()	268
Enrolamiento_model::setvacuna()	268
Enrolamiento_model::setumt()	268
Enrolamiento_model::update_peri_cefa()	271
<i>Actualiza los registros de perimetro cefalico</i>	
Enrolamiento_model::update_regcivil()	271
<i>actualiza el registro civil del paciente</i>	
Estado_tableta_model::getDescripcion()	274
Estado_tableta_model::getById()	273
<i>Obtiene los datos del registro que tiene el ID especificado</i>	
Estado_tableta_model::getId()	274
Estado_tableta_model::getMsgError()	274
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Estado_tableta_model::setId()	275
Estado_tableta_model::setDescription()	274
Estado_tableta_model::getAll()	273
<i>Obtiene todos los registros de la tabla</i>	
Estado_tableta_model	273
<i>Modelo Estado_tableta</i>	
Enrolamiento_model::update_status_tableta()	271
<i>Hace update de la tableta que este sincronizando dependiendo del resultado</i>	
Enrolamiento_model::update_sales()	271
<i>Actualiza los registros de sales de rehidratacion oral</i>	
Enrolamiento_model::update_tutor()	272
<i>Actualiza los datos del tutor del paciente</i>	
Enrolamiento_model::update_umt()	272
<i>Actualiza la unidad medica tratante del paciente</i>	
Enrolamiento_model::valid_card()	272
<i>Este metodo valida que exista un archivo para enviar a la tarjeta por nfc</i>	
Enrolamiento_model::update_vacuna()	272
<i>Actualiza las vacunas del paciente</i>	
Enrolamiento_model::settamiz()	267
Enrolamiento_model::settalla()	266
Enrolamiento_model::setlugarcivil()	260
Enrolamiento_model::setlocalidad()	260
Enrolamiento_model::setlnacimiento()	260
Enrolamiento_model::setmanzana()	261
Enrolamiento_model::setmaterno()	261
Enrolamiento_model::setnacionalidad()	262
Enrolamiento_model::setmaternoT()	261
Enrolamiento_model::setidtutor()	260
Enrolamiento_model::setId()	259
Enrolamiento_model::setfecha_peri_cefa()	258
Enrolamiento_model::setfechacivil()	258
Enrolamiento_model::setfnacimiento()	258
Enrolamiento_model::setfnutricion()	258
Enrolamiento_model::sethemoglobina()	259

Enrolamiento_model::setfvacuna()	259
Enrolamiento_model::setnombre()	262
Enrolamiento_model::setnombreT()	262
Enrolamiento_model::setsales_fecha()	265
Enrolamiento_model::setsales_cantidad()	265
Enrolamiento_model::setsangre()	265
Enrolamiento_model::setsector()	266
Enrolamiento_model::setsexoT()	266
Enrolamiento_model::setsexo()	266
Enrolamiento_model::setreferencia()	264
Enrolamiento_model::setprecurp()	264
Enrolamiento_model::setparto()	263
Enrolamiento_model::setnumero()	262
Enrolamiento_model::setpaterno()	263
Enrolamiento_model::setpaternoT()	263
Enrolamiento_model::setpeso()	264
Enrolamiento_model::setperi_cefa()	264
Enrolamiento_model::getnacionalidad()	240
Enrolamiento_model::getMsgError()	240
Errorlog_model::resetFilter()	155
<i>Elimina todos los filtros registrados</i>	
Errorlog_model::insert()	155
<i>Inserta en la base de datos la informacion del error</i>	
Errorlog_model::getNumRows()	155
<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()	156
<i>Guardar el mensaje de error descriptivo en la base de datos,</i>	
Errorlog_model::setDescription()	156
Errorlog_model::setId_accion()	157
Errorlog_model::setId()	156
Errorlog_model::getId_usuario()	155
Errorlog_model::getId_controlador_accion()	154
Errorlog_model::getAll()	153
<i>Obtiene todos los registros de la tabla Error en caso de existir filtros, estos son aplicados a la consulta</i>	
Errorlog_model::addFilter()	153
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Errorlog_model::getById()	154
<i>Obtiene los datos del registro del Error que tiene el ID especificado</i>	
Errorlog_model::getDescription()	154
Errorlog_model::getId()	154
Errorlog_model::getFecha_hora()	154
Errorlog_model::setId_controlador()	157
Errorlog_model::setId_controlador_accion()	157
Enrolamiento::chechar_session()	203
<i>Revisa si la sesión está activa</i>	
Enrolamiento::categoriacie10_select()	203
<i>Genera los options de un campo tipo select para las categorías de CIE10</i>	
Enrolamiento::cie10_select()	204
<i>Genera los options de un campo tipo select para los CIE10 correspondientes a una categoría</i>	
Enrolamiento::comparar_view()	204

Crea la pagina para ver la informacion de la persona y comprararla con la persona capturada

Enrolamiento::file_to_card()	205
<i>crea un archivo descargable el cual se necesita para el envio por nfc a la tarjeta del paciente</i>	
Enrolamiento::data_tutor()	204
<i>Obtiene inofrmacion del tutor</i>	
Enrolamiento::catalog_select()	203
<i>Genera los options de un campo tipo select</i>	
Enrolamiento::catalog_check()	202
<i>Crea un grupo de radio o check con la informacion de los catalogos</i>	
Enrolamiento	201
<i>Controlador de enrolamiento</i>	
Errorlog_model::setId_usuario()	157
Enrolamiento::addForm()	201
<i>Pase de parametros para la insercion o actualizacion</i>	
Enrolamiento::autocomplete()	201
<i>Crea el autocomplete para facilitar la busqueda de un tutor</i>	
Enrolamiento::brother_found()	202
<i>Este metodo verifica si un paciente comparte un mismo tutor</i>	
Enrolamiento::brothers_search()	202
<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	152
<i>Modelo Errorlog</i>	
Entorno_model::update()	152
<i>Actualiza el objeto actual en la base de datos</i>	
Entorno_model	147
<i>Modelo Entorno</i>	
Errorlog::view()	74
<i>Muestra los datos del registro especificado por el id</i>	
Errorlog::index()	73
<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()	147
<i>Elimina el registro actual de la base de datos</i>	
Entorno_model::getAll()	147
<i>Devuelve todos los registros de la tabla entorno</i>	
Entorno_model::getByName()	148
<i>Devuelve la información de un entorno por su nombre</i>	
Entorno_model::getById()	147
<i>Devuelve la información de un entorno por su ID</i>	
Errorlog	73
<i>Controlador Errorlog</i>	
Entorno::ExistEntornoUpdate()	72
<i>Acción para validar que no exista previamente el entorno a actualizar</i>	
Entorno::index()	71
<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()	70
<i>Acción para eliminar un entorno, recibe el id del entorno a eliminar</i>	

Entorno::insert()	71
<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()	71
<i>Acción para preparar la actualización de un entorno ya existente,</i>	
Entorno:: ExistEntorno()	72
<i>Acción para validar que no exista previamente el entorno a insertar (Esta acción no puede ser accedida desde el navegador)</i>	
Entorno::view()	72
<i>Acción para visualizar de un entorno específico, obtiene el objeto entorno por medio del id proporcionado.</i>	
Entorno_model::getDescripcion()	148
Entorno_model::getDirectorio()	148
Entorno_model::setDirectorio()	151
Entorno_model::setDescripcion()	150
Entorno_model::setHostname()	151
Entorno_model::setId()	151
Entorno_model::setNombre()	152
Entorno_model::setIp()	151
Entorno_model::insert()	150
<i>Inserta en la tabla entorno la información contenida en el objeto</i>	
Entorno_model::getPermissionsByGroup()	150
<i>Obtiene los permisos asignados al grupo</i>	
Entorno_model::getId()	149

Entorno_model::getHostname()	148
Entorno_model::getInfo()	149
<i>Devuelve la información del objeto en forma de string</i>	
Entorno_model::getIp()	149
Entorno_model::getNombre()	150
Entorno_model::getMsgError()	149
<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::ifCarpExists()	205
<i>Valida que la carp del paciente no exista</i>	
Enrolamiento::ifCarpTExists()	206
<i>valida que la carp del tutor no exista</i>	
Enrolamiento_model::getcarpT()	236
Enrolamiento_model::getcarp()	236
Enrolamiento_model::getcp()	236
Enrolamiento_model::getestimulacion_capacitado()	237
Enrolamiento_model::getestimulacion_fecha()	237
Enrolamiento_model::getfconsulta()	237
Enrolamiento_model::getfaccion_nutricional()	237
Enrolamiento_model::getControlConsultas()	236
<i>Obtiene todas las consultas asociadas a un paciente</i>	
Enrolamiento_model::getconsulta()	236
Enrolamiento_model::getCIE10()	235
<i>Obtiene el listado de CIE10 correspondientes a una CIE10</i>	
Enrolamiento_model::getcelularT()	235
Enrolamiento_model::getcodigo_barras()	235
Enrolamiento_model::getcolonia()	235

Enrolamiento_model::getcompaniaT()	236
Enrolamiento_model::getcompania()	235
Enrolamiento_model::getfechacivil()	237
Enrolamiento_model::getfecha_peri_cefa()	237
Enrolamiento_model::getlocalidad()	239
Enrolamiento_model::getlnacimiento()	239
Enrolamiento_model::getlugarcivil()	240
Enrolamiento_model::getmanzana()	240
Enrolamiento_model::getmaternoT()	240
Enrolamiento_model::getmaterno()	240
Enrolamiento_model::getListEnrolamiento()	239
<i>Este metodo retorna el list de las personas enroladas</i>	
Enrolamiento_model::getidtutor()	239
Enrolamiento_model::getfnutricion()	238
Enrolamiento_model::getfnacimiento()	238
Enrolamiento_model::getfolio()	238
<i>Extrae el folio que se anexa en el envio para la tarjeta</i>	
Enrolamiento_model::getfvacuna()	238
Enrolamiento_model::getId()	239
Enrolamiento_model::gethemoglobina()	238
Enrolamiento_model::getcelular()	234
Enrolamiento_model::getCategoriaCIE10()	234
<i>Obtiene el listado de categorias de CIE10</i>	
Enrolamiento::validarForm()	210
<i>valida los datos de entrada en el formulario</i>	
Enrolamiento::vacunacion()	209
<i>Obtiene el historial de vacunacion de un paciente cuyo id es pasado por parametro</i>	
Enrolamiento::update_card()	209
<i>Este metodo actualiza el estado del archivo descargado si fue escrito correctamente o no en la tarjeta</i>	
Enrolamiento::validarisum()	210
<i>valida que el nodo seleccionado en el arbol sea una unidad medica</i>	
Enrolamiento::validate_card()	211
<i>valida que un archivo sea valido para enviar a la tarjeta por nfc</i>	
Enrolamiento_model	229
<i>Modelo Usuario</i>	
Enrolamiento::view()	211
<i>Crea la pagina para ver la infromacion de la persona</i>	
Enrolamiento::update()	209
<i>Crea el fromulario para editar la informacion de la persona</i>	
Enrolamiento::tratamiento_select()	208
<i>Genera los options de un campo tipo select para los tratamientos de consultas</i>	
Enrolamiento::insert()	206
<i>prepara los datos para insertarlos</i>	
Enrolamiento::index()	206
<i>Este es el metodo por default, obtiene el listado de las personas</i>	
Enrolamiento::paciente_similar()	207
<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()	207
<i>Este metodo estrae la informacion del paciente que sera impreso en la tarjeta</i>	
Enrolamiento::searchum()	208

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

Enrolamiento::searchageb()	207
<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()	230
<i>obtiene informacion del tutor para generar el autocomplete</i>	
Enrolamiento_model::cns_insert()	230
<i>Hace insert de las tablas cns_control_x que se reciben en la sincronizacion secuencial</i>	
Enrolamiento_model::getalergias()	233
Enrolamiento_model::getAlergia()	233
<i>Obtiene las alergias asociadas a una persona</i>	
Enrolamiento_model::getaltura()	233
Enrolamiento_model::getByCurp()	233
<i>valida que no se repita la curp en personas y tutor</i>	
Enrolamiento_model::getcalle()	234
Enrolamiento_model::getById()	234
<i>Obtiene la información de la persona</i>	
Enrolamiento_model::getageb()	233
Enrolamiento_model::getAfiliaciones()	232
<i>Obtiene las afiliaciones asociadas a una persona</i>	
Enrolamiento_model::cns_update_visita()	231
Enrolamiento_model::cns_update()	230
<i>Actualiza la tabla especificada</i>	
Enrolamiento_model::data_tutor()	231
<i>obtiene informacion del tutor</i>	
Enrolamiento_model::entorno_x_persona()	231
<i>Este metodo actualiza o inserta los datos que permiten el envio de la informacion a la tarjeta por nfc</i>	
Enrolamiento_model::getafiliacion()	232
Enrolamiento_model::getaccion_nutricional()	232
Entorno	70
<i>Controlador Entorno</i>	

F

Form_validation.php	323
<i>Código fuente</i>	
Form_validation.php	3
<i>CodeIgniter</i>	

G

Grupo_model::getId()	161
Grupo_model::getMsgError()	161
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Grupo_model::getEntornosById()	161
<i>Obtiene el grupo solicitado con sus entornos vinculados</i>	
Grupo_model::getDescripcion()	161
Grupo_model::getById()	160
<i>Obtiene el grupo solicitado</i>	

<u>Grupo_model::getByName()</u>	160
<i>Obtiene el grupo solicitado</i>	
<u>Grupo_model::getNombre()</u>	162
<u>Grupo_model::getNumRows()</u>	162
<i>Obtiene el numero total de grupos</i>	
<u>Grupo_model::setNombre()</u>	163
<u>Grupo_model::update()</u>	163
<i>Actualiza en la base de datos los datos del grupo (datos en propiedades)</i>	
<u>Grupo_model::setId()</u>	163
<u>Grupo_model::setDescripcion()</u>	162
<u>Grupo_model::insert()</u>	162
<i>Inserta en la base de datos los datos del grupo (datos en propiedades)</i>	
<u>Grupo_model::getAll()</u>	160
<i>Obtiene todos los grupos existentes</i>	
<u>Grupo_model::delete()</u>	159
<i>Elimina de la base de datos al grupo (id en propiedades)</i>	
<u>Grupo::index()</u>	75
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	
<u>Grupo::insert()</u>	75
1) Prepara el formulario para la inserción de un grupo nuevo	
2) Realiza las validaciones necesarias sobre cada campo del registro	
<u>Grupo::delete()</u>	75
<i>Solicita la eliminación del grupo recibido</i>	
<u>Grupo</u>	74
<i>Controlador Grupo</i>	
<u>Graph::graph_init()</u>	37
<i>Crea una grafica en el lugar que se llame</i>	
<u>Graph::map()</u>	37
<i>crea un objeto mapa con la ayuda de la api de google</i>	
<u>Grupo::update()</u>	76
1) Prepara el formulario para la modificación de un grupo existente	
2) Realiza las validaciones necesarias sobre cada campo del registro	
<u>Grupo::view()</u>	76
<i>Visualiza los datos del grupo recibido</i>	
<u>Georeferencia_model::process()</u>	159
<i>Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia</i>	
<u>Grupo_model</u>	159
<i>Modelo Grupo</i>	
<u>Georeferencia_model::getMsgError()</u>	158
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
<u>Georeferencia_model</u>	158
<i>Modelo Georeferencia</i>	
<u>Grupo:: ifGroupExists()</u>	76
<i>Callback para validar que un nombre de grupo no se duplique</i>	
<u>Graph</u>	37
<i>Controlador Objetos</i>	

H

Hemoglobina_model::process()	164
<i>Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia</i>	
Hemoglobina_model::getMsgError()	164
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Hemoglobina_model	164
<i>Modelo Hemoglobina</i>	

I

Index::index()	1
Index	1

M

Menu_model::getRuta()	169
Menu_model::hasChild()	169
<i>Verifica si el nodo actual tiene hijos</i>	
Menu_model::insert()	169
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Menu_model::getNumRows()	169
<i>Obtiene el numero total de registros en la tabla Menu en caso de existir filtros, estos son aplicados a la consulta</i>	
Menu_model::getNombre()	168
Menu_model::getId_padre()	168
Menu_model::getId_raiz()	168
Menu_model::getMsgError()	168
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Menu_model::resetFilter()	170
<i>Elimina todos los filtros registrados</i>	
Menu_model::setAtributo()	170
Menu_model::setNombre()	171
Menu_model::setRuta()	171
Menu_model::update()	172
<i>Actualiza los datos del objeto actual</i>	
Menu_model::setId_raiz()	171
Menu_model::setId_padre()	171
Menu_model::setId()	170
Menu_model::setId_controlador()	170
Menu_model::getId_controlador()	168
Menu_model::getId()	167
Menu::index()	77
<i>Lista todos los registros de la menu, con su correspondiente paginación</i>	
Menu::insert()	78
<i>Muestra el formulario para crear un nuevo registro en la menu,</i>	

<i>las variables se obtienen por el metodo POST</i>	
Menu::update()	78
<i>Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos</i>	
Menu::delete()	77
<i>Eliminar el registro especificado por el id</i>	
Menu	77
<i>Controlador Menu</i>	
Menubuilder::build()	35
<i>Construye el menu basandose en los permisos del usuario logeado</i>	
Menubuilder::crearMenu()	36
<i>Función recursiva que recorre todo el árbol de elementos del menu</i>	
Menubuilder::isGranted()	36
<i>Función que valida si se visualiza o no la accion especificada (usada en views)</i>	
Menu::view()	79
<i>Muestra los datos del registro especificado por el id</i>	
Menu_model	165
<i>Modelo Menu</i>	
Menu_model::getAtributo()	167
Menu_model::getById()	167
<i>Obtiene los datos del registro de la menu que tiene el ID especificado</i>	
Menu_model::getByPadre()	167
<i>Obtiene todos los nodos hijos de un padre</i>	
Menu_model::getAll()	166
<i>Obtiene todos los registros de la tabla Menu en caso de existir filtros, estos son aplicados a la consulta</i>	
Menu_model::deleteByFilter()	166
<i>Elimina el conjunto de registros que cumplen con el o los criterios de filtrado</i>	
Menu_model::addFilter()	165
<i>Agrega una nueva regla de filtrado al arreglo de filtros</i>	
Menu_model::delete()	166
<i>Elimina el registro actual de la base de datos</i>	
Menubuilder	35
<i>Menu Builder</i>	

N

Notificacion_model::getTitulo()	278
Notificacion_model::insert()	278
<i>Inserta en la base de datos los datos de la notificación (datos en propiedades)</i>	
Notificacion_model::getNumRows()	278
<i>Obtiene el numero total de notificaciones</i>	
Notificacion_model::getMsgError()	278
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Notificacion_model::getIdsTabletas()	277
Notificacion_model::setContenido()	279
Notificacion_model::setFechaFin()	279
Notificacion_model::setTitulo()	280
Notificacion_model::update()	280
<i>Actualiza en la base de datos los datos de la notificación (datos en propiedades)</i>	
Notificacion_model::setIdsTabletas()	280
Notificacion_model::setId()	280

Notificacion_model::setFechaInicio()	279
Notificacion_model::getId()	277
Notificacion_model::getFechaInicio()	277
Notificacion::update()	213
1) Prepara el formulario para la modificación de una notificación existente	
2) Realiza las validaciones necesarias sobre cada campo del registro	
Notificacion::view()	213
Visualiza los datos de la notificación recibida	
Notificacion::insert()	212
1) Prepara el formulario para la inserción de una notificación nueva	
2) Realiza las validaciones necesarias sobre cada campo del registro	
Notificacion::index()	212
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()	212
Solicita la eliminación de la notificación recibida	
Notificacion_model	275
Modelo Usuario	
Notificacion_model::addFilter()	275
Agrega una nueva regla de filtrado al arreglo de filtros	
Notificacion_model::getContenido()	277
Notificacion_model::getFechaFin()	277
Notificacion_model::getById()	277
Obtiene la notificación solicitada	
Notificacion_model::getAll()	276
Obtiene todas las notificaciones existentes, se puede filtrar por: texto a buscar si se desea	
Notificacion_model::delete()	276
Elimina de la base de datos la notificación (id en propiedades)	
Notificacion	211
Controlador Notificación	

O

Obtenercurp::curp()	40
Consulta si la curp existe en la base de datos de la condusef	
Obtenercurp::calcular_curp()	40
Calcula la curp y el rfc con los datos proporcionados	
Obtenercurp::calcularcurp()	39
Calcula la curp y el rfc con los datos proporcionados	
Obtenercurp::\$estados	39
Arreglo con los estados y sus abreviaturas, sirven para el cálculo de la CURP	
Obtenercurp	38
Controlador Objeto	

P

Permiso_model::setIdControladorAccion()	175
Permiso_model::setId()	175
Permiso_model::setFecha()	175
Permiso_model::insertBatch()	174

<i>Inserta en la base de datos el arreglo de permisos recibido</i>	
Permiso_model::setIdGrupo()	175
Poblacion_model	176
<i>Modelo Poblacion</i>	
Pagination.php	341
<i>Código fuente</i>	
Poblacion_model::process()	177
<i>Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion</i>	
Poblacion_model::getMsgError()	176
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Permiso_model::getPermission()	174
<i>Obtiene el permiso solicitado</i>	
Permiso_model::getMsgError()	174
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Permiso_model	172
<i>Modelo Permiso</i>	
Permiso::index()	80
<i>1) Visualiza los entornos existentes para su selección</i>	
Permiso	79
<i>Controlador Permiso</i>	
Permiso_model::deletePermissions()	173
<i>Elimina de la base de datos los permisos del entorno y grupo recibidos</i>	
Permiso_model::getFecha()	173
Permiso_model::setIdGrupo()	173
Permiso_model::getIdControladorAccion()	173
Permiso_model::getId()	173
Pagination.php	4
<i>CodeIgniter</i>	

R

Reporteador	214
<i>Controlador Reporteador</i>	
Reporteador::index()	214
<i>Visualiza los reportes existentes</i>	
ReglaVacuna_model::update()	189
<i>Actualiza el objeto actual en la base de datos</i>	
ReglaVacuna_model::setRegion()	189
ReglaVacuna_model::setOrdenEsqComp()	188
Reporteador::view()	214
<i>Renderiza la vista del reporte</i>	
Reporte_sincronizacion	215
<i>Controller Usuario</i>	
Reporte_sincronizacion::view()	216
<i>Muestra detallada por cada list del reporte de sincronizacion</i>	
Reporte_sincronizacion::lote_view()	216
<i>Muestra detallada por cada list del reporte de lote de vacunacion</i>	
Reporte_sincronizacion::lote()	215
<i>Genera el ítem del reporte de lote de vacunacion</i>	

Reporte_sincronizacion::index()	215
<i>Este es el metodo por default, crea el listado del reporte de sincronizacion</i>	
ReglaVacuna_model::setObservacionRegion()	188
ReglaVacuna_model::setIdViaVacuna()	188
ReglaVacuna_model::setDiaInicioNacido()	186
ReglaVacuna_model::setDiaInicioPrevio()	186
ReglaVacuna_model::setDiaFinPrevio()	185
ReglaVacuna_model::setDiaFinNacido()	185
ReglaVacuna_model::setAlergias()	185
ReglaVacuna_model::setDosis()	186
ReglaVacuna_model::setEsqComp()	186
ReglaVacuna_model::setIdVacunaPrevio()	188
ReglaVacuna_model::setIdVacuna()	187
ReglaVacuna_model::setId()	187
ReglaVacuna_model::setForzarAplicacion()	187
Reporteador_model	281
<i>Modelo Reporteador</i>	
Reporteador_model::getCensoNominal()	281
Reporte_censo_nominal::\$parto_multiple	285
Reporte_censo_nominal::\$sexo	285
Reporte_censo_nominal::\$nombre	285
Reporte_censo_nominal::\$fecha_nacimiento	285
Reporte_censo_nominal::\$domicilio	285
Reporte_censo_nominal::\$vacunas	286
Reporte_sincronizacion_model	286
<i>Modelo Usuario</i>	
Reporte_sincronizacion_model::get_version()	287
<i>obtiene la ultima version de la apk de las tabletas</i>	
Reporte_sincronizacion_model::getMsgError()	287
Reporte_sincronizacion_model::getListado()	287
<i>Obtiene el resultado de una consulta</i>	
Reporte_sincronizacion_model::getCount()	286
<i>obtiene el numero de registros de una tabla o consulta</i>	
Reporte_censo_nominal::\$curp	284
Reporte_censo_nominal::\$apellido_paterno	284
Reporteador_model::getGrupoVacunas()	282
Reporteador_model::getMsgError()	282
<i>Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)</i>	
Reporteador_model::getEsquemasIncompletos()	282
Reporteador_model::getConcentradoActividades()	282
Reporteador_model::getCoberturaBiologicoListado()	281
<i>Obtiene el reporte de cobertura por tipo de biologico</i>	
Reporteador_model::getSeguimientoRV1RV5()	283
Reporteador_model::getVacunas()	283
Reporte_censo_nominal::\$apellido_materno	284
Reporte_censo_nominal	284
<i>Modelo Reporte_censo_nominal</i>	
Reporteador_model::object_to_array()	283
Reporteador_model::getVacunasByGrupo()	283
ReglaVacuna_model::insert()	185
<i>Inserta en la tabla regla_vacuna la información contenida en el objeto</i>	
ReglaVacuna_model::getRegion()	184
ReglaVacuna::update()	86

<i>Acción para preparar la actualización de una regla ya existente,</i>	87
ReglaVacuna::view()	87
<i>Acción para visualizar información de una regla específica, obtiene el objeto regla_vacuna por medio del id proporcionado.</i>	
ReglaVacuna::insert()	86
<i>Acción para preparar la insercion de nuevas reglas , realiza la validación del formulario del lado cliente</i>	
ReglaVacuna::index()	86
<i>Acción por default del controlador, carga la lista de reglas de vacunas disponibles y una lista de opciones</i>	
<i>No recibe parámetros</i>	
ReglaVacuna::delete()	85
<i>Acción para eliminar una regla, recibe el id de la regla a eliminar</i>	
Raiz_model	177
<i>Modelo Raiz</i>	
Raiz_model::delete()	177
<i>Elimina el registro actual de la base de datos</i>	
Raiz_model::getDescription()	179
Raiz_model::getByld()	178
<i>Devuelve la información de una raiz por su ID</i>	
Raiz_model::getAll()	178
<i>Devuelve todos los registros de la tabla raiz</i>	
Raiz_model::ExistInArbol()	178
<i>Revisa si la raiz pasada como parametro existe en el ASU</i>	
ReglaVacuna	85
<i>Controlador ReglaVacuna</i>	
Raiz::view()	84
<i>Acción para visualizar información de una raiz específica, obtiene el objeto raiz por medio del id proporcionado.</i>	
Raiz::getDataKeyValue()	82
<i>Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro</i>	
Raiz::getDataTreeFromId()	82
<i>Accion para regresar la descripción e informacion adicional de un arreglo de ID's desde el arbol de segmentacion</i>	
Raiz::getChildrenFromLevel()	81
<i>Accion para regresar el arbol de segmentacion determinado, el objeto regresado contiene estructura de arbol y es consumida solamente por peticiones AJAX</i>	
Raiz::delete()	81
<i>Acción para eliminar una raiz, recibe el id de la raiz a eliminar</i>	
Raiz::createasu()	80
<i>Acción para crear el ASU a partir de una raiz</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Raiz::getTreeBlock()	82
<i>Sirve para obtener bloques del arbol de segmentación única ASU</i>	
Raiz::index()	83
<i>Acción por default del controlador, carga la lista de Raices disponibles y una lista de opciones</i>	
<i>No recibe parámetros</i>	
Raiz::updateasu()	84
<i>Acción para actualizar el ASU a partir de una raiz</i>	
<i>Solo se permite su acceso por medio de peticiones AJAX</i>	
Raiz::update()	84

<i>Acción para preparar la actualización de una raiz ya existente,</i>	83
Raiz::insert()	83
<i>Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente</i>	
Raiz::iniciarasu()	83
<i>Crea los archivos JSON necesarios para iniciar el ASU en caché y agilizar su carga</i>	
Raiz_model::getId()	179

Raiz_model::getMsgError()	179
<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>	
ReglaVacuna_model::getForzarAplicacion()	183
ReglaVacuna_model::getId()	183

ReglaVacuna_model::getEsqComp()	183
ReglaVacuna_model::getDosis()	183
ReglaVacuna_model::getDialInicioPrevia()	182
ReglaVacuna_model::getIdVacuna()	183
ReglaVacuna_model::getIdVacunaPrevia()	183
ReglaVacuna_model::getOrdenEsqComp()	184
ReglaVacuna_model::getObservacionRegion()	184
ReglaVacuna_model::getMsgError()	184
<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>	
ReglaVacuna_model::getIdViaVacuna()	183
ReglaVacuna_model::getDialInicioNacido()	182
ReglaVacuna_model::getDiaFinPrevia()	182
Raiz_model::update()	180
<i>Actualiza el objeto actual en la base de datos</i>	
Raiz_model::setId()	180
Raiz_model::setDescripcion()	180
Raiz_model::insert()	179
<i>Inserta en la tabla raiz, la información contenida en el objeto</i>	
ReglaVacuna_model	181
<i>Modelo ReglaVacuna</i>	
ReglaVacuna_model::delete()	181
<i>Elimina el registro actual de la base de datos</i>	
ReglaVacuna_model::getDiaFinNacido()	182
ReglaVacuna_model::getByld()	182
<i>Devuelve la información de una regla de vacuna por su ID</i>	
ReglaVacuna_model::getAll()	181
<i>Devuelve todos los registros de la tabla regla_vacuna</i>	
ReglaVacuna_model::getAlergias()	181
Raiz	80
<i>Controlador Raiz</i>	

S

Servicios::ss_step_6()	223
<i>prepara los datos para la sincronizacion secuencia, envia unicamente aquellos datos</i>	

<i>modificados despues de la ultima sincronizacion de la tableta</i>	
Servicios::ss_step_5()	223
<i>Recibe los datos que genero la tableta para ser almacenados en la base de datos del servidor</i>	
Servicios::Synchronization()	224
<i>Metodo principal al que se le hacen las peticiones y es el que se encarga de distribuir la informacion</i>	
Semana_nacional_model	288
<i>Modelo Tableta</i>	
Semana_nacional_model::delete()	288
<i>Elimina el registro actual de la base de datos</i>	
Servicios::prueba2()	223
Servicios::is_step_4()	222
<i>Prepara la informacion de perssonas con su catalogos transaccionales de cada una</i>	
Servicios::is_step_0()	220
<i>Paso 0 se procesa las peticiones segun la accion:</i>	
Servicios::is_step_1()	221
<i>valida que la tableta este asignada a una unidad medica y que tenga un status valido para la sincronizacion</i>	
Servicios::is_step_2()	221
<i>Valida que la session este activa, genera los catalogos a enviar en la sincronizacion por primera vez</i>	
Servicios::is_step_3()	222
<i>Guarda en la base de datos el estado de la sincronizacion</i>	
Semana_nacional_model::getAll()	288
<i>Obtiene todos los registros de la tabla</i>	
Semana_nacional_model::getById()	289
<i>Obtiene los datos del registro que tiene el ID especificado</i>	
Semana_nacional_model::setDescription()	291
Semana_nacional_model::insert()	290
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Semana_nacional_model::setFecha_fin()	291
Semana_nacional_model::setFecha_inicio()	291
Semana_nacional_model::update()	291
<i>Actualiza los datos del objeto actual</i>	
Semana_nacional_model::getNumRows()	290
<i>Obtiene el numero total de registros en la tabla</i>	
Semana_nacional_model::getMsgError()	290
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Semana_nacional_model::getDescription()	289
Semana_nacional_model::getFecha_fin()	289
Semana_nacional_model::getFecha_inicio()	289
Semana_nacional_model::getId()	290
Servicios::esquema_incompleto()	220
<i>Genera los esquemas incompletos de las personas que correspondan a la unidad medica de la tableta</i>	
Servicios::catalogos_relevantes()	220
<i>Genera los catalogos relevantes por entorno</i>	
Session::is_expired()	45
<i>Check if session is expired</i>	
Session::flashdata()	45

<i>Fetch a specific flashdata item from the session array</i>	
Session::keep_flashdata()	45
<i>Keeps existing flashdata available to next request.</i>	
Session::sess_create()	46
<i>Create Session</i>	
Session::sess_destroy()	46
<i>Destroy session</i>	
Session::all_userdata()	44
<i>Fetch all session data</i>	
Session::\$store	44
Session::\$ci	43
Session::\$flashdata_key	43
Session::\$sess_expiration	44
Session::\$sess_namespace	44
Session::set_flashdata()	46
<i>Add or change flashdata, only available until the next request</i>	
Session::set_userdata()	46
<i>Set value for specific user data element</i>	
Semana_nacional::update()	218
<i>Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos</i>	
Semana_nacional::insert()	218
<i>Muestra el formulario para crear un nuevo registro en la semana_nacional, las variables se obtienen por el metodo POST</i>	
Semana_nacional::view()	218
<i>Muestra los datos del registro especificado por el id</i>	
Servicios	219
<i>Controlador Servicios</i>	
Servicios::actualiza_estado_tableta()	219
<i>Actualiza el estatus de la tableta</i>	
Semana_nacional::index()	217
<i>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</i>	
Semana_nacional::getAll()	217
<i>Devuelve un json con todos los registros de semanas nacional</i>	
Session::unset_userdata()	47
<i>remove array value for specific user data element</i>	
Session::userdata()	47
<i>Get specific user data element</i>	
Semana_nacional	216
<i>Controlador Semana Nacional</i>	
Semana_nacional::delete()	217
<i>Eliminar el registro especificado por el id</i>	
Session	43
<i>CodeIgniter Native Session Library</i>	

T

Tableta_model::setId tes estado tableta()	297
Tableta_model::setId asu um()	296

Tableta_model::setId tipo censo()	297
Tableta_model::setMac()	297
Tableta_model::setPeriodo esq inc()	298
Tableta_model::setIdVersion()	296
Tableta_model::setId()	296
Tableta_model::getPeriodo esq inc()	295
Tableta_model::getUltima actualizacion()	295
Tableta_model::getUsuarios asignados()	296
Tableta_model::insert()	296
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Tableta_model::setUltima actualizacion()	298
Tableta_model::setUsuarios asignados()	298
Tipo censo_model::getMsgError()	300
<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()	301
Tipo censo_model::setId()	301
template.php	346
<i>Código fuente</i>	
Tipo censo_model::getId()	300
Tipo censo_model::getDescripcion()	300
Tableta_model::update()	298
<i>Actualiza los datos del objeto actual</i>	
Tipo censo_model	299
<i>Modelo Tipo_censo</i>	
Tipo censo_model::getAll()	299
<i>Obtiene todos los registros de la tabla</i>	
Tipo censo_model::getById()	300
<i>Obtiene los datos del registro que tiene el ID especificado</i>	
Tableta_model::getNumRows()	295
<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()	295
<i>Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false</i>	
Tableta::setUM()	226
<i>Asignar unidad medica y tipo de censo</i>	
Tableta::update()	226
<i>Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos</i>	
Tableta::uploadFile()	227
<i>Registra tabletas desde un archivo csv</i>	
Tableta::view()	227
<i>Muestra los datos del registro especificado por el id</i>	
Tableta::insert()	226
<i>Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST</i>	
Tableta::index()	225
<i>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 especifico</i>	

Tree	Controlador Objeto	41
Tree::create()	Crea el arbol y lo muestra en la view	41
Tableta	Controlador Tableta	224
Tableta::delete()	Eliminar el registro especificado por el id	225
Tableta::validateMac()	Valida si existe una MAC en la base de datos	227
Tableta model	Modelo Tableta	292
Tableta model::getId asu um()		294
Tableta model::getId tes estado tableta()		294
Tableta model::getId tipo censo()		294
Tableta model::getMac()		294
Tableta model::getIdVersion()		294
Tableta model::getId()		294
Tableta model::delete()	Elimina el registro actual de la base de datos	292
Tableta model::getAll()	Obtiene todos los registros de la tabla	293
Tableta model::getById()	Obtiene los datos del registro que tiene el ID especificado	293
Tableta model::getByMac()	Obtiene los datos del registro la MAC especificada	293
template.php	CodeIgniter	5

U

Usuario model::setApellidoMaterno()		197
Usuario model::setActivo()		197
Usuario model::insert()	Inserta en la base de datos los datos del usuario (datos en propiedades)	196
Usuario model::setApellidoPaterno()		197
Usuario model::setClave()		197
Usuario model::setIdGrupo()		198
Usuario model::setId()		198
Usuario model::setCorreo()		198
Usuario model::get usuario entorno()		196
Usuario model::get permiso entorno()		196
Usuario model::getNombreUsuario()		194
Usuario model::getNombre()		194
Usuario model::getMsgError()	Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)	194
Usuario model::getNumRows()	Obtiene el numero total de usuarios	194
Usuario model::getOnlyActives()	Obtiene todos los usuarios existentes, se puede filtrar por: texto a buscar o solo activos si se desea	195
Usuario model::get_grupo entorno()		196

Usuario model::getuser()	195
Usuario model::setNombre()	199
Usuario model::setNombreUsuario()	199
Usuario tableta model::getTabletasByUsuario()	303
<i>Obtiene el id de todas las tabletas relacionadas con el usuario especificado</i>	
Usuario tableta model::getTableta()	302
Usuario tableta model::getMsgError()	302
<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()	303
Usuario tableta model::getUsuariosByTableta()	303
<i>Obtiene el id de todos los usuarios asignados a la tableta especificada</i>	
Usuario tableta model::setUsuario()	304
Usuario tableta model::setTableta()	304
Usuario tableta model::insert()	303
<i>Inserta en la base de datos, la informacion contenida en el objeto</i>	
Usuario tableta model::delete()	302
<i>Elimina la relacion entre usuario y tableta</i>	
Usuario tableta model	301
<i>Modelo Estado_tableta</i>	
Usuario model::update_user()	200
Usuario model::update_pass()	199
Usuario model::update()	199
<i>Actualiza en la base de datos los datos del usuario (datos en propiedades)</i>	
Usuario tableta	228
<i>Controlador Usuario_tableta</i>	
Usuario tableta::delete()	228
<i>Eliminar un usuario de una tableta especifica</i>	
Usuario tableta::insert()	229
<i>Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST</i>	
Usuario tableta::index()	228
<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()	194
Usuario model::getId()	193
Usuario::login()	90
<i>Ofrece el inicio de sesión</i>	
Usuario::load_update()	90
Usuario::insert()	89
1) Prepara el formulario para la inserción de un usuario nuevo	
2) Realiza las validaciones necesarias sobre cada campo del registro	
Usuario::logout()	90
<i>Termina la sesión</i>	
Usuario::remember()	90
Usuario::token()	91
Usuario::send_mail()	91
Usuario::reset()	90
Usuario::index()	89
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	

<i>búsqueda</i>	89
Usuario::get_token()	89
<i>Acción para obtener el token oculto en el login</i>	
Usuario::cerrar_etab()	88
<i>Acción para cerrar el login en otro sistema</i>	
Usuario::automatic_access()	88
Usuario::\$mas	87
<i>Acción para hacer autologin en otro sistema</i>	
Usuario::delete()	88
<i>Solicita la eliminación del usuario recibido</i>	
Usuario::form_init()	88
Usuario::get_galleta()	89
<i>Obtiene las cookies que se generan en la session</i>	
Usuario::getActivesByGroup()	88
<i>Acción para servir un array de objetos con los usuarios activos por grupo AJAX y devuelve un objeto JSON</i>	
Usuario::update()	91
<i>1) Prepara el formulario para la modificación de un usuario existente 2) Realiza las validaciones necesarias sobre cada campo del registro</i>	
Usuario::update_info()	92
Usuario_model::getApellidoPaterno()	192
Usuario_model::getApellidoMaterno()	192
Usuario_model::getActivo()	192
Usuario_model::getById()	192
<i>Obtiene el usuario solicitado, se puede obtener el registro normal o personalizado para visualización (descripciones en tablas vinculadas)</i>	
Usuario_model::getByUsername()	193
<i>Obtiene el usuario solicitado</i>	
Usuario_model::getgrupo()	193
Usuario_model::getCorreo()	193
Usuario_model::getClave()	193
Usuario_model::getActivesByGroup()	191
<i>Obtiene los usuarios activos del grupo solicitado</i>	
Usuario_model::delete()	191
<i>Elimina de la base de datos al usuario (id en propiedades)</i>	
Usuario_model	189
<i>Modelo Usuario</i>	
Usuario::ifUserExists()	92
<i>Callback para validar que un nombre de usuario no se duplique</i>	
Usuario::view()	92
<i>Visualiza los datos del usuario recibido</i>	
Usuario_model::authenticate()	190
<i>Valida las credenciales recibidas</i>	
Usuario_model::checkCredentials()	190
<i>Verifica que el usuario haya iniciado sesión y además tenga permiso en la acción recibida</i>	
Usuario_model::check_token()	191
Usuario_model::check_data()	190
Usuario	87
<i>Controlador Usuario</i>	